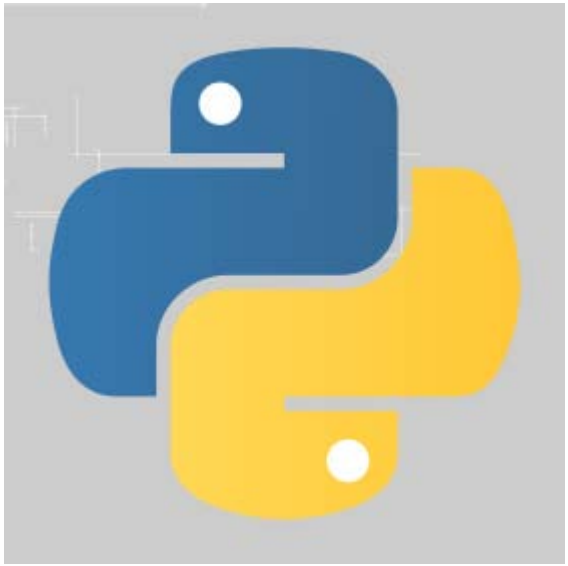
ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΡΥΤΗΟΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΠΙΒΛΕΠΩΝ: ΚΩΝ/ΝΟΣ ΓΙΑΚΟΥΣΤΙΔΗΣ
ΚΑΘΗΓΗΤΗΣ ΕΦΑΡΜΟΓΩΝ

ΜΙΝΤΗ ΦΑΝΗ
05/2890



2012

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

1. ΕΙΣΑΓΩΓΗ ΣΤΗ PYTHON

1.1 Εισαγωγή

Στο πρώτο κεφάλαιο αναφέρομαι στην ιστορική εξέλιξη της Python και τους λόγους που εξελίχθηκε με το πέρασμα του χρόνου στην Python3, με την οποία και θα ασχοληθώ. Παραθέτω τα χαρακτηριστικά της γλώσσας που αποτελούν ταυτόχρονα και πλεονεκτήματά της και τα βήματα που πρέπει να ακολουθήσουμε για να την εγκαταστήσουμε σε Windows, στα οποία και δουλεύω. Θεώρησα ακόμη χρήσιμο να περιγράψω το πώς μπορούμε να δημιουργήσουμε, να αποθηκεύσουμε και να τρέξουμε ένα πρόγραμμα, το κλασσικό, σύντομο πρόγραμμα Hello World, ώστε να έχουμε μία αρχική ιδέα του πώς χρησιμοποιείται το Python κέλυφος (Python Shell) και επιπλέον τα πλεονεκτήματα που έχει αυτός ο ενσωματωμένος επεξεργαστής κώδικα της Python. Τέλος απαριθμώ και περιγράφω σύντομα, οχτώ βασικά σημεία της γλώσσας, απαραίτητα για τη σύνταξη προγραμμάτων, τα οποία θα αναλύσω λεπτομερέστερα στα επόμενα κεφάλαια που θα ακολουθήσουν.

1.2 Ιστορικό

Η Python είναι μία γλώσσα προγραμματισμού, η οποία δημιουργήθηκε από τον Ολλανδό Guido van Rossum το 1990. Το όνομα της γλώσσας προέρχεται από την ομάδα των άγγλων κωμικών Monty Python. Αρχικά χρησιμοποιήθηκε σαν γλώσσα σεναρίων στο λειτουργικό σύστημα Amoeba, ικανή και για κλήσεις συστήματος από τα προγράμματα χρήστη.

1.3 Εξέλιξη σε Python3

Η πρώτη έκδοση της Python ήταν η Python 1.5.2. Με το πέρασμα του χρόνου, αποφασίστηκε πως πρέπει να γίνει μία σημαντική αλλαγή στον τρόπο που η Python χειρίζεται κάποια από τα χαρακτηριστικά της. Για να γίνει αυτή η αλλαγή, έπρεπε να χωριστούν τα χαρακτηριστικά σε αυτά που μπορούσαν να υλοποιηθούν με έναν συμβατό προς τα πίσω τρόπο κ α σε αυτά που δεν μπορούσαν. Τα πρώτα χαρακτηριστικά αποτέλεσαν την έκδοση Python 2.x, ενώ τα δεύτερα αποτέλεσαν λίγο αργότερα την έκδοση Python 3.x.

Η Python 3 είναι ιστορικά η πρώτη γλώσσα προγραμματισμού που σπάει την προς τα πίσω συμβατότητα με τις προηγούμενες εκδόσεις, ώστε να διορθωθούν κάποια λάθη που υπήρχαν σε αυτές. Αυτό σημαίνει πως κώδικας γραμμένος σε παλαιότερες εκδόσεις της Python δεν μπορεί να εκτελεστεί σωστά στην Python 3, αν δε γίνουν κάποιες αλλαγές σε αυτόν.

Ο κύριος στόχος της Python 3 είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της. Διακρίνεται λόγω του ότι έχει πολλές βιβλιοθήκες που διευκολύνουν ιδιαίτερα, αρκετές συνηθισμένες εργασίες και για την ταχύτητα εκμάθησης της.

1.4 Χαρακτηριστικά της Python

- Απλή (Simple)

Η Python είναι μία απλή και μινιμαλιστική γλώσσα προγραμματισμού. Ένα από τα πιο ισχυρά της σημεία είναι η ομοιότητά της με ψευδοκώδικα που επιτρέπει σε έναν προγραμματιστή να συγκεντρώνεται στη λύση του προβλήματος και όχι στην ίδια τη γλώσσα.

- Εύκολη στην εκμάθηση (Easy to learn)

Οι προγραμματιστές που είναι εξοικειωμένοι με τις παραδοσιακές γλώσσες προγραμματισμού είναι εύκολο να μάθουν και την Python κυρίως γιατί α) οι τύποι συσχετίζονται με αντικείμενα και όχι μεταβλητές, β) η Python λειτουργεί σε ένα υψηλό επίπεδο αφαίρεσης λόγω του τρόπου με τον οποίο είναι δομημένη και γ) οι συντακτικοί κανόνες είναι πολύ απλοί.

- Ελεύθερη και ανοιχτού κώδικα (Free and Open Source)

Η Python είναι παράδειγμα Ελεύθερου Λογισμικού και Λογισμικού Ανοιχτού Κώδικα (ΕΛΛΑΚ). Αυτό σημαίνει πως η Python και ο πηγαίος κώδικάς της, είναι διαθέσιμα δωρεάν και έτσι μπορούμε να την τροποποιήσουμε, να τη βελτιώσουμε και να την επεκτείνουμε, σε αντίθεση με ιδιόκτητα λογισμικά στα οποία το κόστος για να γίνουν όλα τα παραπάνω είναι μεγάλο.

- Γλώσσα υψηλού επιπέδου (High-level language)

Όταν γράφουμε προγράμματα σε Python δεν πρέπει να ανησυχούμε για τις χαμηλού επιπέδου λεπτομέρειες όπως η διαχείριση της μνήμης που χρησιμοποιείται από το πρόγραμμά μας.

- Φορητή (Portable)

Επειδή η Python είναι ανοιχτού κώδικα γλώσσα προγραμματισμού, τα προγράμματα σε Python μπορούν να υλοποιηθούν σε πολλές πλατφόρμες. Συγκεκριμένα στο Linux, στα Windows, στο FreeBSD, σε Macintosh, στο Solaris, στο OS/2, στην Amiga, στο AROS, στο AS/400, στο BeOS, στο OS/390, στο z/OS, στο Palm OS, στο QNX, στο VMS, στο Psion, στο Acorn RISC OS, στο VxWorks, σε PlayStation, στο Sharp Zaurus, στα Windows CE και σε PocketPC.

- Διερμηνευόμενη (Interpreted)

Όταν γράφουμε ένα πρόγραμμα σε Python αυτό δεν χρειάζεται μεταγλώττιση, δηλαδή μετατροπή σε δυαδικό κώδικα που κατανοεί ο υπολογιστής. Απλά τρέχουμε το πρόγραμμα απ' ευθείας από τον πηγαίο κώδικα. Εσωτερικά η Python μετατρέπει τον πηγαίο κώδικα σε μία ενδιάμεση μορφή που ονομάζεται bytecode, το μεταφράζει στη γλώσσα του υπολογιστή και το τρέχει. Άρα δεν χρειάζεται να ανησυχούμε για τη μεταγλώττιση του προγράμματος και τη σύνδεση με τις κατάλληλες βιβλιοθήκες και επιπλέον τα προγράμματα είναι εξαιρετικά φορητά. Αυτό δεν συμβαίνει με προγράμματα που είναι γραμμένα σε μεταγλωττιζόμενες γλώσσες όπως η C και C++, όπου η μετατροπή σε δυαδικό κώδικα γίνεται με χρήση ενός μεταγλωττιστή με διάφορες σημαίες και επιλογές. Όταν τρέχουμε το πρόγραμμα ο συνδέτης αντιγράφει το πρόγραμμα στη μνήμη και αρχίζει να το τρέχει.

- Αντικειμενοστραφής (Object Oriented)

Η Python υποστηρίζει τόσο τον διαδικασιοστρεφή προγραμματισμό (procedure-oriented programming), όπου το πρόγραμμα δομείται πάνω σε διαδικασίες ή συναρτήσεις, δηλαδή επαναχρησιμοποιήσιμα κομμάτια κώδικα, όσο και τον αντικειμενοστραφή προγραμματισμό (object-oriented programming), όπου τα προγράμματα δομούνται πάνω σε αντικείμενα, τα οποία συνδυάζουν δεδομένα και λειτουργικότητα.

- Επεκτάσιμη (Extensible)

Η Python μας δίνει τη δυνατότητα να ενσωματώσουμε σε ένα πρόγραμμα Python κομμάτια κώδικα γραμμένα σε άλλη γλώσσα, όπως C/C++ και Java.

- Ενσωματώσιμη (Embeddable)

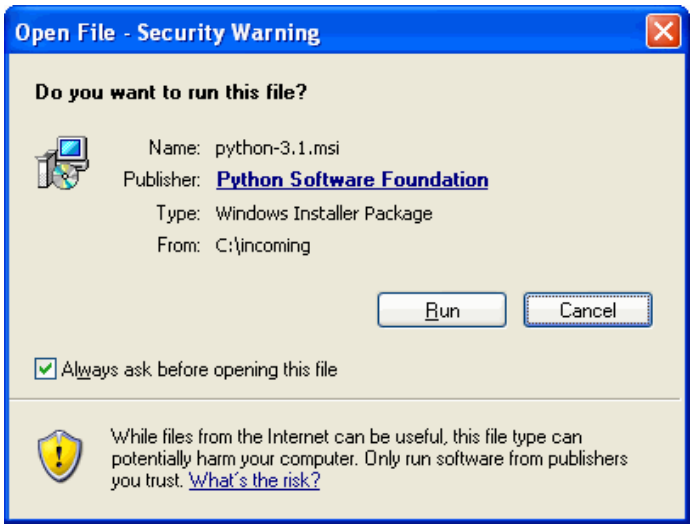
Το χαρακτηριστικό αυτό της Python μας δίνει τη δυνατότητα να ενσωματώσουμε κομμάτια κώδικα Python σε προγράμματα γραμμένα σε άλλη γλώσσα, όπως C/C++ και Java.

- Εκτεταμένες Βιβλιοθήκες (Extensive Libraries)

Η λεγόμενη 'Batteries Included' φιλοσοφία της Python υποστηρίζει ότι με την εγκατάσταση της Python πρέπει να μας παρέχεται οτιδήποτε θα χρειαστούμε για να δουλέψουμε, χωρίς να χρειάζεται να εγκαταστήσουμε πρόσθετες βιβλιοθήκες. Επομένως με την εγκατάσταση της Python παρέχεται και η Πρότυπη βιβλιοθήκη της, η οποία μας βοηθάει στη διαχείριση e-mail, ιστοσελίδων, βάσεων δεδομένων, και σε πράγματα που έχουν να κάνουν με κανονικές εκφράσεις, δημιουργία τεκμηρίωσης, δοκιμές μονάδων, νημάτωση, περιηγητές ιστού, CGI, FTP, email, XML, XML-RPC, HTML, αρχεία WAV, κρυπτογράφηση, ανάπτυξη γραφικών περιβαλλόντων (GUI), Tk και άλλα. Υπάρχουν ακόμη, επιπλέον χρήσιμες βιβλιοθήκες υψηλής ποιότητας όπως η wxPython, η Twisted και η Imaging Library.

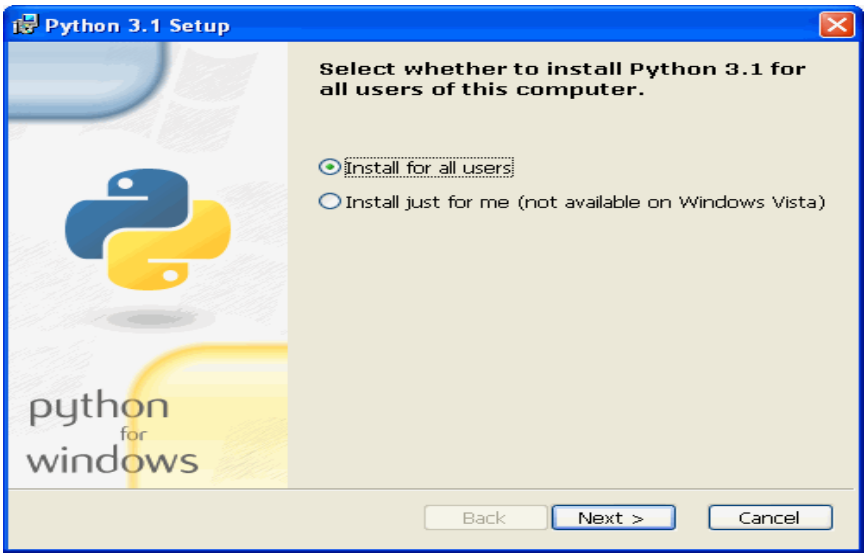
1.5 Εγκατάσταση της Python σε Windows

Για την εγκατάσταση της Python πρέπει να ακολουθήσουμε κάποια συγκεκριμένα βήματα: Στη σελίδα www.python.org, στην επιλογή download, επιλέγουμε την κατάλληλη έκδοση της Python - εγώ επέλεξα την έκδοση 3.1.3 με την οποία θα δουλέψω - για εγκατάσταση σε Windows και προσέχουμε την επιλογή της αρχιτεκτονικής (32-bit ή 64-bit). Αφού κάνουμε τη σωστή επιλογή και αφού ολοκληρωθεί η λήψη, κάνουμε διπλό κλικ στο αρχείο .msi και τα Windows εμφανίζουν μία προειδοποίηση ασφαλείας, δεδομένου ότι είμαστε έτοιμοι να τρέξουμε εκτελέσιμο κώδικα. Πατάμε στο κουμπί Εκτέλεση για να ξεκινήσει η εγκατάσταση της Python 3.



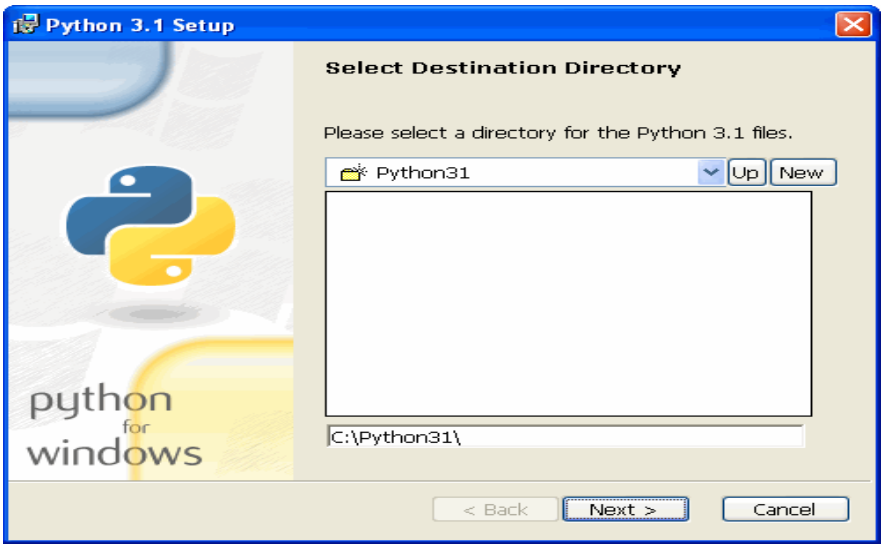
Εικόνα 1.1

Στη πρώτη σελίδα του προγράμματος εγκατάστασης εμφανίζονται δύο επιλογές, σχετικά με το αν θέλουμε να εγκαταστήσουμε τη Python 3 για όλους τους χρήστες (προεπιλογή) ή μόνο για εμάς. Αφήνουμε την προεπιλογή και πατάμε το κουμπί επόμενο.



Εικόνα 1.2

Η επόμενη σελίδα μας προτρέπει να επιλέξουμε φάκελο προορισμού. Η προεπιλογή για όλες τις εκδόσεις της Python 3.1.x είναι C:\Python31\, αλλά μπορούμε να επιλέξουμε έναν άλλο φάκελο προορισμού και εκτός του δίσκου C. Πατάμε το κουμπί Επόμενο.



Εικόνα 1.3

Εδώ έχουμε την επιλογή να εγκαταστήσουμε μόνο κάποια από τα συστατικά της Python 3. Εάν ο χώρος στο δίσκο είναι λίγος τότε μπορούμε να εξαιρέσουμε ορισμένα συστατικά.

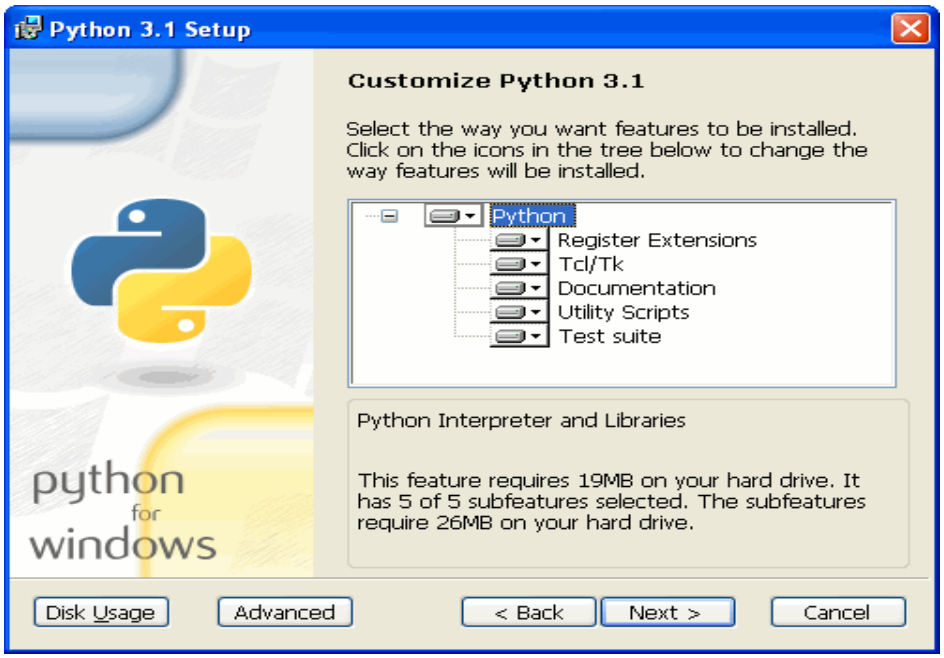
Η επιλογή *Register Extensions* μας επιτρέπει να κάνουμε διπλό κλικ σε Python αρχεία και να τα τρέξουμε. Συνίσταται αλλά δεν απαιτείται.

Η επιλογή *Tcl/Tk* είναι η βιβλιοθήκη γραφικών που χρησιμοποιείται από το κέλυφος της Python, η οποία πρέπει να εγκατασταθεί.

Η επιλογή *Documentation* εγκαθιστά ένα αρχείο βοήθειας που περιλαμβάνει πολλές από τις πληροφορίες του docs.python.org. Συνίσταται σε σύνδεση dial-up ή αν έχουμε περιορισμένη πρόσβαση στο internet.

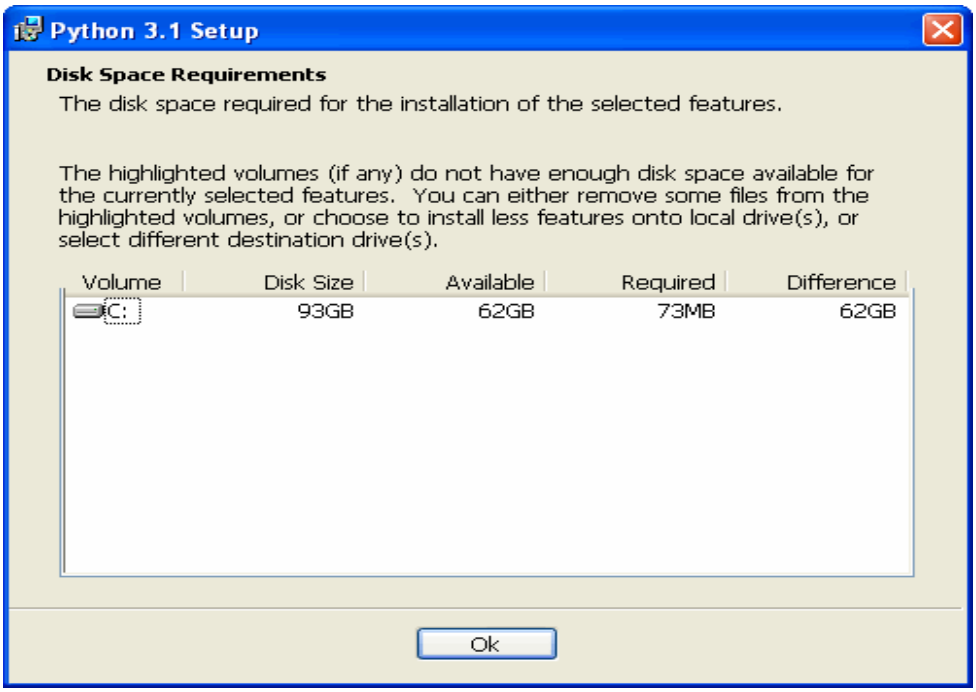
Η επιλογή *Utility Scripts* περιλαμβάνει το αρχείο 2to3.py το οποίο απαιτείται αν θέλουμε να συγχωνέψουμε υπάρχοντα κώδικα Python 2 σε Python 3.

Η επιλογή *Test Suite* είναι μία συλλογή από scripts που χρησιμοποιούνται για να δοκιμάζεται από μόνος του ο Python διερμηνευτής. Επιλογή εντελώς προαιρετική



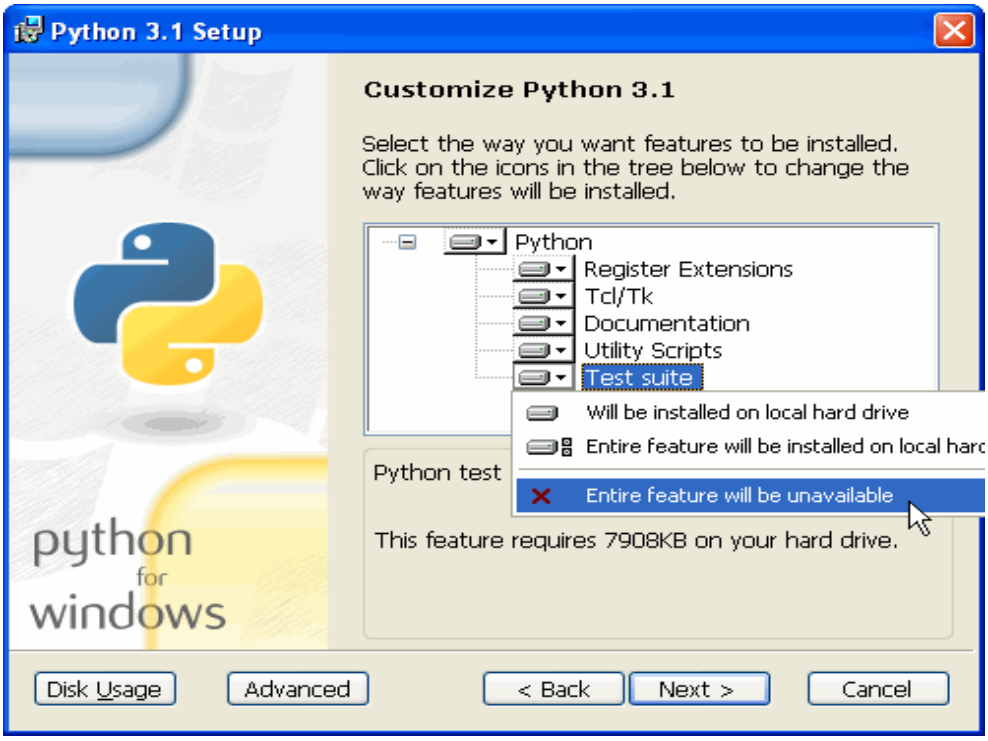
Εικόνα 1.4

Κάνουμε κλικ στο κουμπί Disk Usage όταν δεν είμαστε σίγουροι για τον ελεύθερο χώρο του δίσκου μας. Οπότε και εμφανίζονται με τη σειρά τα γράμματα των δίσκων, πόσος χώρος είναι διαθέσιμος στον καθέναν και πόσος χώρος θα απομείνει διαθέσιμος μετά την εγκατάσταση. Πατάμε το κουμπί OK για να επιστρέψουμε στο παράθυρο διαλόγου Customize Python.



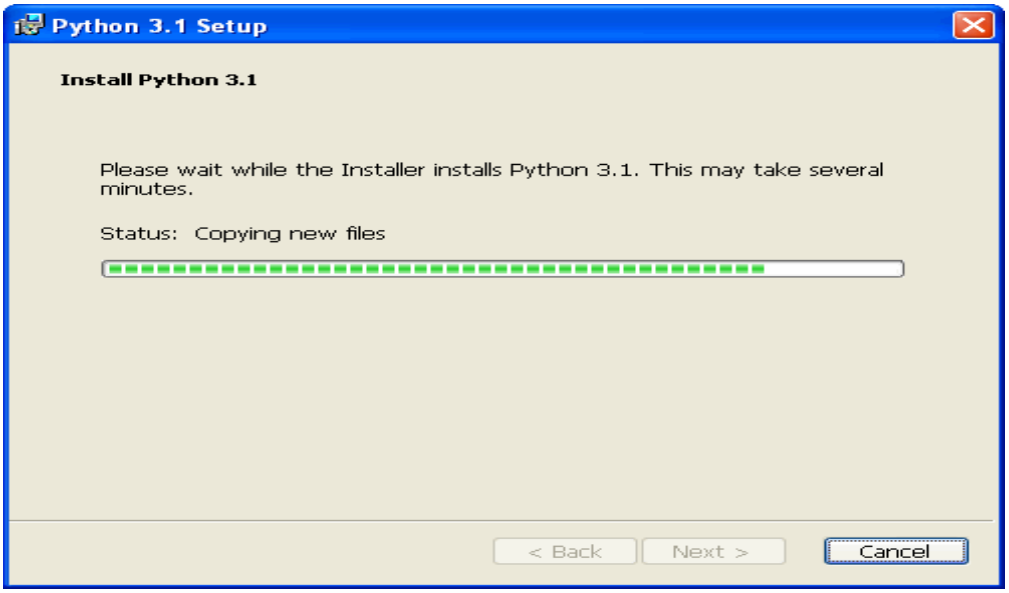
Εικόνα 1.5

Εάν αποφασίσουμε να εξάγουμε μία επιλογή, επιλέγουμε το αναδιπλωμένο κουμπί πριν την επιλογή και μετά «Entire feature will be unavailable». Πατάμε το κουμπί Επόμενο.



Εικόνα 1.6

Το πρόγραμμα εγκατάστασης θα αντιγράψει όλα τα απαραίτητα αρχεία στον φάκελο προορισμού.

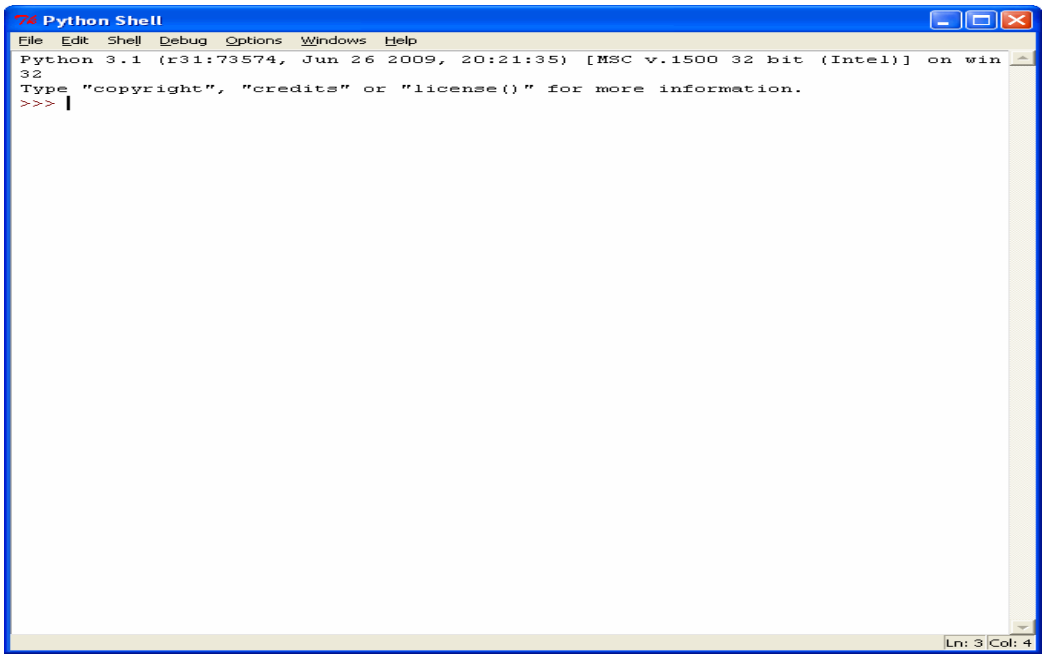


Εικόνα 1.7

Κάνουμε κλικ στο κουμπί Finish για να βγούμε από το πρόγραμμα εγκατάστασης.

Εικόνα 1.8

Στο μενού Έναρξη, όλα τα προγράμματα, στην Python 3.1, επιλέγουμε το IDLE για να τρέξουμε το διαδραστικό κέλυφος της Python (Python Shell).



Εικόνα 1.9

1.6 Επιλογή επεξεργαστή

Για τη συγγραφή προγραμμάτων Python χρειαζόμαστε έναν επεξεργαστή για να γράψουμε τον κώδικα. Η επιλογή ενός καλού επεξεργαστή βοηθάει στην εύκολη και ασφαλή συγγραφή προγραμμάτων. Μία από τις βασικές απαιτήσεις είναι η χρωματική επισήμανση της σύνταξης, όπου όλα τα διαφορετικά τμήματα του προγράμματος χρωματίζονται κατάλληλα, ώστε να μπορούμε να δούμε το πρόγραμμα και να έχουμε μία εικόνα της εκτέλεσής του. Αυτό βοηθάει και στον εντοπισμό συντακτικών λαθών στον κώδικα, αν δεν έχει χρωματιστεί σωστά μία λέξη ή εντολή.

Στα Windows η καλύτερη επιλογή επεξεργαστή είναι το IDLE, που εγκαθίσταται εξορισμού από το πρόγραμμα εγκατάστασης της Python και αυτό είναι που θα χρησιμοποιήσω. Με το IDLE μπορούμε να εξερευνήσουμε τη σύνταξη της Python, να χρησιμοποιήσουμε τις εντολές για βοήθεια και να εντοπίσουμε σφάλματα σε σύντομα προγράμματα. Επιπλέον υποστηρίζει το χρωματισμό της σύνταξης.

Θα πρέπει να αποφύγουμε τη χρήση του Notepad, επειδή δεν κάνει συντακτική επισήμανση και δεν υποστηρίζει τη στοίχιση του κειμένου.

Για έμπειρους προγραμματιστές δύο από τους πιο ισχυρούς επεξεργαστές κώδικα είναι ο Vim και ο Emacs.

1.7 Χρήση του IDLE

Σε αυτήν την ενότητα αναφέρομαι στη διαδικασία που πρέπει να ακολουθήσουμε στον IDLE επεξεργαστή ώστε να εκτελέσουμε εντολές Python αλλά και να δημιουργήσουμε, να αποθηκεύσουμε και να εκτελέσουμε ένα πρόγραμμα

Ο πρώτος τρόπος με τον οποίο μπορούμε να χρησιμοποιήσουμε το IDLE, είναι να εισάγουμε εντολές σε Python και πατώντας το πλήκτρο Enter να δούμε το αποτέλεσμα (έξοδο) άμεσα στην επόμενη γραμμή. Για να κλείσουμε την κονσόλα, πατάμε ctrl-d (σύμβολο EOF - End Of File).

Ο δεύτερος τρόπος είναι να δημιουργήσουμε ένα αρχείο. Επιλέγουμε File → New Window, εισάγουμε μία ακολουθία εντολών που αποτελούν το πρόγραμμά μας και έπειτα επιλέγουμε File → Save για να σώσουμε το αρχείο και ταυτόχρονα του

δίνουμε ένα όνομα. Τα αρχεία σε Python έχουν την κατάληξη .py. Για να τρέξουμε ένα πρόγραμμα σε Python υπάρχουν δύο τρόποι:

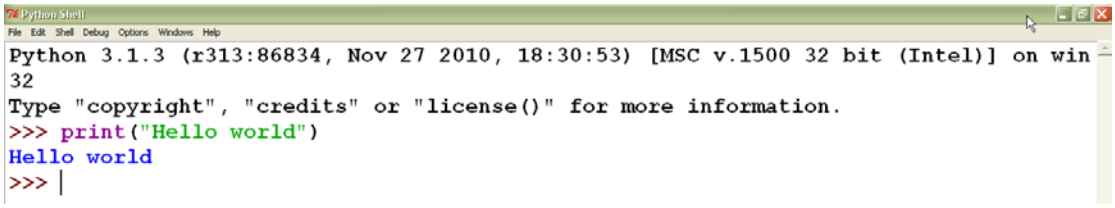
α. Μέσω του IDLE επιλέγοντας από το μενού Run → Run Module ή χρησιμοποιώντας τη συντόμευση πληκτρολογίου F5.

β. Χρησιμοποιώντας τη γραμμή εντολών (Έναρξη → Όλα τα προγράμματα → Βοηθήματα → Γραμμή εντολών). Εισάγουμε τις κατάλληλες εντολές για να μεταβούμε στον φάκελο που είναι αποθηκευμένο το αρχείο και μετά πληκτρολογούμε το όνομα του αρχείου, οπότε και εμφανίζεται η έξοδος του προγράμματος.

1.8 Hello world

Ακολουθώντας τα βήματα της προηγούμενης ενότητας μπορούμε να γράψουμε το πρόγραμμα «hello world» με δύο τρόπους:

Είτε εισάγοντας απευθείας στο Python Shell την εντολή >>> print ('hello world'), οπότε πατώντας enter το αποτέλεσμα εμφανίζεται απευθείας : hello world



Παράδειγμα 1.1

Είτε δημιουργώντας το αρχείο hello.py (File -> New Window) με περιεχόμενο

```
#filename ex1

print("Hello" , "World!")
```

στο οποίο η πρώτη γραμμή είναι σχόλιο και η δεύτερη είναι η ενσωματωμένη συνάρτηση της Python print(), με την οποία μπορούμε να εμφανίσουμε ένα αποτέλεσμα. Για να το τρέξουμε υπάρχουν δύο τρόποι που ήδη έχω αναφέρει. Συγκεκριμένα, αν θέλω να τρέξω το πρόγραμμα στη γραμμή εντολών αλλάζω τον τρέχοντα κατάλογο εργασίας με την εντολή cd c:\Python31, όπου c:\Python31 ο φάκελος στον οποίο είναι αποθηκευμένο το αρχείο και τρέχω το πρόγραμμα

καλώντας το με το όνομά του, c:\Python31\examples>hello.py, οπότε και εμφανίζεται η έξοδος του, Hello World!

cd c:\Python31\examples.

c:\Python31\examples>hello.py

Hello World!

1.9 Βασικά σημεία της Python

Παρακάτω αναφέρω τα οχτώ βασικά σημεία της Python, που είναι απαραίτητα για την σύνταξη ενός προγράμματος σε Python. Κάθε ένα σημείο αναλύεται λεπτομερώς στα επόμενα κεφάλαια.

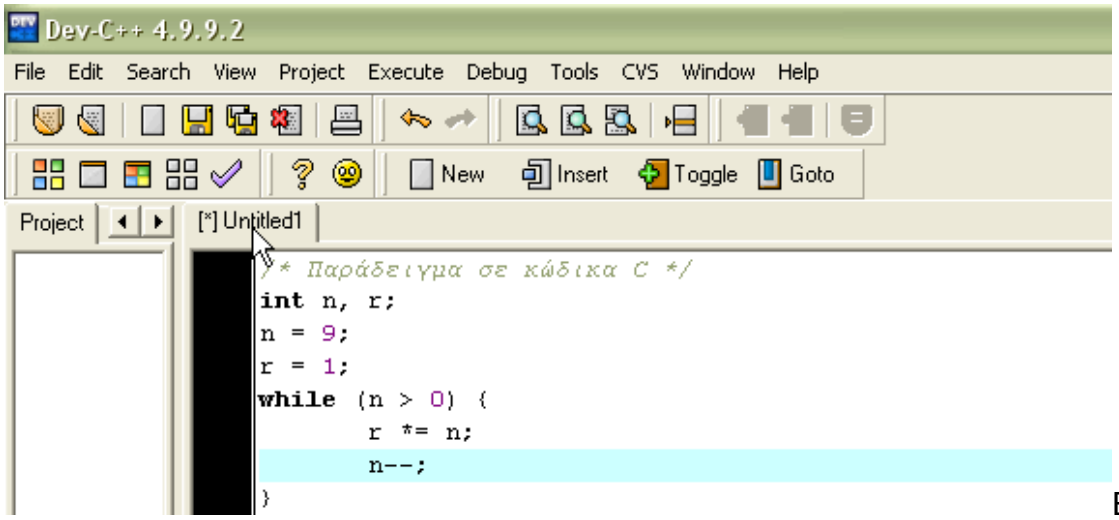
- Τύποι Δεδομένων
- Αναφορές Αντικειμένου
- Συλλογικοί τύποι δεδομένων
- Λογικοί Τελεστές.
- Εντολές ελέγχου ροής
- Αριθμητικοί τελεστές
- Είσοδος/Έξοδος
- Δημιουργία και κλήση συναρτήσεων

2. ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Στο κεφάλαιο αυτό αναφέρονται τα χαρακτηριστικά που διέπουν τα Python προγράμματα όπως η δομή του κώδικα, τα σχόλια, οι μεταβλητές και η σύμβαση ονομάτων που πρέπει να ακολουθούν. Αναφέρονται και αναλύονται οι βασικοί τύποι δεδομένων της Python και οι τελεστές, οι μέθοδοι και οι συναρτήσεις που συνδέονται με τον κάθε τύπο δεδομένων.

2.1 Δομή του κώδικα και σχόλια

Η Python διαφέρει από τις περισσότερες γλώσσες προγραμματισμού στον τρόπο που προσδιορίζει τη δομή ενός μπλοκ και γενικότερα του κώδικα, επειδή χρησιμοποιεί κενά και εσοχές. Οι περισσότερες γλώσσες χρησιμοποιούν τις {} αγκύλες. Παρακάτω υπάρχει το ίδιο κομμάτι κώδικα στη C και στη Python ώστε να γίνει κατανοητή αυτή η διαφορά :



Εικόνα 2.1

Εικόνα 2.2.

Στο κώδικα Python δεν χρησιμοποιούνται αγκύλες. Οι εντολές μετά την εντολή while που βρίσκονται σε εσοχή αποτελούν το σώμα της. Οποιαδήποτε εντολή μετά την εντολή while, είναι στο ίδιο «επίπεδο εσοχής» με αυτή, δεν περιέχεται στο σώμα της.
Επιπλέον, στο κομμάτι κώδικα σε Python οτιδήποτε υπάρχει μετά τον χαρακτήρα #, αποτελεί σχόλιο, εκτός και αν ο χαρακτήρας αυτός βρίσκεται μέσα σε εισαγωγικά, οπότε έχει την κυριολεκτική του σημασία, δηλαδή αποτελεί έναν χαρακτήρα.

2.2 Μεταβλητές

Όπως αναφέρθηκε στην εισαγωγή η Python δεν έχει μεταβλητές αλλά αναφορές αντικειμένων. Το όνομα που δίνουμε σε μία αναφορά αντικειμένου, ονομάζεται αναγνωριστικό ή απλά όνομα. Ένα έγκυρο αναγνωριστικό της Python είναι μία μη κενή ακολουθία χαρακτήρων, οποιουδήποτε μήκους. Ένα τέτοιο αναγνωριστικό πρέπει να υπακούει σε ένα σύνολο κανόνων :

- Ο πρώτος χαρακτήρας μπορεί να είναι οποιοδήποτε γράμμα ASCII (“a”, “b”, … , “A”, “B”, …) και κάτω παύλα (“_”). Οι χαρακτήρες που ακολουθούν μπορεί να είναι οποιοσδήποτε χαρακτήρας – γράμμα και κάτω παύλα ή ψηφίο (0-9) ή τελεία “.”. Τα αναγνωριστικά είναι case sensitive, δηλαδή οι λέξεις TAXRATE, taxrate, Taxrate είναι τρία διαφορετικά αναγνωριστικά.
- Κανένα αναγνωριστικό δεν μπορεί να έχει το ίδιο όνομα με τις λέξεις κλειδιά της Python :

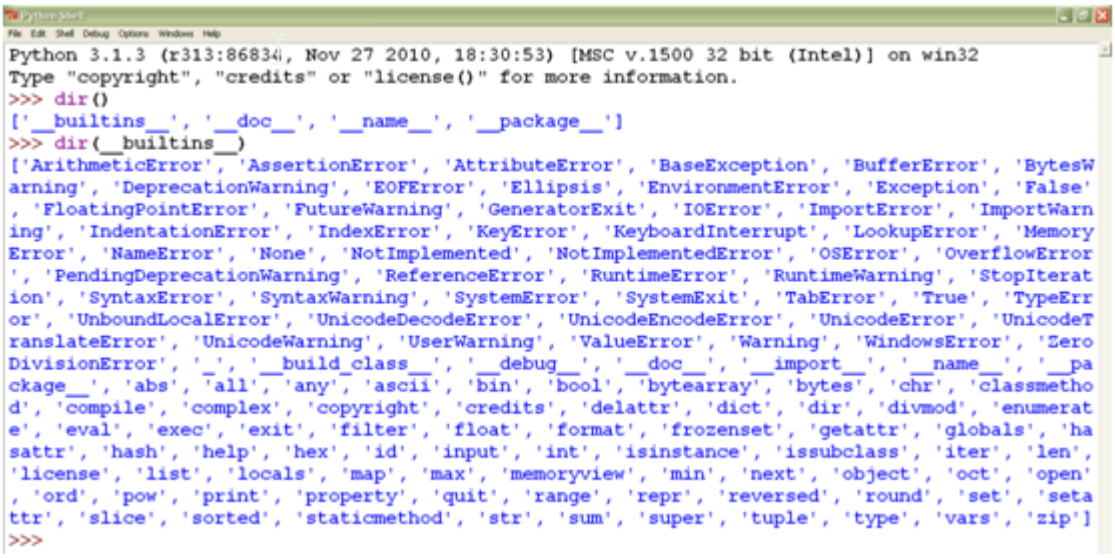
and	continue	except	global	lambda	pass	while
as	def	False	if	None	raise	with
assert	del	finally	import	nonlocal	return	yield
break	elif	for	in	not	True	
class	else	from	is	or	try	

Πίνακας 2.1 Λέξεις κλειδιά της Python

Και ένα σύνολο συμβάσεων:

- Δεν πρέπει να χρησιμοποιούμε τα ονόματα των προκαθορισμένων αναγνωριστικών της Python (χρήση ονομάτων NotImplemented και Ellipsis, ενσωματωμένοι τύποι δεδομένων της Python - int, float, list, str και tuple, ονόματα ενσωματωμένων συναρτήσεων και εξαιρέσεων της Python) για τα

δικά μας αναγνωριστικά. Για να καταλάβουμε πότε ένα αναγνωριστικό ανήκει σε μία από τις προηγούμενες κατηγορίες, η Python παρέχει την ενσωματωμένη συνάρτηση dir(), η οποία επιστρέφει μία λίστα με τις ιδιότητες ενός αντικειμένου. Αν καλείται χωρίς παραμέτρους επιστρέφει τη λίστα με τις ενσωματωμένες ιδιότητες/γνωρίσματα της Python. Αν βάλω ως παράμετρο την __builtins__ τότε εμφανίζεται μία λίστα με περίπου 130 ονόματα, από τα οποία αυτά που ξεκινάνε με κεφαλαίο γράμμα είναι ονόματα ενσωματωμένων εξαιρέσεων της Python, ενώ τα υπόλοιπα αποτελούν ονόματα συναρτήσεων και τύπων δεδομένων. Για παράδειγμα:



Εικόνα 2.3

Η δεύτερη σύμβαση αφορά τη χρήση της κάτω παύλας "_". Ονόματα που αρχίζουν και τελειώνουν με δύο κάτω παύλες, όπως __lt__, δεν πρέπει να χρησιμοποιούνται. Μία κάτω παύλα από μόνη της μπορεί να χρησιμοποιηθεί από μόνη της ως αναγνωριστικό. Μέσα στον διαδραστικό διερμηνευτή ή κέλυφος Python, η κάτω παύλα κρατά το αποτέλεσμα της τελευταίας έκφρασης που αποτιμήθηκε. Μερικά προγράμματα χρησιμοποιούν τη _ στους βρόγχους for…in όταν δεν τους ενδιαφέρει τα αντικείμενα στο βρόγχο:

```
# ex2.2.py
for _ in (0, 1, 2, 3, 4, 5):

    print("Hello!!!")
```

```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
Hello!!!
Hello!!!
Hello!!!
Hello!!!
Hello!!!
Hello!!!
>>>
```

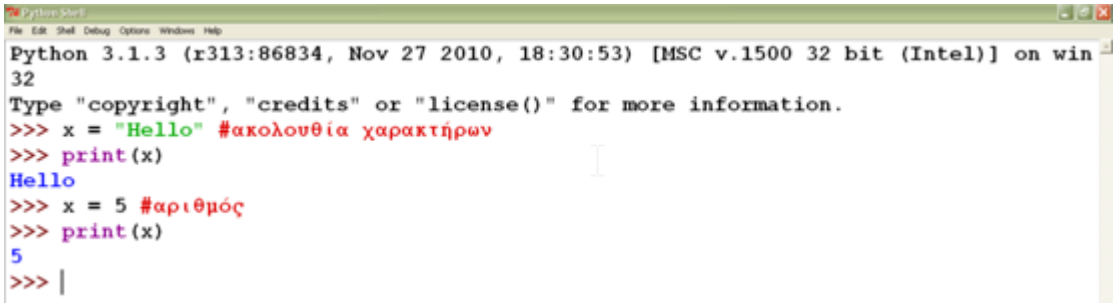
Εικόνα 2.4

Ο πιο εύκολος τρόπος να ελέγξουμε αν κάποιο όνομα είναι έγκυρο, είναι να το πληκτρολογήσουμε στον επεξεργαστή. Για παράδειγμα:

```
Type "copyright", "credits" or "license()" for more information.
>>> stretch-factor = 1 #ο χαρακτήρας - δεν είναι Unicode ή κάτω παύλα
SyntaxError: can't assign to operator
>>> 2miles = 2 # ο πρώτος χαρακτήρας δεν είναι Unicode ή κάτω παύλα
SyntaxError: invalid syntax
>>> str = 3 # νόμιμο αλλά είναι κακλη τακτική επειδή το όνομα ταυτίζεται με τον ενσ
ωματωμένο τύπο της Python, str
>>> l,imptt31 = 4 # περιέχει τον χαρακτήρα , που δεν είναι έγκυρος
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    l,imptt31 = 4 # περιέχει τον χαρακτήρα , που δεν είναι έγκυρος
TypeError: 'int' object is not iterable
>>> l'imptt31 = 4 # όμοια με πριν
SyntaxError: EOL while scanning string literal
>>> l_imptt31 #οκ!
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    l_imptt31 #οκ!
NameError: name 'l_imptt31' is not defined
>>> l_imptt31 = 5 # οκ!! στο προηγούμενο δεν ανατέθηκε τιμή
>>>
```

Εικόνα 2.5

Τέλος, όπως ήδη έγινε αντιληπτό, δεν είναι απαραίτητη η δήλωση ενός τύπου της μεταβλητής πριν χρησιμοποιηθεί, ούτε ένας χαρακτήρας οριοθέτησης, όπως το semicolon (;). Οι μεταβλητές δημιουργούνται αυτόματα, όταν τους ανατίθεται για πρώτη φορά μία τιμή και μπορεί να είναι οποιουδήποτε τύπου δεδομένων. Την επόμενη φορά που θα ανατεθεί στην ίδια μεταβλητή μία τιμή, αυτή μπορεί να είναι διαφορετικού τύπου, οπότε και ο τύπος της μεταβλητής αλλάζει:



Εικόνα 2.6

2.3 Τύποι δεδομένων

Οι βασικοί τύποι δεδομένων της Python είναι αριθμητικοί (numbers) - ακέραιοι (integers), κινητής υποδιαστολής (floats), δεκαδικοί (decimals), μιγαδικοί (complex) και Boolean - και συμβολοσειρές (strings).

2.3.1 Αριθμητικοί τύποι δεδομένων

2.3.1.1 Ακέραιοι αριθμοί - Integers

Οι ακέραιοι τύποι δεδομένων είναι αμετάβλητοι (immutable). Το μέγεθος ενός ακεραίου περιορίζεται μόνο από την μνήμη της μηχανής, οπότε μπορούμε να δημιουργήσουμε ακεραίους με εκατοντάδες ψηφία και να εργαστούμε μαζί τους. Η καθυστέρηση όμως θα είναι μεγαλύτερη σε σχέση με τους ακεραίους που αναπαριστώνται τοπικά από τον επεξεργαστή του μηχανήματος.

Οι ακέραιοι από προεπιλογή γράφονται με βάση το δεκαδικό σύστημα, αλλά μπορούμε να χρησιμοποιήσουμε και το δυαδικό, οχταδικό ή το δεκαεξαδικό σύστημα για να τους αναπαραστήσουμε:

Εικόνα 2.5

Επιπλέον η Python παρέχει ένα σύνολο ενσωματωμένων συναρτήσεων για μετατροπή ακεραίων από τη μορφή ενός συστήματος σε άλλο και ένα σύνολο βασικών συναρτήσεων για εργασίες με ακεραίους, όπως φαίνεται στον επόμενο πίνακα :

bin(i)	Επιστρέφει την δυαδική αναπαράσταση του ακεραίου i σαν ακολουθία χαρακτήρων πχ bin(1980) == ‘0b11110111100’
hex(i)	Επιστρέφει την δεκαεξαδική αναπαράσταση του i σαν ακολουθία χαρακτήρων πχ hex(1980) == ‘0x7bc’
int(x)	Μετατρέπει το αντικείμενο x σε έναν ακέραιο – προκαλεί εξαίρεση ValueError ή TypeError αν ο τύπος δεδομένων του x δεν υποστηρίζει τη μετατροπή σε ακέραιο. Αν το x είναι δεκαδικός αριθμός, το δεκαδικό μέρος αποκόπτεται.
int(s, base)	Μετατρέπει την ακολουθία χαρακτήρων s σε έναν ακέραιο. Αν αποτύχει προκαλεί εξαίρεση λάθους ValueError. Αν η προαιρετική παράμετρος base δίνεται, θα πρέπει να είναι ακέραιος με τιμή μεταξύ 2 έως και 36.
oct(i)	Επιστρέφει την οχταδική αναπαράσταση του i ως ακολουθία χαρακτήρων πχ oct(1980) == ‘0o3674’
abs(x)	Επιστρέφει την απόλυτη τιμή του x
divmod(x, y)	Επιστρέφει το πηλίκο και το υπόλοιπο της διαίρεσης του x με το y σαν πλειάδα δύο ακεραίων.
pow(x, y)	Υψώνει το x στη δύναμη y; Το ίδιο με τον τελεστή **
pow(x, y, z)	Μία πιο γρήγορη εναλλακτική μορφή του (x**y) % z
round(x, n)	Επιστρέφει το x στρογγυλοποιημένο κατά n ψηφία, αν το n είναι αρνητικός ακέραιος ή επιστρέφει το x στρογγυλοποιημένο κατά n δεκαδικές θέσεις, αν το n είναι θετικός ακέραιος ; η επιστρεφόμενη τιμή είναι ίδιου τύπου με το x. Δεν έχει επίδραση σε ακεραίους.
max(a, b, …)	Επιστρέφει το μεγαλύτερο από τους ακεραίους που δέχεται ως παραμέτρους

min(a, b, …)	Επιστρέφει το μικρότερο από τους ακεραίους που δέχεται ως παραμέτρους

Πίνακας 2.2 Ενσωματωμένες συναρτήσεις της Python για ακεραίους

Για να δώσω μία ακέραια τιμή σε μία μεταβλητή, χρησιμοποιώ τον τελεστή ανάθεσης = είτε απευθείας (x =17), είτε καλώντας τον τύπο δεδομένων int ως συνάρτηση (x = int(17)). Η συνάρτηση int() μπορεί να κληθεί με τρεις διαφορετικούς τρόπους ,όσο αφορά το πλήθος των παραμέτρων :

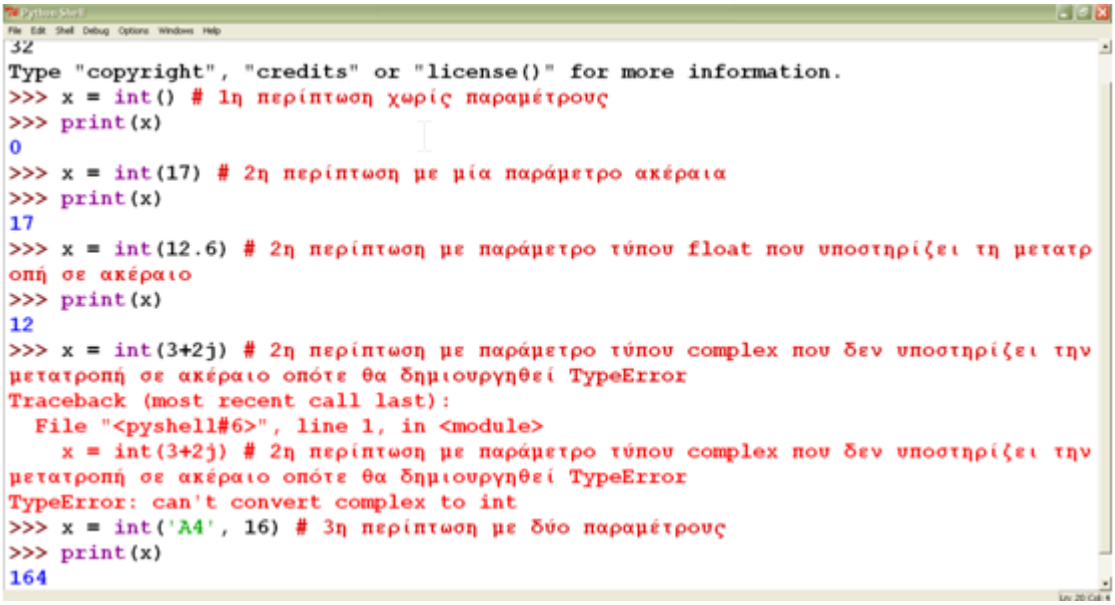
Ο πρώτος τρόπος είναι να καλέσουμε τη συνάρτηση χωρίς παραμέτρους. Σε αυτήν την περίπτωση δημιουργείται ένα αντικείμενο με την προεπιλεγμένη τιμή 0. Όλοι οι ενσωματωμένοι τύποι δεδομένων μπορούν να χρησιμοποιηθούν χωρίς παραμέτρους.

Ο δεύτερος τρόπος είναι να καλέσουμε τη συνάρτηση int() με μία μοναδική παράμετρο. Αν η παράμετρος είναι ίδιου τύπου δημιουργείται ένα νέο αντικείμενο.

Αν η παράμετρος είναι διαφορετικού τύπου γίνεται προσπάθεια για μετατροπή σε ακέραιο. Αν ο τύπος της παραμέτρου υποστηρίζει μετατροπή σε ακέραιο και η μετατροπή αποτύχει προκαλείται η εξαίρεση ValueError, διαφορετικά επιστρέφεται το αντικείμενο αφού υποστεί τη μετατροπή. Αν ο τύπος της παραμέτρου δεν υποστηρίζει μετατροπή προκαλείται η εξαίρεση TypeError. Οι ενσωματωμένοι τύποι δεδομένων float και str υποστηρίζουν τη μετατροπή σε ακέραιο.

Ο τρίτος τρόπος είναι να καλέσουμε την συνάρτηση int() με δύο ή περισσότερες παραμέτρους. Δύο παράμετροι επιτρέπονται όταν η πρώτη είναι ακολουθία χαρακτήρων που αντιπροσωπεύει έναν ακέραιο και η δεύτερη η βάση του συστήματος του αριθμού που αντιπροσωπεύεται με την ακολουθία χαρακτήρων.

Παραδείγματα όλων των περιπτώσεων φαίνονται στην επόμενη εικόνα:



Εικόνα 2.6

Τέλος πρόσθετη λειτουργικότητα παρέχουν οι αριθμητικοί και δυαδικοί τελεστές που επίσης μπορούν να εφαρμοστούν σε ακεραίους:

x + y	Προσθέτει τους ακεραίους x και y
x - y	Αφαιρεί τον y από τον x
x * y	Πολλαπλασιάζει τον x με τον y
x / y	Διαιρεί τον x με τον y; Πάντα παράγει έναν αριθμό κινητής υποδιαστολής
x // y	Διαιρεί τον x με τον y; Αποκόπτει οποιοδήποτε δεκαδικό μέρος, οπότε παράγει πάντα έναν ακέραιο
x % y	Παράγει το υπόλοιπο της διαίρεσης του x με τον y
x ** y	Υψώνει το x στη δύναμη y
- x	Κάνει τον αριθμό x αρνητικό ; αλλάζει το πρόσημό του αν ο x είναι μη μηδενικός ; αν είναι 0 δεν κάνει τίποτα
+ x	Δεν κάνει τίποτε ; χρησιμοποιείται κάποιες φορές για να κάνει πιο ξεκάθαρο τον κώδικα

Πίνακας 2.3 Αριθμητικοί τελεστές ακεραίων

i | j	Δυαδικός τελεστής OR μεταξύ των ακεραίων i και j
i ^ j	Δυαδικός τελεστής XOR μεταξύ των ακεραίων i και j
i & j	Δυαδικός τελεστής AND μεταξύ των ακεραίων i και j
i << j	Μετατόπιση του i αριστερά κατά j bits; όπως i * (2**j) χωρίς έλεγχο υπερχείλισης
i >> j	Μετατόπιση του i δεξιά κατά j bits; όπως i // (2**j) χωρίς έλεγχο υπερχείλισης
~ i	Αντιστρέφει τα bit του ακεραίου i πχ αν x = 17 τότε το ~x == -18

Πίνακας 2.4 Δυαδικοί τελεστές ακεραίων

2.3.1.2 Αριθμοί Κινητής Υποδιαστολής - Floats

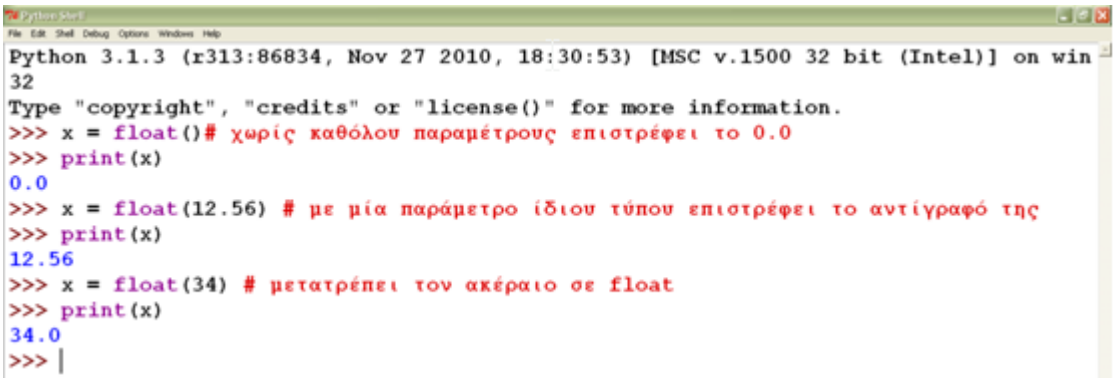
Ο τύπος δεδομένων float είναι ενσωματωμένος στη Python. Είναι διπλής ακριβείας και το εύρος των αριθμών του εξαρτάται από τον compiler (C / C# / Java) που είναι ενσωματωμένος στη Python και για το λόγο αυτό οι αριθμοί κινητής υποδιαστολής έχουν περιορισμένη ακρίβεια. Γράφονται είτε με δεκαδικά ψηφία (14.51), είτε χρησιμοποιώντας εκθετική σημειογραφία (8.9e-4). Η αναπαράστασή τους γίνεται με έναν σταθερό αριθμό από bits, ώστε να υπάρχει ένα όριο στον αριθμό των δεκαδικών ψηφίων που μπορούν να κρατηθούν.

Όταν η Python 3.1 εξάγει έναν αριθμό κινητής υποδιαστολής, στις περισσότερες περιπτώσεις χρησιμοποιεί τον αλγόριθμο του David Gay, ο οποίος εξάγει τα λιγότερα δυνατά ψηφία χωρίς να χάσει ακρίβεια. Αν και αυτός ο αλγόριθμος παράγει μία καλύτερη έξοδο δεν αλλάζει το γεγονός ότι οι υπολογιστές, ανεξάρτητα από τη γλώσσα προγραμματισμού που χρησιμοποιούν, αποθηκεύουν τους αριθμούς κινητής υποδιαστολής ως προσεγγίσεις.

Αν θέλουμε να πετύχουμε μεγαλύτερη ακρίβεια μπορούμε να ακολουθήσουμε δύο προσεγγίσεις. Η πρώτη είναι να χρησιμοποιήσουμε ακεραίους τους οποίους θα

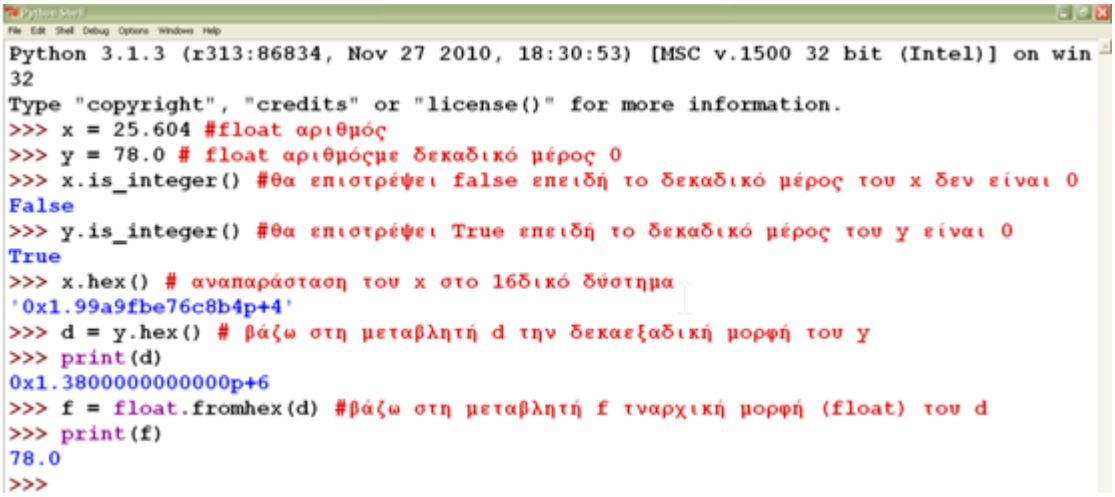
κλιμακώνουμε όταν είναι απαραίτητο. Η δεύτερη προσέγγιση έχει να κάνει με τους δεκαδικούς αριθμούς (decimal) που πλησιάζουν το επίπεδο ακριβείας που προσδιορίζουμε (μέχρι 28 δεκαδικές θέσεις) και μπορούν να αναπαραστήσουν περιοδικούς αριθμούς (πχ 0.1) με ακρίβεια. Για το λόγο αυτό είναι κατάλληλοι για οικονομικούς υπολογισμούς. Οι δεκαδικοί αριθμοί περιγράφονται αναλυτικά στην επόμενη υπό-ενότητα.

Ο τύπος δεδομένων κινητής υποδιαστολής μπορεί επίσης να κληθεί ως συνάρτηση. Η κλήση της συνάρτησης μπορεί να γίνει χωρίς καθόλου παραμέτρους, οπότε επιστρέφει την τιμή 0.0. Αν γίνει κλήση της συνάρτησης με μία παράμετρο ίδιου τύπου - float, επιστρέφει ένα αντίγραφο της παραμέτρου, ενώ αν είναι διαφορετικού τύπου επιχειρεί να μετατρέψει το δοθέν αντικείμενο σε τύπο κινητής υποδιαστολής.



Εικόνα 2.7

Τρεις ενσωματωμένες συναρτήσεις της Python για αριθμούς κινητής υποδιαστολής είναι: α) η συνάρτηση is_integer() η οποία επιστρέφει True αν το δεκαδικό μέρος ενός αριθμού κινητής υποδιαστολής είναι 0, β) η συνάρτηση as_integer_ratio() που χρησιμοποιείται για την κλασματική αναπαράσταση ενός αριθμού κινητής υποδιαστολής, γ) η συνάρτηση hex() για αναπαράσταση ενός αριθμού κινητής υποδιαστολής στο δεκαεξαδικό σύστημα και δ) η συνάρτηση float.fromhex(a), όπου a αριθμός σε δεκαεξαδική μορφή, που κάνει το αντίστροφο της hex().



Εικόνα 2.8

Περισσότερο εξειδικευμένες συναρτήσεις που συνδέονται με αριθμούς κινητής υποδιαστολής, αλλά και γενικότερα με τους αριθμούς και δεν είναι ενσωματωμένες στη Python, ανήκουν στο module math, το οποίο πρέπει να εισάγουμε στην αρχή του προγράμματος μας ώστε να μπορέσουμε να χρησιμοποιήσουμε τις συναρτήσεις του. Για τα modules υπάρχει κεφάλαιο που παρουσιάζονται αναλυτικά.

Μπορούμε να εισάγουμε είτε ολόκληρο το module με την πρόταση *import math* και μετά να χρησιμοποιήσουμε οποιαδήποτε συνάρτηση function μας ενδιαφέρει με την εντολή *math.function,* είτε να εισάγουμε κατευθείαν μόνο τη συνάρτηση function που μας ενδιαφέρει με την πρόταση *from math import function*, όπου function μπορεί να είναι μία από τις συναρτήσεις που παρουσιάζονται στο παρακάτω πίνακα :

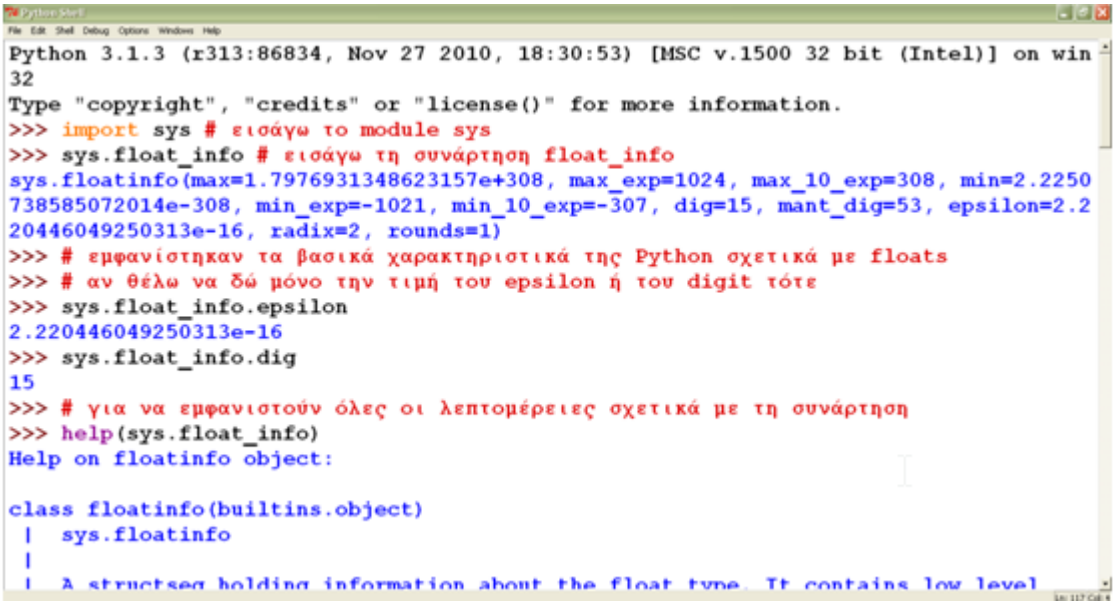
math.acos(x)	Επιστρέφει το τόξο του συνημίτονου του x σε ακτίνια
math.acosh(x)	Επιστρέφει το τόξο του υπερβολικού συνημίτονου του x σε ακτίνια
math.asin(x)	Επιστρέφει το τόξο του ημιτόνου του x σε ακτίνια
math.asinh(h)	Επιστρέφει το τόξο του υπερβολικού ημιτόνου του x σε ακτίνια
math.atan(x)	Επιστρέφει το τόξο της εφαπτομένης του x σε ακτίνια
math.atan2(y, x)	Επιστρέφει το τόξο της εφαπτομένης του y/x σε ακτίνια
math.atanh(x)	Επιστρέφει το τόξο της υπερβολικής εφαπτομένης του x

	σε ακτίνια
math.ceil(x)	Επιστρέφει το άνω όριο του x: ⌈x⌉ πχ το μικρότερο ακέραιο που είναι μεγαλύτερος από ή ίσος με το x σαν ακέραιο math.ceil(5.4) == 6
math.copysign(x, y)	Επιστρέφει το x με το πρόσημο του y
math.cos(x)	Επιστρέφει το συνημίτονο του x σε ακτίνια
math.cosh(x)	Επιστρέφει το υπερβολικό συνημίτονο του x σε ακτίνια
math.degrees(r)	Μετατρέπει τον float r από ακτίνια σε μοίρες
math.e	Χρησιμοποιούμε τη σταθερά e δηλαδή την τιμή 2.7182818284590451
math.exp(x)	Επιστρέφει το e^x ή αλλιώς το math.e**x
math.fabs(x)	Επιστρέφει την απόλυτη τιμή του x
math.factorial(x)	Επιστρέφει το x παραγοντικό : x!
math.floor(x)	Επιστρέφει το κάτω όριο του x : ⌊x⌋ δηλαδή το μεγαλύτερο ακέραιο που είναι μικρότερος από ή ίσος με τον x , math.floor(5.4) == 5
math.fmod(x, y)	Παράγει το υπόλοιπο της διαίρεσης του x με το y; Παράγει καλύτερο αποτέλεσμα από τον τελεστή % για τους floats
math.frexp(x)	Επιστρέφει μία πλειάδα με δύο στοιχεία, την mantissa (ως float) και τον εκθέτη(ως int): x = m x 2^e
math.fsum(i)	Επιστρέφει το άθροισμα των τιμών του iterable i σαν float
math.hypot(x, y)	Επιστρέφει $\sqrt{x^2 + y^2}$
math.isinf(x)	Επιστρέφει True αν ο float x είναι άπειρο +∞ / -∞
math.isnan(x)	Επιστρέφει True αν ο float x δεν είναι αριθμός
math.ldexp(m, e)	Επιστρέφει m x 2^e
math.log(x, b)	Επιστρέφει $\log_b x$; το b είναι προαιρετικό
math.log10(x)	Επιστρέφει $\log_{10} x$
math.loglp(x)	Επιστρέφει $\log_e (1+x)$
math.modf(x)	Επιστρέφει τα τμήματα του αριθμού x ως float πχ

	math.modf(5) == (0.0, 5.0)
math.pi	Επιστρέφει τη σταθερά π = 3.1415926535897931
math.pow(x, y)	Επιστρέφει το x^y σαν float
math.radians(d)	Μετατρέπει τον float d από μοίρες σε ακτίνια
math.sin(x)	Επιστρέφει το ημίτονο του x σε ακτίνια
math.sinh(x)	Επιστρέφει το υπερβολικό ημίτονο του x σε ακτίνια
math.sqrt(x)	Επιστρέφει το √x
math.tan(x)	Επιστρέφει την εφαπτομένη του x σε ακτίνια
math.tanh(x)	Επιστρέφει την υπερβολική εφαπτομένη του x σε ακτίνια
math.trunc(x)	Επιστρέφει το ακέραιο τμήμα του x; το ίδιο με την int(x)

Πίνακας 2.5 Συναρτήσεις του math module

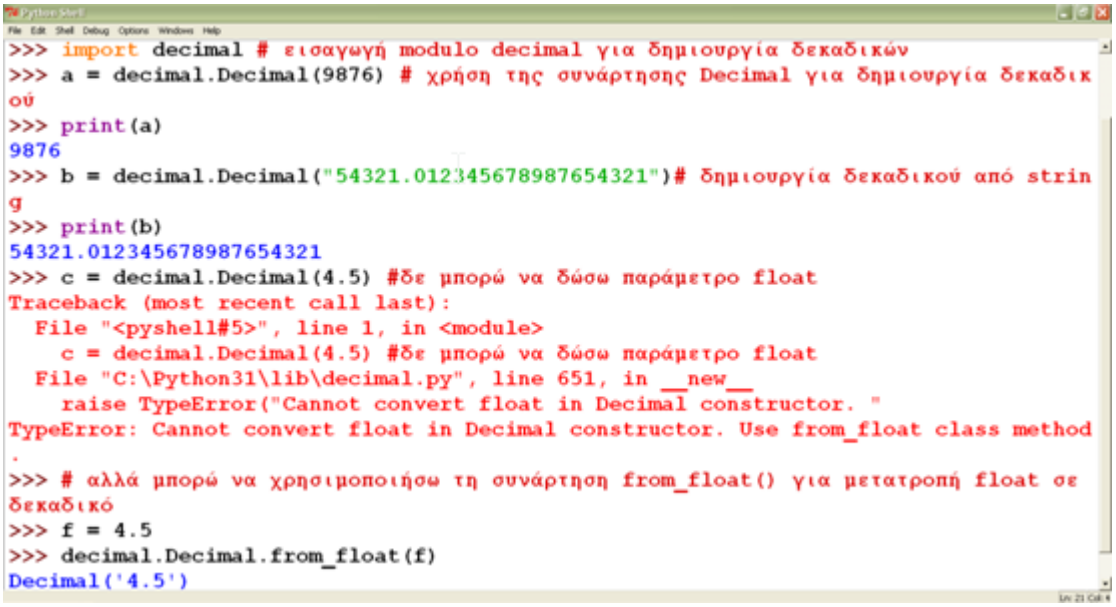
Επιπλέον μία σημαντική συνάρτηση είναι η float_info, η οποία παρέχεται από το module sys, που εισάγεται με τον ίδιο τρόπο που εισάγεται και το module math: import sys ή from sys import float_info. Η συνάρτηση αυτή περιέχει πληροφορίες χαμηλού επιπέδου για την ακρίβεια και την εσωτερική αναπαράσταση των αριθμών κινητής υποδιαστολής. Παράδειγμα, ο αριθμός των σημαντικών ψηφίων που εμφανίζονται, οι μεγαλύτεροι και μικρότεροι αριθμοί κινητής υποδιαστολής που μπορεί να αναπαραστήσει η Python, η μικρότερη διαφορά (που ονομάζεται e) που μπορεί να διακρίνει ανάμεσα σε δύο αριθμούς κινητής υποδιαστολής και άλλα. Για να δούμε όλες αυτές τις πληροφορίες μπορούμε να πληκτρολογήσουμε την εντολή help(sys.float_info). Αν θέλουμε να δούμε την τιμή ενός μόνο χαρακτηριστικού που μας ενδιαφέρει (πχ τη διαφορά epsilon), μπορούμε να χρησιμοποιήσουμε το όνομά του στη πρόταση sys.file_info.όνομα_χαρακτηριστικού (πχ sys.file_info.epsilon).



Εικόνα 2.9

2.3.1.3 Δεκαδικοί Αριθμοί – Decimals

Οι δεκαδικοί αριθμοί είναι αμετάβλητοι και δεν είναι ενσωματωμένοι στη Python. Οι πράξεις με δεκαδικούς είναι πιο αργές αλλά παρέχουν μεγαλύτερη ακρίβεια. Για να δημιουργήσουμε έναν δεκαδικό αριθμό, πρέπει να χρησιμοποιήσουμε τη συνάρτηση Decimal του module decimal. Αυτή η συνάρτηση μπορεί να πάρει σαν παράμετρο έναν ακέραιο ή μία ακολουθία χαρακτήρων αλλά όχι έναν αριθμό κινητής υποδιαστολής επειδή δεν παρέχουν μεγάλη ακρίβεια. Υπάρχει βέβαια η δυνατότητα να μετατρέψουμε έναν αριθμό κινητής υποδιαστολής σε δεκαδικό, με χρήση της συνάρτησης decimal.Decimal.from_float(), η οποία παίρνει σαν παράμετρο έναν αριθμό κινητής υποδιαστολής και επιστρέφει τον δεκαδικό που είναι πιο κοντά στη float παράμετρο.



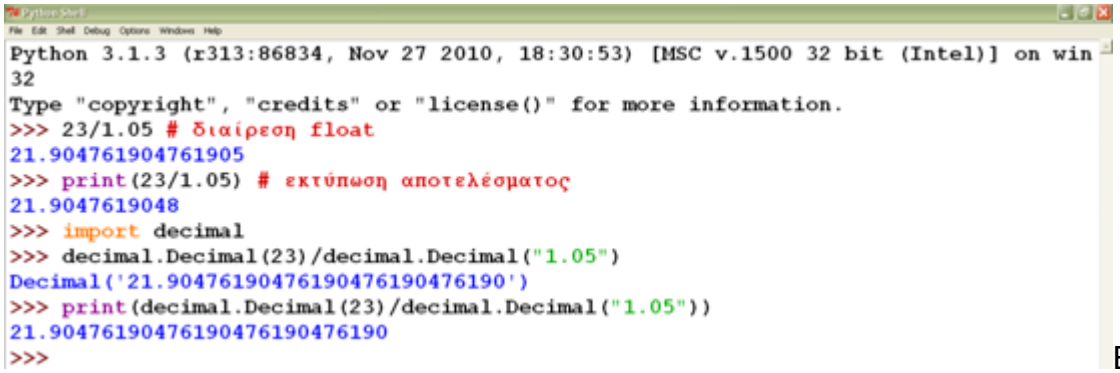
```
>>> import decimal # εισαγωγή modulo decimal για δημιουργία δεκαδικών
>>> a = decimal.Decimal(9876) # χρήση της συνάρτησης Decimal για δημιουργία δεκαδικ
ού
>>> print(a)
9876
>>> b = decimal.Decimal("54321.012345678987654321")# δημιουργία δεκαδικού από strin
g
>>> print(b)
54321.012345678987654321
>>> c = decimal.Decimal(4.5) #δε μπορώ να δώσω παράμετρο float
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    c = decimal.Decimal(4.5) #δε μπορώ να δώσω παράμετρο float
  File "C:\Python31\lib\decimal.py", line 651, in __new__
    raise TypeError("Cannot convert float in Decimal constructor. "
TypeError: Cannot convert float in Decimal constructor. Use from_float class method
.
>>> # αλλά μπορώ να χρησιμοποιήσω τη συνάρτηση from_float() για μετατροπή float σε
δεκαδικό
>>> f = 4.5
>>> decimal.Decimal.from_float(f)
Decimal('4.5')
```

Εικόνα 2.10

Όλες οι ενσωματωμένες αριθμητικές συναρτήσεις της Python και οι αριθμητικοί τελεστές που υπάρχουν αντίστοιχα στους πίνακες 2.2 και 2.6 μπορούν να χρησιμοποιηθούν και με δεκαδικούς, αλλά με κάποιες επισημάνσεις.

Αν ο τελεστής ** έχει στα αριστερά του έναν δεκαδικό, τότε στα δεξιά του πρέπει να έχει ακέραιο. Αν η πρώτη παράμετρος της συνάρτησης pow(x, y, z) είναι δεκαδικός αριθμός τότε η δεύτερη και τρίτη προαιρετικές παράμετροι πρέπει να είναι ακέραιοι αριθμοί.

Οι μαθηματικές συναρτήσεις των module math και cmath δεν είναι κατάλληλες για να χρησιμοποιηθούν με δεκαδικούς πέρα από ελάχιστες εξαιρέσεις. Για παράδειγμα για να υπολογίσουμε το e^x όπου x αριθμός κινητής υποδιαστολής, γράφουμε math.exp(x), ενώ αν ο x είναι δεκαδικός γράφουμε x.exp() – πολύ καλύτερη από την πρόταση decimal.Decimal.exp(x). Διαφορά στην ακρίβεια μεταξύ αριθμών κινητής υποδιαστολής και δεκαδικών :



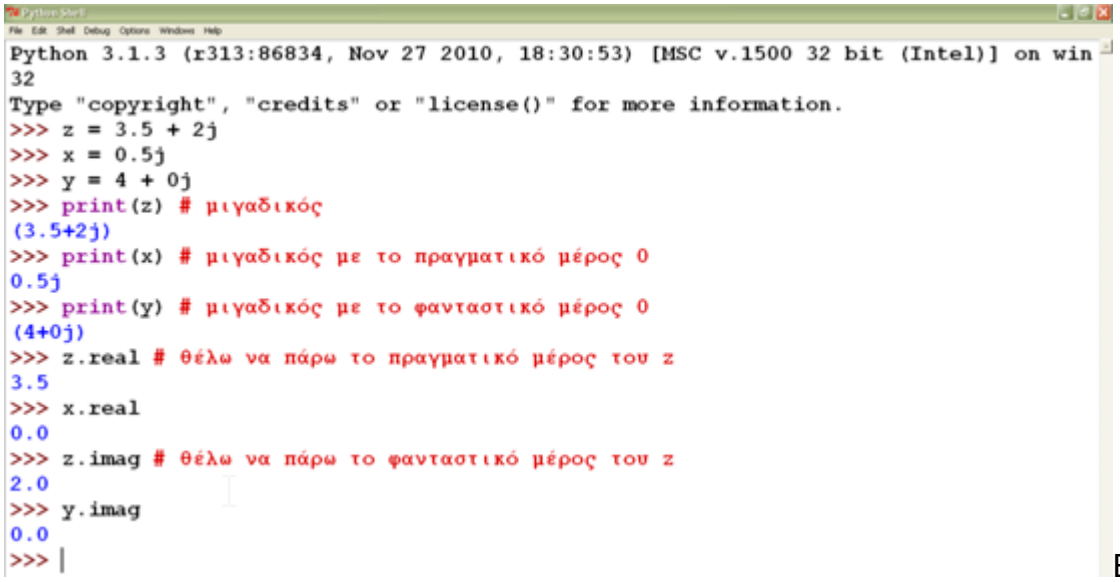
Εικόνα 2.11

Αν και η διαίρεση με δεκαδικούς δίνει περισσότερο ακριβή αποτελέσματα, σε αυτό το παράδειγμα η διαφορά φαίνεται μόνο στο $15^{ο}$ ψηφίο. Ακόμη σε αυτό το παράδειγμα βλέπουμε πως η εκτύπωση του αποτελέσματος της διαίρεσης δύο δεκαδικών, είναι σκέτος αριθμός σε μορφή ακολουθίας χαρακτήρων σε αντίθεση με την απλή πράξη που το αποτέλεσμα είναι δεκαδικός.

Όλα τα Python αντικείμενα έχουν δύο μορφές εξόδου. Η μορφή ακολουθίας χαρακτήρων είναι σχεδιασμένη για να διαβάζεται από ανθρώπους και η δεύτερη μορφή είναι σχεδιασμένη να παράγει έξοδο που τροφοδοτείται σε έναν Python επεξεργαστή, ο οποίος αναπαράγει το αντικείμενο.

2.3.1.4 Μιγαδικοί Αριθμοί – Complex

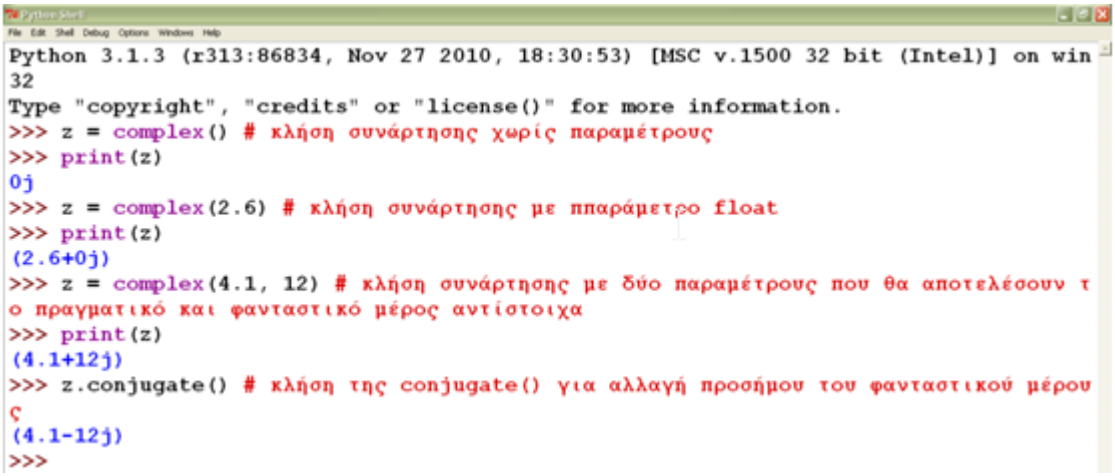
Ένας μιγαδικός αριθμός είναι αμετάβλητος και αποτελείται από το πραγματικό και το φανταστικό μέρος. Το πραγματικό και φανταστικό μέρος ενώνονται με το πρόσημο + ή – και το φανταστικό μέρος ακολουθείται από ένα j. Αν το πραγματικό μέρος είναι 0 μπορούμε να το παραλείψουμε εντελώς. Η Python εμφανίζει τους μιγαδικούς αριθμούς σε παρενθέσεις. Για να πάρω μόνο το πραγματικό ή μόνο το φανταστικό μέρος ενός μιγαδικού αριθμού χρησιμοποιώ τις ιδιότητες real και imag αντίστοιχα. Η Python τα εμφανίζει σαν αριθμούς κινητής υποδιαστολής.



Εικόνα 2.12

Ο τύπος complex μπορεί να κληθεί ως συνάρτηση. Χωρίς παραμέτρους επιστρέφει 0j. Με μία παράμετρο που είναι μιγαδικός αριθμός επιστρέφει ένα αντίγραφο της παραμέτρου και με μία παράμετρο διαφορετικού τύπου επιχειρεί να μετατρέψει το δοθέν αντικείμενο σε μιγαδικό. Όταν χρησιμοποιείται μόνο ένας αριθμός κινητής υποδιαστολής, το φανταστικό μέρος γίνεται 0j.

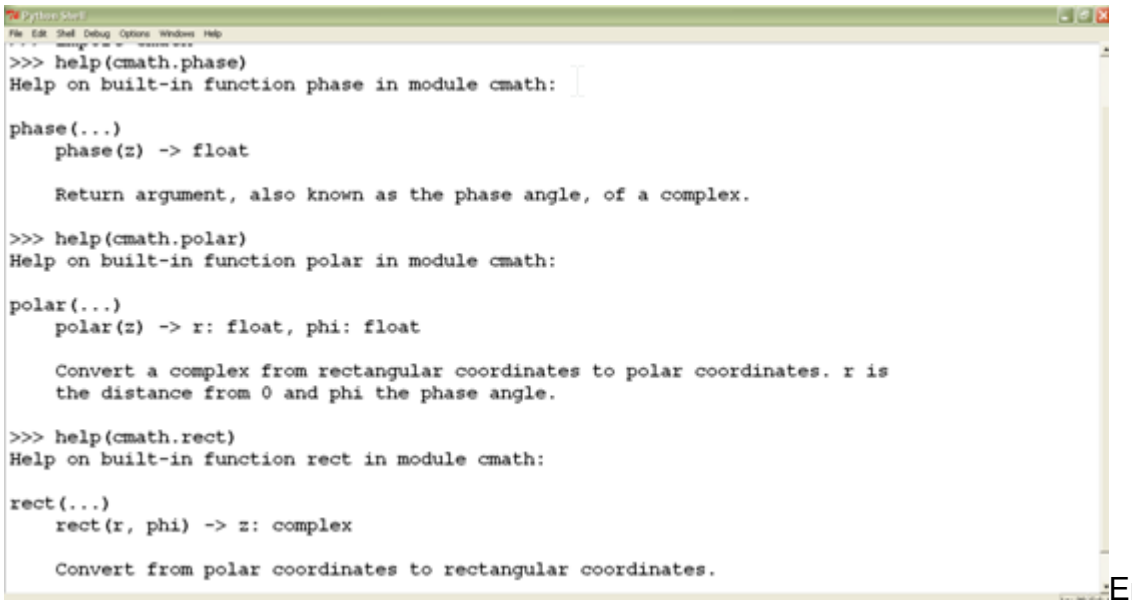
Εκτός από τους τελεστές //, % και τις συναρτήσεις divmod() και pow(), όλοι οι υπόλοιποι τελεστές και συναρτήσεις μπορούν να χρησιμοποιηθούν με τους μιγαδικούς αριθμούς . Επιπλέον οι μιγαδικοί αριθμοί χρησιμοποιούν τη μέθοδο conjugate(), η οποία αλλάζει το πρόσημο του φανταστικού μέρους.



Εικόνα 2.13

Οι συναρτήσεις στο module math δεν εφαρμόζονται πάνω σε μιγαδικούς. Οι χρήστες μιγαδικών αριθμών μπορούν να εισάγουν το module cmath που παρέχει

τις περισσότερες από τις τριγωνομετρικές και λογαριθμικές συναρτήσεις που υπάρχουν και στο math (acos, acosh, asin, asinh, atan, atanh, cos, cosh, e, exp, log, log10, pi, sin, sinh, sqrt, tan, tanh) και επιπλέον κάποιες ειδικές συναρτήσεις για μιγαδικούς αριθμούς όπως οι cmath.phase(), cmath.polar(), cmath.rect και τις σταθερές cmath.pi, cmath.e που κρατάνε τις ίδιες τιμές με τις αντίστοιχες μαθηματικές.



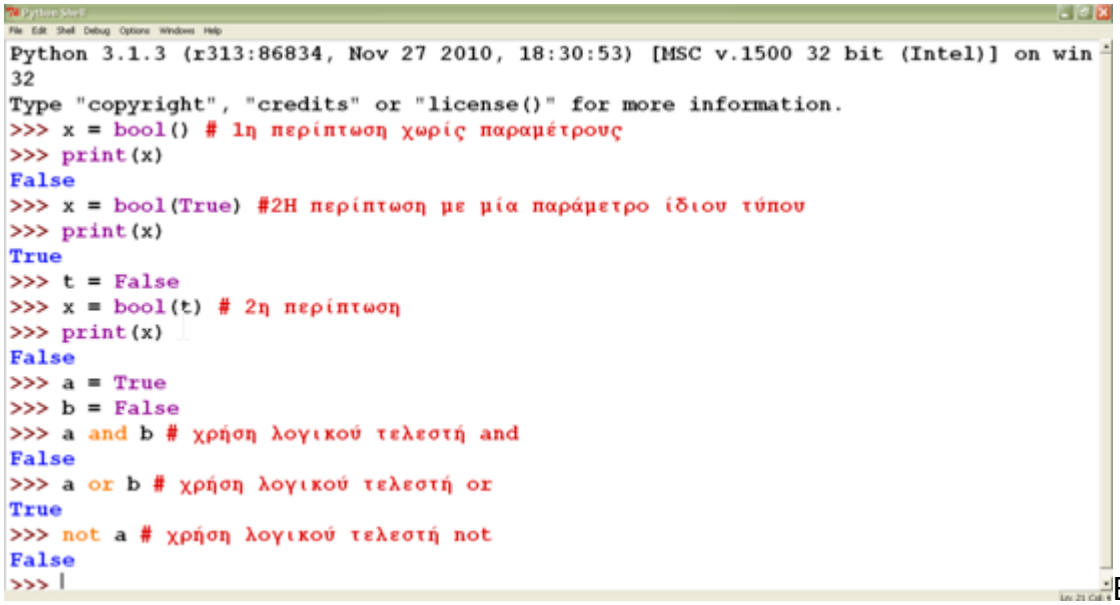
Εικόνα 2.14

2.3.1.5 Boolean

Ο Boolean τύπος δεδομένων μπορεί να πάρει δύο τιμές, αληθής (True) ή ψευδής (False) και συμπεριφέρεται ακριβώς όπως το 1 και 0 αντίστοιχα. Όπως και οι υπόλοιποι τύποι δεδομένων της Python (built-in, library, custom), μπορεί να κληθεί σαν συνάρτηση bool().

Η κλήση της χωρίς καθόλου παραμέτρους επιστρέφει False. Η κλήση της με μία μοναδική παράμετρο τύπου Boolean επιστρέφει ένα αντίγραφο της παραμέτρου και με μία μοναδική παράμετρο διαφορετικού τύπου, επιδιώκει να μετατρέψει το δοθέν αντικείμενο σε Boolean. Όλοι οι τύποι δεδομένων που είναι ενσωματωμένοι στη Python ή ανήκουν στην στάνταρ βιβλιοθήκη της, μπορούν να μετατραπούν και να παράγουν μία Boolean τιμή, όπως επίσης είναι εύκολο να κάνουμε Boolean μετατροπές από προσαρμοσμένους τύπους δεδομένων. Ακόμη είναι δυνατόν να χρησιμοποιηθούν οι λογική τελεστές and και or που χρησιμοποιούν λογική

βραχυκυκλώματος και επιστρέφουν την μεταβλητή που καθορίζει το αποτέλεσμα και ο τελεστής not που επιστρέφει πάντα True ή False.



Εικόνα 2.15

2.3.2 Ακολουθίες χαρακτήρων – Strings

Οι ακολουθίες χαρακτήρων είναι αμετάβλητος τύπος δεδομένων και κρατάει μία ακολουθία Unicode χαρακτήρων. Μπορεί να κληθεί ως συνάρτηση για δημιουργία αντικειμένων string. Η κλήση της συνάρτησης str() χωρίς παραμέτρους επιστρέφει ένα άδειο string. Η κλήση της με μία παράμετρο ίδιου τύπου επιστρέφει ένα αντίγραφο της παραμέτρου, ενώ με μία παράμετρο διαφορετικού τύπου, επιστρέφει την παράμετρο σε μορφή string.

Η συνάρτηση str() μπορεί ακόμη να χρησιμοποιηθεί σαν συνάρτηση μετατροπής, οπότε και παίρνει τρεις παραμέτρους. Η πρώτη παράμετρος είναι τύπου string ή ο τύπος της έχει τη δυνατότητα να μετατραπεί σε string. Οι δύο επόμενες παράμετροι είναι προαιρετικές, όπου η μία προσδιορίζει την κωδικοποίηση που θα χρησιμοποιήσουμε για την αναπαράσταση της πρώτης παραμέτρου και η άλλη τον τρόπο διαχείρισης των λαθών που προκύπτουν από την κωδικοποίηση – τιμές : strict, replace και ignore.

Οι τιμές των string περικλείονται μέσα σε μονά ή διπλά εισαγωγικά, αρκεί αυτά που χρησιμοποιούμε στην αρχή, να είναι ίδια με αυτά που χρησιμοποιούμε στο τέλος.

x = “Hello World!” ή x = ‘Hello World!’

Δεν μπορούμε να βάλουμε διπλά εισαγωγικά μέσα σε ένα string που περικλείεται σε διπλά εισαγωγικά και το ίδιο ισχύει για τα μονά εισαγωγικά. Πρέπει για να γίνουν δεκτά με την κυριολεκτική τους σημασία να προηγηθεί μία ανάποδη κάθετος (backslash).

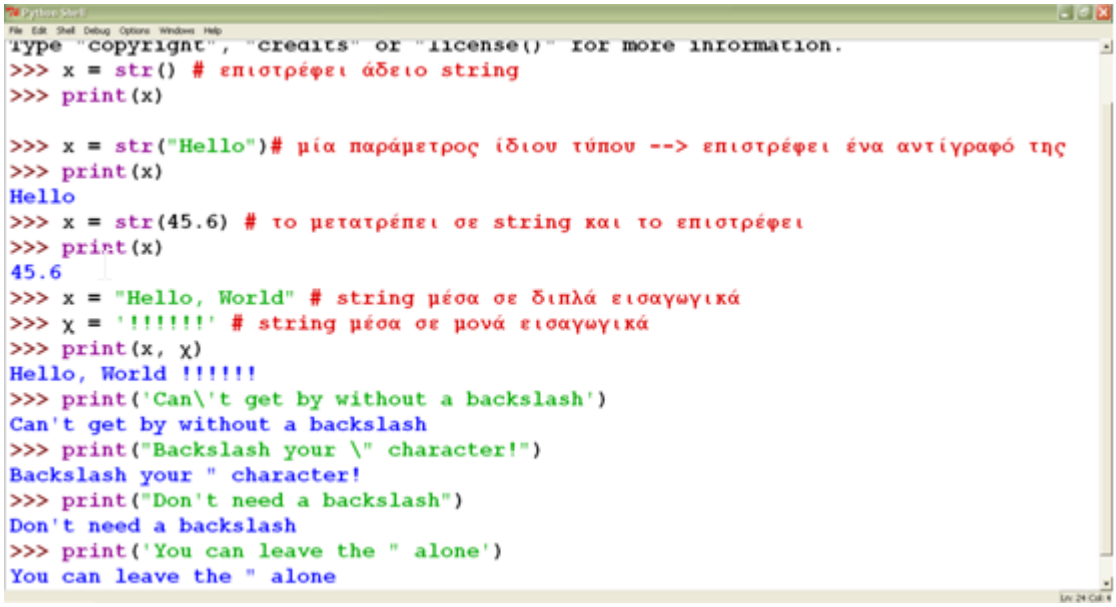
x = ‘Can\’t get by without a backslash’

x = “Backslash your \” character!”

Σε αντίθεση, μπορώ να εισάγω μονά εισαγωγικά μέσα σε ένα string που περικλείεται σε διπλά, και διπλά εισαγωγικά σε ένα string που περικλείεται σε μονά εισαγωγικά.

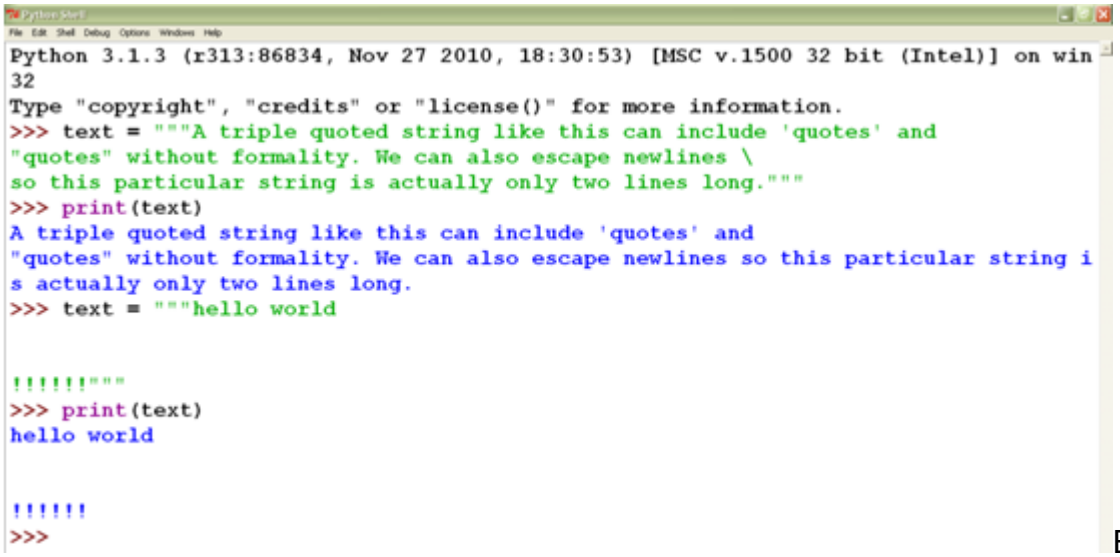
x = “Don’t need a backslash”

x = ‘You can leave the “ alone’



Εικόνα 2.16

Δεν μπορούμε να χωρίσουμε ένα string σε δύο γραμμές, γιατί τότε προκαλείται λάθος. Για να γίνει αυτό η Python παρέχει τη δυνατότητα να χρησιμοποιήσουμε τριπλά διπλά εισαγωγικά (δηλαδή τρεις φορές συνεχόμενα το σύμβολο “) για να εισάγουμε σε αυτά ένα string. Επιπλέον μέσα στα τριπλά εισαγωγικά μας δίνεται η δυνατότητα να χρησιμοποιήσουμε μονά ή διπλά εισαγωγικά χωρίς να προηγούνται backslashes και το ίδιο ισχύει για τις νέες γραμμές.



Εικόνα 2.17

Οι ιδιότητες των strings, οι λειτουργίες που βασίζονται σε αυτά και άλλα σημαντικά χαρακτηριστικά αναλύονται στο επόμενο κεφάλαιο.

3. ΑΚΟΛΟΥΘΙΕΣ ΧΑΡΑΚΤΗΡΩΝ – STRINGS

3.1 Τεμαχισμός συμβολοσειρών

Τα strings είναι ακολουθίες χαρακτήρων και επομένως μπορούμε να εξάγουμε έναν χαρακτήρα ή ένα υπό-σύνολο χαρακτήρων από αυτές χρησιμοποιώντας τον τελεστή εξαγωγής στοιχείων [].

Πρώτη επομένως περίπτωση είναι η εξαγωγή ενός μόνο χαρακτήρα. Ο δείκτης θέσης σε μία ακολουθία χαρακτήρων ξεκινά από το 0 μέχρι το μήκος του -1. Υπάρχει επίσης η δυνατότητα να χρησιμοποιήσουμε και αρνητικούς δείκτες. Ένας αρνητικός δείκτης ξεκινάει να μετράει από τον τελευταίο χαρακτήρα προς τον πρώτο. Έστω το string s = “Light ray”.

s[-9]	s[-8]	s[-7]	s[-6]	s[-5]	s[-4]	s[-3]	s[-2]	s[-1]
L	i	g	h	t		r	a	y
s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]

Οι αρνητικοί δείκτες είναι ιδιαίτερα χρήσιμοι και ειδικά ο δείκτης -1 που μας δίνει πάντα τον τελευταίο χαρακτήρα σε μία ακολουθία χαρακτήρων. Αν προσπαθήσουμε να προσπελάσουμε έναν δείκτη εκτός ορίων ή μία κενή ακολουθία χαρακτήρων θα προκληθεί η εξαίρεση IndexError. Ο τελεστής τεμαχισμού έχει τρεις τρόπους σύνταξης:

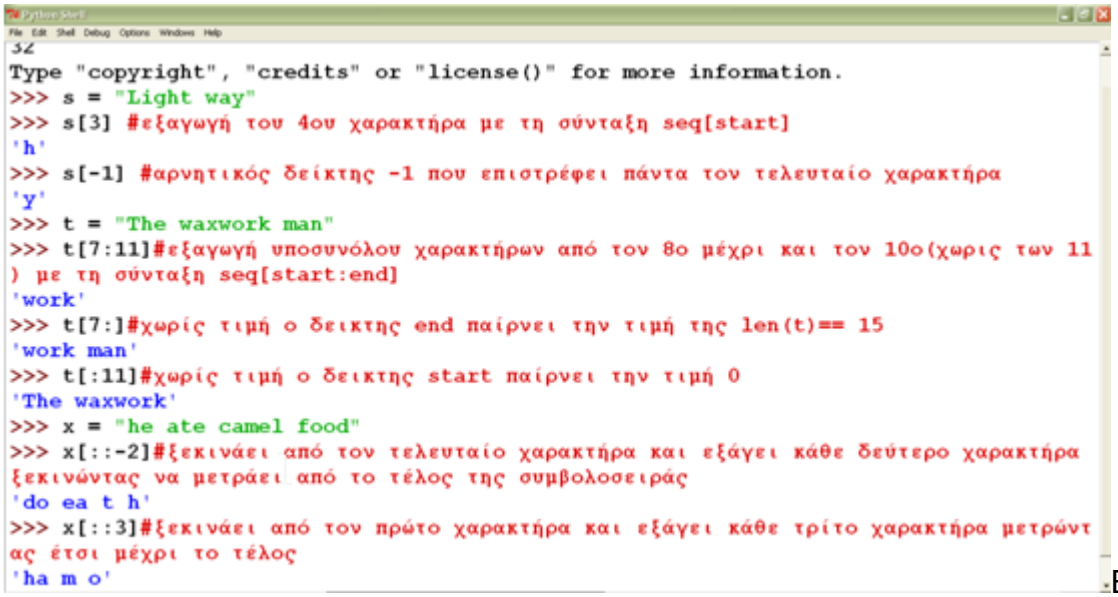
- seq[start]
- seq[start:end]
- seq[start:end:step]

Η seq μπορεί να είναι οποιαδήποτε ακολουθία χαρακτήρων – θα μπορούσε να είναι επίσης πλειάδα ή λίστα αλλά το κεφάλαιο αυτό αφορά ειδικά τον τύπο δεδομένων string. Οι τιμές start, end και step πρέπει να είναι όλοι ακέραιοι ή μεταβλητές που κρατάνε ακεραίους.

Η σύνταξη seq[start] εξάγει το στοιχείο της ακολουθίας που βρίσκεται στη θέση start. Η σύνταξη seq[start:end] εξάγει το κομμάτι που ξεκινά από τη θέση start και τελειώνει στη θέση end. Μπορούμε να παραλείψουμε έναν από τους δύο

ακέραιους δείκτες ή και τους δύο. Αν παραλείψουμε το δείκτη start, αυτός θα πάρει την προεπιλεγμένη τιμή 0. Αν παραλείψουμε το δείκτη end, τότε ο δείκτης θα πάρει την τιμή της συνάρτησης len(seq). Αν παραλείψουμε και τους δύο δείκτες η πρόταση s[:] είναι στην ουσία ίδια με την πρόταση s[0:len(s)] και εξάγει ολόκληρη την ακολουθία χαρακτήρων, δηλαδή ένα αντίγραφό της.

Η σύνταξη seq[start:end:step] είναι παρόμοια με τη σύνταξη seq[start:end] με τη διαφορά ότι αντί να εξάγουμε κάθε χαρακτήρα που βρίσκεται μεταξύ των θέσεων start και end, εξάγουμε τους χαρακτήρες που η θέση τους ορίζεται από το δείκτη step (βήμα) μεταξύ των θέσεων start και end. Και εδώ μπορούμε να παραλείψουμε έναν από τους δύο ακέραιους δείκτες ή και τους δύο. Αν παραλείψουμε το δείκτη start, αυτός θα πάρει την προεπιλεγμένη τιμή 0. Αν παραλείψουμε το δείκτη end, τότε ο δείκτης θα πάρει την τιμή της συνάρτησης len(seq). Αν χρησιμοποιήσουμε τους δύο πρώτους δείκτες και παραλείψουμε το βήμα, από προεπιλογή παίρνει τιμή 1 αλλά αυτό δεν έχει νόημα. Το βήμα δεν επιτρέπεται να πάρει τιμή 0. Στις εικόνες 3.1 και 3.2 εμφανίζεται ένα σύνολο παραδειγμάτων:



Εικόνα 3.1

```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> x = "he ate camel food"
>>> x[-1:2:-2]#τρόπος εξαγωγής χαρακτήρων με τη σύνταξη seq[start:end:step] που παρ
άγει το ίδιο αποτέλεσμα με την x[:2:-2]
'do ea t'
>>> x[0:-5:3]#παράγει το ίδιο αποτέλεσμα με την x[:-5:3]
'ha m'
>>> x[::-1]#αν το βήμα είναι -1 τότε κάθε χαρακτήρας εξάγεται από την ακολουθία με
τη σειρά, από το τέλος μέχρι την αρχή και παράγει την ακολουθία ανεστραμμένη
'doof lemac eta eh'
>>> t = "The waxwork man"
>>> t = t[:12]+"wo"+[12:]#τρόπος για να εισάγουμε ένα κομμάτι string μέσα σε μία ακ
ολουθία είναι να συνδυάσουμε τον τεμαχισμό με την ένωση ακολουθιών με τον τελεστή +
. Το ίδιο αποτέλεσμα παράγει και η t[:]+t[7:9]+s[12:]
SyntaxError: invalid syntax
>>> t = t[:12]+"wo"+t[12:]
>>>
>>> t
'The waxwork woman'
>>>
```

Εικόνα 3.2

Μπορεί να μεταχειριζόμαστε τις ακολουθίες χαρακτήρων σαν λίστες χαρακτήρων αλλά δεν είναι λίστες χαρακτήρων. Η κυριότερη διαφορά των ακολουθιών χαρακτήρων είναι ότι σε αντίθεση με τον τύπο δεδομένων λίστα, που θα δούμε στο επόμενο κεφάλαιο, δεν μπορούν να τροποποιηθούν. Αν προσπαθήσουμε να προσθέσουμε ή να αναθέσουμε έναν ή περισσότερους χαρακτήρες με κώδικα, όπως string.append('c') ή string[0] = 'H', τότε θα προκληθεί λάθος. Αυτό που μπορούμε να κάνουμε είναι να πάρουμε ένα κομμάτι της ακολουθίας χαρακτήρων, όπως είδαμε παραπάνω και να το τροποποιήσουμε. Ένας άλλος τρόπος είναι η χρήση της μεθόδου list() που παίρνει ως παράμετρο ένα string και το μετατρέπει σε λίστα χαρακτήρων. Ακολουθεί ένα παράδειγμα:

```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> txt = "Lady Rose"
>>> charList = list(txt) # μετατρέπει το string txt σε λίστα και την αποθηκεύει στη
μεταβλητή charList
>>> print(charList)
['L', 'a', 'd', 'y', ' ', 'R', 'o', 's', 'e']
>>> charList[4:] = []# διαγράφει τους χαρακτήρες από τη θέση 5 και μετά
>>> charList.reverse()
>>> txt = charList
>>> print(txt)
['y', 'd', 'a', 'L']
>>>
```

Εικόνα 3.3

3.2 Τελεστές και μέθοδοι συμβολοσειρών

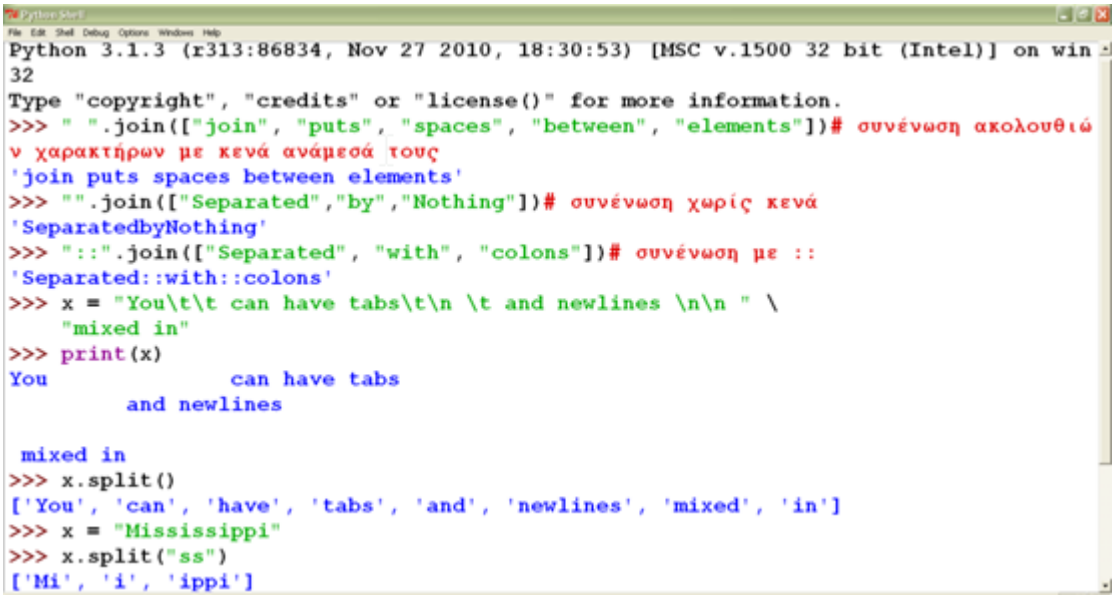
Κάποιοι από τους τελεστές που χρησιμοποιούνται με ακολουθίες χαρακτήρων είναι ο τελεστής συνένωσης +, ο τελεστής πολλαπλασιασμού * και ο τελεστής ελέγχου ύπαρξης in μιας ακολουθίας χαρακτήρων μέσα σε μία άλλη. Τα αποτελέσματα εφαρμογής τους εμφανίζονται στην παρακάτω εικόνα:

```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> s = "Hello" + "World!" # τελεστής συνένωσης
>>> print(s)
HelloWorld!
>>> t = "="*5 # τελεστής πολλαπλασιασμού του string
>>> print(t)
=====
>>> t*=2 #μπορώ να χρησιμοποιήσω και αυτήν τη μορφή
>>> print(t)
==========
>>> "nice" in "Have a nice day"
True
>>> "h" in "cat"
False
>>> "α" in "α"
True
>>> #ο τελεστής in επιστρέφει True μόνο αν το string στα αριστερά περιέχεται ή είνα
ι ίσο με το string στα δεξιά, διαφορετικά επιστρέφει False
```

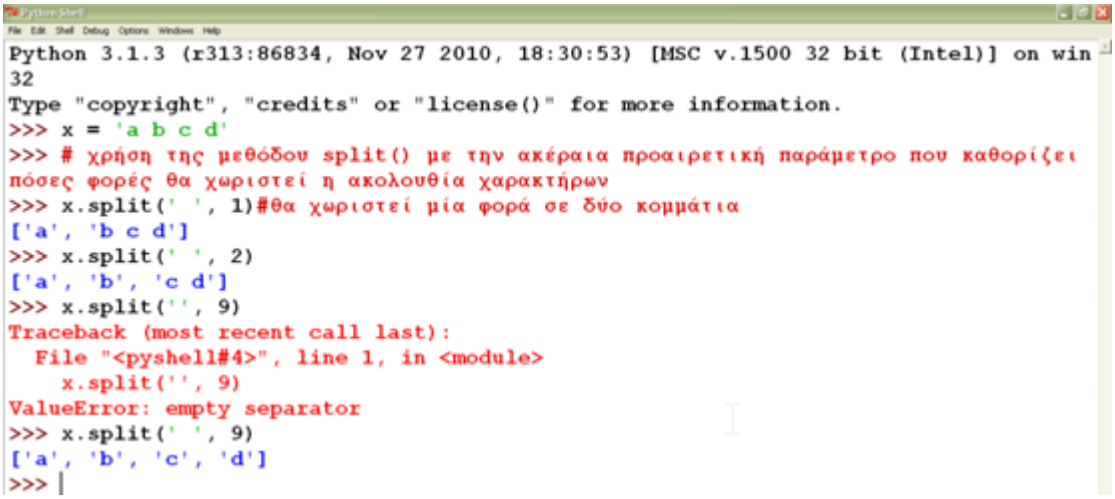
Εικόνα 3.3

Η χρήση του τελεστή συνένωσης ακολουθιών χαρακτήρων, δε συνίσταται όταν έχουμε να κάνουμε με πολλές ακολουθίες χαρακτήρων. Για να ενώσουμε πολλές ακολουθίες χαρακτήρων είναι προτιμότερο να χρησιμοποιούμε τη μέθοδο str.join(). Η str.join() παίρνει σαν παράμετρο ένα σύνολο από ακολουθίες χαρακτήρων, τις ενώνει σε μία ακολουθία και την επιστρέφει. Το str στη μέθοδο join είναι ένα string που καθορίζει το διαχωριστικό ανάμεσα στις ακολουθίες χαρακτήρων που θα ενωθούν. Η μέθοδος str.join() μπορεί ακόμη να χρησιμοποιηθεί με την ενσωματωμένη συνάρτηση reversed() για να αντιστρέψει μία ακολουθία χαρακτήρων.

Το αντίθετο κάνει η μέθοδος str.split() που παίρνει σαν παράμετρο μία μοναδική ακολουθία χαρακτήρων, τη χωρίζει σε μικρότερες και τις επιστρέφει σε μία λίστα. Παίρνει επιπλέον μία δεύτερη προαιρετική παράμετρο, που είναι ένας ακέραιος ο οποίος καθορίζει σε πόσα κομμάτια θα χωριστεί η ακολουθία (Εικόνα 3.5).

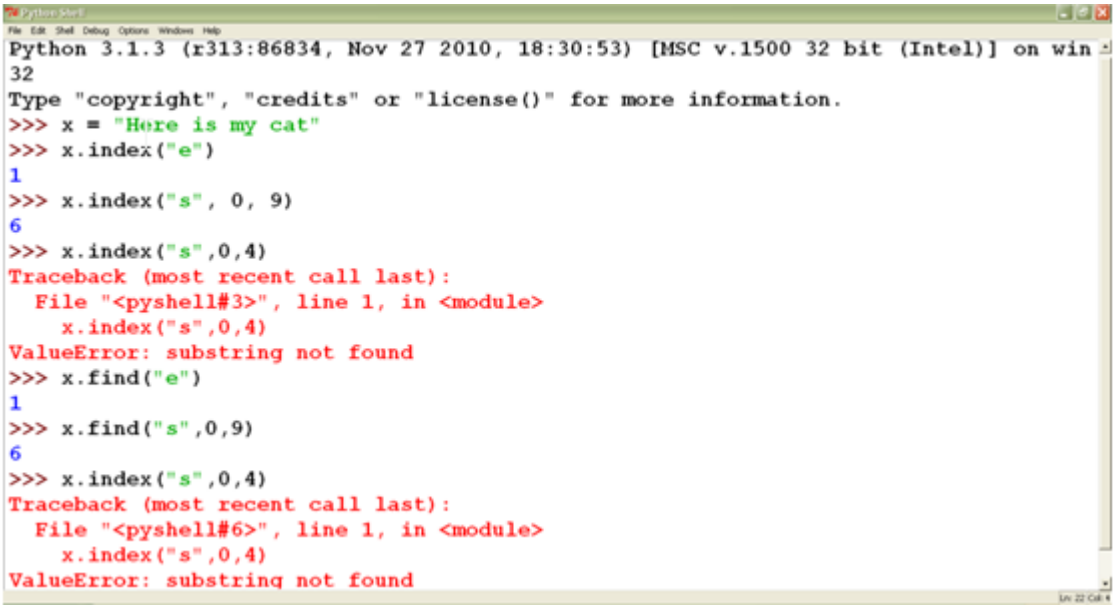


Εικόνα 3.4



Εικόνα 3.5

Σε περίπτωση που θέλουμε να βρούμε τη θέση ενός string μέσα σε ένα άλλο, έχουμε στη διάθεσή μας δύο μεθόδους να επιλέξουμε. Η πρώτη είναι η str.index(), η οποία επιστρέφει τη θέση της συμβολοσειράς ή προκαλεί την εξαίρεση ValueError σε περίπτωση αποτυχίας. Η δεύτερη μέθοδος είναι η str.find(), η οποία επιστρέφει έναν δείκτη που δείχνει στη θέση της συμβολοσειράς ή -1 σε περίπτωση αποτυχίας. Και οι δυο παίρνουν σαν πρώτη παράμετρο το string που πρέπει να ψάξουν και δύο επιπλέον προαιρετικές παραμέτρους. Η πρώτη προαιρετική παράμετρος είναι η θέση αρχής στο string από την οποία θα αρχίσει να ψάχνει και η δεύτερη προαιρετική παράμετρος η θέση τέλους της αναζήτησης.



Εικόνα 3.6

s.capitalize()	Επιστρέφει ένα αντίγραφο του string s με το πρώτο γράμμα κεφαλαίο
s.center(width, char)	Επιστρέφει ένα αντίγραφο του string s στο κέντρο ενός string μήκους width, γεμισμένο με κενά ή προαιρετικά με το string char μήκους 1
s.count(t, start, end)	Επιστρέφει των αριθμό εμφανίσεων του string t στο string s ή στο κομμάτι του string s που ορίζεται από τους δείκτες start και end.
s.encode(encoding, err)	Επιστρέφει ένα αντικείμενο byte που αντιπροσωπεύει την προεπιλεγμένη κωδικοποίηση για την αναπαράσταση των string ή την κωδικοποίηση encoding που ορίζουμε και τον τρόπο err αντιμετώπισης των λαθών
s.endswith(x, start, end)	Επιστρέφει True αν το string s ή το κομμάτι του string s που καθορίζεται από τους δείκτες start και end, τελειώνει με το sting x ή με οποιοδήποτε string της πλειάδας x.
s.expandtabs(size)	Επιστρέφει ένα αντίγραφο του string s με τα tabs να αντικαθίστανται με κενά πολλαπλασιασμένα επί 8 ή με το

	size αν προσδιορίζεται.
s.find(t, start, end)	Επιστρέφει την πιο αριστερή θέση του t που θα εντοπίσει στο string s ή στο κομμάτι του string s που προσδιορίζεται από τους δείκτες start και end ή -1 αν δεν το εντοπίσει.
s.format(…)	Επιστρέφει ένα αντίγραφο του string s μορφοποιημένο ανάλογα με τις δοθείσες παραμέτρους.
s.index(t, start, end)	Επιστρέφει την πιο αριστερή θέση του t που θα εντοπίσει στο string s ή στο κομμάτι του string s που προσδιορίζεται από τους δείκτες start και end ή προκαλεί την εξαίρεση ValueError αν δεν το εντοπίσει. Για αναζήτηση από δεξιά χρησιμοποιώ την str.rindex()
s.isalnum()	Επιστρέφει True αν το string s δεν είναι κενό και κάθε χαρακτήρας σε αυτό είναι αλφαριθμητικός.
s.isalpha()	Επιστρέφει True αν το string s δεν είναι κενό και κάθε χαρακτήρας σε αυτό είναι αλφαβητικός.
s.isdecimal()	Επιστρέφει True αν το string s δεν είναι κενό και κάθε χαρακτήρας σε αυτό είναι ένα δεκαδικό Unicode ψηφίο .
s.isdigit()	Επιστρέφει True αν το string s δεν είναι κενό και κάθε χαρακτήρας σε αυτό είναι ASCII ψηφίο.
s.isidentifier()	Επιστρέφει True αν το string s δεν είναι κενό και είναι έγκυρο αναγνωριστικό.
s.islower()	Επιστρέφει True αν όλοι οι χαρακτήρες του string s είναι πεζοί.
s.isnumeric()	Επιστρέφει True αν το string s δεν είναι κενό και όλοι οι χαρακτήρες του είναι αριθμητικοί Unicode όπως ένα ψηφίο ή κλάσμα.

s.isprintable()	Επιστρέφει True αν το string s είναι κενό ή κάθε χαρακτήρας σε αυτό πρόκειται να εκτυπωθεί και περιλαμβάνει κενό, αλλά όχι αλλαγή γραμμής.
s.isspace()	Επιστρέφει True αν το string s δεν είναι κενό και όλοι οι χαρακτήρες του είναι κενά (whitespaces)
s.istitle()	Επιστρέφει True αν το string s είναι title cased και όχι κενό πχ upper-
s.isupper()	Επιστρέφει True αν όλοι οι χαρακτήρες του string s είναι σε κεφαλαία.
s.join(seq)	Ενώνει τα strings που δίνονται ως παράμετροι με το string str ανάμεσά τους σε ένα string και το επιστρέφει.
s.ljust(width, char)	Επιστρέφει ένα αντίγραφο του string s στοιχισμένο αριστερά σε ένα string μήκους width, γεμισμένο με κενά ή προαιρετικά με το string char μήκους 1
s.lower()	Επιστρέφει ένα αντίγραφο του string s με όλους τους κεφαλαίους χαρακτήρες σε πεζούς.
s.maketrans()	Συνδυάζεται με την str.translate()
s.partition(t)	Χωρίζει το string s σε τρία τμήματα με βάση το string t και επιστρέφει μία πλειάδα τριών strings. Αν το t δεν βρίσκεται στο string s, επιστρέφει το s και δύο κενά strings.
s.replace(t, u, n)	Επιστρέφει ένα αντίγραφο του string s με όλες / συγκεκριμένο αριθμό n των εμφανίσεων του string t να έχει αντικατασταθεί με το string u.
s.split(t, n)	Επιστρέφει μία λίστα από n strings που έχουν προκύψει από τον τεμαχισμό του string t.
s.splitlines(f)	Επιστρέφει μία λίστα με τις γραμμές που παράγονται από

	τον τεμαχισμό του string s στα σημεία τερματισμού της γραμμής. Αν το f είναι True τότε στο τέλος των string εμφανίζει και τους χαρακτήρες τερματισμού.
s.startswith(x, start, end)	Επιστρέφει True αν το string s ή το κομμάτι του s που προσδιορίζεται από τους δείκτες start και end, ξεκινάει με το string x ή με οποιοδήποτε από τα strings της πλειάδας x. Διαφορετικά επιστρέφει False.
s.strip(chars)	Επιστρέφει ένα αντίγραφο του s αφού διαγράψει τα κενά από την αρχή και το τέλος του string. Η str.lstrip() αφαιρεί μόνο από την αρχή και η str.rstrip() αφαιρεί μόνο από το τέλος.
s.swapcase()	Επιστρέφει ένα αντίγραφο του s αφού μετατρέψει τους κεφαλαίους χαρακτήρες σε πεζούς και τους πεζούς σε κεφαλαίους.
s.title()	Επιστρέφει ένα αντίγραφο του s όπου το πρώτο γράμμα κάθε λέξης είναι κεφαλαίο και όλα τα υπόλοιπα πεζά.
s.translate()	Συνδυάζεται με την str.maketrans().
s.upper()	Επιστρέφει ένα αντίγραφο του string s με κεφαλαίους χαρακτήρες.
s.zfill(w)	Γεμίζει το string s με μηδενικά στα αριστερά, για να γεμίσει ένα πεδίο συγκεκριμένου μεγέθους width. Το string s δεν αποκόπτεται ποτέ.

Πίνακας 3.1 Μέθοδοι των string

Δύο μέθοδοι που δεν αναφέρθηκαν καθόλου στον πίνακα είναι οι int() και float() με τις οποίες μπορούμε να μετατρέψουμε τις ακολουθίες χαρακτήρων σε ακέραιους αριθμούς και αριθμούς κινητής υποδιαστολής αντίστοιχα. Αν περάσουμε σε αυτές σαν παράμετρο ένα string που δεν μπορεί να ερμηνευτεί σαν αριθμός, τότε

προκαλείται η εξαίρεση ValueError. Επιπλέον μπορούμε να περάσουμε στη μέθοδο int() μία δεύτερη προαιρετική παράμετρο, που θα προσδιορίζει τη βάση του συστήματος που θα χρησιμοποιηθεί κατά την διερμηνεία του string εισόδου:

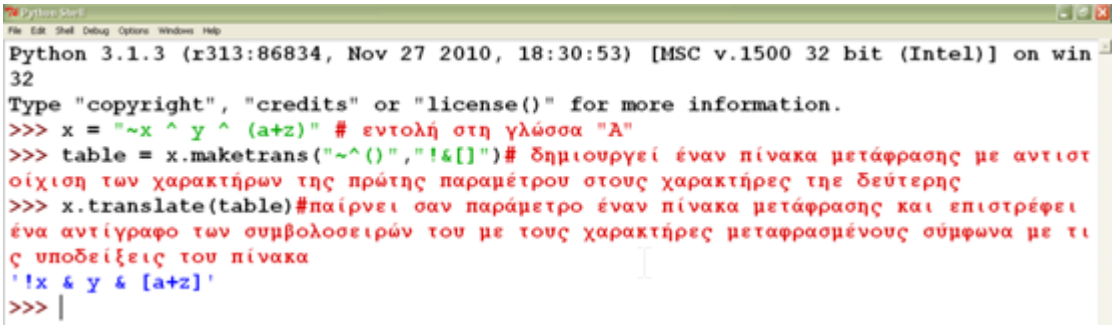
```
>>> int("120")# μετατροπή string 120 σε ακέραιο
120
>>> int("120.4")# δεν μπορώ να δώσω παράμετρο float στην int μέθοδο
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    int("120.4")# δεν μπορώ να δώσω παράμετρο float στην int μέθοδο
ValueError: invalid literal for int() with base 10: '120.4'
>>> int('120',8)#ερμηνεύει το string 120 στο οχταδικό σύστημα
80
>>> int('120',2)#ερμηνεύει το string 120 στο δυαδικό σύστημα
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    int('120',2)#ερμηνεύει το string 120 στο δυαδικό σύστημα
ValueError: invalid literal for int() with base 2: '120'
>>> int('110',2)
6
>>> #θα πρέπει το string να μπορεί να μετατραπεί στο δυαδικό σύστημα αλλιώς προκαλε
ίται λάθος
>>> float("129.56")
129.56
>>> float("xxx")#δεν μπορεί να μετατρέψει χαρακτήρες σε αριθμό
Traceback (most recent call last):
```

Εικόνα 3.7

Μία από τις μεθόδους που αναφέρεται στον Πίνακα 3.1 είναι η s.replace(t, u, n), που χρησιμοποιείται για να αντικαταστήσει τις εμφανίσεις ενός ή πολλών χαρακτήρων του string t με αυτά που έχουμε ορίσει στη δεύτερη παράμετρο u.

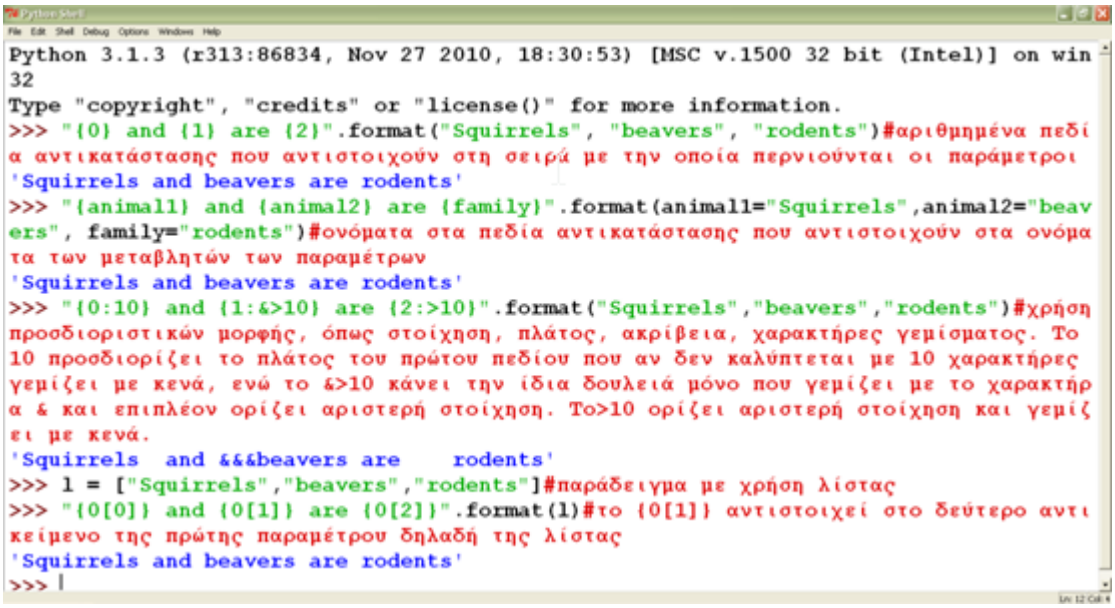
Οι μέθοδοι str.maketrans() και str.translate() που αναφέρονται αλλά δεν περιγράφονται στον πίνακα, μπορούν να χρησιμοποιηθούν μαζί για να μεταφράσουν τους χαρακτήρες μέσα σε ένα string σε άλλους χαρακτήρες. Αν και χρησιμοποιούνται σπάνια παρέχουν, όποτε τις χρειαζόμαστε, μεγάλη ευκολία.

Για παράδειγμα έστω ότι θέλουμε να μεταφράσουμε την εντολή μίας γλώσσας προγραμματισμού «Α» σε μία γλώσσα προγραμματισμού «Β». Η εντολή της «Α» χρησιμοποιεί τα σύμβολα ~ για λογικό όχι, ^ για λογικό και αγκύλες ( ) ενώ στη γλώσσα «Β» τα αντίστοιχα σύμβολα είναι !, & και [ ]. Με τη μέθοδο replace θα έπρεπε να κάνουμε τις αλλαγές μία προς μία. Ένας πιο αποδοτικός τρόπος είναι:



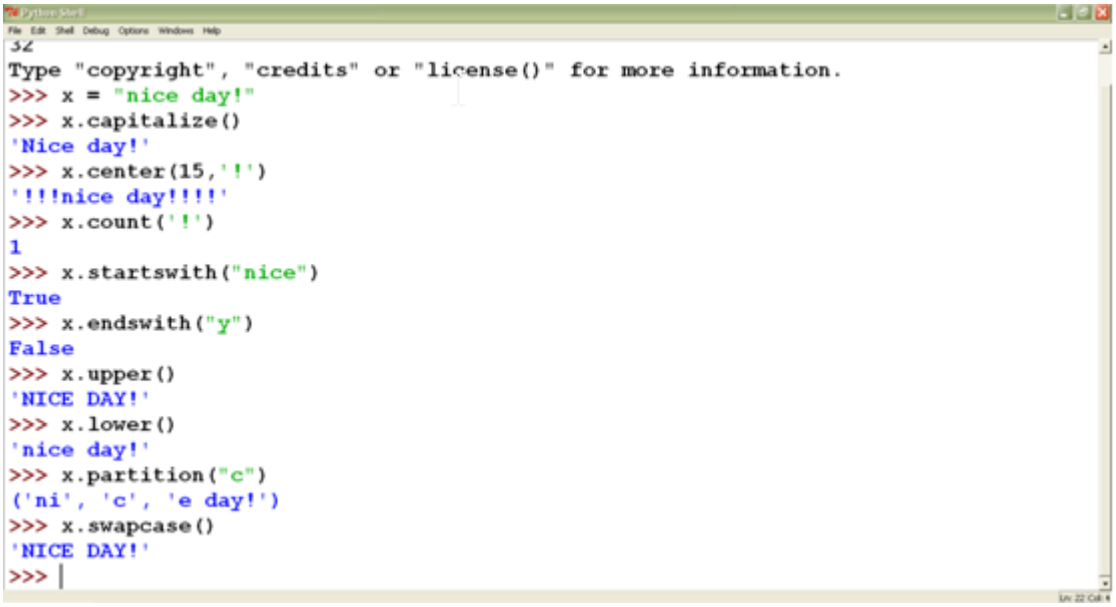
Εικόνα 3.8

Μία ακόμη μέθοδος που θα πρέπει να αναλυθεί είναι η str.format() με την οποία μπορούμε να μορφοποιήσουμε μία ακολουθία χαρακτήρων. Η μέθοδος format αντιστοιχεί τα τμήματα ενός string που βρίσκονται μέσα σε { }, με τις τιμές των παραμέτρων της μεθόδου. Αν χρειαστεί να χρησιμοποιήσουμε την κυριολεκτική σημασία των συμβόλων { και }, τότε τα γράφουμε στη μορφή {{ και }}. Υπάρχουν τέσσερις τρόποι αντιστοίχησης των τμημάτων αντικατάστασης με τις παραμέτρους που φαίνονται στα παρακάτω παραδείγματα:

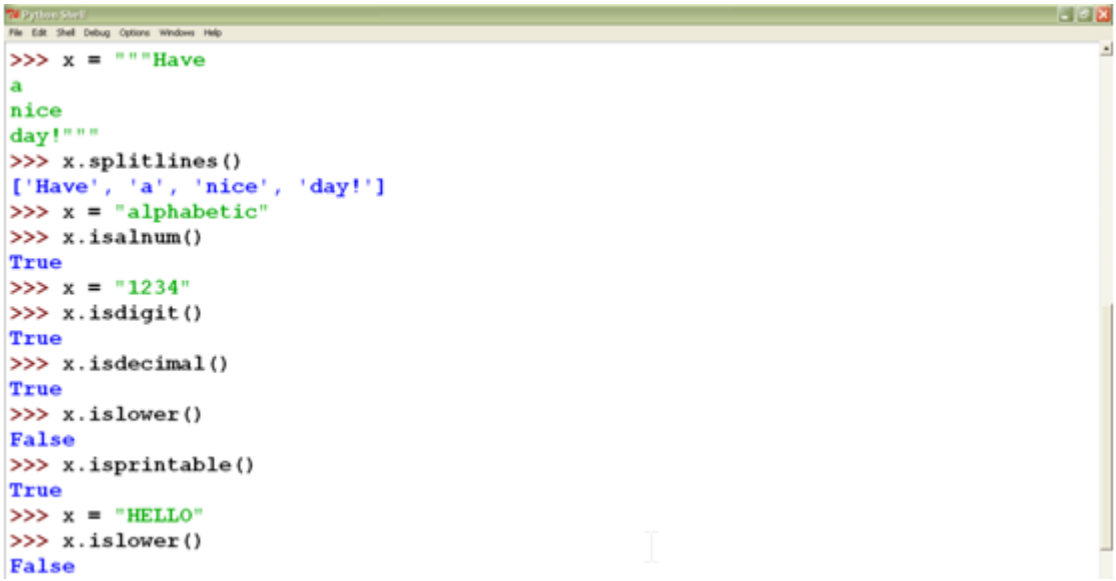


Εικόνα 3.9

Ακολουθούν παραδείγματα που χρησιμοποιούν τις μεθόδους του πίνακα 3.1 για κατανόηση της λειτουργίας τους πάνω στις ακολουθίες χαρακτήρων.



Εικόνα 3.10



Εικόνα3.11

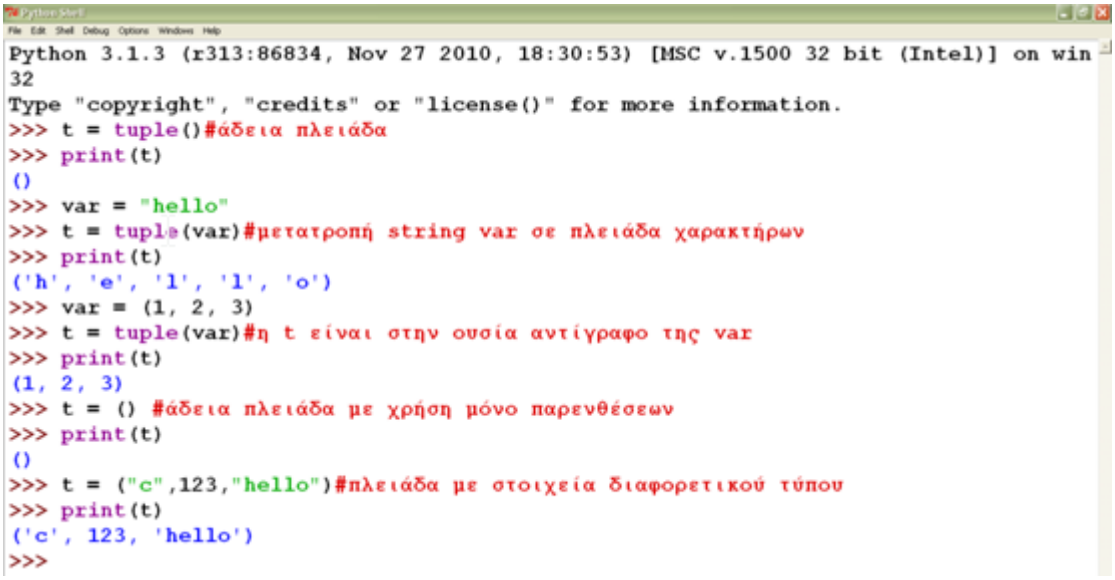
4. ΣΥΛΛΟΓΙΚΟΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Σε αυτό το κεφάλαιο θα δούμε πώς συγκεντρώνουμε στοιχεία δεδομένων χρησιμοποιώντας τους συλλογικούς τύπων δεδομένων της Python, δηλαδή τις πλειάδες, τις λίστες, τα σύνολα και τα λεξικά.

4.1 Πλειάδες – tuples

Πλειάδα είναι μία διατεταγμένη ακολουθία που περιέχει μηδέν ή περισσότερα αντικείμενα που μπορεί να είναι διαφορετικού τύπου. Οι πλειάδες είναι αμετάβλητες, οπότε δεν μπορούμε να αντικαταστήσουμε ή να διαγράψουμε στοιχεία τους. Αν θέλουμε να κάνουμε κάτι τέτοιο μπορούμε να μετατρέψουμε την πλειάδα σε λίστα με τη μέθοδο list() ή απλά να χρησιμοποιήσουμε εξαρχής τον τύπο δεδομένων λίστα. Οι πλειάδες υποστηρίζουν την ίδια σύνταξη τεμαχισμού με τις ακολουθίες χαρακτήρων. Αυτό κάνει ευκολότερη την εξαγωγή στοιχείων από μία πλειάδα.

Για τη δημιουργία μίας πλειάδας μπορούμε να καλέσουμε τη συνάρτηση tuple(). Όταν καλείται χωρίς παραμέτρους δημιουργεί και επιστρέφει μία άδεια πλειάδα. Όταν καλείται με μία παράμετρο ίδιου τύπου επιστρέφει ένα ακριβές αντίγραφο της παραμέτρου, ενώ αν η παράμετρος είναι διαφορετικού τύπου προσπαθεί να τη μετατρέψει στ πλειάδα. Η συνάρτηση tuple() δεν δέχεται παραπάνω από μία παράμετρο. Μπορούμε ακόμη να δημιουργήσουμε μία πλειάδα με χρήση παρενθέσεων. Οι παρενθέσεις χωρίς περιεχόμενο ( ) δημιουργούν μία κενή πλειάδα, ενώ για πλειάδες με ένα μόνο στοιχείο απαραίτητο είναι το κόμμα (2,).

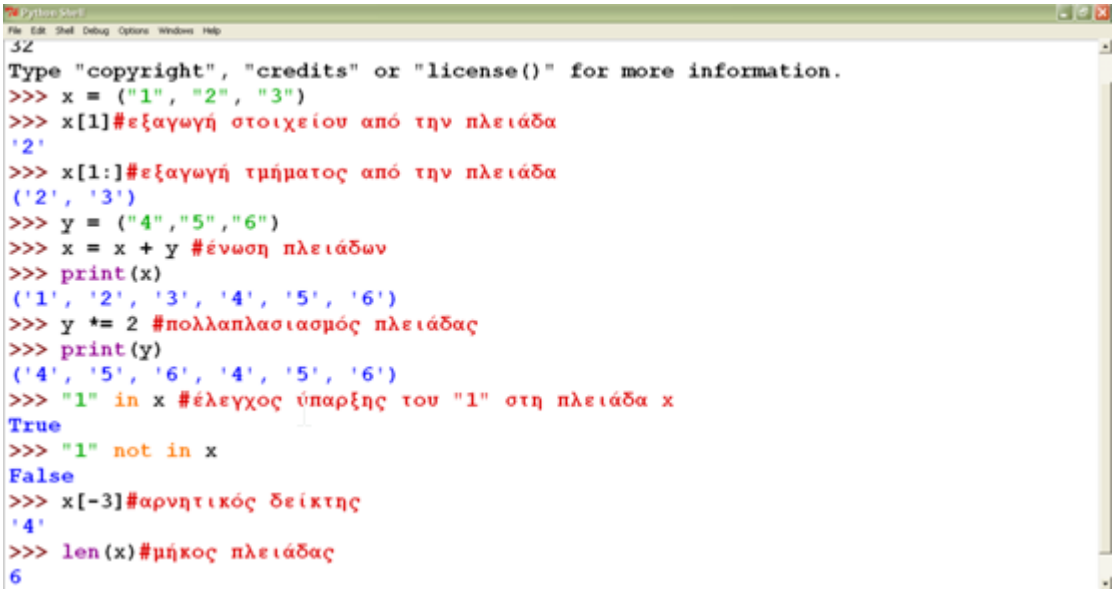


Εικόνα 4.1

Οι πλειάδες δεικτοδοτούνται αλλά σε κάθε θέση έχουμε μία αναφορά αντικειμένου και όχι έναν χαρακτήρα, όπως συμβαίνει με τα strings. Και εδώ υποστηρίζονται οι αρνητικοί δείκτες.

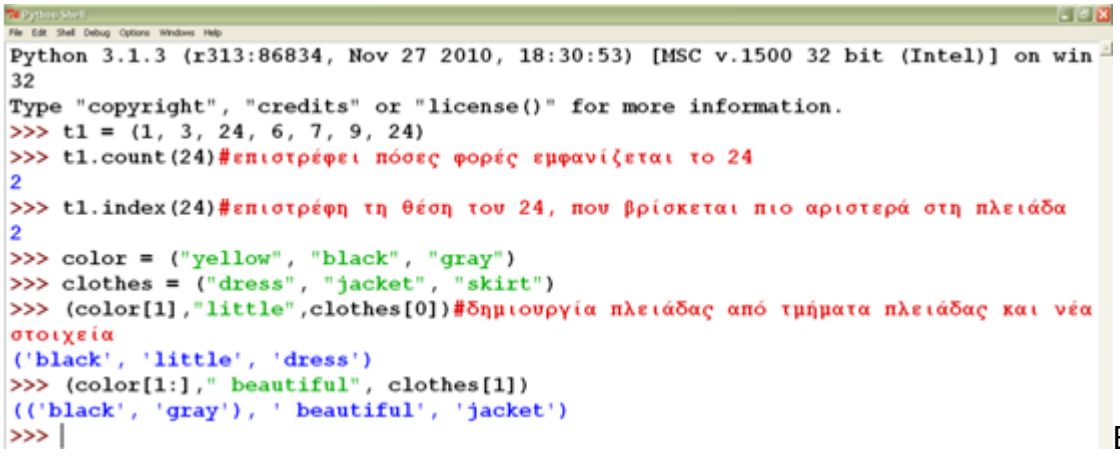
t[-5]	t[-4]	t[-3]	t[-2]	t[-1]
‘yellow’	‘black’	122	‘green’	30.28
t[0]	t[1]	t[2]	t[3]	t[4]

Οι τελεστές που εφαρμόζονται στις πλειάδες είναι ο τελεστής ένωσης +, ο τελεστής πολλαπλασιασμού *, ο τελεστής εξαγωγής [ ] και οι τελεστές in και not in για έλεγχο ύπαρξης ενός συγκεκριμένου στοιχείου στην πλειάδα. Αν και οι πλειάδες είναι αμετάβλητες, οι τελεστές += και *= μπορούν να εφαρμοστούν σε αυτές Η Python δεν τροποποιεί την πλειάδα, αλλά δημιουργεί παρασκηνιακά μία νέα πλειάδα για να κρατήσει το αποτέλεσμα. Αυτό συμβαίνει και με τα strings που είναι αμετάβλητα.



Εικόνα 4.2

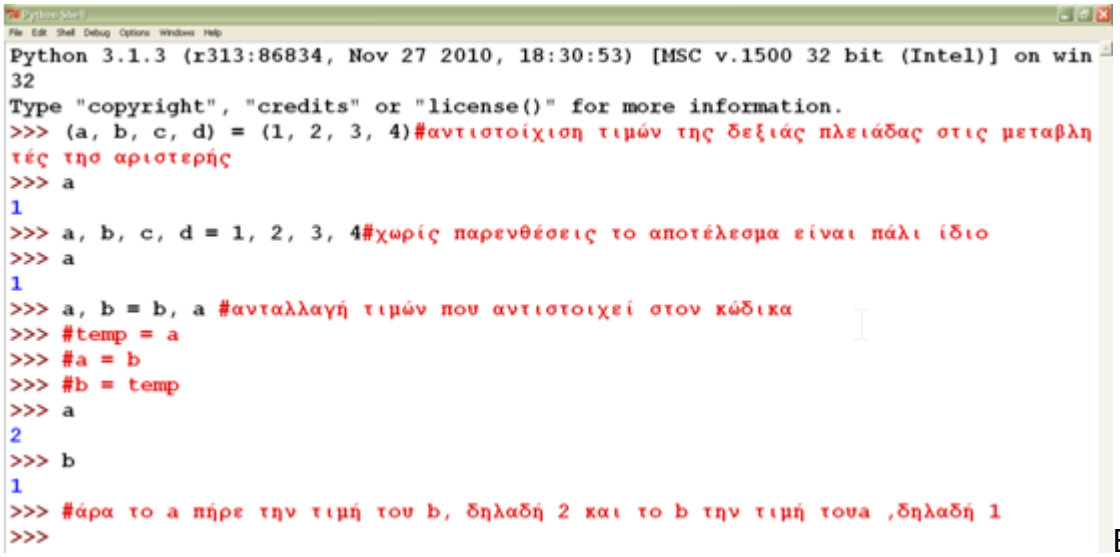
Οι πλειάδες παρέχουν δύο μόνο δικές τους μεθόδους. Η πρώτη είναι η t.count(x), η οποία μετράει και επιστρέφει τον αριθμό των αντικειμένων x που υπάρχουν στη πλειάδα t. Η δεύτερη μέθοδος είναι η t.index(x), η οποία επιστρέφει τη θέση του στοιχείου x, που βρίσκεται πιο αριστερά στη πλειάδα t ή προκαλεί την εξαίρεση ValueError αν το στοιχείο x δεν υπάρχει στη πλειάδα. Επιπλέον οι πλειάδες μπορούν να συγκριθούν χρησιμοποιώντας τους τελεστές σύγκρισης <, <=, !=, >=, > με την σύγκριση να γίνεται στοιχείο προς στοιχείο.



Εικόνα 4.3

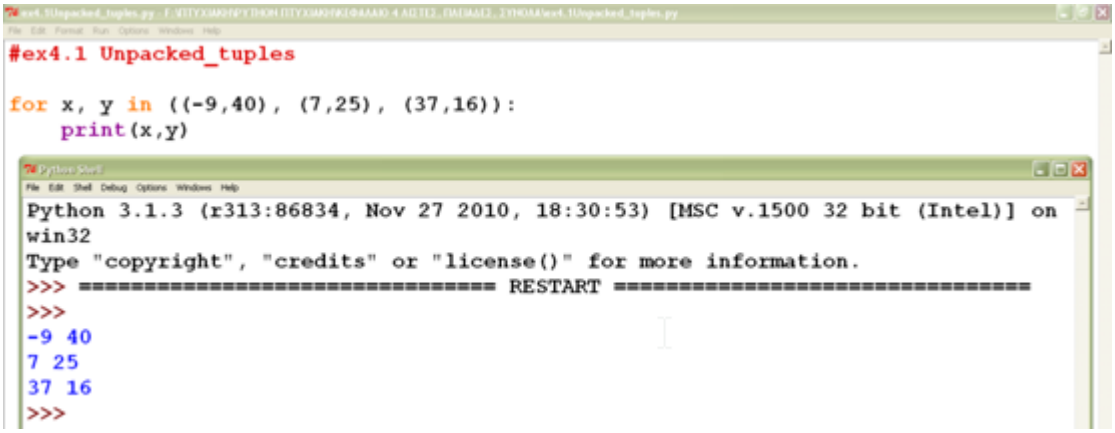
Η Python επιτρέπει τις πλειάδες να εμφανίζονται στην αριστερή πλευρά ενός τελεστή εκχώρησης, ώστε οι μεταβλητές της πλειάδας αυτής να παίρνουν τις τιμές της πλειάδας στη δεξιά πλευρά. Τότε λέμε ότι οι τιμές της δεξιάς πλειάδας

πακετάρονται σε μία πλειάδα και ξε-πακετάρονται στις μεταβλητές της αριστερής. Αυτό μπορεί να γίνει και χωρίς τη χρήση παρενθέσεων. Αυτή η ιδιότητα είναι πολύ χρήσιμη για ανταλλαγή τιμών.



Εικόνα 4.4

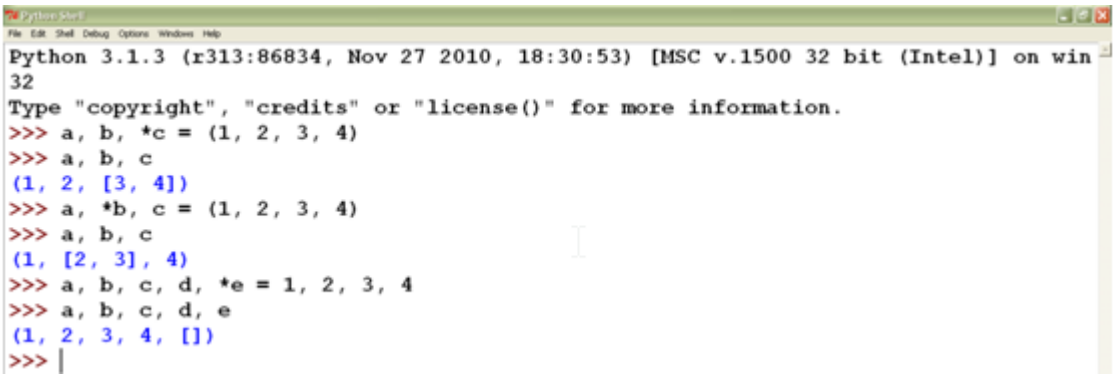
Παράδειγμα 4.1 προσπέλασης μίας πλειάδας, που περιέχει πλειάδες δύο στοιχείων και ξε – πακετάρισμα των τιμών τους στις μεταβλητές x και y.



Παράδειγμα 4.1

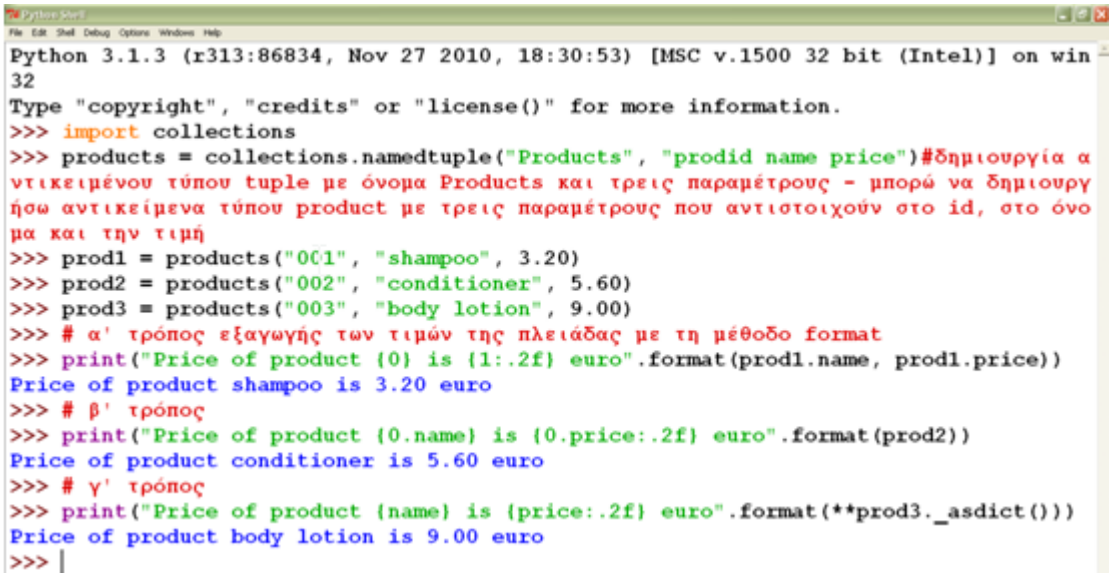
Ένα ακόμη χαρακτηριστικό σχετικό με το ξε-πακετάρισμα των πλειάδων, επιτρέπει ένα στοιχείο να «σημαδεύεται» από έναν αστερίσκο *, για να απορροφά οποιαδήποτε αριθμητικά στοιχεία δεν αντιστοιχούν-πακετάρονται σε κάποια μεταβλητή. Το «σημαδεμένο» στοιχείο λαμβάνει όλα τα πλεονάζοντα στοιχεία σε

μία λίστα, ενώ αν δεν υπάρχουν καθόλου πλεονάζοντα στοιχεία, τότε λαμβάνει μία κενή λίστα. Παράδειγμα χρήσης του:



Εικόνα 4.5

Η Python μας δίνει την δυνατότητα να δώσουμε ονόματα σε πλειάδες. Μία ονοματισμένη πλειάδα έχει την ίδια απόδοση και χαρακτηριστικά με μία πλειάδα χωρίς όνομα, με τη διαφορά ότι έχουμε τη δυνατότητα να αναφερόμαστε στα στοιχεία της πλειάδας με τα ονόματά τους και όχι με τον δείκτη θέσης. Αυτό το χαρακτηριστικό μας επιτρέπει να δημιουργήσουμε συλλογές στοιχείων δεδομένων. Το module collections παρέχει τη μέθοδο namedtuple() η οποία δημιουργεί πλειάδες με προσαρμοσμένους τύπους δεδομένων.



Εικόνα 4.6

Υπάρχουν τρεις τρόποι εξαγωγής στοιχείων μιας πλειάδας :

α)Τρόπος προσπέλασης και μορφοποίησης των τιμών των στοιχείων της πλειάδας με χρήση της μεθόδου format() για την αντιστοίχιση τους στα πεδία για αντικατάσταση.

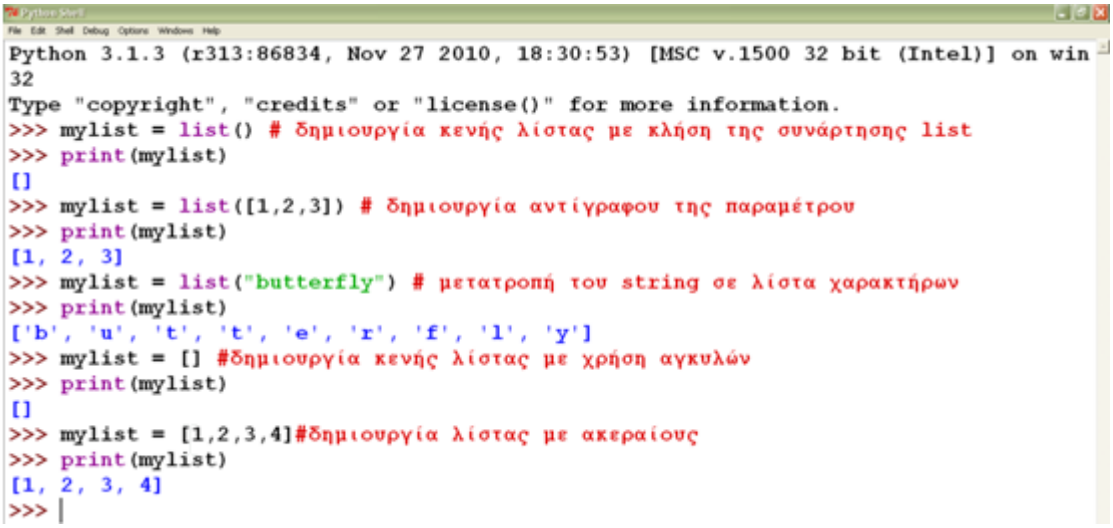
β)Δίνουμε στη μέθοδο format() μία μοναδική παράμετρο, που είναι το όνομα της πλειάδας και προσδιορίζουμε τα πεδία για αντικατάσταση με τα ονόματα των στοιχείων. Αυτός ο κώδικας είναι πιο ευανάγνωστος από το να χρησιμοποιούμε απλά παραμέτρους θέσης.

γ)Η ιδιωτική μέθοδος _asdict() της πλειάδας επιστρέφει τα ζεύγη κλειδιά – τιμές, όπου κλειδί είναι το όνομα ενός στοιχείου της πλειάδας και τιμή είναι η αντίστοιχη τιμή του. Αυτό ονομάζεται mapping unpacking και το κάνουμε για να περάσουμε τα ζεύγη κλειδιά-τιμές ως παραμέτρους στη μέθοδο str.format().

4.2 Λίστες – lists

Λίστα είναι μία διατεταγμένη ακολουθία που περιέχει μηδέν ή περισσότερα αντικείμενα που μπορεί να είναι διαφορετικού τύπου συμπεριλαμβανομένων τύπων ακολουθίας όπως λίστες και πλειάδες. Οι λίστες είναι μεταβλητές, οπότε μπορούμε να αντικαταστήσουμε, να εισάγουμε ή να διαγράψουμε οποιοδήποτε από τα στοιχεία τους ή ακόμα και τμήματά τους. Οι λίστες υποστηρίζουν την ίδια σύνταξη τεμαχισμού με τις ακολουθίες χαρακτήρων και τις πλειάδες. Αυτό κάνει εύκολη την εξαγωγή στοιχείων από μία λίστα.

Για τη δημιουργία μίας λίστας μπορούμε να καλέσουμε τη συνάρτηση list().Όταν καλείται χωρίς παραμέτρους δημιουργεί και επιστρέφει μία άδεια λίστα. Όταν καλείται με μία παράμετρο ίδιου τύπου επιστρέφει ένα ακριβές αντίγραφο της παραμέτρου, ενώ αν η παράμετρος είναι διαφορετικού τύπου προσπαθεί να τη μετατρέψει σε λίστα. Η συνάρτηση list() δεν δέχεται παραπάνω από μία παράμετρο. Μπορούμε ακόμη να δημιουργήσουμε μία λίστα με χρήση αγκυλών []. Οι αγκύλες χωρίς περιεχόμενο δημιουργούν μία κενή λίστα ενώ για μία λίστα με ένα ή περισσότερα στοιχεία πρέπει μέσα στις αγκύλες να υπάρχει μία ακολουθία στοιχείων χωρισμένα με κόμμα.

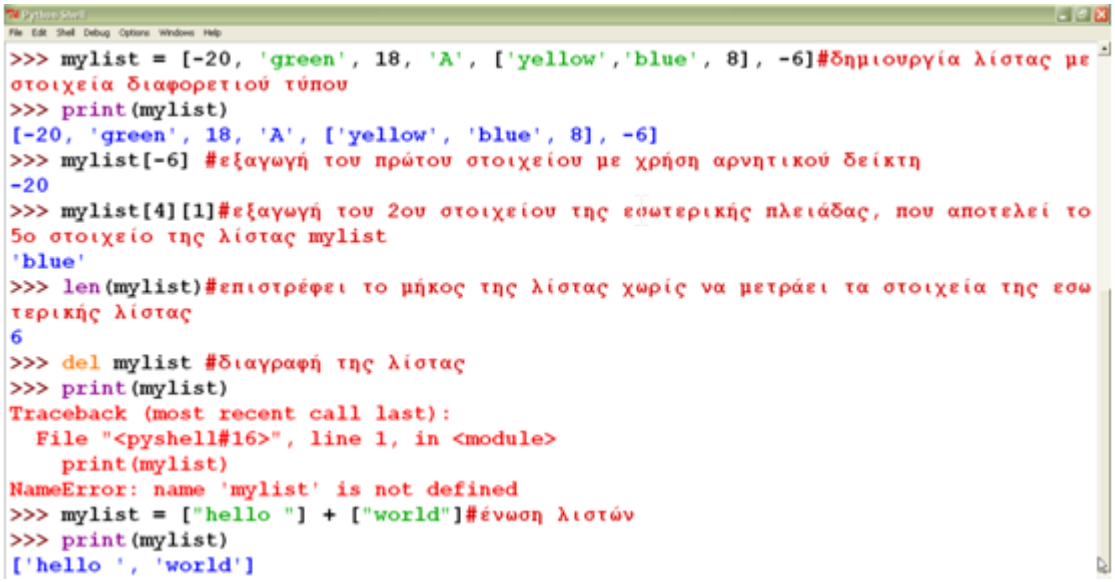


Εικόνα 4.7

Οι λίστες δεικτοδοτούνται αλλά σε κάθε θέση έχουμε μία αναφορά αντικειμένου και όχι έναν χαρακτήρα, όπως συμβαίνει με τα strings. Και εδώ υποστηρίζονται οι αρνητικοί δείκτες.

L[-6]	L[-5]	L[-4]	L[-3]	L[-2]	L[-1]
-20	‘green’	18	‘A’	[‘yellow’, ‘blue’, 8]	-6
L[0]	L[1]	L[2]	L[3]	L[4]	L[5]

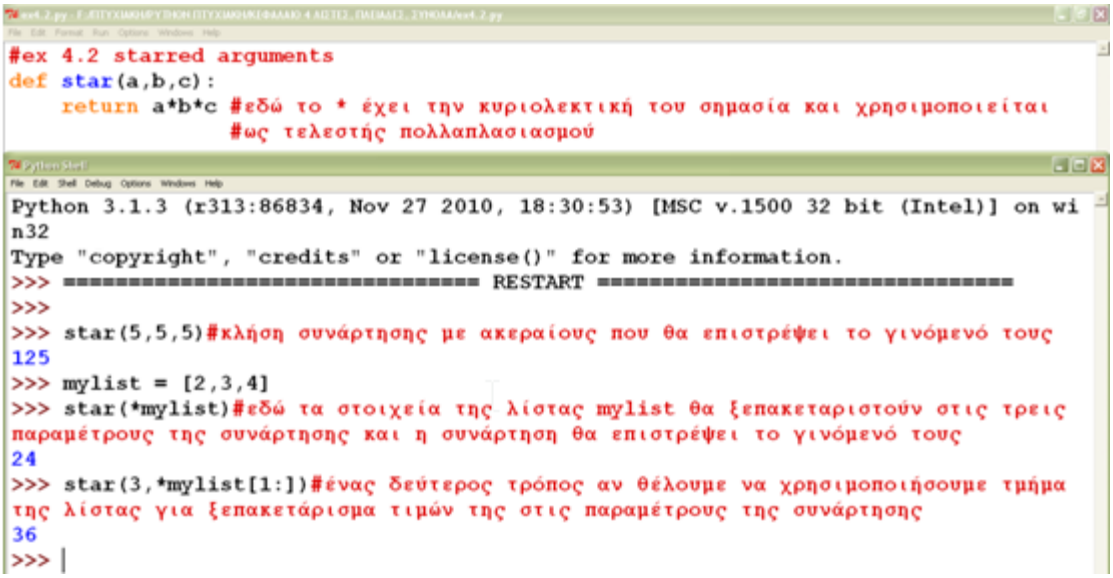
Οι τελεστές που εφαρμόζονται στις λίστες είναι ο τελεστής ένωσης +, ο τελεστής πολλαπλασιασμού *, ο τελεστής εξαγωγής [ ] και οι τελεστές in και not in για έλεγχο ύπαρξης ενός συγκεκριμένου στοιχείου στη λίστα. Επιπλέον οι λίστες μπορούν να χρησιμοποιηθούν με τις ενσωματωμένες συναρτήσεις len(), min() και max() και την εντολή del για διαγραφή. Τέλος οι λίστες μπορούν να συγκριθούν με τους τελεστές σύγκρισης <, <=, >, >=, !=, με την σύγκριση να γίνεται στοιχείο προς στοιχείο.



```
>>> mylist = [-20, 'green', 18, 'A', ['yellow','blue', 8], -6]#δημιουργία λίστας με
στοιχεία διαφορετιού τύπου
>>> print(mylist)
[-20, 'green', 18, 'A', ['yellow', 'blue', 8], -6]
>>> mylist[-6] #εξαγωγή του πρώτου στοιχείου με χρήση αρνητικού δείκτη
-20
>>> mylist[4][1]#εξαγωγή του 2ου στοιχείου της εσωτερικής πλειάδας, που αποτελεί το
5ο στοιχείο της λίστας mylist
'blue'
>>> len(mylist)#επιστρέφει το μήκος της λίστας χωρίς να μετράει τα στοιχεία της εσω
τερικής λίστας
6
>>> del mylist #διαγραφή της λίστας
>>> print(mylist)
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    print(mylist)
NameError: name 'mylist' is not defined
>>> mylist = ["hello "] + ["world"]#ένωση λιστών
>>> print(mylist)
['hello ', 'world']
```

Εικόνα 4.8

Σε μερικές περιπτώσεις θέλουμε να εξάγουμε δύο ή περισσότερα κομμάτια από μία λίστα σε μία φορά, οπότε ο τελεστής εξαγωγής [] δεν είναι κατάλληλος. Για το λόγο αυτό κάνουμε ξεπακετάρισμα ακολουθίας και χρησιμοποιούμε αστεροειδή εκφράσεις (χαρακτήρας * μπροστά από μία μεταβλητή).



```
#ex 4.2 starred arguments
def star(a,b,c):
    return a*b*c #εδώ το * έχει την κυριολεκτική του σημασία και χρησιμοποιείται
                 #ως τελεστής πολλαπλασιασμού
```

```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on wi
n32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
>>> star(5,5,5)#κλήση συνάρτησης με ακεραίους που θα επιστρέψει το γινόμενό τους
125
>>> mylist = [2,3,4]
>>> star(*mylist)#εδώ τα στοιχεία της λίστας mylist θα ξεπακεταριστούν στις τρεις
παραμέτρους της συνάρτησης και η συνάρτηση θα επιστρέψει το γινόμενό τους
24
>>> star(3,*mylist[1:])#ένας δεύτερος τρόπος αν θέλουμε να χρησιμοποιήσουμε τμήμα
της λίστας για ξεπακετάρισμα τιμών της στις παραμέτρους της συνάρτησης
36
>>>
```

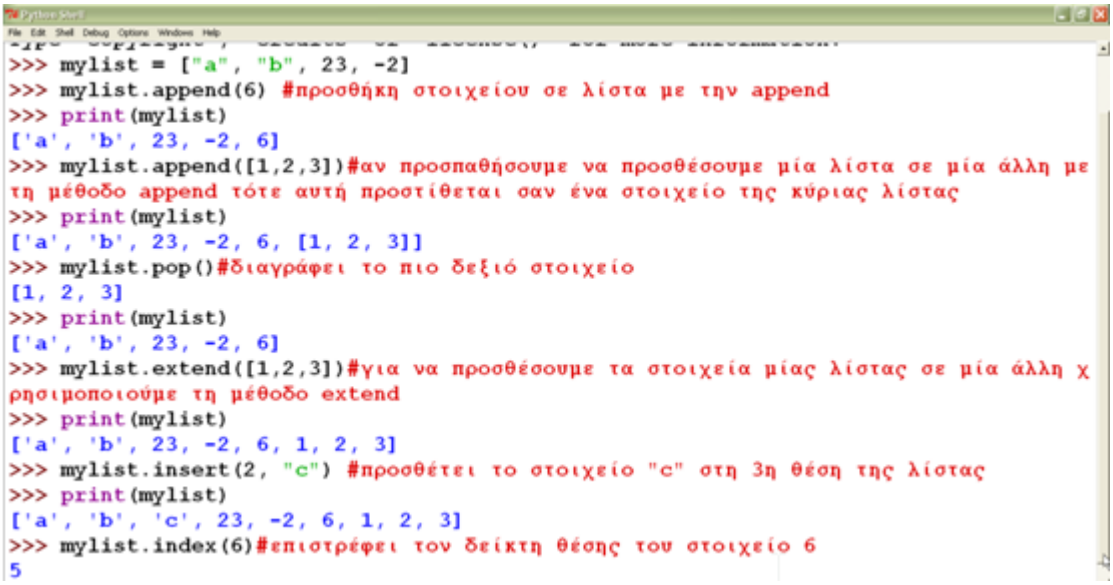
Παράδειγμα 4.2

Στον Πίνακα 4.1 απαριθμούνται οι μέθοδοι των λιστών που εκτελούν διάφορες λειτουργίες πάνω σε αυτές και η Εικόνα 4.12 εμφανίζει παραδείγματα χρήσης τους.

L.append(x)	Προσθέτει το στοιχείο x στο τέλος της λίστας L

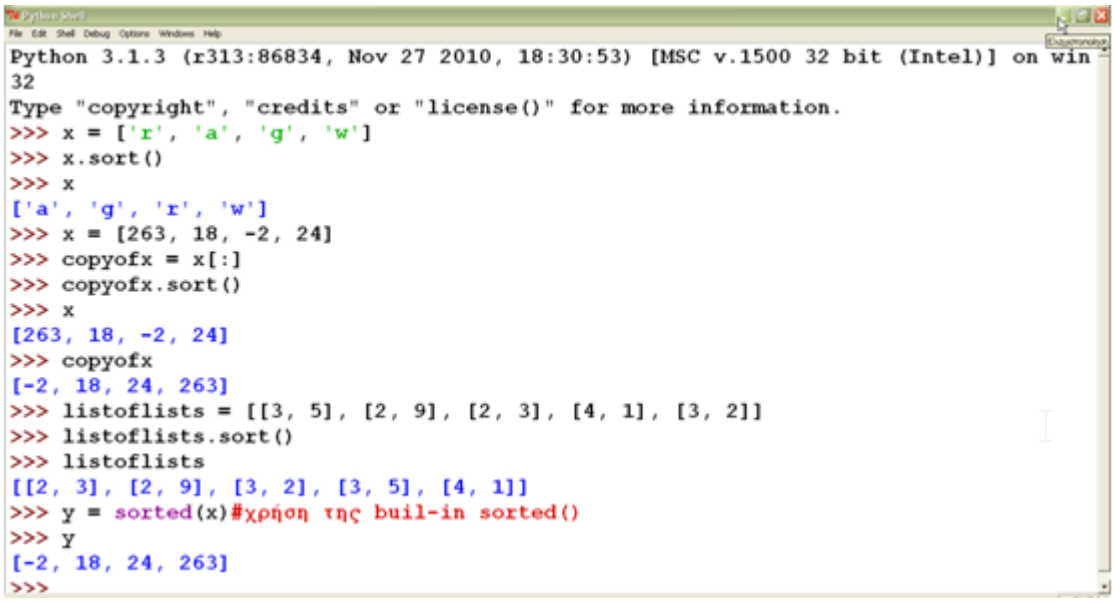
L.count(x)	Επιστρέφει έναν ακέραιο που εκφράζει πόσες φορές εμφανίζεται το στοιχείο x στη λίστα L
L.extend(m)	Προσθέτει όλα τα στοιχεία της λίστας m στο τέλος της λίστας L. Το ίδιο αποτέλεσμα έχει και η χρήση του τελεστή += : L += m
L.index(x, start, end)	Επιστρέφει τον δείκτη της θέσης του στοιχείου x που βρίσκεται πιο αριστερά στη λίστα L ή στο τμήμα της λίστας που καθορίζεται από τους δείκτες start:end διαφορετικά προκαλεί εξαίρεση ValueError
L.insert(i, x)	Προσθέτει το στοιχείο x στη θέση I της λίστας L.
L.pop()	Επιστρέφει και διαγράφει το στοιχείο που βρίσκεται πιο δεξιά στη λίστα L.
L.pop(i)	Επιστρέφει και διαγράφει το στοιχείο που βρίσκεται στη θέση i στη λίστα L.
L.remove(x)	Διαγράφει τη πρώτη εμφάνιση του στοιχείου x στη λίστα L ή προκαλεί εξαίρεση ValueError αν δεν βρει το στοιχείο x.
L.reverse()	Αντιστρέφει τα στοιχεία της λίστας L.
L.sort	Ταξινομεί τη λίστα L και δέχεται τις ίδιες παραμέτρους με την ενσωματωμένη συνάρτηση sorted().

Πίνακας 4.1 Μέθοδοι λιστών



Εικόνα 4.9

Με τη μέθοδο sort μπορούμε να ταξινομήσουμε μία λίστα. Αν θέλουμε να ταξινομήσουμε τη λίστα χωρίς να αλλάξουμε την αρχική πρέπει να δημιουργήσουμε ένα αντίγραφο της. Η μέθοδος sort μπορεί να ταξινομήσει οτιδήποτε αρκεί όλα τα αντικείμενα στη λίστα να έχουν τύπους που να μπορούν να συγκριθούν. Αυτό σημαίνει πως αν χρησιμοποιήσουμε τη μέθοδο sort για μία λίστα που περιέχει ακεραίους και ακολουθίες χαρακτήρων, θα προκαλέσει εξαίρεση. Από την άλλη μπορούμε να ταξινομήσουμε μία λίστα από λίστες με την ταξινόμηση να γίνεται με βάση το πρώτο στοιχείο κάθε λίστας και μετά με το δεύτερο.



Εικόνα 4.10

4.3 Σύνολα – Sets

Σύνολο είναι μία μη ταξινομημένη συλλογή αντικειμένων που περιέχει μηδέν ή περισσότερα μοναδικά αντικείμενα και χρησιμοποιείται σε περιπτώσεις όπου τα κύρια πράγματα που μας ενδιαφέρουν για ένα στοιχείο είναι αν αποτελεί μέλος ενός συνόλου και αν είναι μοναδικό μέσα στο σύνολο. Τα στοιχεία σε ένα σύνολο πρέπει να είναι αμετάβλητα (immutable) και να μπορούν να κατακερματιστούν (hashable). Άρα στοιχεία τύπου δεδομένων int, float, string και tuple, μπορούν να είναι μέλη ενός συνόλου αλλά στοιχεία τύπου list, set και dictionary δε μπορούν.

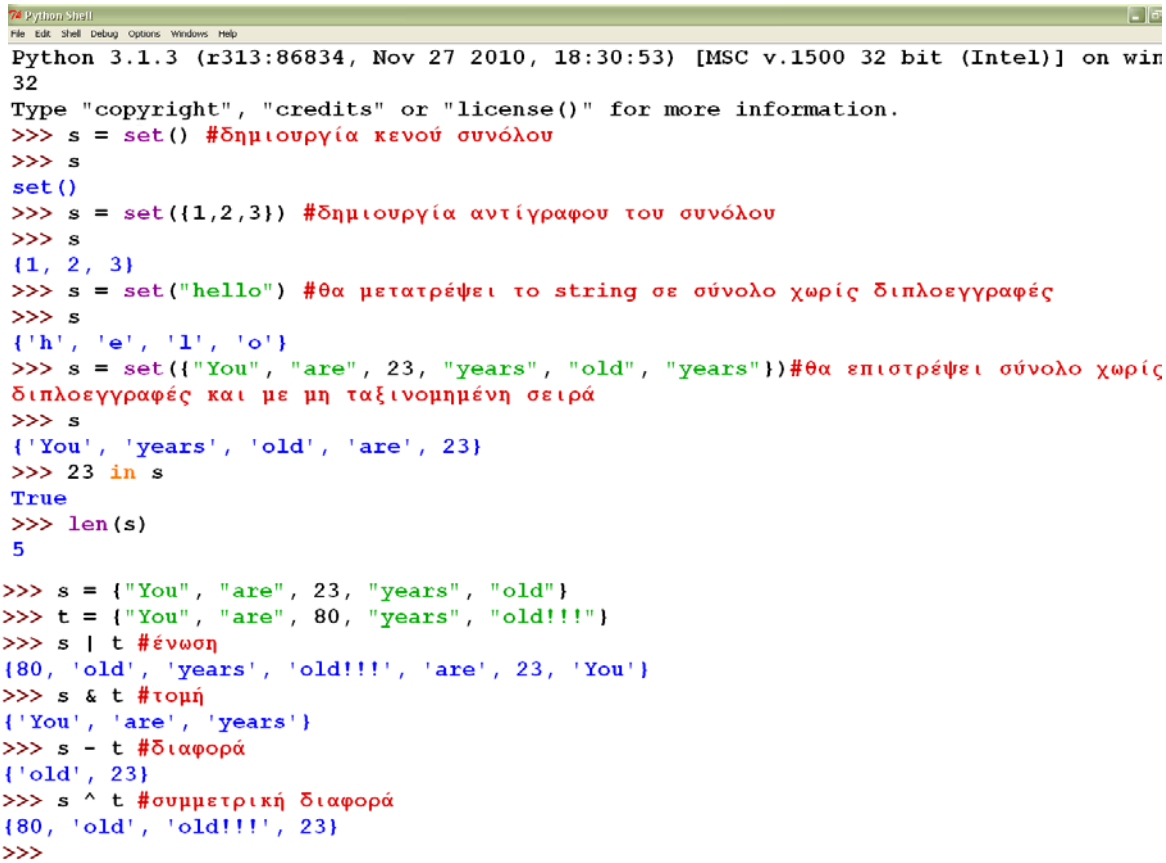
Τα σύνολα αν και περιέχουν αμετάβλητα στοιχεία είναι μεταβλητά, οπότε μπορούμε να εισάγουμε ή να διαγράψουμε οποιοδήποτε από τα στοιχεία τους.

Επειδή όμως δεν είναι ταξινομημένα δεν μπορούμε να χρησιμοποιήσουμε την έννοια του δείκτη θέσης, οπότε δεν μπορούμε να τα «τεμαχίσουμε».
Για τη δημιουργία ενός συνόλου μπορούμε να καλέσουμε τη συνάρτηση set().Όταν καλείται χωρίς παραμέτρους δημιουργεί και επιστρέφει ένα κενό σύνολο. Όταν καλείται με μία παράμετρο ίδιου τύπου επιστρέφει ένα ακριβές αντίγραφο του συνόλου, ενώ αν η παράμετρος είναι διαφορετικού τύπου προσπαθεί να τη μετατρέψει σε σύνολο. Η συνάρτηση set() δεν δέχεται παραπάνω από μία παράμετρο. Μπορούμε ακόμη να δημιουργήσουμε ένα μη κενό σύνολο με χρήση άγκιστρων {} τα οποία περιέχουν μία ακολουθία στοιχείων. Δεν μπορούμε να δημιουργήσουμε ένα κενό σύνολο με χρήση άδειων άγκιστρων γιατί χρησιμοποιούνται για δημιουργία λεξικών, που θα δούμε παρακάτω.
Τα σύνολα υποστηρίζουν την ενσωματωμένη συνάρτηση len() και τους τελεστές in και not in. Επιπλέον υποστηρίζουν τους τελεστές συνόλων ένωση (|), τομή (&), διαφορά (-) και συμμετρική διαφορά (^).

s.difference(t) / s-t	Επιστρέφει ένα νέο σύνολο με όλα τα στοιχεία του συνόλου s που δε βρίσκονται στο σύνολο t
s.difference_update(t) / s -= t	Διαγράφει τα στοιχεία του συνόλου s που είναι ίδια με τα στοιχεία του συνόλου t
s.intersection(t) / s&t	Επιστρέφει ένα νέο σύνολο με τα κοινά στοιχεία του συνόλου s και του συνόλου t
s.intersection_update(t) / s&= t	Κάνει το σύνολο s να περιέχει την τομή του ίδιου και του συνόλου t
s.symmetric_difference(t) / s ^ t	Επιστρέφει ένα νέο σύνολο με όλα τα στοιχεία του συνόλου s και του συνόλου t εκτός των στοιχείων που είναι κοινά και στα δύο σύνολα
s.symmetric_difference_update(t) / s ^= t	Κάνει το σύνολο s να περιέχει τη συμμετρική διαφορά του ίδιου και του συνόλου t
s.union(t) / s|t	Επιστρέφει ένα νέο σύνολο με όλα τα στοιχεία του συνόλου s και όλα τα στοιχεία του συνόλου t που δεν βρίσκονται στο

	σύνολο s
s.update(t) / s |= t	Προσθέτει κάθε στοιχείο του συνόλου t που δεν υπάρχει στο σύνολο s, στο σύνολο s

Πίνακας 4.2 Τελεστές συνόλων

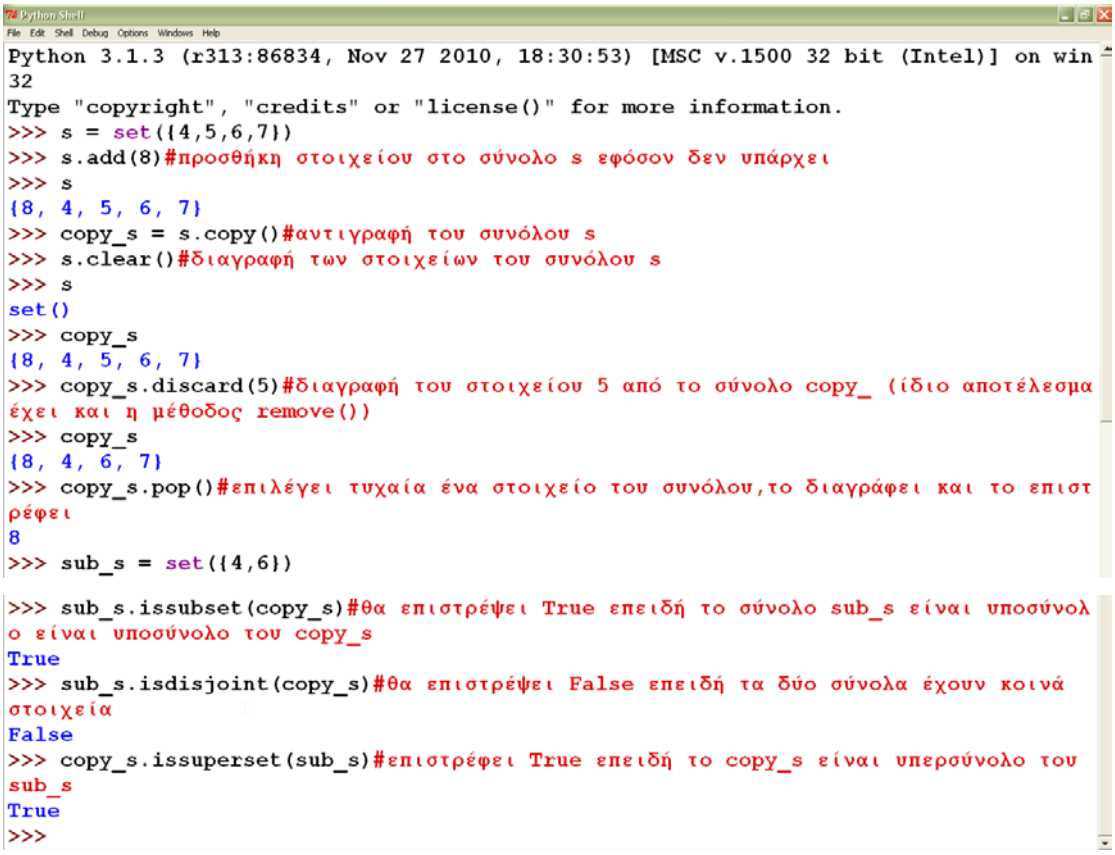


Εικόνα 4.11

s.add(x)	Προσθέτει το στοιχείο x στο σύνολο s αν δεν υπάρχει ήδη
s.clear()	Διαγράφει όλα τα στοιχεία του συνόλου s
s.copy()	Επιστρέφει ένα αντίγραφο του συνόλου s
s.discard(x)	Διαγράφει το στοιχείο x από το σύνολο s, αν υπάρχει
s.isdisjoint(t)	Επιστρέφει True αν τα σύνολα s και t δεν έχουν κοινά στοιχεία
s.issubset(t) / s<=t	Επιστρέφει True αν το σύνολο s είναι ίσο με το σύνολο t ή υποσύνολο του t
s.issuperset(t) / s>=t	Επιστρέφει True αν το σύνολο s είναι ίσο με το σύνολο t ή υπερσύνολο του t
s.pop()	Επιστρέφει και διαγράφει ένα τυχαίο στοιχείο του συνόλου s ή προκαλεί εξαίρεση KeyError αν το σύνολο s είναι κενό

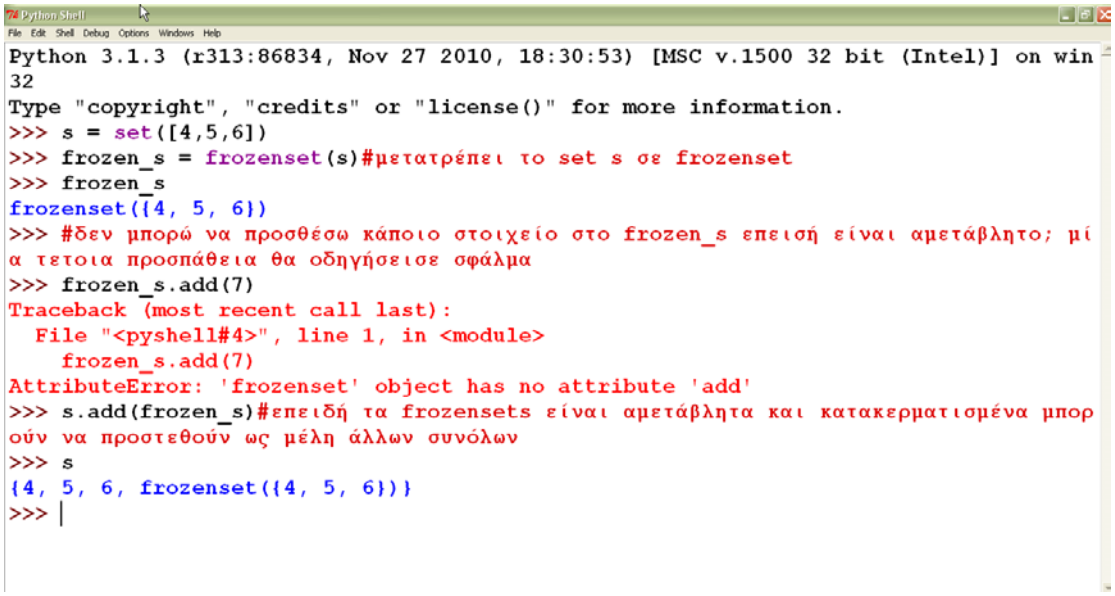
s.remove(x)	Διαγράφει το στοιχείο x από το σύνολο s ή προκαλεί εξαίρεση KeyError αν το στοιχείο x δεν είναι μέσα στο s

Πίνακας 4.3 Μέθοδοι συνόλων



Εικόνα 4.12

Επειδή τα σύνολα είναι αμετάβλητα και κατακερματισμένα (hashable) δεν μπορούν να ανήκουν σε άλλα σύνολα. Για το λόγο αυτό υπάρχει ένας επιπλέον τύπος συνόλου, ο frozenset, που δεν μπορεί να αλλάξει μετά την δημιουργία του. Επομένως τα frozensets είναι αμετάβλητα και κατακερματισμένα και άρα μπορούν να είναι μέλη άλλων συνόλων:



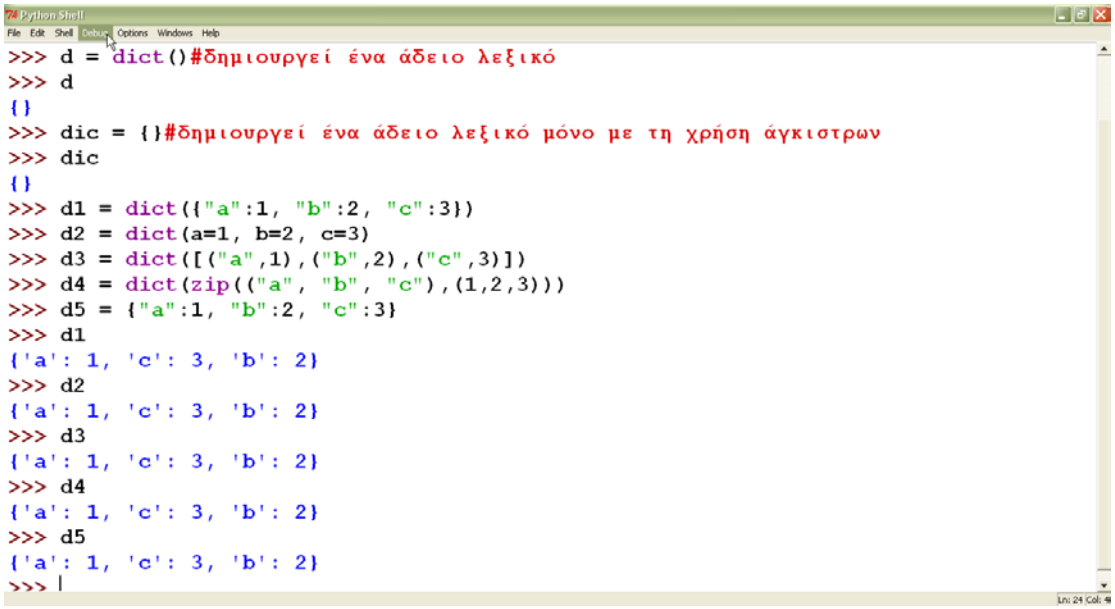
Εικόνα 4.13

4.4 Λεξικά – Dictionaries

Για να μπορέσουμε να καταλάβουμε την έννοια των λεξικών, πρέπει να σκεφτούμε τις λίστες. Ο λίστες μπορούν να αποθηκεύσουν αντικείμενα οποιουδήποτε τύπου, οι τιμές τους προσπελαύνονται μέσω ακεραίων που ονομάζονται δείκτες και είναι αυστηρά ταξινομημένες με βάση τη θέση τους στη λίστα, επειδή οι δείκτες που προσπελαύνουν αυτές τις τιμές είναι διαδοχικοί ακέραιοι.

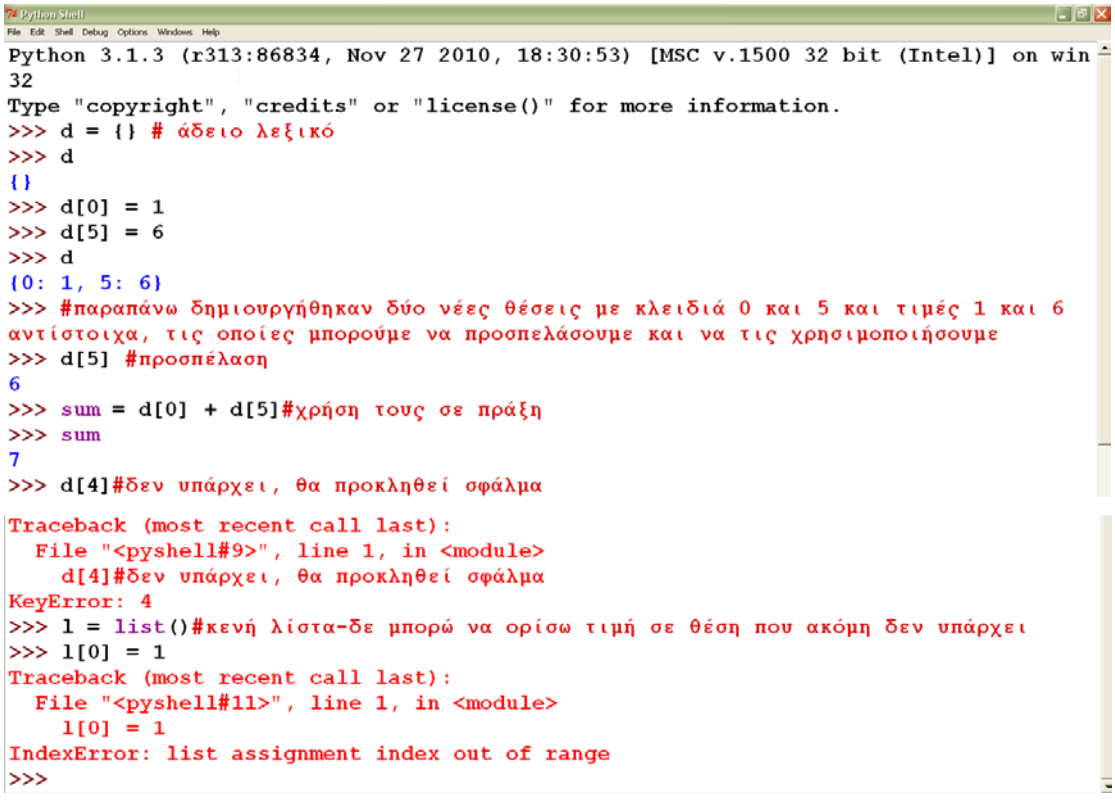
Τα λεξικά μπορούν να αποθηκεύσουν αντικείμενα οποιουδήποτε τύπου, οι τιμές τους προσπελαύνονται μέσω ακεραίων, συμβολοσειρών ή άλλων Python αντικειμένων που ονομάζονται κλειδιά (keys). Με άλλα λόγια και οι λίστες και τα λεξικά παρέχουν δεικτοδοτημένη πρόσβαση σε αυθαίρετες τιμές, αλλά το σύνολο των αντικειμένων που μπορούν να χρησιμοποιηθούν σαν δείκτες λεξικών είναι πολύ μεγαλύτερο. Τέλος οι τιμές που είναι αποθηκευμένες σε ένα λεξικό δεν είναι αυστηρά ταξινομημένες, επειδή τα κλειδιά των λεξικών δεν είναι μόνο αριθμοί.

Για τη δημιουργία ενός λεξικού μπορούμε να καλέσουμε τη συνάρτηση dict().Όταν καλείται χωρίς παραμέτρους δημιουργεί και επιστρέφει ένα άδειο λεξικό. Μπορούμε ακόμη να δημιουργήσουμε ένα άδειο λεξικό με χρήση άγκιστρων {}. Αν θέλουμε να δημιουργήσουμε ένα λεξικό με περιεχόμενο θα πρέπει να βάλουμε ως παραμέτρους στη συνάρτηση ζεύγη τιμών χωρισμένα με κόμμα. Η πρώτη τιμή αποτελεί το κλειδί – key κ α η δεύτερη την τιμή του κ λιδ ού. Ακόμη θα μπορούσαμε να βάλουμε ως παράμετρο μία λίστα με ζεύγη τιμών. Παραδείγματα δημιουργίας λεξικού με διάφορους τρόπους που παράγουν το ίδιο αποτέλεσμα:



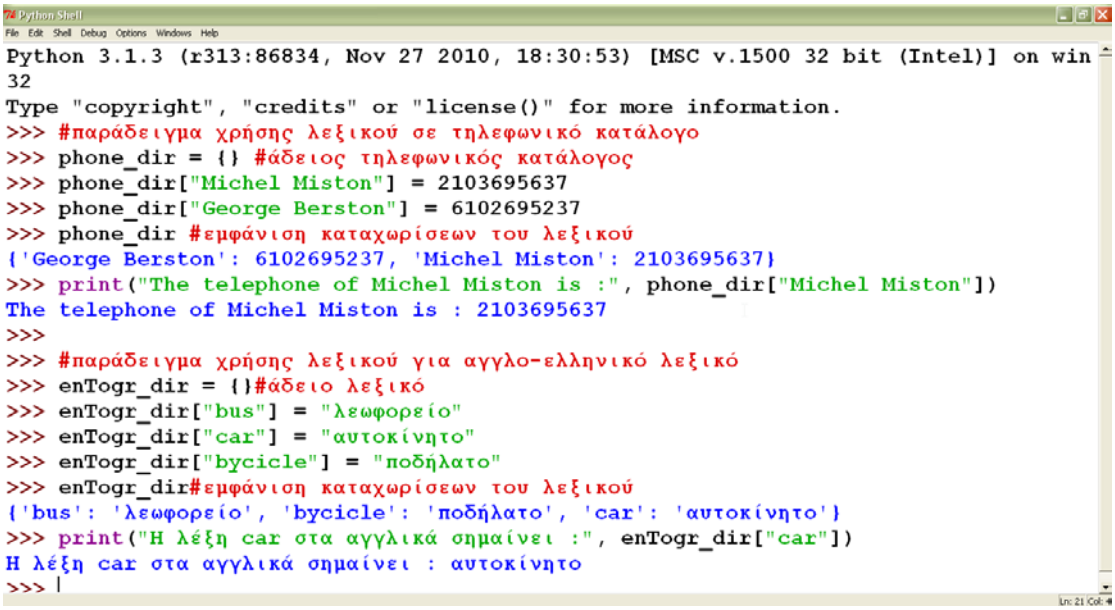
Εικόνα 4.14

Τα λεξικά μας δίνουν τη δυνατότητα να αναθέσουμε τιμές σε θέσεις που δεν υπάρχουν. Νέες θέσεις δημιουργούνται όποτε είναι απαραίτητο. Κάτι τέτοιο δε συμβαίνει με τις λίστες, οι οποίες σε τέτοιες περιπτώσεις προκαλούν σφάλμα επειδή στη Python είναι μη εφικτό να προσθέσουμε μία τιμή στη θέση μίας λίστας που ακόμη δεν υπάρχει. Αν υπάρχουν τιμές σε ένα λεξικό, μπορούμε να τις προσπελάσουμε και να τις χρησιμοποιήσουμε:



Εικόνα 4.15

Το πιο δυνατό χαρακτηριστικό των λεξικών είναι ότι τα κλειδιά είναι λιγότερο περιορισμένα σε σχέση με τους δείκτες μίας λίστας – μπορεί να είναι αριθμοί, συμβολοσειρές ή αντικείμενα Python. Αυτό είναι χρήσιμο σε εφαρμογές που ανταποκρίνονται στον πραγματικό κόσμο όπως τηλεφωνικοί κατάλογοι και βιβλία μετάφρασης.

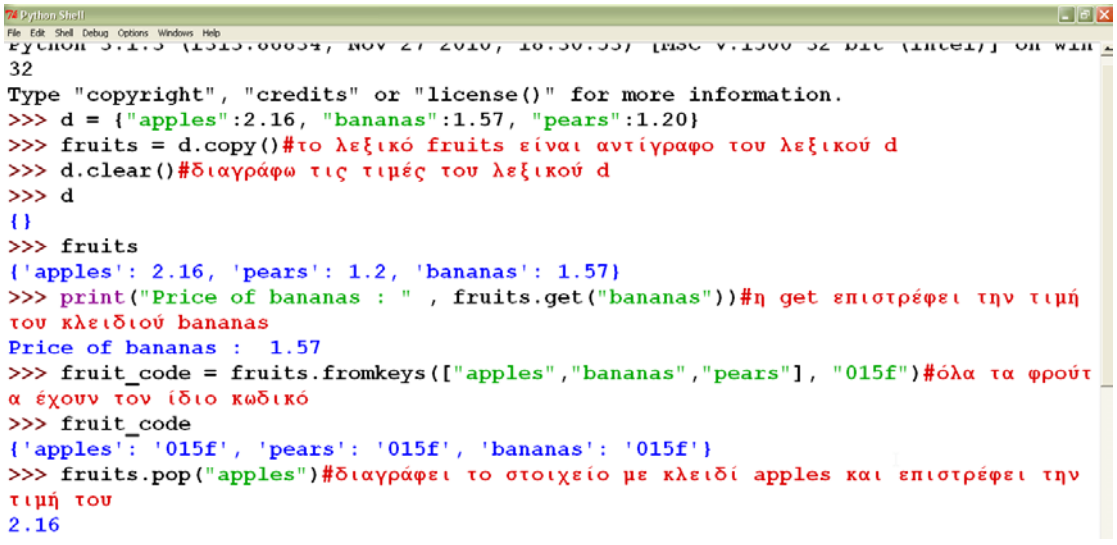


Εικόνα 4.16

d.clear()	Διαγράφει όλα τα στοιχεία του λεξικού d.
d.copy()	Επιστρέφει ένα ακριβές αντίγραφο του λεξικού d.
d.fromkeys(s, v)	Επιστρέφει ένα λεξικό του οποίου τα κλειδιά αντιστοιχίζονται στα στοιχεία της ακολουθίας s των οποίων οι τιμές είναι None ή v αν δίνεται.
d.get(k)	Επιστρέφει την τιμή του κλειδιού k ή None αν το κλειδί δεν βρίσκεται στο λεξικό d.
d.get(k, v)	Επιστρέφει την τιμή του κλειδιού k, ή την τιμή v αν το k δε βρίσκεται μέσα στο λεξικό d.
d.items()	Επιστρέφει ένα read-only αντικείμενο που κρατάει όλα τα ζεύγη (key, value) του λεξικού d.
d.keys()	Επιστρέφει ένα read-only αντικείμενο που κρατάει όλα τα κλειδιά - keys του λεξικού d.

d.pop(k)	Επιστρέφει την τιμή του κλειδιού k και διαγράφει το στοιχείο του οποίου το κλειδί είναι k ή προκαλεί την εξαίρεση KeyError αν το κλειδί k δε βρίσκεται στο λεξικό d.
d.pop(k, v)	Επιστρέφει την τιμή του κλειδιού k και διαγράφει το στοιχείο του οποίου το κλειδί είναι k ή επιστρέφει την τιμή v αν το κλειδί k δε βρίσκεται στο λεξικό d.
d.popitem()	Διαγράφει και επιστρέφει αυθαίρετα ένα ζευγάρι (key, value) από το λεξικό d ή προκαλεί την εξαίρεση KeyError αν το λεξικό d είναι άδειο.
d.setdefault(k, v)	Έχει το ίδιο αποτέλεσμα με τη μέθοδο d.get(), με τη διαφορά ότι αν το κλειδί k δε βρίσκεται μέσα στο λεξικό d, ένα νέο στοιχείο εισάγεται με κλειδί k και τιμή None ή v αν δίνεται.
d.update(a)	Προσθέτει στο λεξικό d κάθε ζευγάρι (key, value) από το a που δε βρίσκεται στο λεξικό d. Για κάθε κλειδί που βρίσκεται και στο d και στο a, αντικαθιστά την αντίστοιχη τιμή του d με αυτήν του a.
d.values()	Επιστρέφει ένα read-only αντικείμενο που κρατάει όλες τις τιμές του λεξικού d.
d.len()	Επιστρέφει το πλήθος των ζευγών του λεξικού d.
del(d[k])	Διαγράφει το ζεύγος με κλειδί k του λεξικού d.

Πίνακας 4.4 Dictionary methods



```
>>> fruits
{'pears': 1.2, 'bananas': 1.57}
>>> fruits.setdefault("oranges",1.19)#αν το στοιχείο με κλειδί oranges δεν υπάρχει
στο λεξικό fruits τότε δημιουργεί ένα νέο στοιχείο ('oranges':1.19)
1.19
>>> fruits
{'oranges': 1.19, 'pears': 1.2, 'bananas': 1.57}
>>> fruits.popitem()#διαγρέφει αυθαίρετα ένα στοιχείο και το επιστρέφει
('oranges', 1.19)
>>> fruits
{'pears': 1.2, 'bananas': 1.57}
>>> newFruitsandPrices = {"carrots":0.40,"lemons":0.91,"bananas":1.60}
>>> fruits.update(newFruitsandPrices)#ενημερώνω το λεξικό fruits με νέα στοιχεία το
υ λεξικού newFruitsandPrices και νέες τιμές
>>> fruits
{'bananas': 1.6, 'lemons': 0.91, 'carrots': 0.4, 'pears': 1.2}
>>>
```

Εικόνα 4.17

Συγκεκριμένα για τις μεθόδους items(), keys() και values(), όλες επιστρέφουν μία όψη του λεξικού - dictionary view, το οποίο είναι ένα read-only iterable αντικείμενο που κρατάει τα στοιχεία, τα κλειδιά ή τις τιμές του λεξικού αντίστοιχα.
Μπορούμε να χειριστούμε τα dictionary views σαν iterables αλλά με δύο σημαντικές διαφορές. Αν γίνουν αλλαγές στο λεξικό, το view αλλάζει ώστε να αντιπροσωπεύει τις αλλαγές του λεξικού ενώ τα key και item views υποστηρίζουν κάποιες λειτουργίες συνόλων – τομή(&), ένωση(|), διαφορά (-) και συμμετρική διαφορά (^).

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> fruits = {"apples":2.16, "bananas":1.57, "pears":1.20}
>>> i = fruits.items()#στοιχεία του λεξικού fruits
>>> i
dict_items([('apples', 2.16), ('pears', 1.2), ('bananas', 1.57)])
>>> k = fruits.keys()#κλειδιά του λεξικού fruits
>>> k
dict_keys(['apples', 'pears', 'bananas'])
>>> v = fruits.values()#τιμές του λεξικού fruits
>>> v
dict_values([2.16, 1.2, 1.57])
```

Εικόνα 4.18

Συχνά τα λεξικά χρησιμοποιούνται για να μετράνε την εμφάνιση μοναδικών αντικειμένων. Το παρακάτω παράδειγμα χρησιμοποιεί ένα λεξικό για να μετρήσει πόσες φορές εμφανίζεται μία λέξη σε ένα αρχείο. Παραθέτει σε λίστα κάθε λέξη και αριθμό - που αντιπροσωπεύει πόσες φορές εμφανίστηκε - σε αλφαβητική σειρά για όλα τα αρχεία που δίνονται στη command line.

5. ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ ΡΟΗΣ

Η Python παρέχει ένα ολοκληρωμένο σύνολο προτάσεων ελέγχου ροής των προγραμμάτων που μπορούμε να τις χωρίσουμε σε εντολές λήψεις αποφάσεων και βρόγχους επανάληψης.

5.1 Εντολή if-else-if

Η σύνταξη της εντολής if για τη λήψη αποφάσεων είναι η παρακάτω:

```
if συνθήκη 1:
  body1
elif συνθήκη 2:
  body2
elif συνθήκη3:
  body3
……
elif συνθήκη (n-1):
  body(n-1)
else:
  body(n)
```

Η πρόταση if ελέγχει μία έκφραση γνωστή ως συνθήκη. Ανάλογα με το ποια συνθήκη αποτιμάται σε True εκτελείται και το αντίστοιχο κομμάτι κώδικα body. Αν καμία συνθήκη δεν αποτιμάται σε True τότε εκτελείται ο κώδικας στο else. Τα κομμάτια κώδικα body αποτελούνται από μία ή περισσότερες εντολές Python που είναι χωρισμένες με νέες γραμμές και είναι στο ίδιο επίπεδο οδόντωσης. Ο Python επεξεργαστής χρησιμοποιεί αυτό το επίπεδο για να τις οριοθετήσει. Κανένα άλλο σύμβολο οριοθέτησης δεν είναι απαραίτητο, όπως αγκύλες ή παρενθέσεις. Μπορούμε να παραλείψουμε τα μέρη elif ή else ή και τα δύο. Αν μία συνθήκη δεν βρίσκει κομμάτι κώδικα να εκτελέσει (καμία συνθήκη δεν είναι True και δεν υπάρχει μέρος else) δεν κάνει τίποτα.

Το κομμάτι body μετά την εντολή if απαιτείται. Αν δε θέλουμε να εκτελείται καμία πράξη όταν ισχύει μία συνθήκη μπορούμε να χρησιμοποιήσουμε την εντολή pass που εξασφαλίζει κράτηση θέσης για το κομμάτι body.

```
if x < 5:
    pass
else:
    x = 5
```

Μία ακόμη εντολή για τη λήψη αποφάσεων που υπάρχει σε άλλες γλώσσες προγραμματισμού είναι η case/switch. Στη Python **δεν** υπάρχει εντολή case/switch..

5.2 Βρόγχος while

Η σύνταξη του βρόγχου επανάληψης while είναι η παρακάτω:

```
while συνθήκη:
    body
else:
    post-code
```

Ο βρόγχος while εκτελεί το κομμάτι κώδικα body εφόσον η συνθήκη αποτιμάται σε True. Αν η συνθήκη αποτιμάται σε False, εκτελείται μόνο το κομμάτι κώδικα post-code. Τα κομμάτια κώδικα body αποτελούνται από μία ή περισσότερες εντολές Python που είναι χωρισμένες με νέες γραμμές και είναι στο ίδιο επίπεδο οδόντωσης. Ο Python επεξεργαστής χρησιμοποιεί αυτό το επίπεδο για να τις οριοθετήσει. Κανένα άλλο σύμβολο οριοθέτησης δεν είναι απαραίτητο, όπως αγκύλες ή παρενθέσεις. Μπορούμε να παραλείψουμε το μέρος else του βρόγχου while.

Δύο ειδικές εντολές που μπορούν να χρησιμοποιηθούν στο κομμάτι body του βρόγχου while, είναι οι break και continue. Η εκτέλεση της εντολής break τερματίζει το βρόγχο while και ακόμα και το κομμάτι post-code δεν εκτελείται. Η εκτέλεση της εντολής continue προκαλεί την προσπέραση του body, η συνθήκη αποτιμάται ξανά και ο βρόγχος μπορεί να προχωρήσει κανονικά.

5.3 Βρόγχος for

Ο βρόγχος for στη Python διαφέρει από τους βρόγχους που συναντάμε σε άλλες γλώσσες προγραμματισμού. Χρησιμοποιείται με αντικείμενα που μπορούν να

κρατήσουν μία ακολουθία τιμών και εκτελείτε τόσες φορές, όσες είναι και οι τιμές που περιέχονται στο αντικείμενο. Δηλαδή ένας βρόγχος for μπορεί να εκτελείται για κάθε στοιχείο σε μία λίστα, πλειάδα ή ακολουθία χαρακτήρων. Ακόμη το αντικείμενο μπορεί να είναι μία ειδική συνάρτηση όπως η range και η generator που αναλύονται παρακάτω. Η σύνταξη του βρόγχου επανάληψης for είναι η παρακάτω:

```
for item in sequence:
   body
else:
   post-code
```

Το κομμάτι κώδικα body θα εκτελεστεί για κάθε ένα στοιχείο της ακολουθίας sequence. Η μεταβλητή item παίρνει σαν τιμή, με τη σειρά, την τιμή κάθε ενός από τα στοιχεία που απαρτίζουν την ακολουθία sequence. Το κομμάτι else είναι προαιρετικό και χρησιμοποιείται σπάνια. Οι εντολές break και continue μπορούν επίσης να χρησιμοποιηθούν στο σώμα ενός βρόγχου for. Αν η εντολή break εκτελεστεί τερματίζει απευθείας το βρόγχο for. Αν η εντολή continue εκτελεστεί προκαλεί διακοπή του body και βρόγχος προχωράει κανονικά με το επόμενο αντικείμενο.

5.3.1 Η συνάρτηση range

Η συνάρτηση range δημιουργεί και επιστρέφει ακολουθίες αριθμών και έχει τρεις μορφές σύνταξης:

a. *range(number):* δημιουργεί μία ακολουθία αριθμών από το 0 μέχρι το number-1.

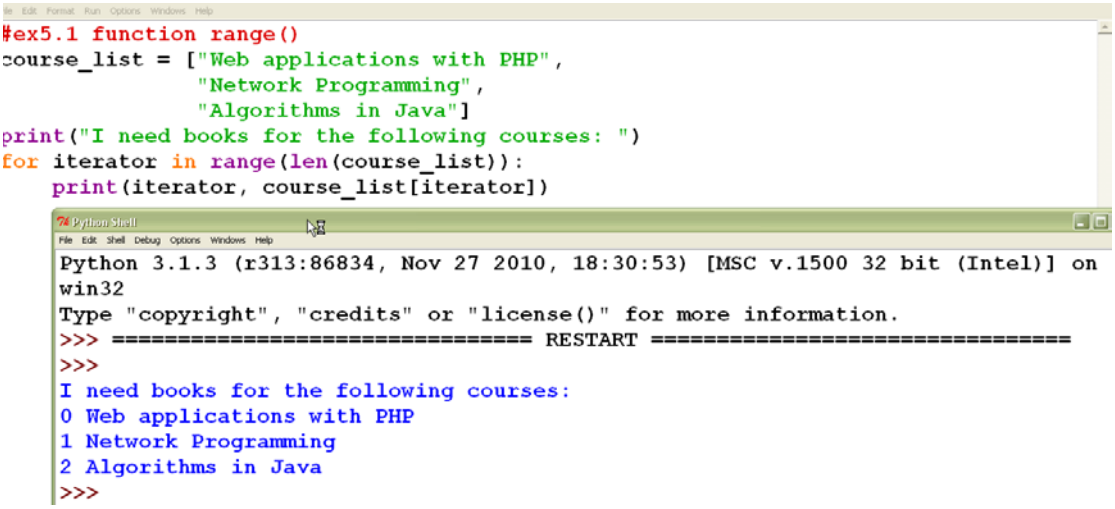
b.*range(from, to):* δημιουργεί μία ακολουθία αριθμών από την τιμή της μεταβλητής from μέχρι την τιμή της μεταβλητής to -1.

c. *range(from, to, step):* δημιουργεί μία ακολουθία αριθμών από την τιμή της μεταβλητής from μέχρι την τιμή της μεταβλητής to -1 με βήμα step.

Η συνάρτηση range μπορεί να χρησιμοποιηθεί με το βρόγχο for όταν θέλουμε η μεταβλητή item να πάρει με τη σειρά ακέραιους δείκτες που αντιστοιχούν στις θέσεις των στοιχείων ενός ακολουθιακού τύπου δεδομένων πχ λίστα και μαζί με τη συνάρτηση len() μας δίνει τη δυνατότητα να απαριθμούμε τα αποτελέσματα που

τυπώνονται σε έναν βρόγχο επανάληψης for. Παρακάτω υπάρχει ένα παράδειγμα χρήσης της μεθόδου range στο βρόγχο for.

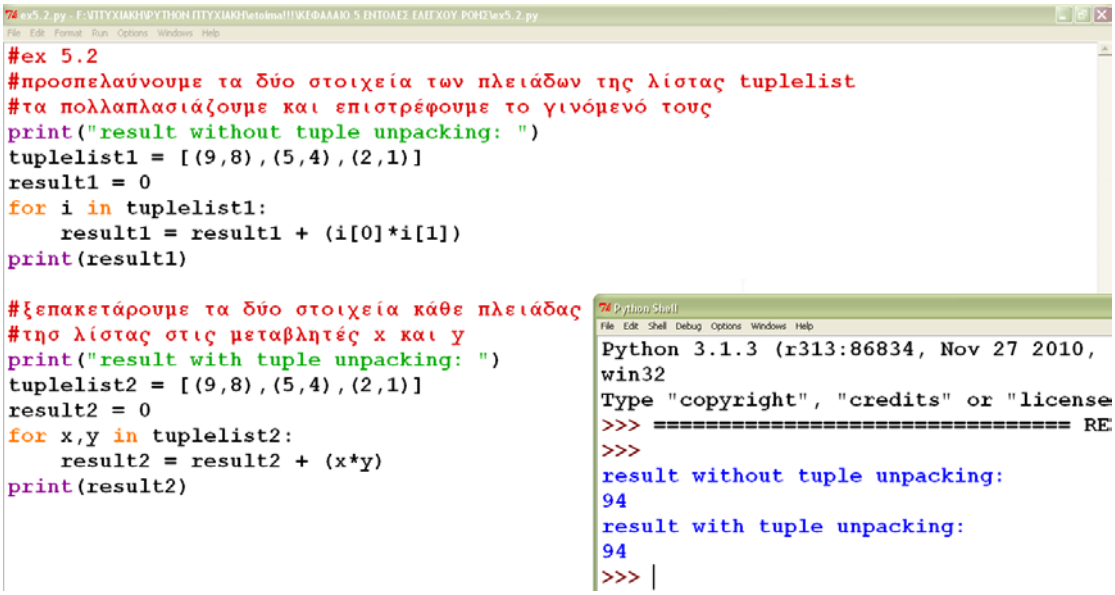
Η μεταβλητή iterator θα πάρει την τιμή των στοιχείων που βρίσκονται μέσα στην λίστα course_list. Αντί να πούμε <for iterator in course_list>, περικλείουμε τη λίστα course_list στη συνάρτηση len() που θα μας επιστρέψει τον αριθμό τον στοιχείων της λίστας course_list. Η len() με τη σειρά της περικλείεται στη συνάρτηση range() η οποία θα μας επιστρέψει μια ακολουθία αριθμών με τόσα στοιχεία όσα και αυτά της λίστας course_list. Στη συνέχεια η εντολή print τυπώνει τον αριθμό της θέσης του στοιχείου στη λίστα και το στοιχείο που βρίσκεται στην εκάστοτε θέση της λίστας.



Παράδειγμα 5.1

5.3.2 Tuple unpacking

Ο όρος tuple unpacking έχει να κάνει με τη χρήση πλειάδων και το ξε-πακετάρισμα των τιμών τους σε μεταβλητές στο βρόγχο for, παρέχοντας έτσι πιο καθαρό κώδικα.. Τα ακόλουθα κομμάτια κώδικα παράγουν ακριβώς το ίδιο αποτέλεσμα.



```
#ex 5.2
#προσπελαύνουμε τα δύο στοιχεία των πλειάδων της λίστας tuplelist
#τα πολλαπλασιάζουμε και επιστρέφουμε το γινόμενό τους
print("result without tuple unpacking: ")
tuplelist1 = [(9,8),(5,4),(2,1)]
result1 = 0
for i in tuplelist1:
    result1 = result1 + (i[0]*i[1])
print(result1)

#ξεπακετάρουμε τα δύο στοιχεία κάθε πλειάδας
#της λίστας στις μεταβλητές x και y
print("result with tuple unpacking: ")
tuplelist2 = [(9,8),(5,4),(2,1)]
result2 = 0
for x,y in tuplelist2:
    result2 = result2 + (x*y)
print(result2)
```

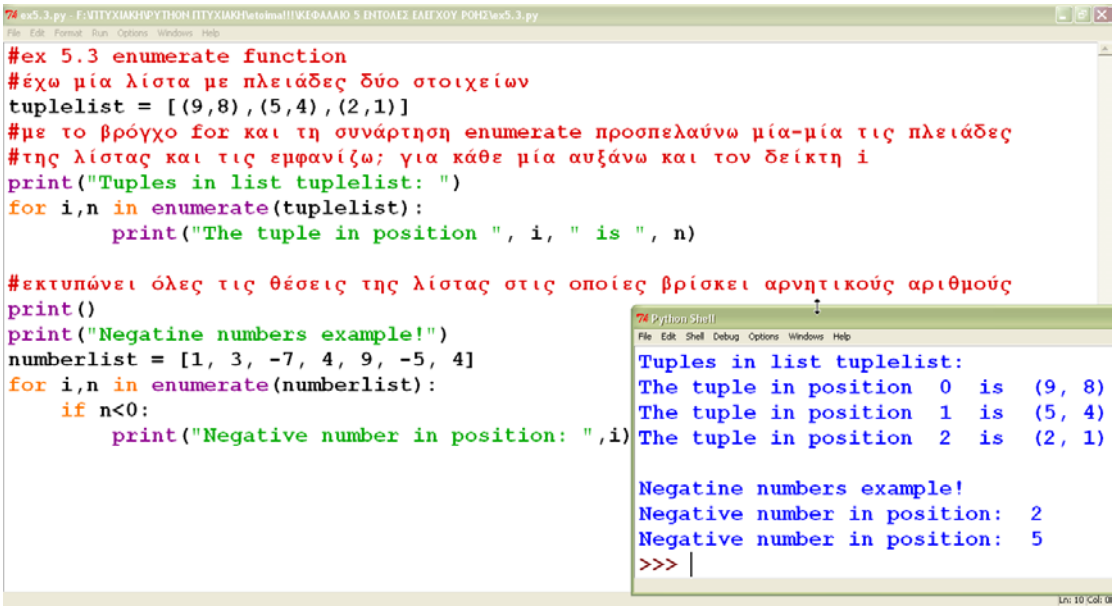
```
Python 3.1.3 (r313:86834, Nov 27 2010,
win32
Type "copyright", "credits" or "license
>>> ========== RE
>>>
result without tuple unpacking:
94
result with tuple unpacking:
94
>>>
```

Παράδειγμα 5.2

Στη δεύτερη περίπτωση χρησιμοποιήσαμε μία πλειάδα x,y αμέσως μετά τη λέξη κλειδί for, αντί για μία μονή μεταβλητή. Σε κάθε επανάληψη του βρόγχου for, το x περιλαμβάνει το στοιχείο 0 της τρέχουσας πλειάδας της λίστας και το y το στοιχείο 1 της τρέχουσας πλειάδας της λίστας. Αυτό σημαίνει ότι κάθε στοιχείο της λίστας περιμένουμε να είναι μία πλειάδα συγκεκριμένου μεγέθους.

5.3.2.1 Η συνάρτηση enumerate()

Η συνάρτηση enumerate επιστρέφει ζεύγη της μορφής (index, item). Μπορεί να συνδυαστεί με το tuple unpacking για να προσπελάσουμε και να εμφανίσουμε τα στοιχεία και τον δείκτη κάθε στοιχείου ενός iterable αντικειμένου πχ μίας λίστας ή πλειάδας. Η χρήση της συνάρτησης enumerate είναι παρόμοια με αυτή της range αλλά έχει το πλεονέκτημα ότι ο κώδικας είναι πιο «καθαρός» και πιο κατανοητός. Παρακάτω υπάρχει ένα παράδειγμα για να κατανοήσουμε πως ακριβώς λειτουργεί η συνάρτηση enumerate:



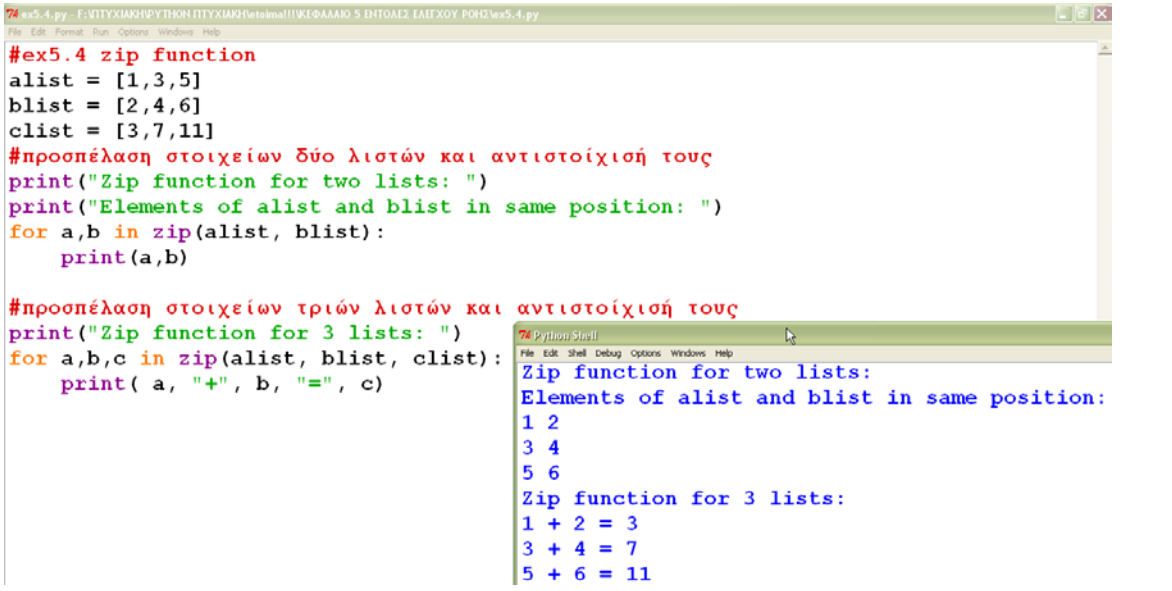
```
#ex 5.3 enumerate function
#έχω μία λίστα με πλειάδες δύο στοιχείων
tuplelist = [(9,8),(5,4),(2,1)]
#με το βρόγχο for και τη συνάρτηση enumerate προσπελαύνω μία-μία τις πλειάδες
#της λίστας και τις εμφανίζω; για κάθε μία αυξάνω και τον δείκτη i
print("Tuples in list tuplelist: ")
for i,n in enumerate(tuplelist):
        print("The tuple in position ", i, " is ", n)

#εκτυπώνει όλες τις θέσεις της λίστας στις οποίες βρίσκει αρνητικούς αριθμούς
print()
print("Negatine numbers example!")
numberlist = [1, 3, -7, 4, 9, -5, 4]
for i,n in enumerate(numberlist):
    if n<0:
        print("Negative number in position: ",i)
```

```
Tuples in list tuplelist:
The tuple in position  0  is  (9, 8)
The tuple in position  1  is  (5, 4)
The tuple in position  2  is  (2, 1)

Negatine numbers example!
Negative number in position:  2
Negative number in position:  5
>>>
```

Παράδειγμα 5.3

5.3.2.2 Η συνάρτηση zip()

Η συνάρτηση zip προσπελαύνει ταυτόχρονα δύο ή περισσότερα iterable αντικείμενα πχ λίστες και συνδυάζει τα αντίστοιχα στοιχεία τους με βάση τη θέση τους. Στο παρακάτω παράδειγμα χρησιμοποιούμε τη συνάρτηση zip σε έναν βρόγχο for για να προσπελάσουμε, να συνδυάσουμε και να ξεπακετάρουμε τα στοιχεία των λιστών στις μεταβλητές.

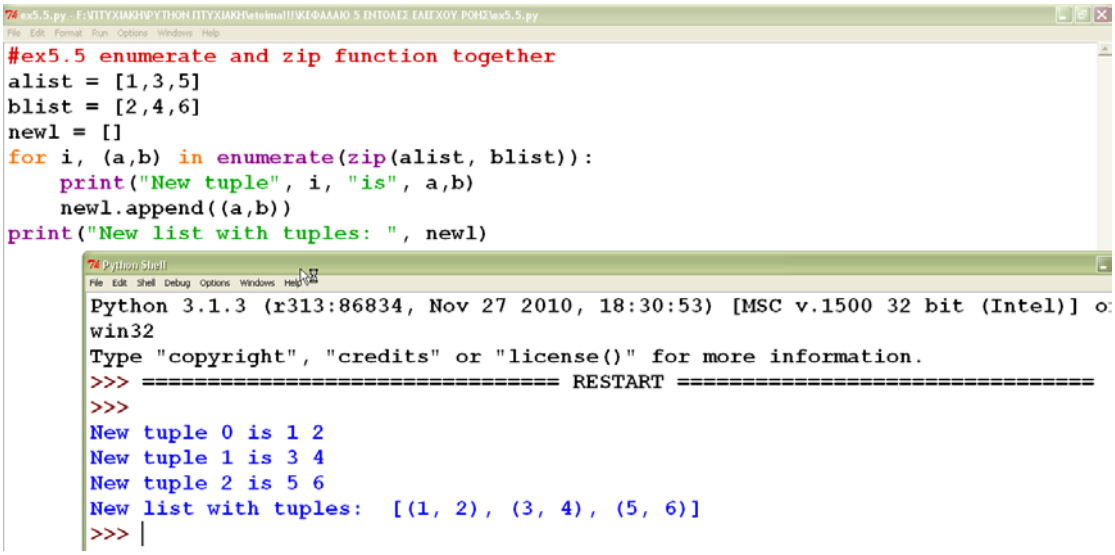


```
#ex5.4 zip function
alist = [1,3,5]
blist = [2,4,6]
clist = [3,7,11]
#προσπέλαση στοιχείων δύο λιστών και αντιστοίχισή τους
print("Zip function for two lists: ")
print("Elements of alist and blist in same position: ")
for a,b in zip(alist, blist):
    print(a,b)

#προσπέλαση στοιχείων τριών λιστών και αντιστοίχισή τους
print("Zip function for 3 lists: ")
for a,b,c in zip(alist, blist, clist):
    print( a, "+", b, "=", c)
```

```
Zip function for two lists:
Elements of alist and blist in same position:
1 2
3 4
5 6
Zip function for 3 lists:
1 + 2 = 3
3 + 4 = 7
5 + 6 = 11
```

Παράδειγμα 5.4

Τέλος ένα παράδειγμα που χρησιμοποιεί τις συναρτήσεις enumerate και zip μαζί :



Παράδειγμα 5.5

5.3 Εντολές, μπλοκ και οδόντωση - Statements, blocks, indentation

Οι εντολές στη Python μπορεί να είναι εντολές ανάθεσης, κλήσεις συναρτήσεων, η συνάρτηση print, η εντολή δέσμευσης θέσης pass και η εντολή del.

Ένα Python μπλοκ αποτελείται από μία ή περισσότερες εντολές που χωρίζονται με νέες γραμμές. Πολλές εντολές μπορούν να τοποθετηθούν στην ίδια γραμμή αν χωρίζονται με semicolon(;). Εντολές που βρίσκονται στο ίδιο μπλοκ έχουν το ίδιο επίπεδο οδόντωσης.

Η οδόντωση έχει να κάνει με την εσοχή στην οποία πρέπει να τοποθετούνται οι εντολές για να δομείται ο κώδικας, αφού η Python δε χρησιμοποιεί άγκιστρα ή παρενθέσεις. Αν η εσοχή έχει μπει σε ένα σημείο που δεν είναι απαραίτητο ή έχουμε βάλει τις εντολές ενός μπλοκ σε διαφορετικά επίπεδα οδόντωσης – εσοχές τότε προκαλείται εξαίρεση. Για κάθε επίπεδο οδόντωσης η προτεινόμενη χρήση εσοχών είναι 4κενά. Βέβαια ο διερμηνευτής της Python βάζει αυτόματα τις εσοχές και το μόνο που έχουμε να κάνουμε όταν γράφουμε προγράμματα σε Python είναι να πληκτρολογούμε τις εντολές που θέλουμε.

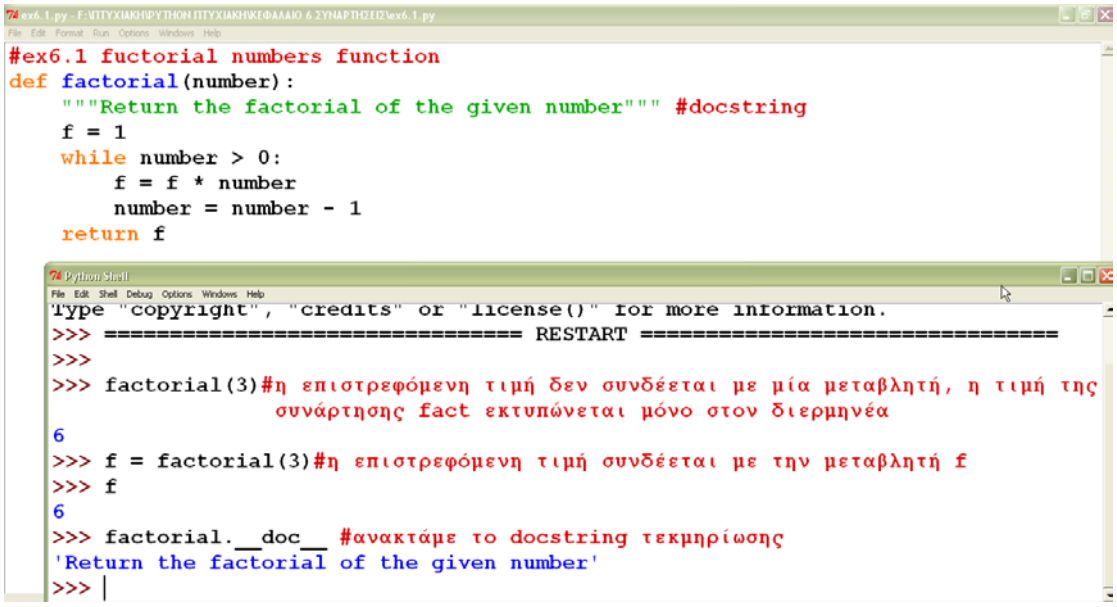
6. ΣΥΝΑΡΤΗΣΕΙΣ

6.1 Ορισμός συναρτήσεων

Οι συναρτήσεις είναι ένας τρόπος για να παρέχουμε λειτουργικότητα στις εφαρμογές μας. Στη Python υπάρχουν τέσσερα είδη συναρτήσεων: οι καθολικές (global) συναρτήσεις, οι τοπικές (local) συναρτήσεις, οι συναρτήσεις lambda και οι μέθοδοι. Η Python παρέχει πολλές ενσωματωμένες συναρτήσεις, ενώ η στάνταρ βιβλιοθήκη και οι εξωτερικές βιβλιοθήκες ακόμα περισσότερες. Η βασική σύνταξη για μία Python συνάρτηση είναι :

```
def όνομα_συνάρτησης(παράμετρος1, παράμετρος2, …) :
    σώμα
```

Ακολουθεί παράδειγμα της συνάρτησης factorial που υπολογίζει το παραγοντικό ενός αριθμού. Στη συνάρτηση αυτή η δεύτερη γραμμή είναι ένα docstring. Τα docstrings χρησιμοποιούνται για να περιγράψουν την εξωτερική συμπεριφορά μιας συνάρτησης και τις παραμέτρους που δέχεται, σε αντίθεση με τα σχόλια που τεκμηριώνουν την εσωτερική πληροφορία για το πώς δουλεύει ο κώδικας. Είναι προαιρετικές συμβολοσειρές που δηλώνονται αμέσως μετά τη δήλωση της συνάρτησης και εισάγονται συνήθως σε τριπλά εισαγωγικά για να επιτρέπεται η συγγραφή πολλών γραμμών τεκμηρίωσης. Μπορούμε να αποκτήσουμε την τιμή τους με την εντολή *όνομασυνάρησης.__doc __.*



Παράδειγμα 6.1

Στην Python όλες οι συναρτήσεις επιστρέφουν μία τιμή. Αν δεν υπάρχει πρόταση return να εκτελεστεί, τότε επιστρέφεται η ειδική τιμή Python, None.

6.2 Παράμετροι συναρτήσεων

Οι παράμετροι στις συναρτήσεις είναι προαιρετικές. Οι περισσότερες όμως συναρτήσεις χρειάζονται παραμέτρους. Η Python είναι ευέλικτη και παρέχει τρεις τρόπους για να ορίσουμε τις παραμέτρους μίας συνάρτησης.

6.2.1 Παράμετροι θέσης – Positional Parameters

Ο απλούστερος τρόπος για να περάσουμε παραμέτρους σε μία Python συνάρτηση είναι με βάση τη θέση τους. Στον ορισμό της συνάρτησης, ορίζουμε ονόματα μεταβλητών για κάθε παράμετρο. Όταν καλούμε τη συνάρτηση οι τιμές που δίνουμε αντιστοιχίζονται στα ονόματα των παραμέτρων της με τη σειρά. Ο αριθμός των τιμών κατά την κλήση της πρέπει να είναι ίδιος με τον αριθμό των παραμέτρων στον ορισμό της συνάρτησης διαφορετικά προκαλείται εξαίρεση TypeError. Η ακόλουθη συνάρτηση υπολογίζει το x^y:

```
#ex6.2 positional parameters
def power(x,y):
    p = 1
    while y > 0:
        p = p * x
        y = y - 1
    return p
```

```
>>> ================================ RESTART ================================
>>>
>>> power(4,5)#η τιμή 4 αντιστοιχει στη 1η παράμετρο x και
                η τιμή 5 αντιστοιχεί στη 2η παράμετρο y
1024
>>> power(5,4)#αν αλλάξω τη θέση των τιμών το αποτέλεσμα θα είναι διαφορετικό
625
>>> power(5)#θα προκληθεί εξαίρεση TypeError
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    power(5)#θα προκληθεί εξαίρεση TypeError
TypeError: power() takes exactly 2 positional arguments (1 given)
>>>
```

Παράδειγμα 6.2

6.2.1.1 Προεπιλεγμένες τιμές

Προεπιλεγμένες τιμές μπορούν να δοθούν σε οποιονδήποτε αριθμό παραμέτρων κατά τον ορισμό της συνάρτησης και πρέπει να ανατίθενται τελευταίες στη λίστα παραμέτρων.

```
#ex6.3 default parameters
def power(x, y = 2):
    p = 1
    while y > 0:
        p = p * x
        y = y - 1
    return p
```

```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>> power(3,3)#η συνάρτηση power υπολογίζει το αποτέλεσμα
                σύμφωνα με τις τιμές που δώσαμε
27
>>> power(3)#εφόσον λείπει η δεύτερη τιμή θα χρησιμοποιηθεί η default τιμή της
            παραμέτρου y που δώσαμε στον ορισμό της συνάρτησης και δεν θα
            προκληθεί εξαίρεση
9
>>>
```

Παράδειγμα 6.3

6.2.2 Πέρασμα παραμέτρων ονομαστικά

Με αυτόν τον τρόπο περνάμε στην κλήση της συνάρτησης μαζί με την τιμή, και το όνομα της παραμέτρου. Επειδή οι τιμές δηλώνονται μαζί με τα ονόματα των

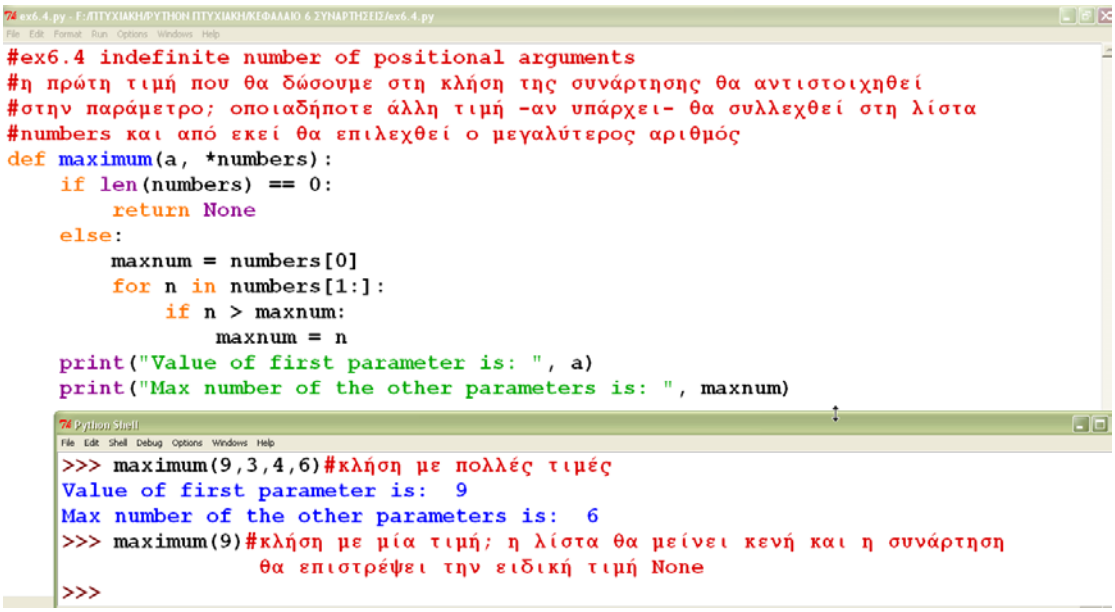
παραμέτρων, η σειρά τους δεν έχει σημασία. Ο τρόπος αυτός που περνάμε τις παραμέτρους καλείται keyword passing.

6.2.3 Μεταβλητός αριθμός παραμέτρων

Οι συναρτήσεις στην Python μπορούν να διαχειριστούν μεταβλητό αριθμό παραμέτρων. Αυτό μπορεί να γίνει με δύο διαφορετικούς τρόπους.

α. Απροσδιόριστος αριθμός παραμέτρων θέσης

Αν στην τελευταία παράμετρο μιας συνάρτησης βάλουμε το * μπροστά από το όνομά της, τότε αυτή η παράμετρος αποτελεί μία λίστα στην οποία συλλέγονται όλοι οι παράμετροι που δεν αντιστοιχούν σε παραμέτρους. Αυτός είναι και ένας απλό τρόπος να βρούμε τον μεγαλύτερο από μία λίστα αριθμών, όπως στο παρακάτω παράδειγμα:

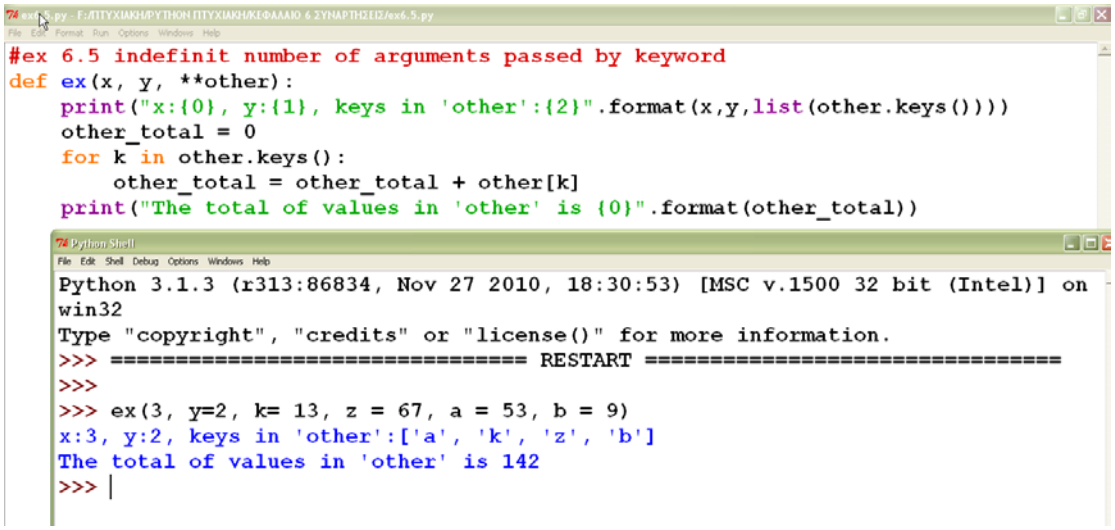


Παράδειγμα 6.4

β. Απροσδιόριστος αριθμός παραμέτρων ονομαστικά

Αν στην τελευταία παράμετρο μία συνάρτησης βάλουμε τα ** μπροστά από το όνομά της, τότε αυτή η παράμετρος αποτελεί ένα λεξικό στο οποίο συλλέγονται όλοι οι παράμετροι που θα περάσουμε με αυθαίρετο τρόπο με χρήση keyword, όταν καλέσουμε τη συνάρτηση. Για κάθε μία παράμετρο που θα εισάγεται στο λεξικό το κλειδί θα είναι το όνομα της ονομαστικής παραμέτρου και η τιμή, η τιμή

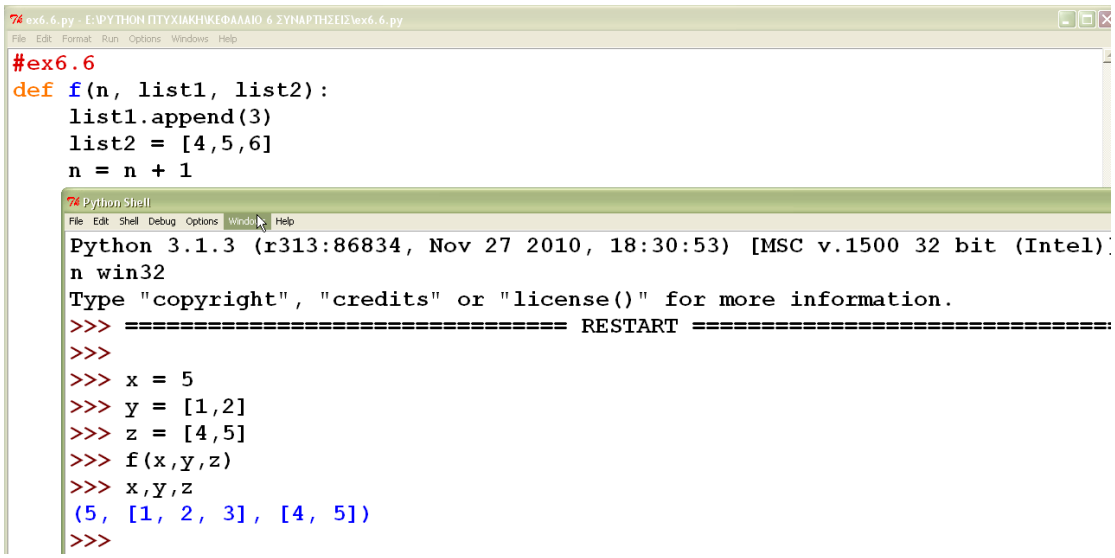
της ονομαστικής παραμέτρου. Βέβαια είναι περιττό να βάλουμε ονομαστικές παραμέτρους που τα ονόματά τους δεν αντιστοιχούν σε ονόματα παραμέτρων στον ορισμό της συνάρτησης.



Παράδειγμα 6.5

6.2.4 Μεταβλητοί τύποι δεδομένων ως παράμετροι

Οι παράμετροι περνιούνται σαν αναφορές αντικειμένων. Όταν σε μία συνάρτηση περνάμε αμετάβλητους τύπους δεδομένων (πλειάδες, ακολουθίες χαρακτήρων, αριθμούς) ως παραμέτρους, τότε οτιδήποτε γίνεται με αυτές τις παραμέτρους μέσα στη συνάρτηση δεν έχει επίπτωση έξω από τη συνάρτηση. Αν όμως περάσουμε μεταβλητούς τύπους δεδομένων (λίστες, λεξικά) ως παραμέτρους, τότε οποιαδήποτε αλλαγή γίνει σε αυτούς τους τύπους δεδομένων επηρεάζει οτιδήποτε σχετίζεται με αυτούς έξω από τη συνάρτηση.



Παράδειγμα 6.6

6.3 Είδη μεταβλητών και ανάθεση συναρτήσεων σε μεταβλητές

6.3.1 Είδη μεταβλητών

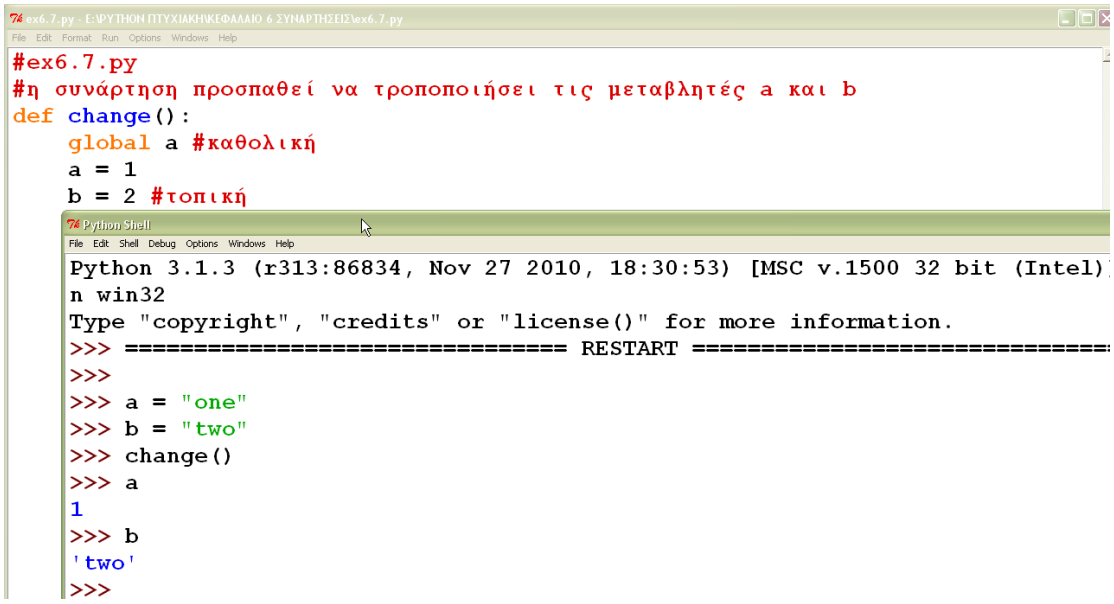
Υπάρχουν τριών ειδών μεταβλητές

- Τοπικές μεταβλητές (local)

Στο παράδειγμα 6.1 στη συνάρτηση fact, οι μεταβλητές r και n είναι τοπικές σε οποιαδήποτε κλήση της συνάρτησης. Οι αλλαγές σε αυτές γίνονται όταν η συνάρτηση εκτελείται και δεν έχει καμία επίπτωση σε μεταβλητές έξω από τη συνάρτηση. Οποιεσδήποτε μεταβλητές στη λίστα παραμέτρων της συνάρτησης και οι μεταβλητές που δημιουργούνται μέσα στη συνάρτηση με ανάθεση πχ r = 1, είναι τοπικές στη συνάρτηση.

- Καθολικές μεταβλητές (global)

Μπορούμε να μετατρέψουμε μία μεταβλητή σε καθολική δηλώνοντάς την πριν χρησιμοποιηθεί με τη λέξη κλειδί global. Οι καθολικές μεταβλητές μπορούν να προσπελαστούν και να τροποποιηθούν από την συνάρτηση. Υφίστανται και έξω από την συνάρτηση και μπορούν να προσπελαστούν και να τροποποιηθούν και από άλλες συναρτήσεις που τις ορίζουν καθολικές ή από κώδικα που δεν είναι μέσα σε μία συνάρτηση.



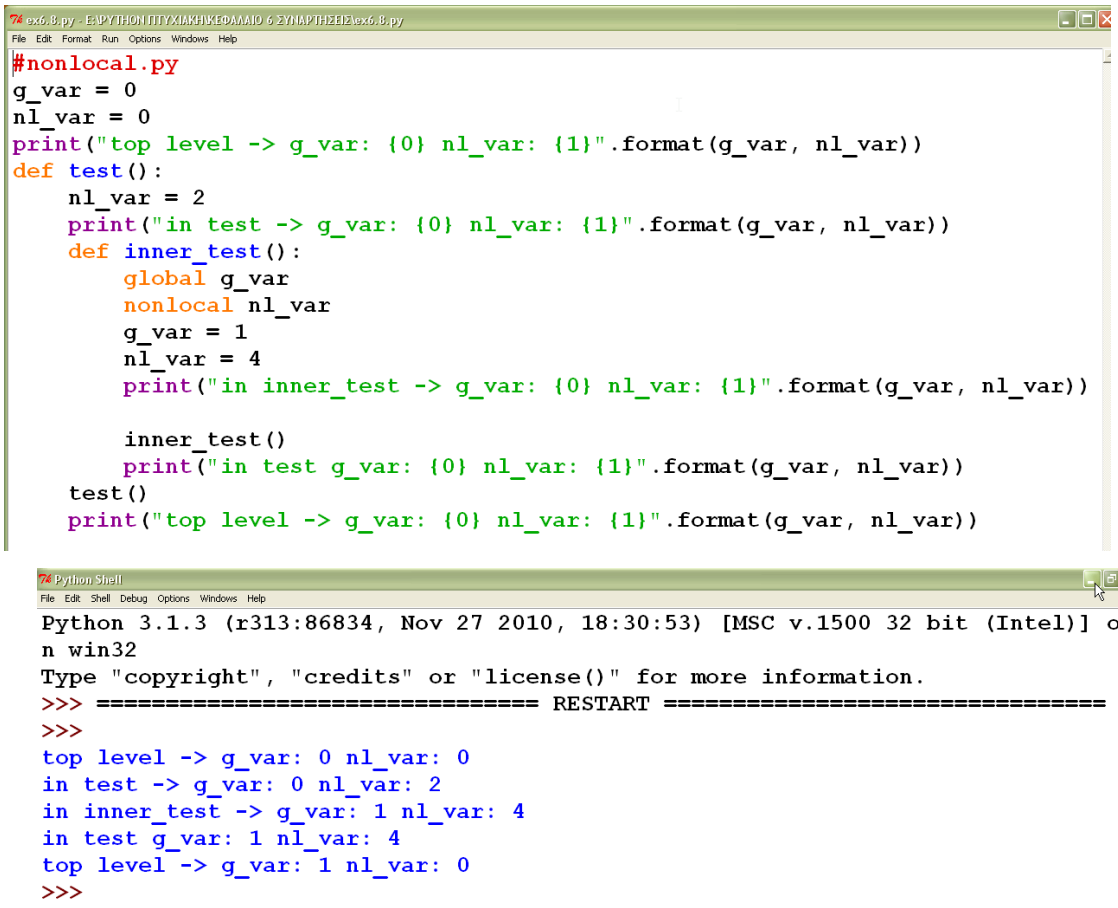
Παράδειγμα 6.7

Η ανάθεση στην μεταβλητή a μέσα στη συνάρτηση είναι ανάθεση στη καθολική μεταβλητή a που υπάρχει και εκτός της fun. Επειδή όμως είναι καθολική, η συνάρτηση μπορεί να αλλάξει την τιμή της και να της δώσει την τιμή 1 αντί “one”.

Το ίδιο δεν συμβαίνει με την μεταβλητή b η οποία είναι τοπική στην συνάρτηση fun οπότε η τιμή της θα παραμείνει ως έχει και δεν θα μεταβληθεί.

- Μη τοπικές μεταβλητές (nonlocal)

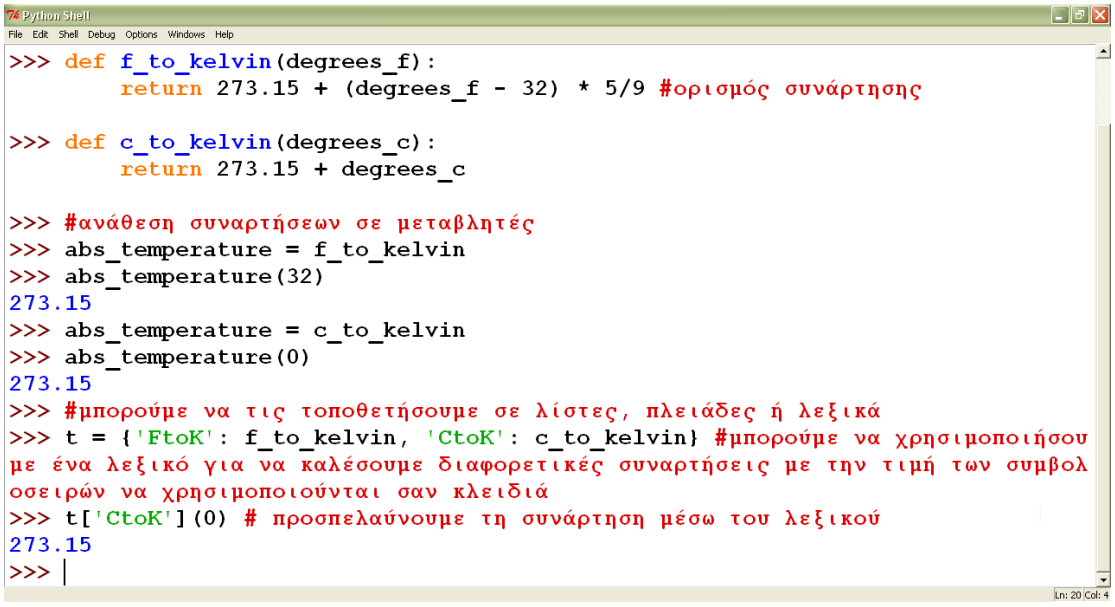
Παρόμοια με την καθολική είναι και μία μη τοπική μεταβλητή, που προκαλεί ένα προσδιοριστικό να αναφέρεται σε μία συγκεκριμένη μεταβλητή. Το σημαντικό είναι ότι οι καθολικές μεταβλητές χρησιμοποιούνται για μία υψηλού επιπέδου (top-level) μεταβλητή, ενώ η μη τοπικές μεταβλητές μπορούν να αναφέρονται σε οποιαδήποτε μεταβλητή που περικλείει το πεδίο εφαρμογής. (enclosing scope), όπως στο παράδειγμα παρακάτω :



Παράδειγμα 6.8

6.3.2 Ανάθεση συναρτήσεων σε μεταβλητές.

Μπορούμε να αναθέσουμε τις συναρτήσεις, όπως οποιοδήποτε άλλο Python αντικείμενο, σε μεταβλητές όπως στο επόμενο παράδειγμα:



Εικόνα 6.1

6.4 Συναρτήσεις lambda

Οι συναρτήσεις lambda είναι μικρές συναρτήσεις που μπορούμε να ορίσουμε εσωτερικά σε μία συνάρτηση με την παρακάτω σύνταξη:

lambda <parameters> : expression

Οι παράμετροι (parameters) είναι προαιρετικές. Η έκφραση (expression) δεν μπορεί να περιέχει εντολές πχ if…elsif ούτε μία εντολή return ή yield. Το αποτέλεσμα μίας lambda συνάρτησης είναι μία ανώνυμη (anonymous) συνάρτηση, όπως καλείται, και επιστρέφει ουσιαστικά το αποτέλεσμα της έκφρασης, το οποίο μπορεί να ανατεθεί σε μία μεταβλητή.

Σαν παράμετρο στη συνάρτηση αυτή μπορώ να περάσω μία πλειάδα αλλά ο διερμηνευτής δεν θα ξε-πακετάρει τις τιμές της σε ξεχωριστές παραμέτρους. Θα πρέπει να αναφερθούμε σε κάθε παράμετρο με τον δείκτη θέσης.

>>>e = [“HOW”, “ARE”, “YOU”]

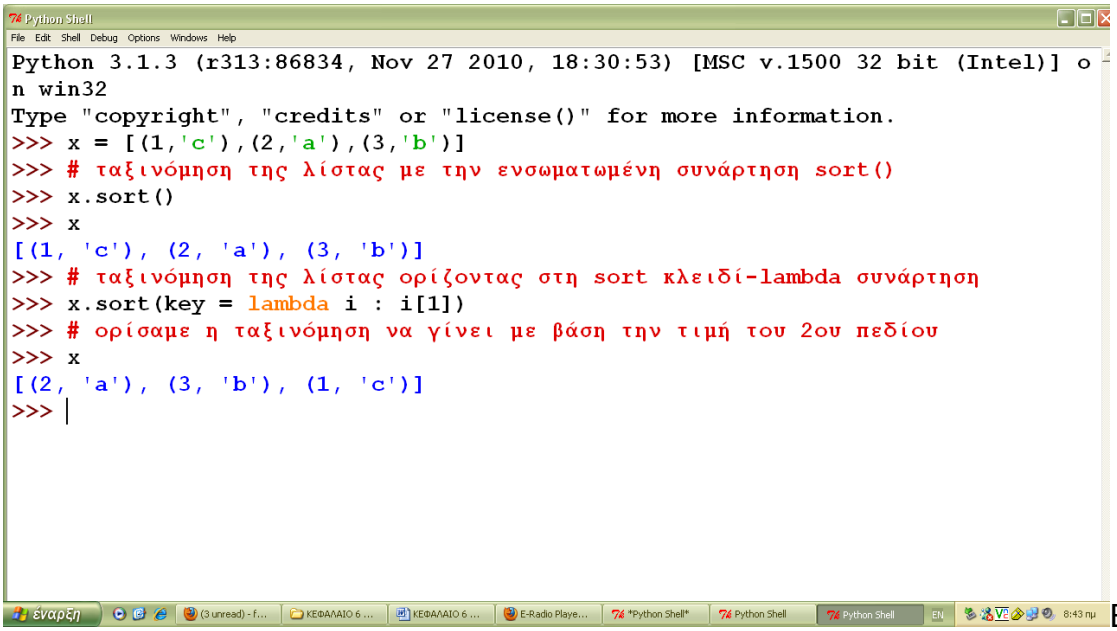
>>>s = lambda e : (e[2].lower, e[1])

>>>s

“you”, “ARE”

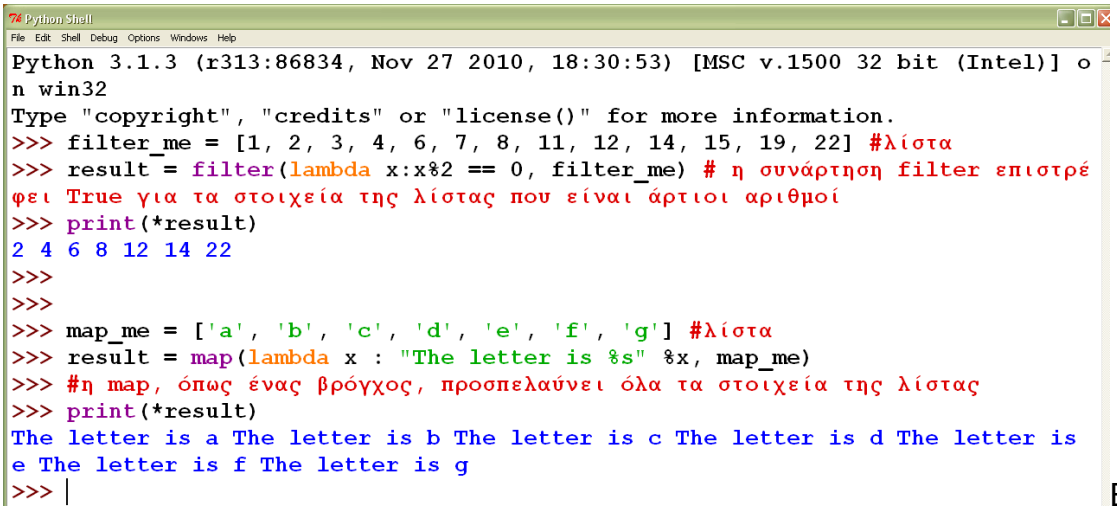
Παράδειγμα 6.9

Η συνάρτηση αυτή μπορεί να περαστεί ως παράμετρος στη built-in συνάρτηση sort() και αποτελεί το κλειδί για την ταξινόμηση. Παράδειγμα της συνάρτησης sort() χωρίς και με τη συνάρτηση lambda :



Εικόνα 6.2

Τέλος η συνάρτηση lambda χρησιμοποιείται πολύ συχνά με τις ενσωματωμένες συναρτήσεις map και filter. Η συνάρτηση map() χρησιμοποιείται σε περιπτώσεις που χρειαζόμαστε να γίνει μία συγκεκριμένη ενέργεια σε κάθε στοιχείο μίας λίστας και η συνάρτηση filter() παίρνει και αυτή σαν παράμετρο μία λίστα και διαγράφει στοιχεία της με βάση κάποιο κριτήριο που έχουμε προσδιορίσει.

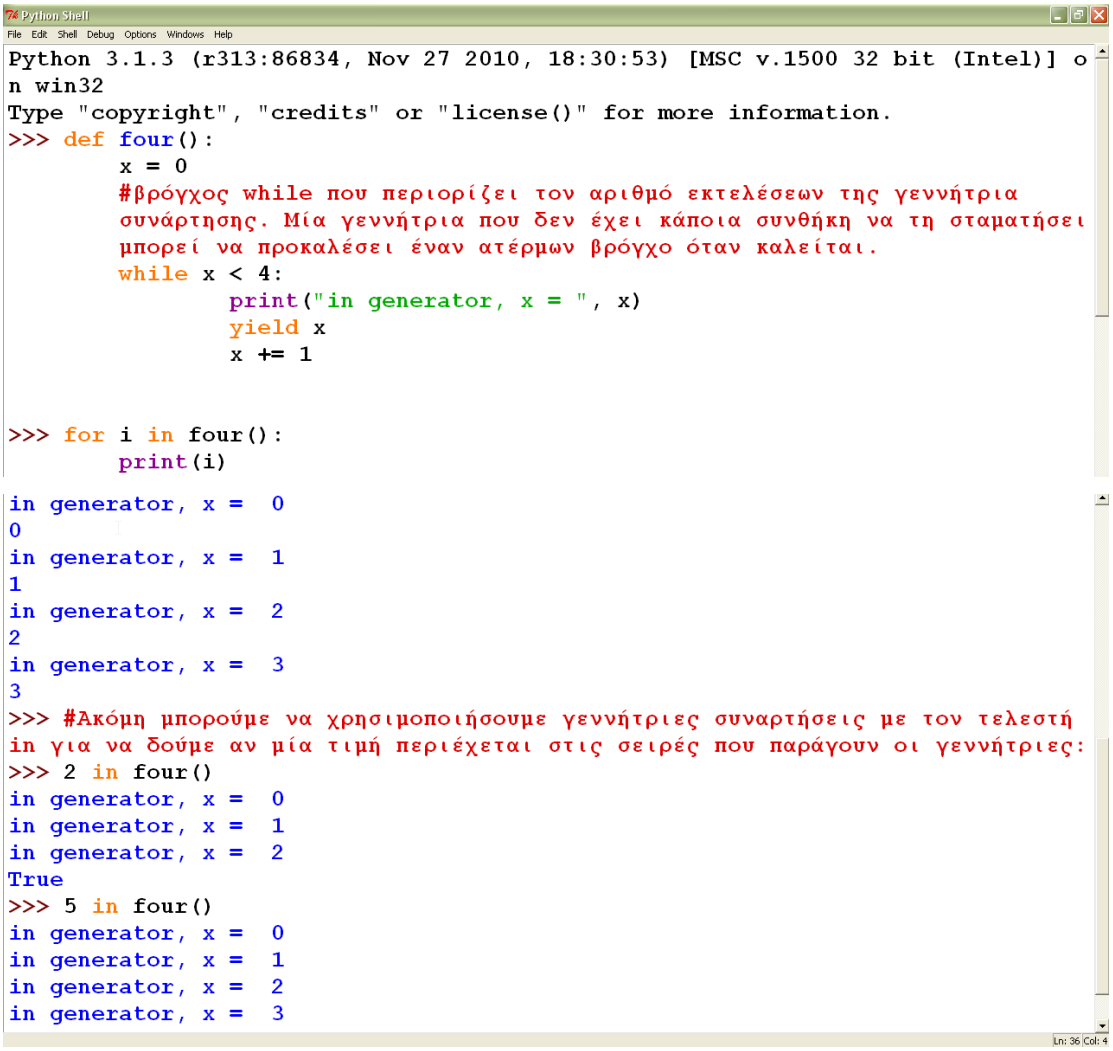


Εικόνα 6.3

6.5 Γεννήτριες συναρτήσεις

Η γεννήτρια συνάρτηση είναι ένας ειδικός τύπος συνάρτησης που μπορούμε να χρησιμοποιήσουμε για να προσδιορίσουμε της δικές μας επαναλήψεις. Μία

γεννήτρια συνάρτηση, επιστρέφει την τιμή που προκύπτει μετά την επανάληψη χρησιμοποιώντας την παραγόμενη λέξη κλειδί. Όταν δεν υπάρχουν άλλα προσδιοριστικά, μία κενή πρόταση return ή μία ροή παύσης στο τέλος της συνάρτησης τερματίζουν τις επαναλήψεις. Οι τοπικές μεταβλητές σε μία γεννήτρια συνάρτηση σώζονται από τη μία κλήση στην επόμενη, σε αντίθεση με τις κανονικές συναρτήσεις :



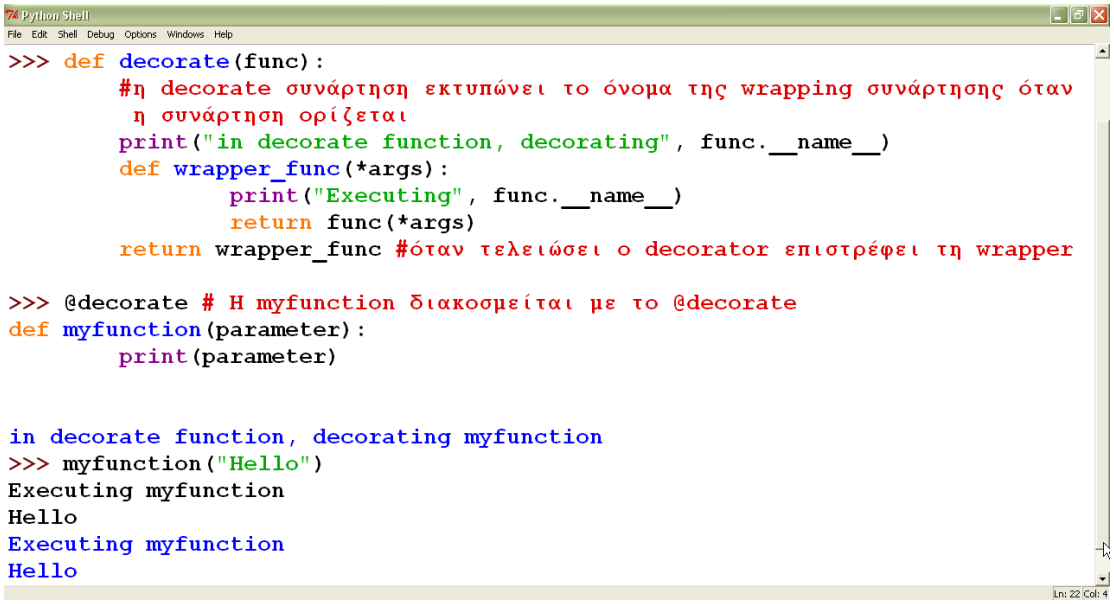
Εικόνα 6.4

6.6. Decorators

Ένας decorator είναι μία συνάρτηση που παίρνει ως παράμετρο μία συνάρτηση ή μέθοδο και επιστρέφει μία νέα συνάρτηση ή μέθοδο που είναι ουσιαστικά η προηγούμενη με επιπρόσθετη λειτουργικότητα.

Οι συναρτήσεις αυτές μπορούν να περαστούν σαν παράμετροι σε άλλες συναρτήσεις και να επιστραφούν με την πρόταση return. Για να χρησιμοποιήσουμε

έναν decorator πρέπει να προσδιορίσουμε τη συνάρτηση που θα καλύψει ή θα «διακοσμήσει» άλλες συναρτήσεις και μετά να χρησιμοποιήσουμε το @ ακολουθούμενο από τον decorator αμέσως πριν η συνάρτηση κάλυψης προσδιοριστεί. Η διακοσμητική συνάρτηση πρέπει να πάρει μία συνάρτηση ως παράμετρο και να επιστρέψει συνάρτηση, όπως στη συνέχεια.

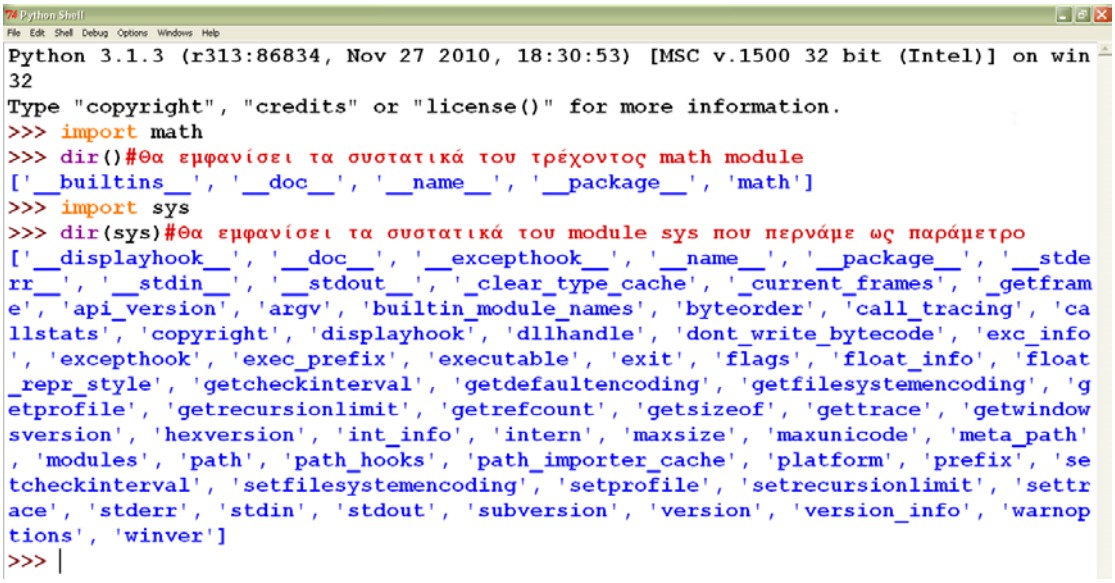


Εικόνα 6.5

7. MODULES – ΠΑΚΕΤΑ – ΒΙΒΛΙΟΘΗΚΕΣ

7.1 Modules

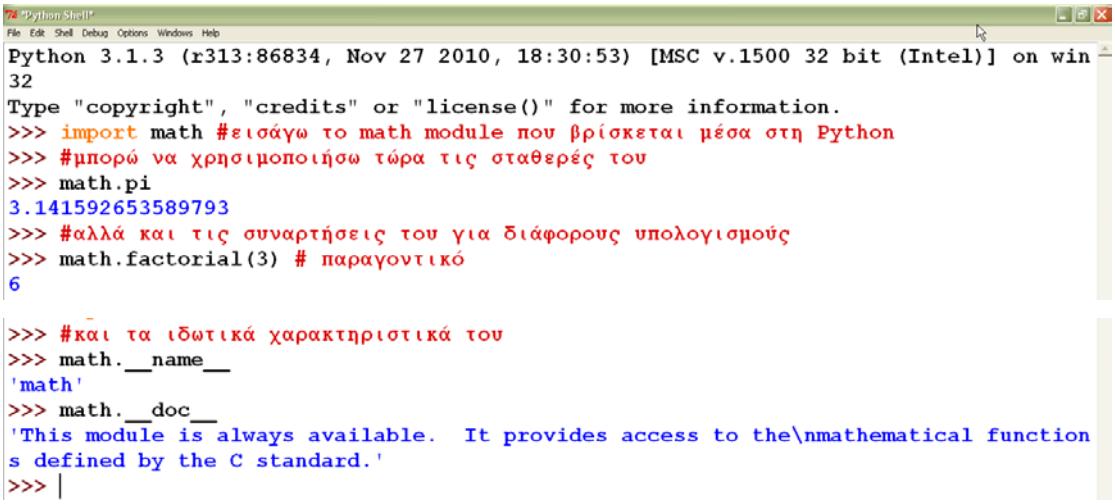
Ένα module είναι ένα αρχείο κώδικα με κατάληξη .py. Η διαφορά του με ένα πρόγραμμα είναι ότι το module δεν τρέχει, αλλά εισάγετε σε ένα πρόγραμμα για να χρησιμοποιηθεί από αυτό. Η εισαγωγή ενός module γίνεται με την εντολή import. Το όνομά του προέρχεται από το όνομα του αρχείου. Η χρήση ενός συστατικού (συνάρτηση, σταθερά, μεταβλητή) του module γίνεται μέσω του ονόματος του module και έτσι αποφεύγεται η σύγκρουση ονομάτων. Αυτό μας επιτρέπει να δημιουργήσουμε μία διαφορετική μεταβλητή μέσα στο πρόγραμμα με το ίδιο όνομα. Αν θέλουμε να δούμε τα συστατικά (συναρτήσεις, κλάσεις, μεταβλητές) που ενσωματώνονται μέσα σε ένα module χρησιμοποιούμε τη συνάρτηση dir. Αν δώσουμε στη συνάρτηση dir το όνομα ενός module μας επιστρέφει μία λίστα με τα συστατικά που ορίζονται στο συγκεκριμένο module, διαφορετικά χωρίς παράμετρο μας επιστρέφει τη λίστα με τα συστατικά του τρέχοντος module.



Εικόνα 7.1

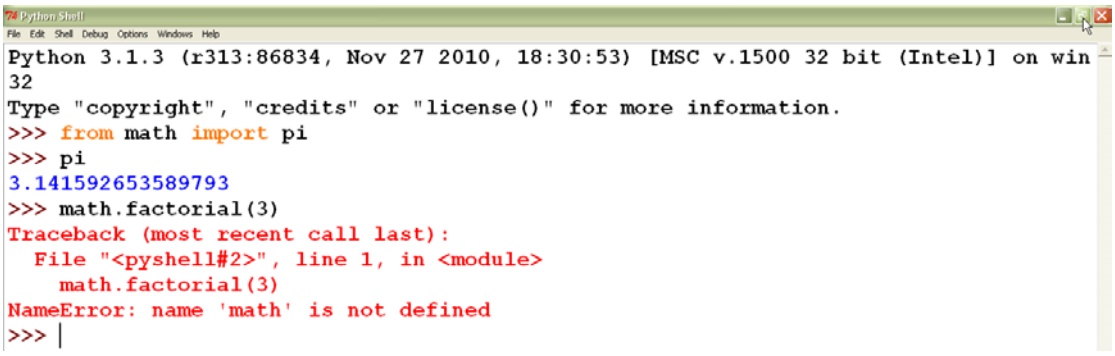
Η εντολή import έχει τρεις τρόπους σύνταξης :

1. import modulename : μπορούμε να χρησιμοποιήσουμε οποιοδήποτε συστατικό του module μέσω του ονόματός του:



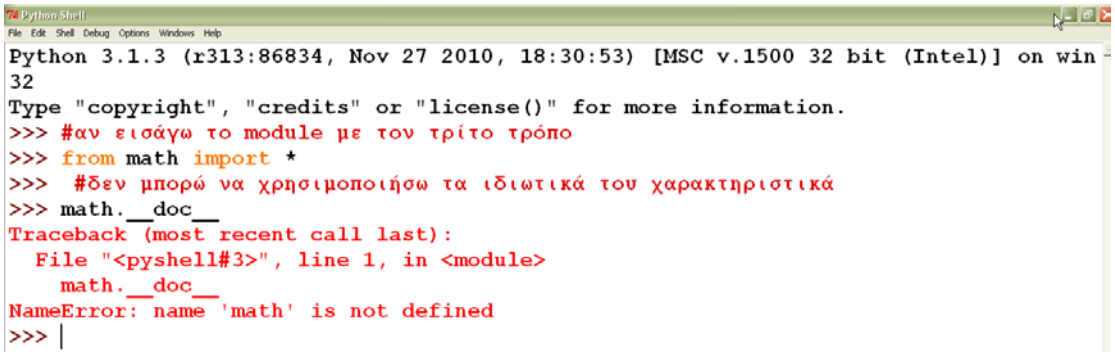
Εικόνα 7.2

2. from modulename import name1, name2, name3,… : μπορούμε να χρησιμοποιήσουμε μόνο κάποιο από τα συστατικά (name1, name2, name3,…) που εισάγουμε και απευθείας με το όνομά του χωρίς να προηγείται ρο όνομα του module:



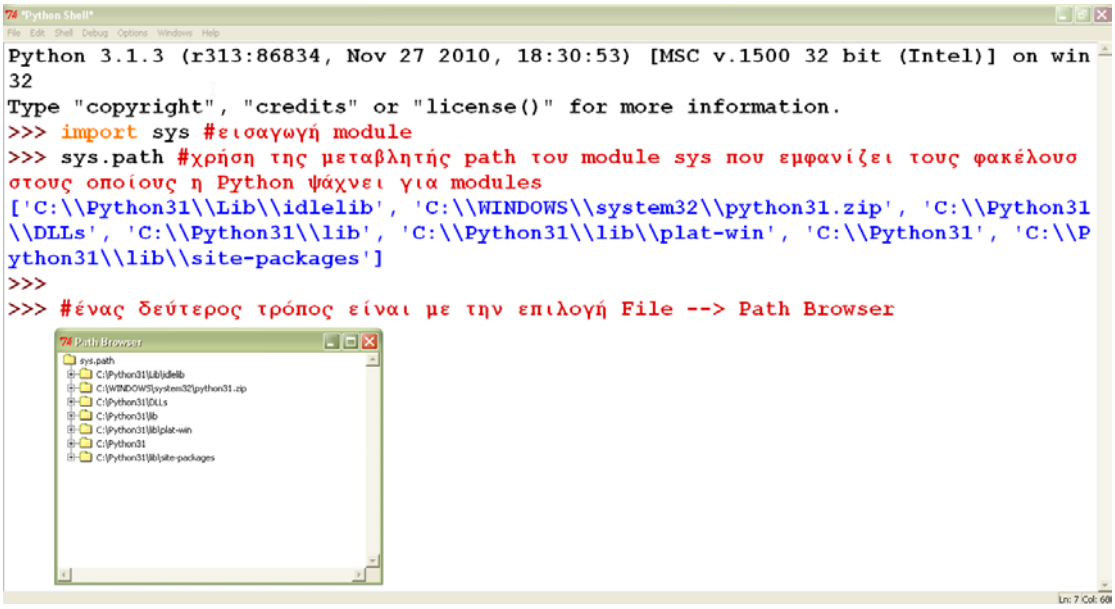
Εικόνα 7.3

3. from modulename import * : εισάγει όλα τα δημόσια συστατικά του module, δηλαδή αυτά που δεν ξεκινάνε με underscore και μπορούμε να τα χρησιμοποιήσουμε απευθείας. Η ιδιαιτερότητα εδώ είναι ότι αν μέσα στο module υπάρχει μία λίστα με όνομα _all_ τότε τα ονόματα που υπάρχουν σε αυτήν εισάγονται είτε ξεκινάνε με underscore είτε όχι. Με αυτόν τον τρόπο αν δύο modules ορίζουν ένα ίδιο συστατικό και τα εισάγουμε και τα δύο με αυτόν τον τρόπο θα καταλήξουμε σε σύγκρουση ονομάτων και το συστατικό του δεύτερου module θα αντικαταστήσει το συστατικό του πρώτου.



Εικόνα 7.4

Ένα module γραμμένο σε Python είναι ενσωματωμένο σε αυτήν και ο διερμηνευτής της Python γνωρίζει ότι θα το βρει σε έναν από του φακέλους που εμφανίζει η μεταβλητή sys.path στους οποίους ψάχνει για διαθέσιμα modules. Ένας άλλος τρόπος να δούμε αυτούς τους φακέλους είναι από την επιλογή File → Path Browser. Ένα module γραμμένο σε άλλη γλώσσα πχ C / C++ δεν είναι ενσωματωμένο αλλά εισάγεται με τον ίδιο τρόπο.



Εικόνα 7.5

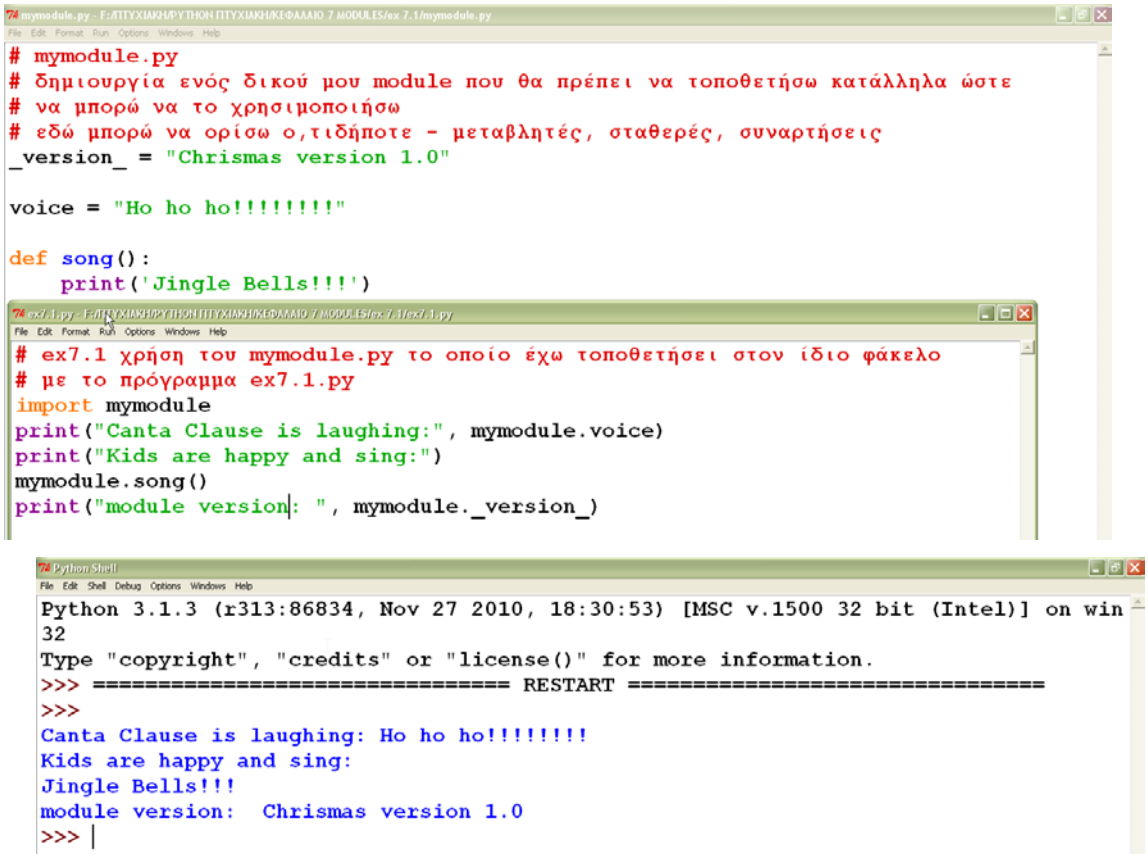
Ακόμη μας δίνεται οι δυνατότητα να δημιουργήσουμε τα δικά μας modules αρκεί να είναι αρχεία με κατάληξη .py και να τοποθετούνται κατάλληλα.

Υπάρχουν τρεις τρόποι να τοποθετήσουμε τα modules ώστε να είναι διαθέσιμα στα προγράμματά μας διαφορετικά δημιουργείται εξαίρεση ImportError :

1. Τοποθέτηση των modules μέσα σε φακέλους που συνήθως η Python ψάχνει για modules στη λίστα του sys.path.

2. Τοποθέτηση των modules που χρησιμοποιούνται από ένα Python πρόγραμμα μέσα στον ίδιο φάκελο με το πρόγραμμα.

3. Δημιουργία φακέλου που θα κρατάει τα modules και τροποποίηση της μεταβλητής sys.path ώστε να περιλαμβάνει αυτόν το νέο φάκελο, που είναι πιο σωστή επιλογή για site-specific modules που θα χρησιμοποιηθούν σε περισσότερα από ένα προγράμματα σε ένα site.



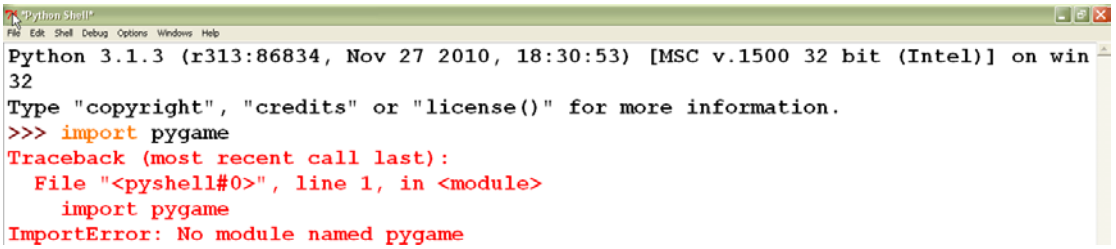
```
# mymodule.py
# δημιουργία ενός δικού μου module που θα πρέπει να τοποθετήσω κατάλληλα ώστε
# να μπορώ να το χρησιμοποιήσω
# εδώ μπορώ να ορίσω ο,τιδήποτε - μεταβλητές, σταθερές, συναρτήσεις
_version_ = "Chrismas version 1.0"

voice = "Ho ho ho!!!!!!!!"

def song():
    print('Jingle Bells!!!')
```

```
# ex7.1 χρήση του mymodule.py το οποίο έχω τοποθετήσει στον ίδιο φάκελο
# με το πρόγραμμα ex7.1.py
import mymodule
print("Canta Clause is laughing:", mymodule.voice)
print("Kids are happy and sing:")
mymodule.song()
print("module version: ", mymodule._version_)
```

```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
Canta Clause is laughing: Ho ho ho!!!!!!!!
Kids are happy and sing:
Jingle Bells!!!
module version:  Chrismas version 1.0
>>>
```

Παράδειγμα 7.1 Δημιουργία δικού μας module

Τέλος υπάρχουν και τα third-party modules που πρέπει να τα «κατεβάσουμε» και να τα τοποθετήσουμε σε έναν φάκελο στη διαδρομή αναζήτησης για modules ώστε να είναι διαθέσιμα για εισαγωγή στα προγράμματά μας. Παράδειγμα ενός third-party module είναι το pygame, το οποίο χρησιμοποιείται για προγραμματισμό παιχνιδιών σε Python και θα το δούμε σε επόμενο κεφάλαιο. Πριν εγκαταστήσουμε το pygame, η Python δεν το αναγνωρίζει και προκαλεί σφάλμα:



Μετά την εγκατάστασή του η Python το αναγνωρίζει και μπορούμε να χρησιμοποιήσουμε οποιοδήποτε από τα συστατικά του:

Εικόνα 7.6

7.2 Πακέτα

Ένα πακέτο είναι ένας φάκελος που περιέχει ένα σύνολο από modules και ένα αρχείο __init__.py που ορίζει ότι ο φάκελος αυτός είναι ένα πακέτο και περιέχει modules. Το όνομα ενός πακέτου προέρχεται από το όνομα του κύριου φακέλου του πακέτου. Έστω ότι τα modules Bmp.py, Jpeg.py, Png.py, Tiff.py και Xpm.py για ανάγνωση και εγγραφή ποικίλων μορφών αρχείων γραφικών. Μπορούμε να δημιουργήσουμε ένα φάκελο Graphics μέσα στο φάκελο του προγράμματος μας και να βάλουμε σε αυτόν τα παραπάνω modules και το αρχείο __init__.py οπότε ο φάκελος Graphics αποτελεί ένα πακέτο.

Η εισαγωγή του module Bmp.py του πακέτου Graphics γίνεται με την πρόταση:

```
import Graphics.Bmp
image = Graphics.Bmp.load("img.bmp")
```

```
Graphics/
        __init__.py
        Bmp.py
        Jpeg.py
        Png.py
        Tiff.py
        Xpm.py
```

Ένα πακέτο μπορεί να περιέχει υπό-πακέτα. Η εισαγωγή ενός module ενός υπό-πακέτου γίνεται με την πρόταση:

```
import Graphics.Vector.Eps
image = Graphics.Vector.Eps.load("img.eps")
```

```
Graphics/
        __init__.py
        Bmp.py
        Jpeg.py
        Png.py
        Tiff.py
        Xpm.py
        Vector/
                __init__.py
                Eps.py
                Svg.py
```

7.3 Βιβλιοθήκες

Η στάνταρ βιβλιοθήκη της Python, ακολουθεί τη φιλοσοφία «batteries included» και περιέχει ενσωματωμένους τύπους δεδομένων και σταθερές, ενσωματωμένες συναρτήσεις και εξαιρέσεις και μεγάλο μέρος της λειτουργικότητας που προσφέρει προέρχεται από τα modules και τα πακέτα που περιέχονται σε αυτήν.

Μερικά από τα σημαντικότερα modules και πακέτα παρουσιάζονται παρακάτω και είναι χωρισμένα σε ενότητες.

string	Εισάγετε όταν θέλουμε να μορφοποιήσουμε ακολουθίες χαρακτήρων. Σημαντικές σταθερές είναι οι string.ascii_letters και string.hexdigits και η κλάση string.Formatter.
re	Χρήσιμο για χειρισμό κανονικών εκφράσεων.
struct	Παρέχει συναρτήσεις για packing και unpacking αριθμών, Booleans και strings σε και από χρησιμοποιώντας τη δυαδική αναπαράστασή τους. Χρήσιμο σε περιπτώσεις που θέλουμε να αντλήσουμε ή να στείλουμε δεδομένα από ή σε μία βιβλιοθήκη

	χαμηλότερου επιπέδου γραμμένη σε γλώσσα C.
difflib	Παρέχει κλάσεις και μεθόδους για σύγκριση ακολουθιών πχ strings.
textwrap	Για μορφοποίηση κειμένου. Ορίζει το πλάτος των γραμμών και τις εσοχές.

Πίνακας 7.1 Modules για χειρισμό strings

calendar, datetime	Προσφέρουν λειτουργίες για να χειριστούμε δεδομένα τύπου date, time και calendar.
time	Χρησιμοποιούμε αυτό το module για χειρισμό timestamps.
collections	Παρέχει τους συλλογικούς τύπους δεδομένων λεξικό και πλειάδα με τις συναρτήσεις collections.defaultdict και collections.namedtuple αντίστοιχα. Επιπλέον παρέχει τους τύπους collections.UserList, collections.UserDict και collections.deque.
array	Το module αυτό παρέχει τον ακολουθιακό τύπο δεδομένων array στον οποίο μπορούμε να αποθηκεύσουμε αριθμούς ή χαρακτήρες ίδιου τύπου.
shed	Παρέχει χρονοπρογραμματισμό γεγονότων.
queue	Παρέχει συγχρονισμό κλάσεων queue.
copy	Χρησιμοποιείται για λειτουργίες αντιγραφής.
pprint	Χρησιμοποιείται για εκτύπωση δεδομένων
bisect	Παρέχει συναρτήσεις για αναζήτηση σε ταξινομημένες ακολουθίες (πχ ταξινομημένες λίστες) και για εισαγωγή στοιχείων ώστε να διατηρείται η σειρά ταξινόμησης.
heapq	Παρέχει συναρτήσεις για μετατροπή ακολουθιών (πχ λίστα) σε σωρό.

Πίνακας 7.2 Modules για χειρισμό διάφορων τύπων δεδομένων

decimal	Παρέχει τους αριθμούς τύπου decimal – δεκαδικοί αριθμοί.
fraction	Παρέχει τους αριθμούς τύπου fraction – λογικοί αριθμοί.
math	Παρέχει τις στάνταρ μαθηματικές συναρτήσεις.
cmath	Παρέχει συναρτήσεις για μιγαδικούς αριθμούς.

random	Χρησιμοποιείται για την παραγωγή τυχαίων αριθμών.

Πίνακας 7.3 Αριθμητικά και Μαθηματικά Modules

os	Παρέχει λειτουργίες για χειρισμό ονομάτων διαδρομών (pathnames).
shutil	Παρέχει υψηλού επιπέδου συναρτήσεις για χειρισμό αρχείων και φακέλων όπως οι shutil.copy(), shutil.copytree(), shutil.move() και shutil.rmtree().
tempfile	Χρησιμοποιείται για δημιουργία προσωρινών αρχείων και φακέλων.
filecmp	Χρησιμοποιείται για σύγκριση αρχείων (με τη συνάρτηση filecmp.cmp()) και σύγκριση φακέλων (με τη συνάρτηση filecmp.cmpfiles()).
fileinput	Προσπελαύνει τις γραμμές μιας ροής εισόδου πολλαπλών γραμμών.
linecache	Παρέχει τυχαία πρόσβαση σε γραμμές κειμένου.
cvs	Παρέχει τη δυνατότητα για ανάγνωση και εγγραφή αρχείων cvs.

Πίνακας 7.4 Modules για χειρισμό αρχείων και φακέλων

bz2	Για χειρισμό .bz2 αρχείων.
gzip	Για χειρισμό .gz αρχείων.
tarfile	Για χειρισμό .tar, .tgz αρχείων.
zipfile	Για χειρισμό .zip αρχείων.
aifc	Για AIFF (Audio Interchange File Format)
configparser	Για ανάγνωση και εγγραφή .ini αρχείων.
cvc	Για ανάγνωση και εγγραφή Comma Seperatored Value δεδομένων που παρέχει πχ το excel

Πίνακας 7.5 Modules για διάφορους τύπους αρχείων και κωδικοποιήσεις

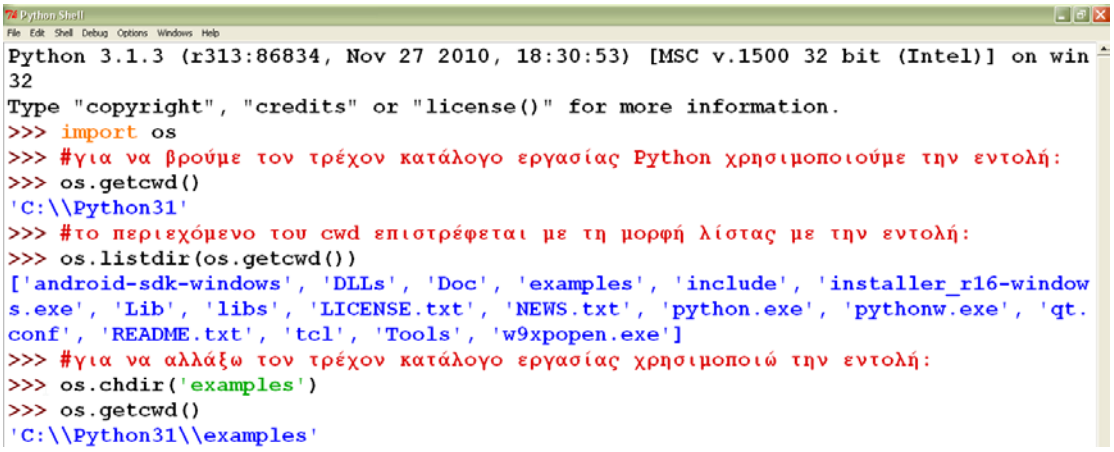
Πέρα από τα modules και τα πακέτα της στάνταρ βιβλιοθήκης υπάρχουν και εξωτερικά modules που παρέχουν λειτουργικότητα που δεν μας παρέχει η Python και θα τα δούμε σε επόμενα κεφάλαια.

8. ΔΙΑΧΕΙΡΙΣΗ ΣΥΣΤΗΜΑΤΟΣ ΑΡΧΕΙΩΝ ΜΕΣΩ PYTHON

Η Python παρέχει ένα σύνολο συναρτήσεων και σταθερών που διευκολύνουν την διαχείριση διαδρομών και λειτουργιών στο σύστημα αρχείων με έναν τρόπο ανεξάρτητο από το επιλεγμένο λειτουργικό σύστημα μέσω του os module.

8.1 Διαδρομές

Όλα τα λειτουργικά συστήματα αναφέρονται σε αρχεία και φακέλους με συμβολοσειρές, οι οποίες καλούνται διαδρομές / μονοπάτια (paths / pathnames).
Υπάρχουν δύο τύποι διαδρομών.
α)Οι **απόλυτες διαδρομές** που προσδιορίζουν την ακριβή θέση ενός αρχείου σε ένα σύστημα αρχείων, ξεκινώντας από τη ρίζα του συστήματος. Στα Windows μία διαδρομή που ξεκινάει με ένα όνομα δίσκου ακολουθούμενο με μία \ και μετά ένα μονοπάτι είναι απόλυτη διαδρομή: C:\Program Files\Doom.
β)Οι **σχετικές διαδρομές** που προσδιορίζουν τη θέση του αρχείου σχετικά με κάποιο άλλο σημείο στο σύστημα αρχείων, το οποίο δεν προσδιορίζεται στην σχετική διαδρομή.
Οι σχετικές διαδρομές χρειάζονται περιεχόμενο για να προσκολληθούν σε αυτό. Ο απλούστερος τρόπος είναι να προσθέσουμε τη σχετική διαδρομή σε μία ήδη υπάρχουσα απόλυτη διαδρομή. Ο δεύτερος τρόπος είναι δια μέσου μίας ρητής αναφοράς στον τρέχοντα κατάλογο εργασίας (current working directory).
Ο φάκελος στον οποίο το πρόγραμμα Python βρίσκεται όταν εκτελείται, καλείται **τρέχον κατάλογος εργασίας** για το πρόγραμμα αυτό. Η Python από προεπιλογή χρησιμοποιεί τον τρέχοντα κατάλογο εργασίας όταν συναντήσει μία σχετική διαδρομή ως παράμετρο σε μία εντολή.
Στα Windows μία διαδρομή που δεν ξεκινάει ούτε με γράμμα δίσκου, ούτε με \ είναι σχετική διαδρομή : mydirectory\letters\business.
Μία διαδρομή που ξεκινάει με \\ ακολουθούμενο από το όνομα ενός server είναι μία διαδρομή για πηγή δικτύου. Οτιδήποτε άλλο μπορεί να θεωρηθεί ως άκυρη διαδρομή.



Εικόνα 8.1

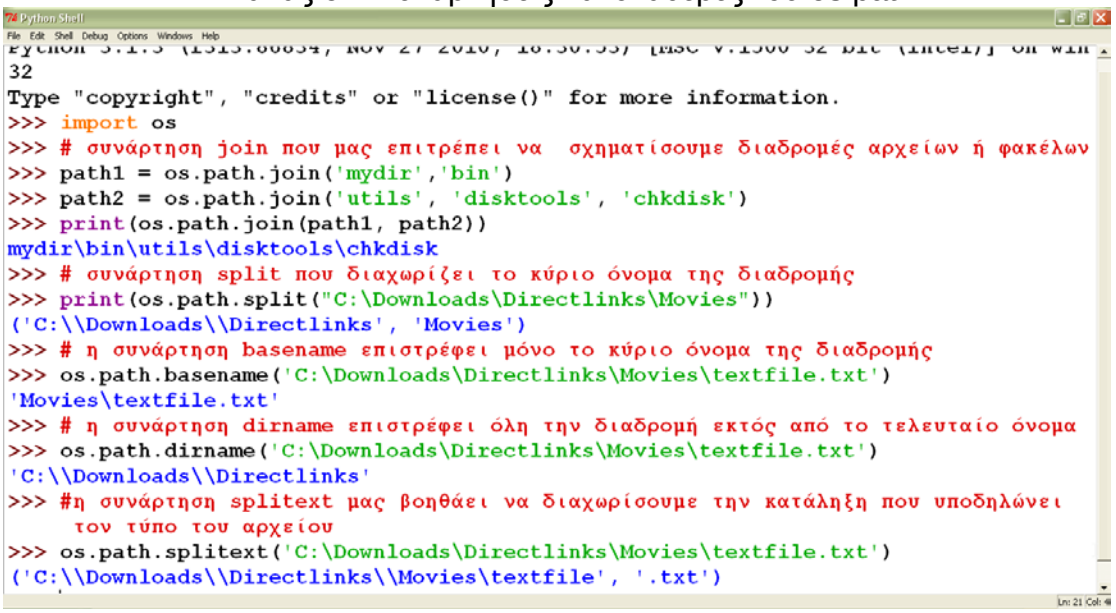
8.2 Χειρισμός Διαδρομών

Η Python παρέχει έναν αριθμό συναρτήσεων και σταθερών μέσω του sub-module **os.path**, το οποίο μπορούμε να χρησιμοποιήσουμε για να χειριστούμε τις διαδρομές.

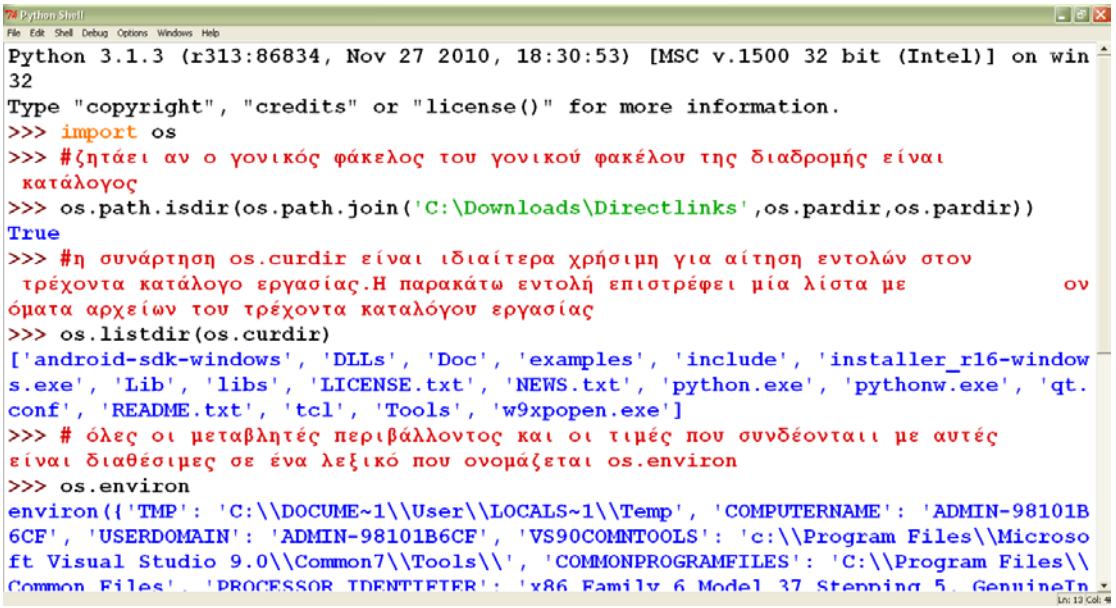
os.path.join(n1,n2,…)	η συνάρτηση join, μας επιτρέπει να σχηματίσουμε διαδρομές αρχείων ή φακέλων χωρίς να ανησυχούμε για τις συμβάσεις που επιβάλλει το λειτουργικό σύστημα όσο αφορά τους διαχωριστές των ονομάτων
os.path.split(path)	η εντολή split επιστρέφει μία πλειάδα δύο στοιχείων, διαχωρίζοντας το κύριο όνομα της διαδρομής από το υπόλοιπο της διαδρομής
os.path.basename(path)	η συνάρτηση basename επιστρέφει μόνο το κύριο όνομα της διαδρομής
os.path.dirname(path)	η συνάρτηση dirname επιστρέφει όλη την διαδρομή εκτός από το τελευταίο όνομα
os.path.splitext(path)	η συνάρτηση splitext μας βοηθάει να διαχωρίσουμε την κατάληξη που υποδηλώνει τον τύπο του αρχείου
os.curdir, os.pardir	σταθερές οι οποίες προσδιορίζουν το σύμβολο που χρησιμοποιείται από το λειτουργικό σύστημα για τους δείκτες που προσδιορίζουν τους καταλόγους και τους γονικούς καταλόγους
os.name	επιστρέφει το όνομα του Python module που έχει εισαχθεί

	για να διαχειριστεί τις λεπτομέρειες του λειτουργικού συστήματος

Πίνακας 8.1 Συναρτήσεις και σταθερές του os.path



Εικόνα 8.2

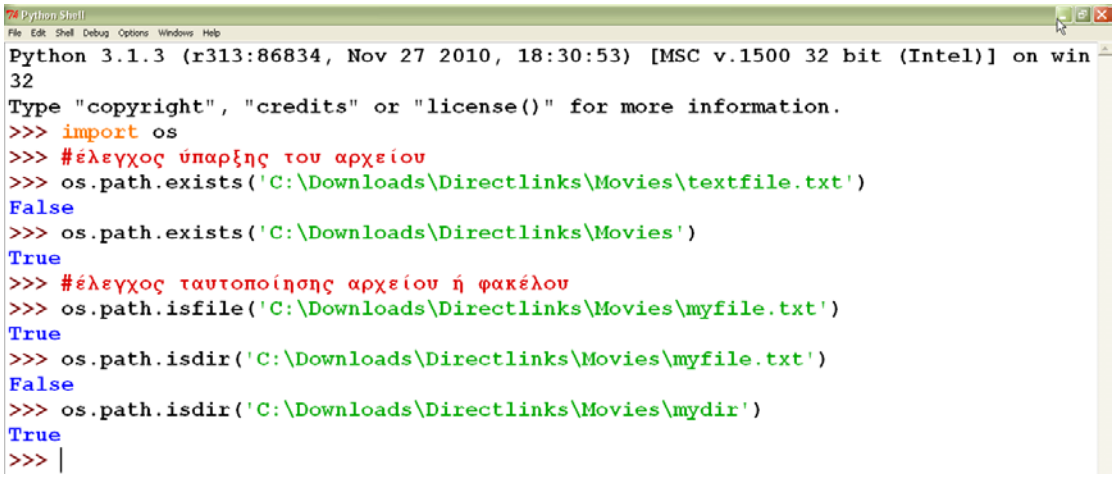


Εικόνα 8.3

os.path.exists(path)	επιστρέφει true αν η παράμετρός της είναι μία διαδρομή που αντιστοιχεί σε κάτι που υπάρχει στο σύστημα αρχείων
os.path.isfile(path)	επιστρέφει true αν και μόνο αν η διαδρομή που

	δίνεται προσδιορίζει ένα κανονικό αρχείο δεδομένων
os.path.isdir(path)	επιστρέφει true αν και μόνο αν η διαδρομή που δίνεται ως παράμετρος προσδιορίζει έναν κατάλογο
os.path.islink os.path.ismount	επιστρέφουν true αν μία διαδρομή προσδιορίζει ένα αρχείο που είναι ένας σύνδεσμος ή σημείο αναφοράς
os.path.samefile(path1, path2)	επιστρέφει true αν και μόνο αν οι δύο διαδρομές δείχνουν στο ίδιο αρχείο
os.path.isabs(path)	επιστρέφει true αν η παράμετρός της είναι μία απόλυτη διαδρομή
os.path.getsize(path) os.path.getmtime(path) os.path.getatime(path)	επιστρέφουν το μέγεθος, την ημερομηνία της τελευταίας τροποποίησης του αρχείου και την ημερομηνία της τελευταίας πρόσβασης του αρχείου αντίστοιχα

Πίνακας 8.2 Συναρτήσεις για άντληση πληροφοριών σχετικά με αρχεία και διαδρομές

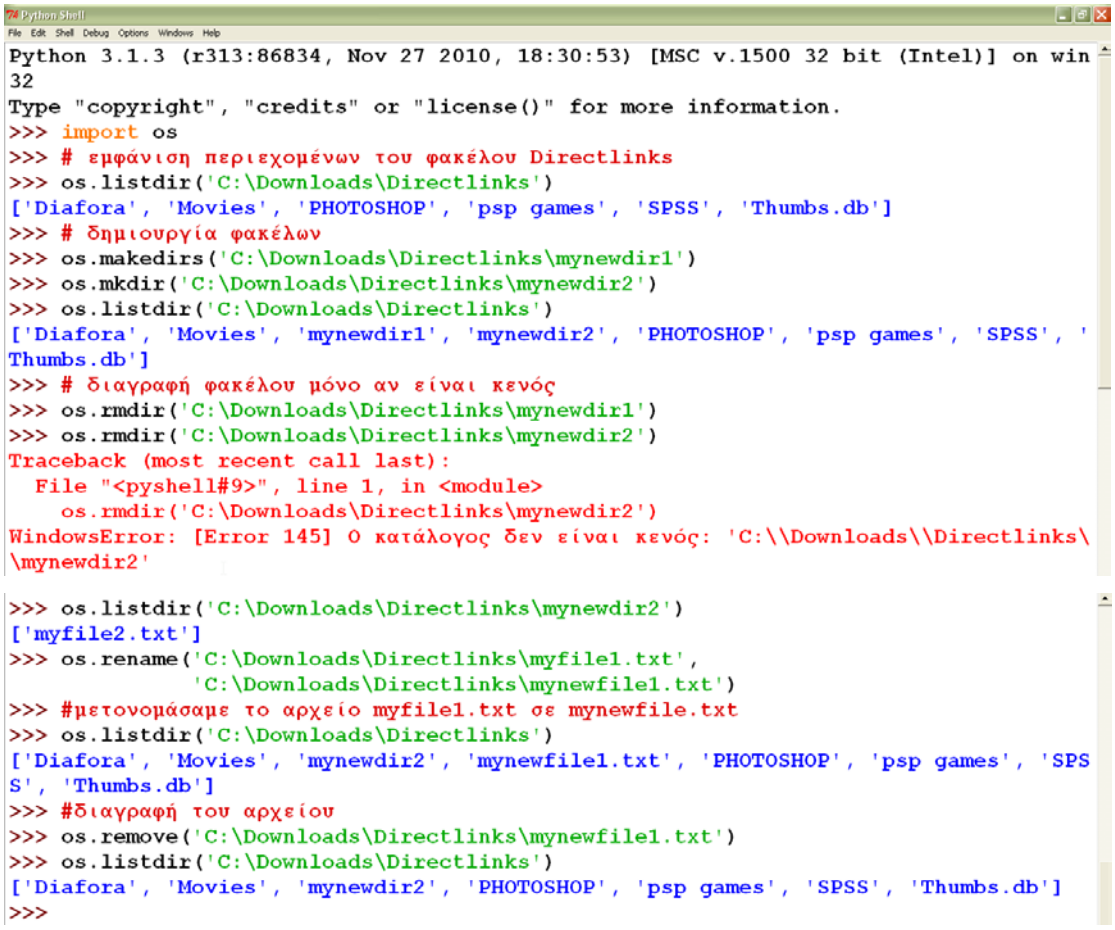


Εικόνα 8.4

Ο παρακάτω πίνακας παρουσιάζει συναρτήσεις του os module για περισσότερες λειτουργίες σχετικά με αρχεία και φακέλους και δύο επιπλέον module, το glob και shutil.

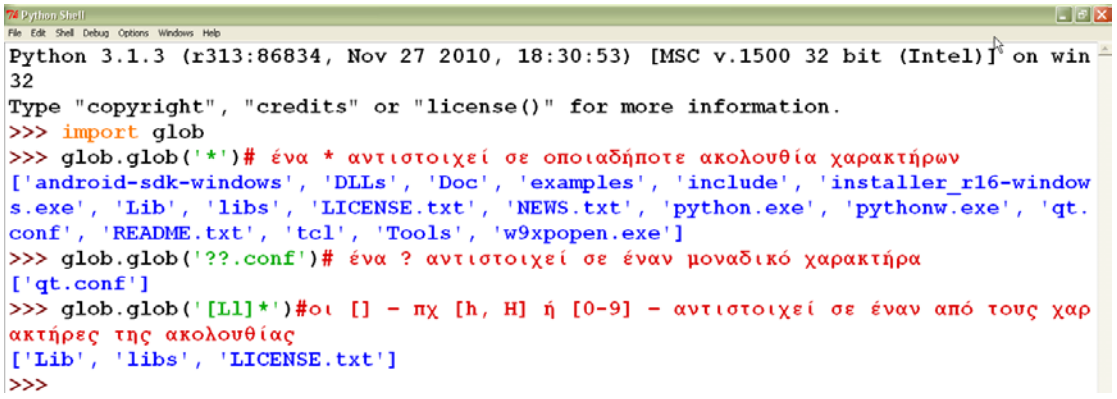
os.rename(old_path, new_path)	για να μετακινήσουμε ή να μετονομάσουμε ένα αρχείο ή κατάλογο
os.remove(path)	για να διαγράψουμε αρχεία δεδομένων καταλόγων (όχι καταλόγους)
os.makedirs(path) os.mkdir(path)	για να δημιουργήσουμε έναν φάκελο. Η διαφορά είναι ότι η os.mkdir δεν δημιουργεί οποιονδήποτε απαραίτητο ενδιάμεσο φάκελο σε αντίθεση με την os.makedir που το κάνει
os.rmdir(path)	για διαγραφή φακέλου. Μπορεί να διαγράψει μόνο άδειους φακέλους, διαφορετικά προκαλείται εξαίρεση
glob.glob(pattern)	επιστρέφει μία λίστα με τα περιεχόμενα του τρέχοντα καταλόγου εργασίας, που αντιστοιχούν στο pathname pattern
shutil.copytree	φτιάχνει αναδρομικά αντίγραφα όλων των αρχείων και υπό-φακέλων ενός φακέλου, διατηρώντας την κατάσταση πρόσβασης και της πληροφορίες κατάστασης.
shutil.rmtree	για διαγραφή μη κενών φακέλων. Διαγράφει αναδρομικά όλα τα αρχεία και τους υπό-φακέλους ενός φακέλου

Πίνακας 8.3 Περισσότερες λειτουργίες



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import os
>>> # εμφάνιση περιεχομένων του φακέλου Directlinks
>>> os.listdir('C:\Downloads\Directlinks')
['Diafora', 'Movies', 'PHOTOSHOP', 'psp games', 'SPSS', 'Thumbs.db']
>>> # δημιουργία φακέλων
>>> os.makedirs('C:\Downloads\Directlinks\mynewdir1')
>>> os.mkdir('C:\Downloads\Directlinks\mynewdir2')
>>> os.listdir('C:\Downloads\Directlinks')
['Diafora', 'Movies', 'mynewdir1', 'mynewdir2', 'PHOTOSHOP', 'psp games', 'SPSS', '
Thumbs.db']
>>> # διαγραφή φακέλου μόνο αν είναι κενός
>>> os.rmdir('C:\Downloads\Directlinks\mynewdir1')
>>> os.rmdir('C:\Downloads\Directlinks\mynewdir2')
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    os.rmdir('C:\Downloads\Directlinks\mynewdir2')
WindowsError: [Error 145] Ο κατάλογος δεν είναι κενός: 'C:\\Downloads\\Directlinks\
\mynewdir2'
>>> os.listdir('C:\Downloads\Directlinks\mynewdir2')
['myfile2.txt']
>>> os.rename('C:\Downloads\Directlinks\myfile1.txt',
              'C:\Downloads\Directlinks\mynewfile1.txt')
>>> #μετονομάσαμε το αρχείο myfile1.txt σε mynewfile.txt
>>> os.listdir('C:\Downloads\Directlinks')
['Diafora', 'Movies', 'mynewdir2', 'mynewfile1.txt', 'PHOTOSHOP', 'psp games', 'SPS
S', 'Thumbs.db']
>>> #διαγραφή του αρχείου
>>> os.remove('C:\Downloads\Directlinks\mynewfile1.txt')
>>> os.listdir('C:\Downloads\Directlinks')
['Diafora', 'Movies', 'mynewdir2', 'PHOTOSHOP', 'psp games', 'SPSS', 'Thumbs.db']
>>>
```

Εικόνα 8.5 Περισσότερες λειτουργίες του os module



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import glob
>>> glob.glob('*')# ένα * αντιστοιχεί σε οποιαδήποτε ακολουθία χαρακτήρων
['android-sdk-windows', 'DLLs', 'Doc', 'examples', 'include', 'installer_r16-window
s.exe', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe', 'pythonw.exe', 'qt.
conf', 'README.txt', 'tcl', 'Tools', 'w9xpopen.exe']
>>> glob.glob('??.conf')# ένα ? αντιστοιχεί σε έναν μοναδικό χαρακτήρα
['qt.conf']
>>> glob.glob('[Ll]*')#οι [] - πχ [h, H] ή [0-9] - αντιστοιχεί σε έναν από τους χαρ
ακτήρες της ακολουθίας
['Lib', 'libs', 'LICENSE.txt']
>>>
```

Εικόνα 8.6 glob και shutil module

Μία πολλή χρήσιμη συνάρτηση που χρησιμοποιούμε για να μετακινηθούμε μέσα σε ένα ολόκληρο δέντρο καταλόγων και δεν αναφέρεται παραπάνω είναι η **os.walk**. Η συνάρτηση αυτή επιστρέφει για κάθε κατάλογο που εντοπίζει, τη ρίζα ή

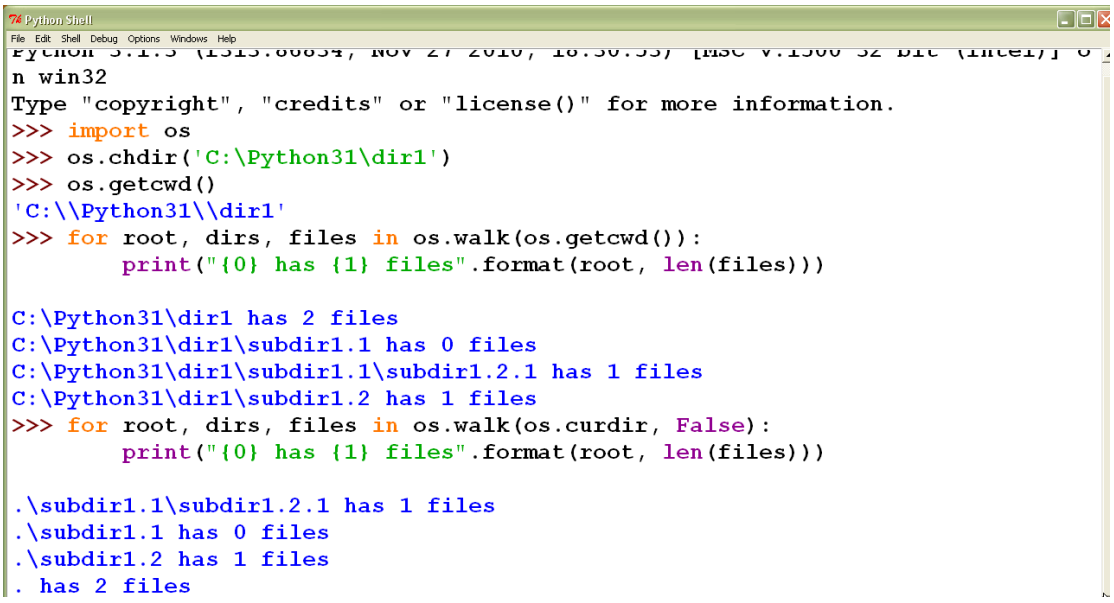
τη διαδρομή του φακέλου, μία λίστα με τους υποκαταλόγους του και μία λίστα με τα αρχεία του.

Δέχεται σαν παράμετρο τη διαδρομή του καταλόγου από όπου θα ξεκινήσει και επιπλέον τρεις προαιρετικές παραμέτρους :

os.walk(directory, topdown = True, onerror = None, followlinks = False)

Όταν η παράμετρος topdown είναι True(default τιμή), τα αρχεία σε κάθε φάκελο προσπελαύνονται πριν τους υποκαταλόγους του, ενώ όταν είναι false οι υπό-φάκελοι του κάθε καταλόγου προσπελαύνονται πρώτοι, δίνοντας μία άποψη του δέντρου από κάτω προς τα πάνω. Η παράμετρος onerror δίνει τη δυνατότητα στη συνάρτηση να διαχειρίζεται οποιαδήποτε λάθη προκύπτουν από την κλήση της os.listdir, που αγνοούνται από προεπιλογή. Η παράμετρος followlinks έχει default τιμή False που σημαίνει ότι η συνάρτηση os.walk δεν προχωράει σε φακέλους που αποτελούν symbolic links εκτός και αν θέσουμε τη παράμετρο στην τιμή True.

Για να καταλάβουμε την os.walk μπορούμε να περάσουμε επαναληπτικά πάνω από ένα δέντρο και να εκτυπώσουμε τις τιμές που επιστρέφονται για κάθε φάκελο:



Εικόνα 8.7 Χρήση της os.walk

8.3 Λειτουργίες πάνω σε αρχεία

8.3.1 Άνοιγμα και κλείσιμο αρχείου

Μπορούμε να ανοίξουμε ένα αρχείο με την ενσωματωμένη συνάρτηση open(). Η πρώτη παράμετρος της open είναι το όνομα του αρχείου που θέλουμε να ανοίξουμε, όταν πρόκειται για αρχείο που βρίσκεται στον τρέχον κατάλογο

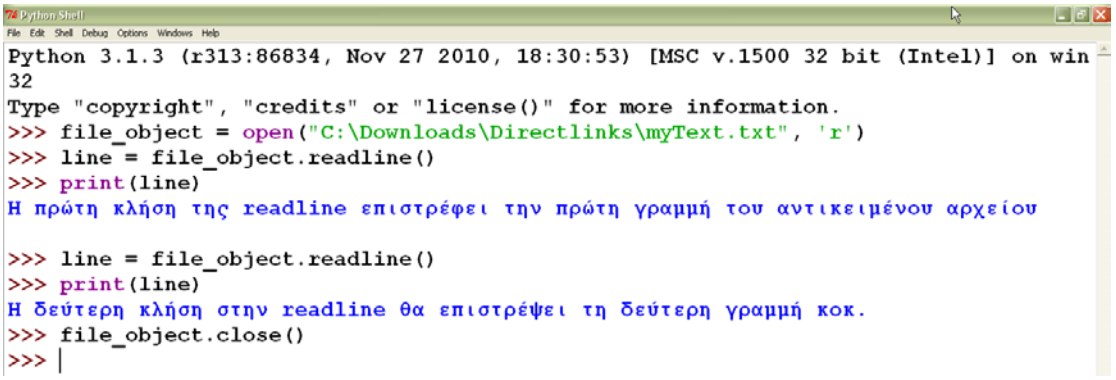
εργασίας ή η διαδρομή του αρχείου όταν είναι εκτός. Η δεύτερη παράμετρος είναι μία συμβολοσειρά που προσδιορίζει το λόγο που το αρχείο θα ανοίξει. Οι τιμές που μπορεί να πάρει είναι:

‘r’ - άνοιγμα του αρχείου για ανάγνωση (default τιμή)

‘w’ – άνοιγμα του αρχείου για εγγραφή

‘a’ – άνοιγμα του αρχείου για πρόσθεση δεδομένων στα ήδη υπάρχοντα στο αρχείο.

Ένα αρχείο που ανοίγει πρέπει και να κλείνει με τη συνάρτηση close.



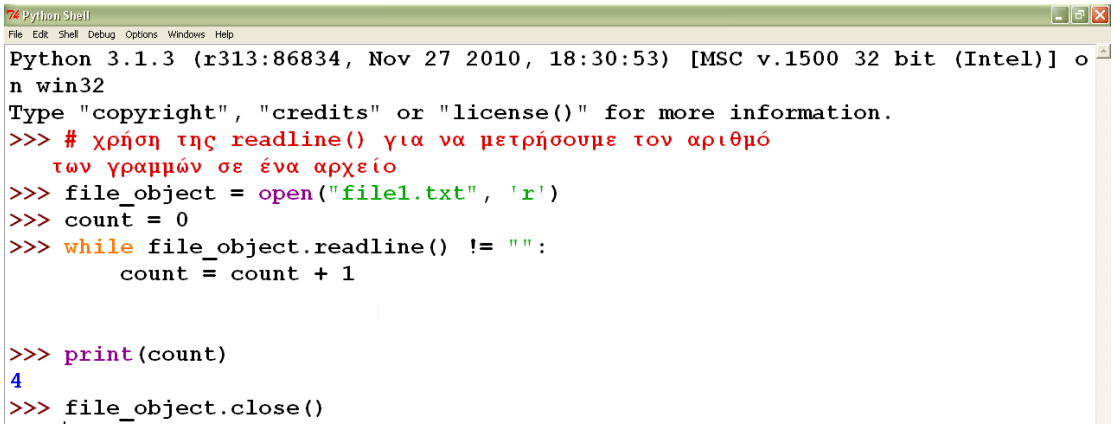
Εικόνα 8.8 Άνοιγμα και κλείσιμο αρχείου

8.3.2 Ανάγνωση από αρχείο

Υπάρχουν τρεις τρόποι για να διαβάσουμε δεδομένα από ένα αρχείο:

- Συνάρτηση readline()

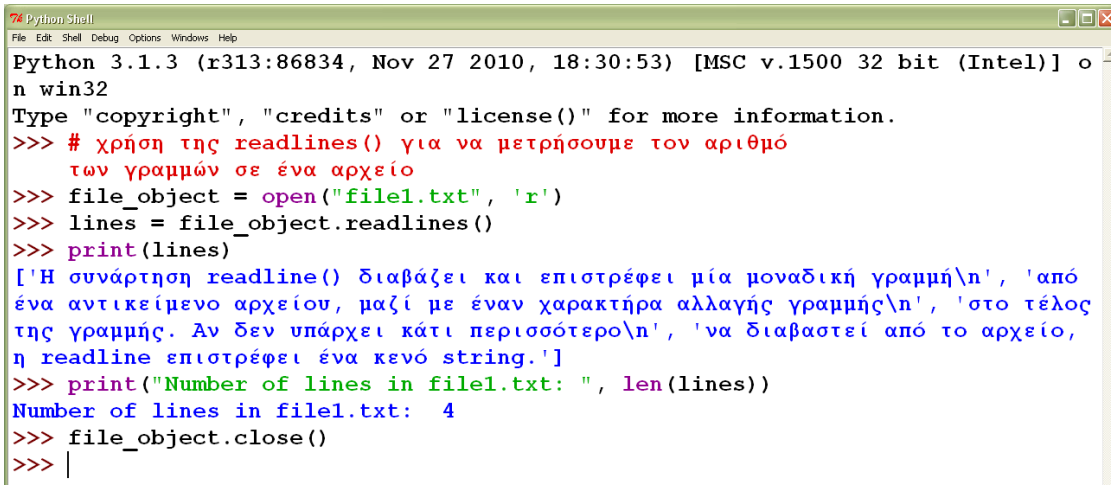
Η συνάρτηση readline() διαβάζει και επιστρέφει μία μοναδική γραμμή από ένα file object, μαζί με έναν χαρακτήρα αλλαγής γραμμής στο τέλος της γραμμής.



Εικόνα 8.9 Ανάγνωση από αρχείο με τη readline()

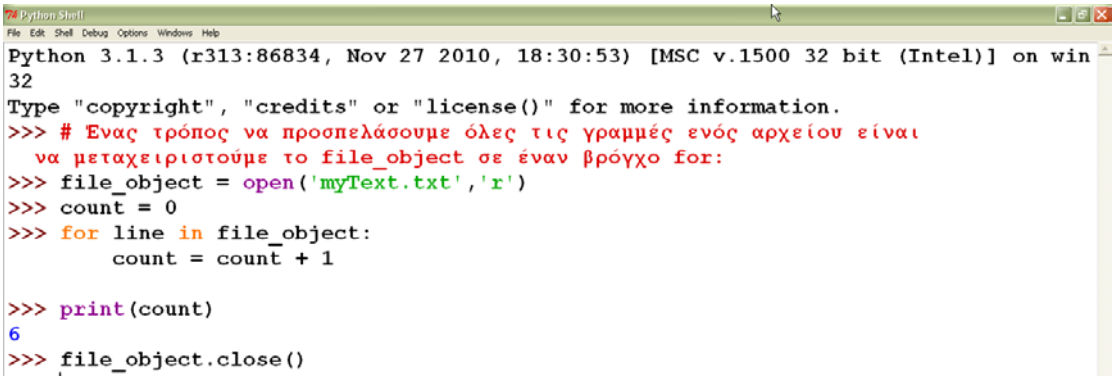
- Μέθοδος readlines()

Η μέθοδος readlines() διαβάζει όλες τις γραμμές ενός αρχείου και τις επιστρέφει σαν μία λίστα συμβολοσειρών, ένα string ανά γραμμή :



Εικόνα 8.10 Ανάγνωση από αρχείο με τη readlines()

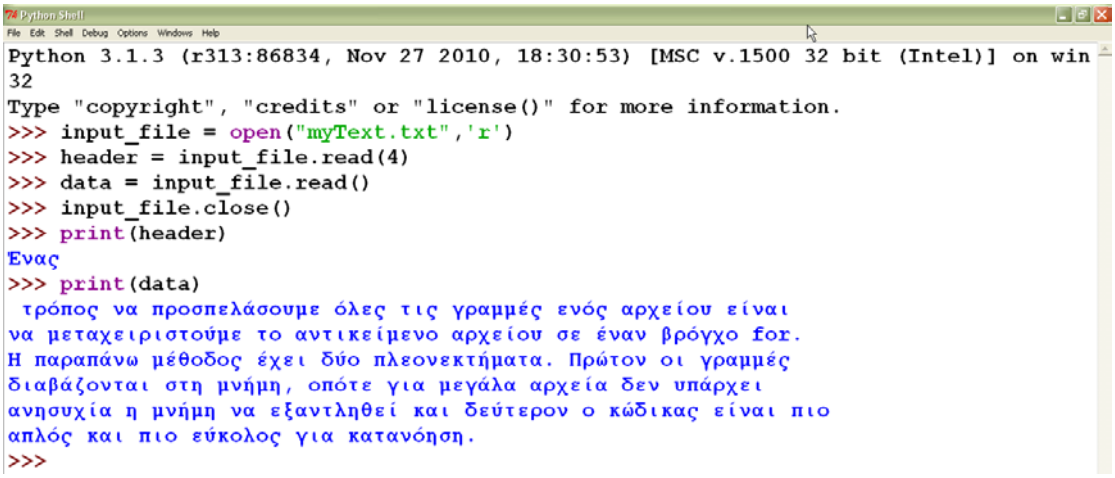
Ένας τρόπος να προσπελάσουμε όλες τις γραμμές ενός αρχείου είναι να μεταχειριστούμε το file_object σε έναν βρόγχο for. Αυτή η μέθοδος έχει δύο πλεονεκτήματα. Πρώτον οι γραμμές διαβάζονται στη μνήμη, οπότε για μεγάλα αρχεία δεν υπάρχει ανησυχία η μνήμη να εξαντληθεί. Δεύτερον ο κώδικας είναι πιο απλός και κατανοητός.



Εικόνα 8.11 Προσπέλαση αρχείου μέσω του file_object

- Μέθοδος read()

Με τη μέθοδο read() διαβάζουμε όλα τα δεδομένα ενός αρχείου και τα αποθηκεύουμε σε ένα αντικείμενο string. Η μέθοδος read() χωρίς παραμέτρους, διαβάζει όλο το αρχείο ενώ με μία ακέραια παράμετρο, διαβάζει τον αριθμό των δεδομένων που ορίζει η παράμετρος και τα επιστρέφει.

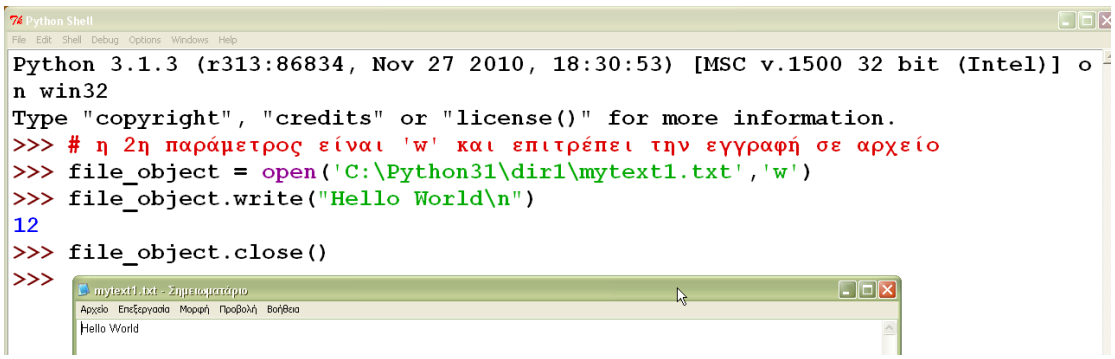


Εικόνα 8.12 Ανάγνωση αρχείου με τη μέθοδο read()

8.3.3 Εγγραφή σε αρχείο

- Μέθοδος write()

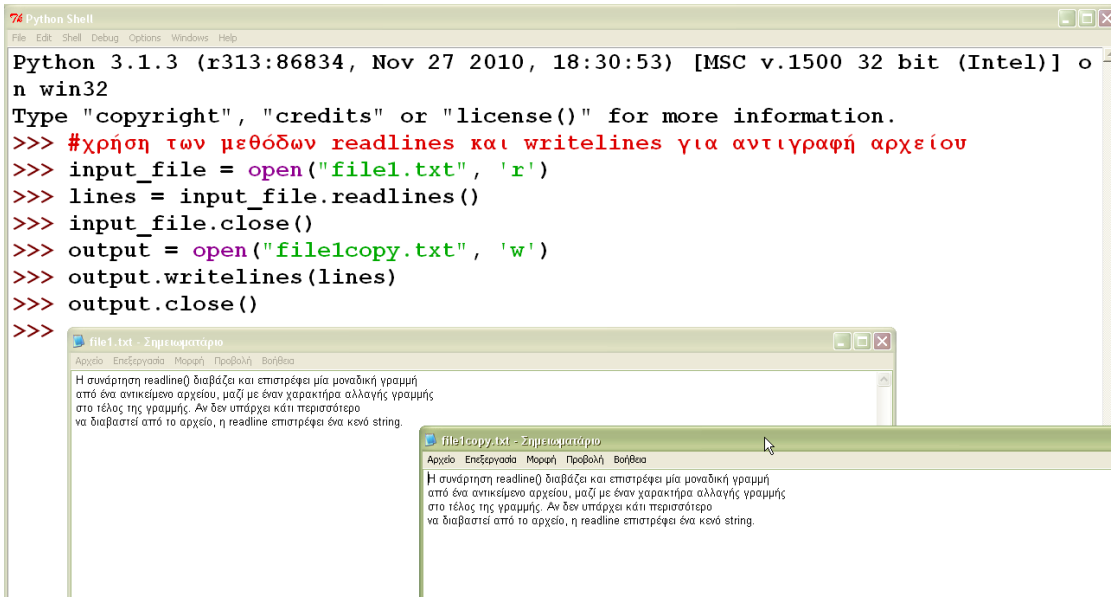
Η μέθοδος write() γράφει μία συμβολοσειρά σε ένα αρχείο. Η συμβολοσειρά πρέπει να περιλαμβάνει τους ειδικούς χαρακτήρες για να γίνει αλλαγή γραμμής.



Εικόνα 8.13 Εγγραφή σε αρχείο με τη μέθοδο write()

- Μέθοδος writelines()

Η μέθοδος writelines() παίρνει σαν παράμετρο μία λίστα συμβολοσειρών και τις γράφει την μία μετά την άλλη, στο δοθέν file object, χωρίς να γράφει νέες γραμμές. Αν οι συμβολοσειρές στη λίστα τελειώνουν με χαρακτήρα αλλαγής γραμμής, γράφονται σαν νέες γραμμές, διαφορετικά συγχωνεύονται αποδοτικά στο αρχείο. Η writelines είναι μία ακριβής αντιστροφή της readlines και σε συνδυασμό μπορούν να χρησιμοποιηθούν για να δημιουργήσουν το αντίγραφο ενός αρχείου:

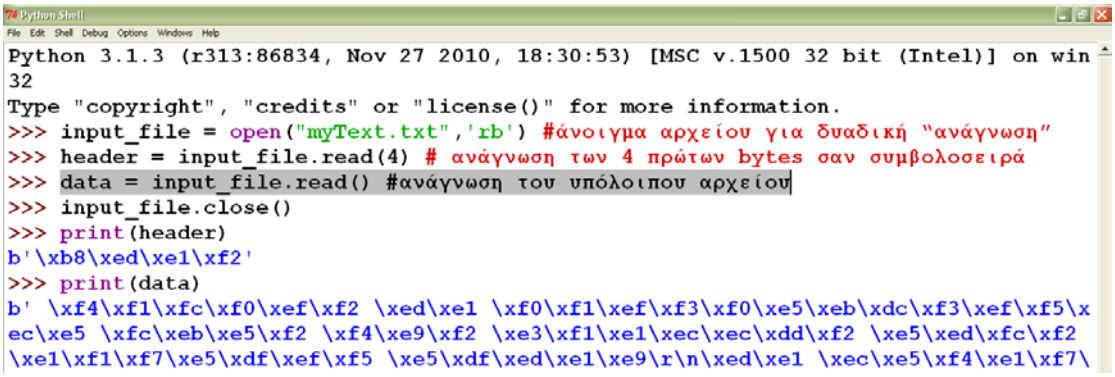


Εικόνα 8.14 Αντιγραφή αρχείου

8.3.4 Άνοιγμα αρχείων σε δυαδική μορφή

Για να ανοίξουμε ένα αρχείο σε δυαδική μορφή προσθέτουμε το χαρακτήρα ‘b’ στη δεύτερη παράμετρο (πχ open(“file”, ‘rb’) ή (“file”, ‘wb’)).

Έτσι μπορούμε να προσπελάσουμε τα δεδομένα σαν byte ακολουθία, χωρίς μετάφραση ή κωδικοποίηση κειμένου, αφού η open() θα επιστρέψει ένα αντικείμενο byte αντί για string όταν διαβάσει το αρχείο.



Εικόνα 8.15 Άνοιγμα αρχείου σε δυαδική μορφή

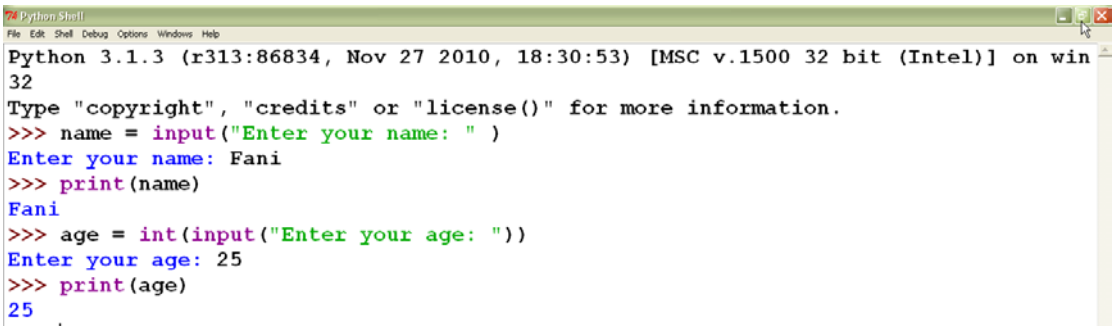
Τα αρχεία που ανοίγουν σε δυαδική μορφή αντιμετωπίζονται μόνο με bytes και όχι με strings. Για να χρησιμοποιήσουμε τα δεδομένα σαν συμβολοσειρές, πρέπει να αποκωδικοποιήσουμε τα αντικείμενα bytes σε αντικείμενα string. Αυτό είναι χρήσιμο όταν έχουμε να κάνουμε με πρωτόκολλα δικτύου, όπου η ροές δεδομένων εμφανίζονται σαν αρχεία σε δυαδική μορφή.

8.4 Είσοδος - έξοδος και ανακατεύθυνση

Για να διαβάσουμε είσοδο από το πληκτρολόγιο και να εμφανίσουμε τη συμβολοσειρά στην οθόνη χρησιμοποιούμε την ενσωματωμένη συνάρτηση input().

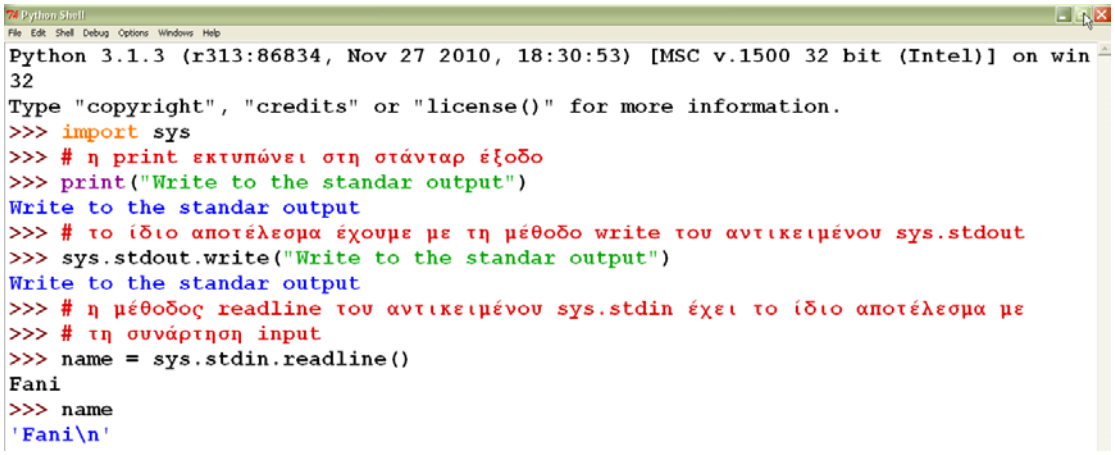
Για να διαβάσουμε αριθμούς με χρήση της input, χρειάζεται να μετατρέψουμε το string που επιστρέφει η input, στον σωστό αριθμητικό τύπο.

Η input διαβάζει από τη στάνταρ είσοδο και γράφει στη στάνταρ έξοδο.



Εικόνα 8.16

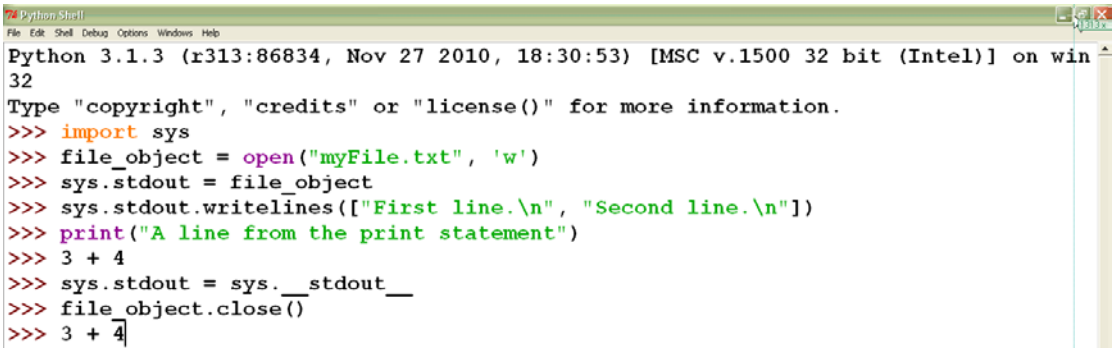
Επιπλέον μπορούμε να χειριστούμε την είσοδο και έξοδο σαν αντικείμενα. Το sys module παρέχει τις ιδιότητες sys.stdin, sys.stdout και sys.stderr τις οποίες χειριζόμαστε σαν ειδικά file_objects με τις δικές τους μεθόδους. Για τη sys.stdin έχουμε τις μεθόδους read, readline, readlines. Για την sys.stdout και sys.stderr μπορούμε να χρησιμοποιήσουμε την συνάρτηση print καθώς και τις write και writellines, που λειτουργούν όπως ακριβώς και με τα άλλα αρχεία :



Εικόνα 8.17

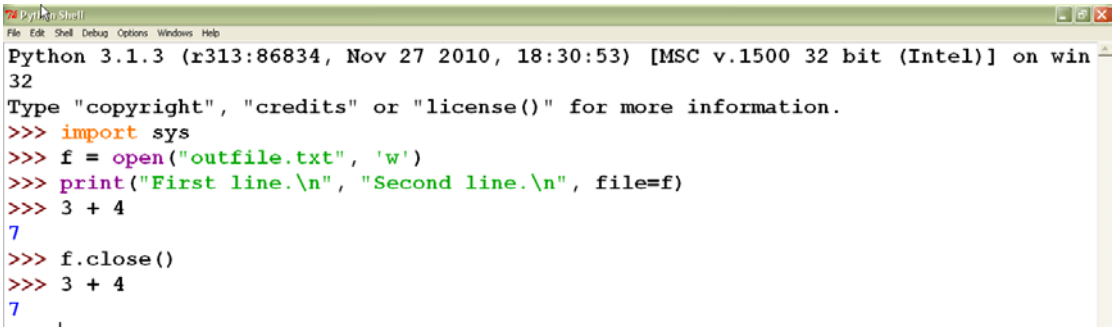
Μπορούμε να ανακατευθύνουμε τη στάνταρ είσοδο ώστε να διαβάσει από ένα αρχείο. Αντίστοιχα η έξοδος ή τα λάθη μπορούν να γραφτούν σε ένα αρχείο.

Μπορούμε ακόμη να αποθηκεύσουμε τα δεδομένα στις αρχικές τους τιμές με χρήση των sys.__stdin__, sys.__stdout__, and sys.__stderr__:



Παράδειγμα 8.18

Η συνάρτηση print μπορεί επίσης να ανακατευθυνθεί σε οποιοδήποτε αρχείο χωρίς να αλλάξουμε την κανονική έξοδο:



Παράδειγμα 8.19

9. Python και αντικειμενοστραφής προγραμματισμός

9.1 Βασικές έννοιες

9.1.1 Αντικείμενα και κλάσεις

Οτιδήποτε στη Python είναι ένα αντικείμενο και μπορούμε να πάρουμε τον τύπο του με τη συνάρτηση type(). Η κλάση ορίζει έναν νέο τύπο δεδομένων με τη λέξη-κλειδί class.

```
class ClassName:
        body
```

Το σώμα είναι μία λίστα με Python εντολές, αναθέσεις τιμών και ορισμό συναρτήσεων. Το σώμα μπορεί να είναι κενό αλλά τότε πρέπει να βάλουμε την εντολή pass. Τα ονόματα των κλάσεων είναι στη μορφή CapCase.
Μετά τον προσδιορισμό της κλάσης μπορούμε να δημιουργήσουμε ένα αντικείμενο τύπου κλάσης που ονομάζεται στιγμιότυπο (instance) και δημιουργείται καλώντας το όνομα της κλάσης σαν συνάρτηση:
instance = className()

Η ιδιότητα __class_ που είναι διαθέσιμη για όλα τα Python στιγμιότυπα, επιστρέφει την κλάση στην οποία το στιγμιότυπο ανήκει. Για παράδειγμα :

```
>>> Circle
<class '__main__.Circle'>
>>> c.__class__
<class '__main__.Circle'>
```

Η κλάση με όνομα Circle αναπαρίσταται εσωτερικά από μία αφηρημένη δομή δεδομένων η οποία είναι ένα στιγμιότυπο της κλάσης Circle. Αυτό μας επιτρέπει να αναφερθούμε στην τιμή της Circle.pi χωρίς να αναφερθούμε ρητά στο όνομα της κλάσης Circle.

```
>>> c.__class__.pi
3.1415899999999999
```

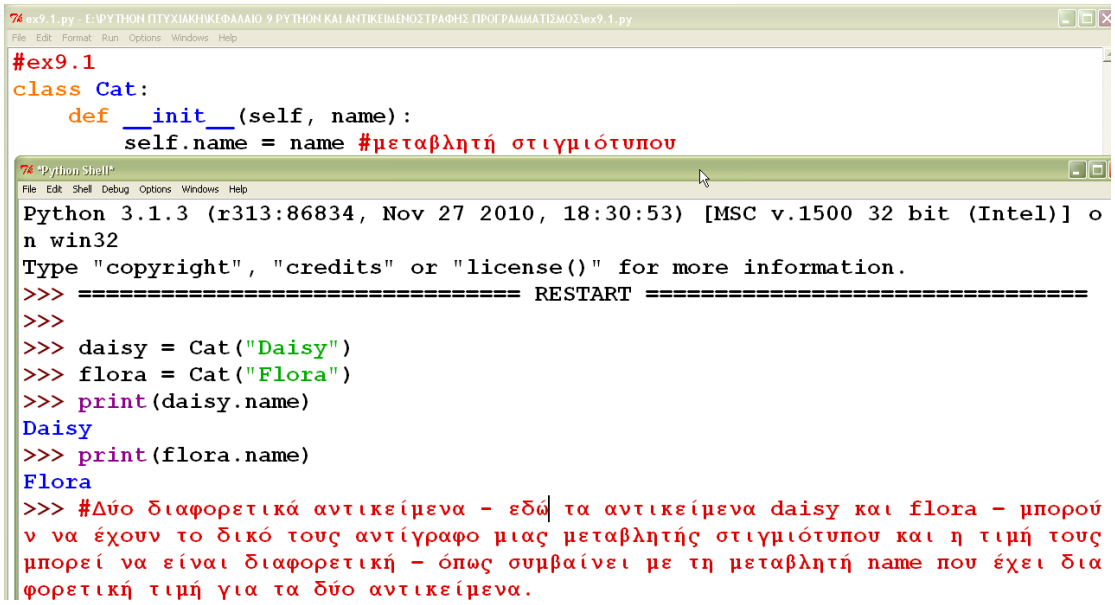
Μέσα σε μία κλάση ορίζουμε μεταβλητές και μεθόδους.

9.1.2 Μεταβλητές

Οι μεταβλητές χωρίζονται σε δύο κατηγορίες:

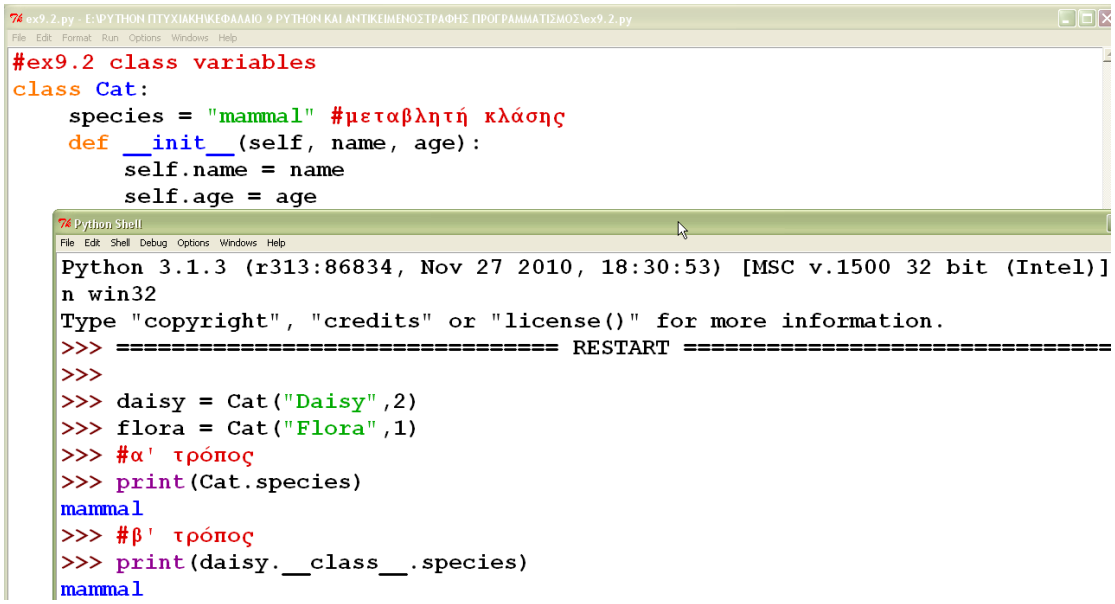
α. **Μεταβλητές στιγμιότυπου** (instance variables): είναι οι μεταβλητές που ανήκουν στο αντικείμενο και μπορούμε να τις αρχικοποιήσουμε με χρήση της μεθόδου __init__( ). Η μέθοδος __init( )__ παίρνει σαν πρώτη παράμετρο μία αναφορά στο αντικείμενο που από σύμβαση ονομάζεται self και μετά ορίζουμε τις υπόλοιπες ιδιότητες. Η μέθοδος __init( )__ τρέχει κάθε φορά που δημιουργείται ένα στιγμιότυπο της κλάσης και είναι παρόμοια με έναν δομητή στη Java αλλά αυτό που κάνει είναι να αρχικοποιεί τις μεταβλητές του αντικείμενου, όχι να τις δομεί.

Η αναφορά σε μία μεταβλητή στιγμιότυπου γίνεται μέσω του αντικειμένου διαφορετικά πρόκειται για μία τοπική μεταβλητή. Αυτό είναι διαφορετικό στη C++ και Java όπου οι μεταβλητές στιγμιότυπου αναφέρονται μόνο με το όνομά τους.



Παράδειγμα 9.1

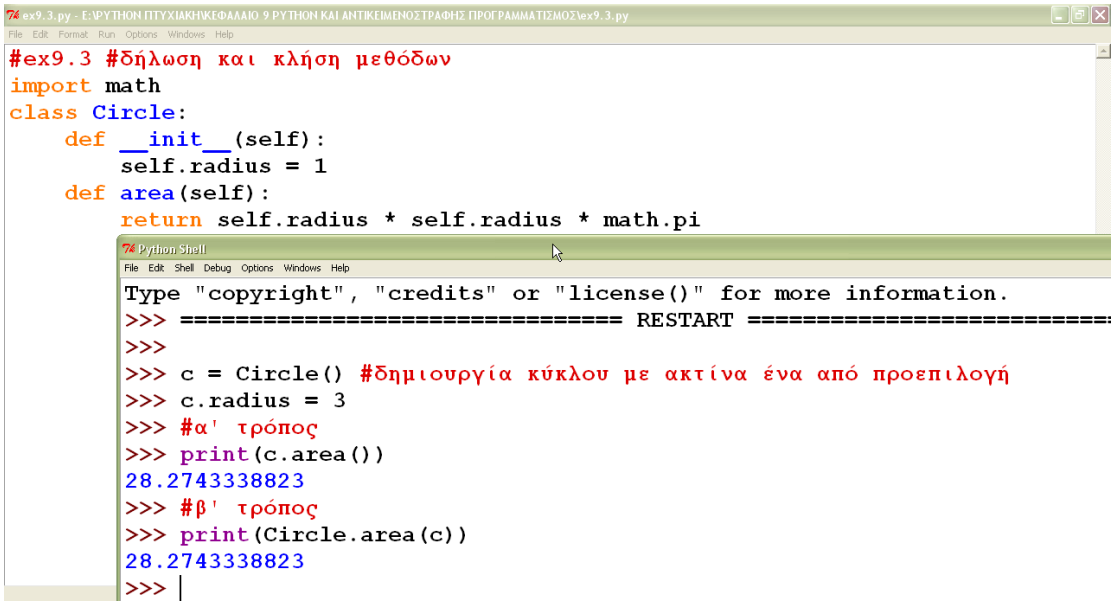
β. **Μεταβλητές κλάσης** (class variables): οι μεταβλητές κλάσης ορίζονται μέσα στο σώμα της κλάσης – όχι στη συνάρτηση __init__( ) - και είναι ίδιες για όλα τα στιγμιότυπα της κλάσης, δηλαδή μπορούν να τις χρησιμοποιήσουν όλα τα στιγμιότυπα και προσπελαύνονται μέσω του ονόματος της κλάσης ή μέσω της ιδιότητας _class_.



Παράδειγμα 9.2

9.1.3 Μέθοδοι

Μία μέθοδος είναι μία συνάρτηση που ορίζεται μέσα στο σώμα μίας κλάσης. Οι μέθοδοι καλούνται για στιγμιότυπα της κλάσης. Η πρώτη παράμετρος μίας μεθόδου είναι πάντα το αντικείμενο – self – για το οποίο καλείται. Είναι ιδιαίτερα χρήσιμες στην ενθυλάκωση.



Παράδειγμα 9.3

Οι μέθοδοι μπορούν να κληθούν με παραμέτρους, αν στον ορισμό της η συνάρτηση δέχεται αυτές τις παραμέτρους. Στο παράδειγμα αυτό η μέθοδος

__init__() δέχεται μία επιπλέον παράμετρο ώστε να μπορούμε να δημιουργήσουμε κύκλους χωρίς να χρειάζεται να θέσουμε μία τιμή στη μεταβλητή radius αλλά να χρησιμοποιείται η προεπιλεγμένη τιμή της 1.

Παραπάνω βλέπουμε ότι υπάρχουν δύο τρόποι κλήσεις μίας μεθόδου :

α. Bounded οπότε η μέθοδος καλείται μέσω του αντικειμένου – c.area().

b. Unbounded οπότε η μέθοδος καλείται μέσω της κλάσης που περιέχει τη μέθοδο και το στιγμιότυπο δίνεται ρητά σαν παράμετρος στη μέθοδο - Circle.area(c). Ο τρόπος αυτός είναι πιο σπάνιος.

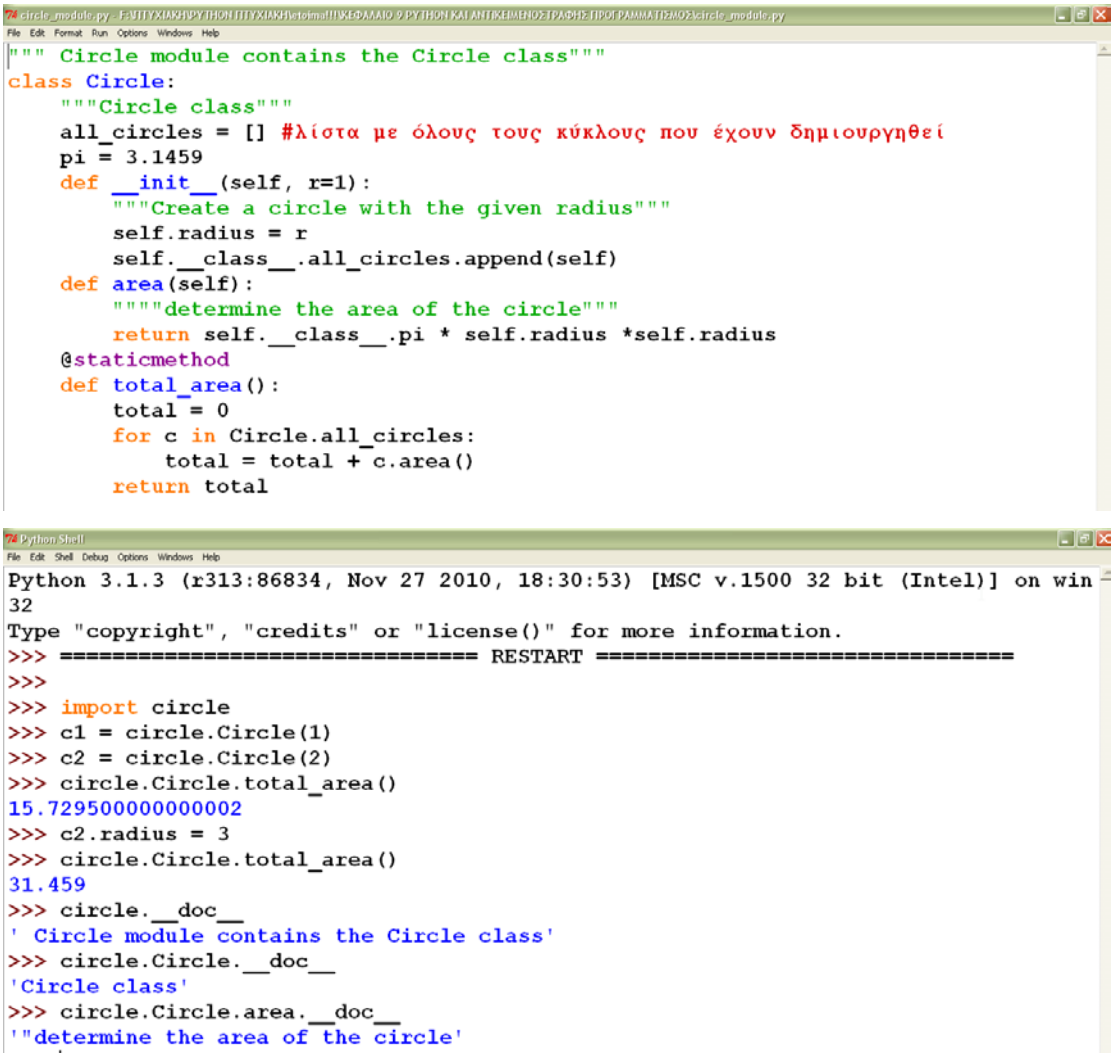
Δεν υπάρχει κάτι μαγικό στην κλήση μεθόδων στην Python. Όταν γίνεται κλήση συνάρτησης με τη μορφή instance.method(arg1, arg2, …), η Python τη μετατρέπει σε κανονική συνάρτηση χρησιμοποιώντας τους ακόλουθους κανόνες:

1. Ψάχνει για το όνομα της μεθόδου στο χώρο ονομάτων των στιγμιότυπων. Αν μία μέθοδος έχει αλλάξει ή έχει προστεθεί για αυτό το στιγμιότυπο, καλείται κατά προτίμηση πάνω από τις μεθόδους στην κλάση ή στην υπέρ-κλάση.
2. Αν η μέθοδος δεν βρεθεί στο χώρο ονομάτων , ψάχνει στον τύπο κλάσης class του στιγμιότυπου και ψάχνει εκεί για τη μέθοδο. Στο προηγούμενο παράδειγμα η class είναι η Circle – ο τύπος του στιγμιότυπου c.
3. Αν η μέθοδος πάλι δε βρεθεί, ψάχνει για τη μέθοδο στις superclasses.
4. Όταν η μέθοδος βρεθεί, γίνεται μία κλήση σε αυτή χρησιμοποιώντας το στιγμιότυπο σαν πρώτη παράμετρο της συνάρτησης και μετακινεί όλες τις υπόλοιπες παραμέτρους στην κλήση της μεθόδου κατά μία θέση δεξιά. Οπότε η κλήση instance.method(arg1,arg2, . . .) γίνεται class.method(instance, arg1, arg2, …).

9.1.3.1 Στατικές Μέθοδοι

Οι Python κλάσεις έχουν και αυτές μεθόδους αντίστοιχες με τις static μεθόδους της Java.

Όπως ακριβώς στη Java, έτσι και στη Python μπορούμε να καλέσουμε στατικές μεθόδους μέσω ενός στιγμιότυπου κλάσης, ακόμα και αν κανένα στιγμιότυπο της κλάσης δεν έχει δημιουργηθεί. Για να δημιουργήσουμε μία στατική μέθοδο στη Python χρησιμοποιούμε τον προσδιοριστή @staticmethod, ο οποίος χρησιμοποιείται πριν τον ορισμό μιας συνάρτησης.



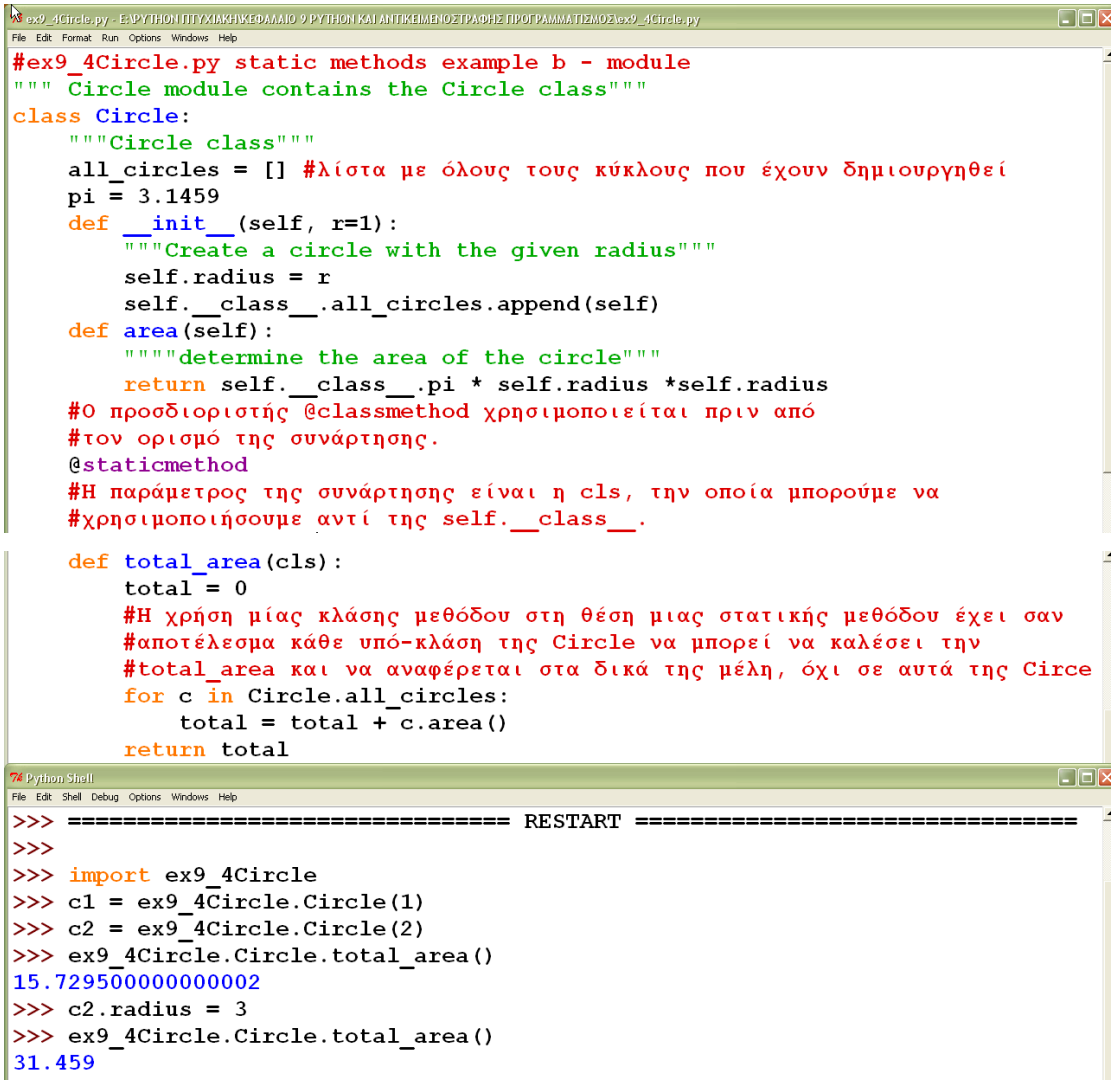
```
""" Circle module contains the Circle class"""
class Circle:
    """Circle class"""
    all_circles = [] #λίστα με όλους τους κύκλους που έχουν δημιουργηθεί
    pi = 3.1459
    def __init__(self, r=1):
        """Create a circle with the given radius"""
        self.radius = r
        self.__class__.all_circles.append(self)
    def area(self):
        """determine the area of the circle"""
        return self.__class__.pi * self.radius *self.radius
    @staticmethod
    def total_area():
        total = 0
        for c in Circle.all_circles:
            total = total + c.area()
        return total
```

```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>> import circle
>>> c1 = circle.Circle(1)
>>> c2 = circle.Circle(2)
>>> circle.Circle.total_area()
15.729500000000002
>>> c2.radius = 3
>>> circle.Circle.total_area()
31.459
>>> circle.__doc__
' Circle module contains the Circle class'
>>> circle.Circle.__doc__
'Circle class'
>>> circle.Circle.area.__doc__
'"determine the area of the circle'
```

Εικόνα 9.1

9.1.3.2 Κλάσεις Μεθόδων

Οι κλάσεις μεθόδων είναι παρόμοιες με τις στατικές μεθόδους από την άποψη ότι μπορούμε να τις καλέσουμε πριν τη δημιουργία ενός στιγμιότυπου της κλάσης ή χρησιμοποιώντας το στιγμιότυπο μιας κλάσης. Η διαφορά είναι ότι οι κλάσεις των μεθόδων περνούν ρητά σαν πρώτη παράμετρο την κλάση, στην οποία ανήκουν, οπότε μπορούμε να τις κωδικοποιήσουμε πιο απλά.



```
#ex9_4Circle.py static methods example b - module
""" Circle module contains the Circle class"""
class Circle:
    """Circle class"""
    all_circles = [] #λίστα με όλους τους κύκλους που έχουν δημιουργηθεί
    pi = 3.1459
    def __init__(self, r=1):
        """Create a circle with the given radius"""
        self.radius = r
        self.__class__.all_circles.append(self)
    def area(self):
        """determine the area of the circle"""
        return self.__class__.pi * self.radius *self.radius
    #Ο προσδιοριστής @classmethod χρησιμοποιείται πριν από
    #τον ορισμό της συνάρτησης.
    @staticmethod
    #Η παράμετρος της συνάρτησης είναι η cls, την οποία μπορούμε να
    #χρησιμοποιήσουμε αντί της self.__class__.
    def total_area(cls):
        total = 0
        #Η χρήση μίας κλάσης μεθόδου στη θέση μιας στατικής μεθόδου έχει σαν
        #αποτέλεσμα κάθε υπό-κλάση της Circle να μπορεί να καλέσει την
        #total_area και να αναφέρεται στα δικά της μέλη, όχι σε αυτά της Circe
        for c in Circle.all_circles:
            total = total + c.area()
        return total
```

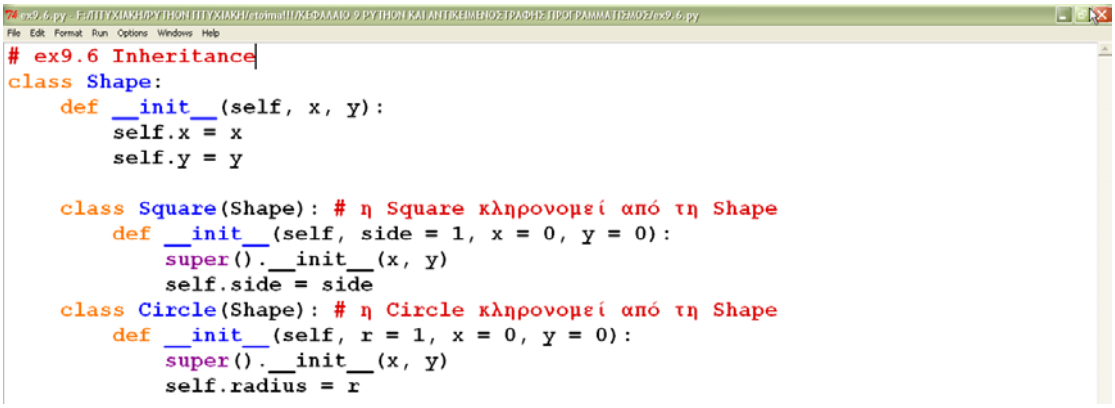
```
>>> ================================ RESTART ================================
>>>
>>> import ex9_4Circle
>>> c1 = ex9_4Circle.Circle(1)
>>> c2 = ex9_4Circle.Circle(2)
>>> ex9_4Circle.Circle.total_area()
15.729500000000002
>>> c2.radius = 3
>>> ex9_4Circle.Circle.total_area()
31.459
```

Παράδειγμα 9.4

9.2 Κληρονομικότητα

Η κληρονομικότητα στη Python είναι πιο εύκολη και περισσότερο ευέλικτη από ότι η κληρονομικότητα στις compiled γλώσσες προγραμματισμού όπως η Java και η C++. Αυτό συμβαίνει επειδή η δυναμική φύση της Python δεν επιβάλλει πολλούς περιορισμούς στη γλώσσα. Η κληρονομικότητα είναι ένας τρόπος να σχηματίζουμε νέες κλάσεις που χρησιμοποιούν κλάσεις που έχουν ήδη οριστεί. Οι νέες κλάσεις ονομάζονται derived classes και υπερκαλύπτουν ή επεκτείνουν τη λειτουργικότητα των κλάσεων που έχουν ήδη οριστεί και ονομάζονται base classes.

Τα πλεονεκτήματα που παρουσιάζει η κληρονομικότητα είναι η επαναχρησιμοποίηση κώδικα και η μείωση της πολυπλοκότητας του προγράμματος.



```
# ex9.6 Inheritance
class Shape:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    class Square(Shape): # η Square κληρονομεί από τη Shape
        def __init__(self, side = 1, x = 0, y = 0):
            super().__init__(x, y)
            self.side = side
    class Circle(Shape): # η Circle κληρονομεί από τη Shape
        def __init__(self, r = 1, x = 0, y = 0):
            super().__init__(x, y)
            self.radius = r
```

Παράδειγμα 9.6 Κληρονομικότητα

Δύο σημεία που πρέπει να προσέξουμε είναι ότι:

1. Πρέπει να δώσουμε σαν παράμετρο στην κλάση το όνομα της κλάσης από την οποία θα κληρονομήσει στοιχεία. Στο παραπάνω παράδειγμα η Circle και η Square κληρονομούν από τη Shape.

2. Οι κλάσεις πρέπει να κάνουν ρητή κλήση της μεθόδου __init__ των κλάσεων από τις οποίες κληρονομούν. Για να γίνει αυτό στη Python πρέπει να χρησιμοποιήσουμε τη συνάρτηση super που ξεκαθαρίζει ποια κληρονομούμενη κλάση να χρησιμοποιήσει. Αντί της super, μπορούμε να καλέσουμε την μέθοδο __init__ της κλάσης Shape: Shape.__init__(self, x, y) η οποία θα καλέσει τη συνάρτηση αρχικοποίησης Shape με το στιγμιότυπο να είναι αρχικοποιημένο.

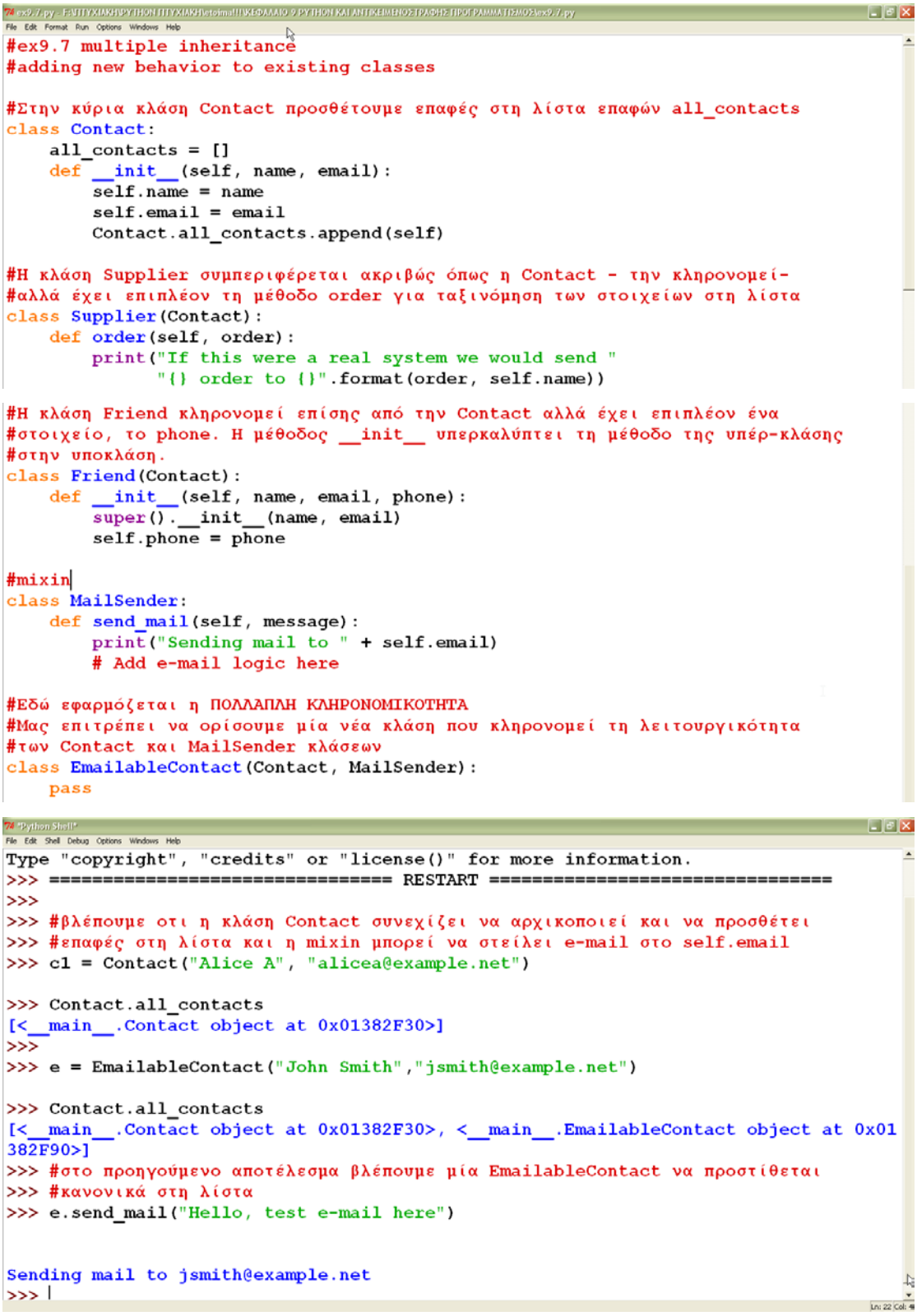
9.2.1 Πολλαπλή Κληρονομικότητα

Πολλαπλή κληρονομικότητα είναι η δυνατότητα μίας υποκλάσης να κληρονομεί στοιχεία από περισσότερες από μία γονικές κλάσεις και να έχει πρόσβαση στη λειτουργικότητα και των δύο. Ο κώδικας έχει τη μορφή:

class NN(class1, class2, class3, …)

Η απλούστερη και περισσότερο χρήσιμη μορφή της πολλαπλής κληρονομικότητας καλείται mixin. Σύμφωνα με αυτή μία υπέρ-κλάση δεν μπορεί να υπάρχει από μόνη της αλλά πρέπει να κληρονομείται από μία άλλη κλάση για να παρέχει επιπρόσθετη λειτουργικότητα. Στο παρακάτω παράδειγμα θεωρούμε την κλάση Contact που μας επιτρέπει την αποστολή e-mail. Θέλουμε να της προσθέσουμε λειτουργικότητα που επιτρέπει αποστολή e-mail στο self.email. Επειδή η αποστολή e-mail είναι συνηθισμένη λειτουργία και χρησιμοποιείται σε

περισσότερες από μία κλάσεις μπορούμε να γράψουμε μία απλή mixin κλάση που θα κάνει το e-mailing για μας:



```
#ex9.7 multiple inheritance
#adding new behavior to existing classes

#Στην κύρια κλάση Contact προσθέτουμε επαφές στη λίστα επαφών all_contacts
class Contact:
    all_contacts = []
    def __init__(self, name, email):
        self.name = name
        self.email = email
        Contact.all_contacts.append(self)

#Η κλάση Supplier συμπεριφέρεται ακριβώς όπως η Contact - την κληρονομεί-
#αλλά έχει επιπλέον τη μέθοδο order για ταξινόμηση των στοιχείων στη λίστα
class Supplier(Contact):
    def order(self, order):
        print("If this were a real system we would send "
              "{} order to {}".format(order, self.name))

#Η κλάση Friend κληρονομεί επίσης από την Contact αλλά έχει επιπλέον ένα
#στοιχείο, το phone. Η μέθοδος __init__ υπερκαλύπτει τη μέθοδο της υπέρ-κλάσης
#στην υποκλάση.
class Friend(Contact):
    def __init__(self, name, email, phone):
        super().__init__(name, email)
        self.phone = phone

#mixin
class MailSender:
    def send_mail(self, message):
        print("Sending mail to " + self.email)
        # Add e-mail logic here

#Εδώ εφαρμόζεται η ΠΟΛΛΑΠΛΗ ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ
#Μας επιτρέπει να ορίσουμε μία νέα κλάση που κληρονομεί τη λειτουργικότητα
#των Contact και MailSender κλάσεων
class EmailableContact(Contact, MailSender):
    pass
```

```
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>> #βλέπουμε οτι η κλάση Contact συνεχίζει να αρχικοποιεί και να προσθέτει
>>> #επαφές στη λίστα και η mixin μπορεί να στείλει e-mail στο self.email
>>> c1 = Contact("Alice A", "alicea@example.net")

>>> Contact.all_contacts
[<__main__.Contact object at 0x01382F30>]
>>>
>>> e = EmailableContact("John Smith","jsmith@example.net")

>>> Contact.all_contacts
[<__main__.Contact object at 0x01382F30>, <__main__.EmailableContact object at 0x01
382F90>]
>>> #στο προηγούμενο αποτέλεσμα βλέπουμε μία EmailableContact να προστίθεται
>>> #κανονικά στη λίστα
>>> e.send_mail("Hello, test e-mail here")


Sending mail to jsmith@example.net
>>>
```

Παράδειγμα 9.7 Πολλαπλή Κληρονομικότητα

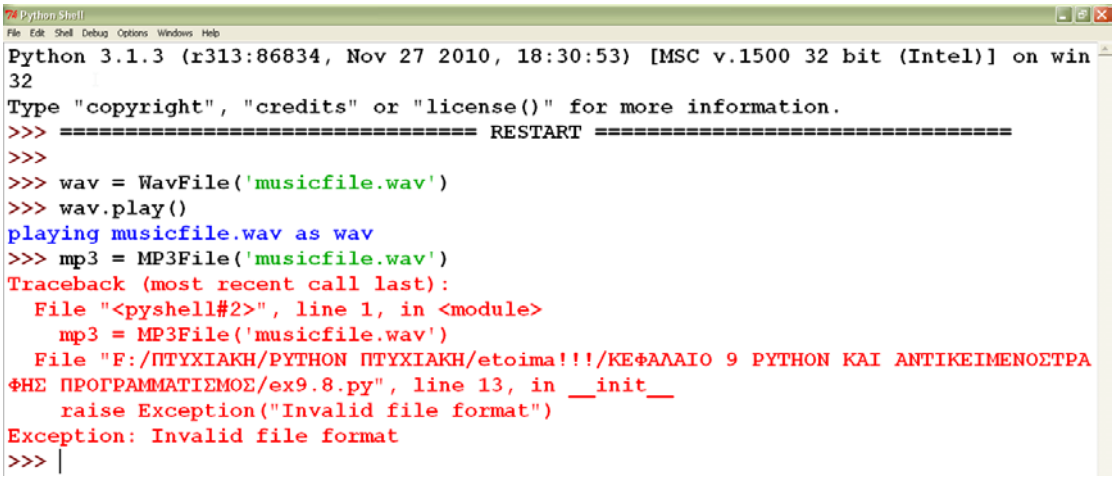
9.3 Πολυμορφισμός

Ο πολυμορφισμός αναφέρεται στη δυνατότητα που έχει ένα αντικείμενο να ανήκει σε διαφορετικούς τύπους ώστε να μπορεί να χρησιμοποιεί τις μεθόδους που αναφέρονται σε αυτούς. Είναι δηλαδή η διαδικασία χρήσης μίας μεθόδου με διαφορετικούς τρόπους για διαφορετική είσοδο δεδομένων. Ο πολυμορφισμός χρησιμοποιείται σε συνδυασμό με την κληρονομικότητα και σε ενσωματωμένους τύπους δεδομένων της Python.

Για παράδειγμα ας θεωρήσουμε ένα πρόγραμμα που αναπαράγει αρχεία ήχου. Το πρόγραμμα αυτό πρέπει να λάβει ένα AudioFile αντικείμενο και να το αναπαράγει με τη μέθοδο play : audio_file.play(). Η μέθοδος play είναι υπεύθυνη να αποσυμπιέσει τον ήχο και να τον δρομολογήσει στην κάρτα ήχου και στα ηχεία. Η διαδικασία διαφέρει ανάλογα με τον τύπο του αρχείου ήχου (.wav, .mp3, .wma κλπ) και για να απλοποιηθεί μπορούμε να χρησιμοποιήσουμε την κληρονομικότητα σε συνδυασμό με τον πολυμορφισμό.

```
#ex9.8 Polymorphism
#Κάθε τύπος αρχείου ήχου αντιπροσωπεύεται με μία υποκλάση της κλάσης AudioFile
#Εδώ οι υποκλάσεις είναι οι WavFile και MP3File. Κάθε υποκλάση έχει μία μέθοδο
#play() αλλά υλοποιείται διαφορετικά για κάθε αρχείο  ώστε να γίνει σωστά η
#διαδικασία αποσυμπίεσης του ήχου. Το media player αντικείμενο που αναπαράγει
#τον ήχο δε χρειάζεται να γνωρίζει σε ποια υποκλάση ανήκει η μέθοδος play. Αυτό
#είναι δουλειά του Audiofile και εδώ είναι που γίνεται η χρήση του πολυμορφισμού
class AudioFile:
    #Η μέθοδος __init__ της γονικής κλάσης μπορεί να έχει πρόσβαση στη μεταβλητή
    #ext των υποκλάσεων. Έτσι λειτουργεί ο πολυμορφισμός.
    def __init__(self, filename):
        if not filename.endswith(self.ext):
            raise Exception("Invalid file format")
        self.filename = filename

class MP3File(AudioFile):
    ext = "mp3"
    def play(self):
        print("playing {} as mp3".format(self.filename))
```

```
class WavFile(AudioFile):
    ext = "wav"
    def play(self):
        print("playing {} as wav".format(self.filename))

#Ο media player μπορεί να αναπαράγει το αντικείμενο FlacFile όπως ακριβώς κάνει
#με ένα αντικείμενο που επεκτείνει την AudioFile. Αυτό αποτελεί μειονέκτημα της
#Python στον πολυμορφισμό και είναι αποτέλεσμα του duck typing που μας επιτρέπει
#να χρησιμοποιήσουμε οποιοδήποτε αντικείμενο εμφανίζει την απαιτούμενη συμπεριφορά
# - δομή, χωρίς να αποτελεί υποκλάση. Αυτό βέβαια η Python το θεωρεί ασήμαντο.
class FlacFile:
    def __init__(self, filename):
        if not filename.endswith(".flac"):
            raise Exception("Invalid file format")
        self.filename = filename
    def play(self):
        print("playing {} as flac".format(self.filename))
```



Παράδειγμα 9.8 Πολυμορφισμός

10.ΧΡΗΣΗ ΤΗΣ PYTHON ΣΕ ΑΛΛΟΥΣ ΤΟΜΕΙΣ

10.1 Python και network programming

Η στάνταρ βιβλιοθήκη της Python παρέχει έναν αριθμό από modules για να αποκτήσουμε πρόσβαση στα στάνταρ internet πρωτόκολλα (HTTP, FTP) καθώς και για να υλοποιήσουμε clients και servers για αυτά.

Ένα πρωτόκολλο επικοινωνίας είναι ένα σύνολο κανόνων συμφωνημένων και από τα δυο επικοινωνούντα μέρη, που εξυπηρετούν την ανταλλαγή πληροφοριών μεταξύ τους. Οι κανόνες αυτοί καθορίζουν τη μορφή, το χρόνο και τη σειρά μετάδοσης των πληροφοριών στο δίκτυο.

10.1.1 Πρωτόκολλο HTTP (Hyper Text Transport Protocol)

Το πρωτόκολλο μεταφοράς υπερκειμένου (HTTP) είναι χτισμένο πάνω στο TCP και χρησιμοποιείται μεταξύ ενός φυλλομετρητή ιστού (web browser) και ενός διακομιστή ιστού (web server) για απευθείας επικοινωνία μεταξύ τους. Ο web browser στέλνει μία αίτηση για να ανακτήσει μία συγκεκριμένη ιστοσελίδα (web page) με την εντολή GET (μορφή: GET <URL διεύθυνση> <έκδοση του HTTP>) και ο web server στέλνει ως απάντηση το αντίγραφο της ιστοσελίδας.

10.1.1.1 socket module

Η Python παρέχει υποστήριξη του διαδικτυακού πρωτοκόλλου HTTP μέσω του χαμηλού επιπέδου socket module πάνω στο οποίο βασίζονται τα υψηλού επιπέδου δικτυακά πρωτόκολλα. Το socket αντικείμενο μοιάζει με ένα αρχείο και παρέχει σύνδεση μεταξύ δύο εφαρμογών. Μπορούμε να διαβάσουμε από και να γράψουμε στο ίδιο socket. Όταν γράφουμε κάτι στη μία πλευρά του socket, αυτό στέλνεται στην εφαρμογή που βρίσκεται στην άλλη πλευρά του. Όταν διαβάζουμε από ένα socket, στην ουσία διαβάζουμε τα δεδομένα που η εφαρμογή στην άλλη πλευρά έχει στείλει. Αν μία εφαρμογή προσπαθήσει να διαβάσει το περιεχόμενο ενός socket ενώ η εφαρμογή στην άλλη πλευρά δεν έχει στείλει δεδομένα, τότε απλά τίθεται σε κατάσταση αναμονής. Αν οι εφαρμογές και στις δύο πλευρές αναμένουν για τα ίδια δεδομένα χωρίς να στέλνουνε άλλα τότε θα βρίσκονται σε αναμονή για πολύ μεγάλο χρονικό διάστημα

Για να δημιουργήσουμε ένα socket αντικείμενο χρησιμοποιούμε τη μέθοδο socket του module:
socket.socket(family, type, protocol) όπου

- η πρώτη παράμετρος ελέγχει το OSI πρωτόκολλο επιπέδων και παίρνει μία από τις τιμές AF_UNIX, AF_INET ή AF_INET6 που αποτελούν σταθερές του socket module. Η AF_UNIX είναι ένα μοναδικό string, η AF_INET είναι μία πλειάδα της μορφής (host, port) και χρησιμοποιείται για IPv4 διευθυνσιοδότηση. Οι IPv4 διευθύνσεις είναι οι γνωστές IP διευθύνσεις. Τέλος η τιμή AF_INET6 είναι μία πλειάδα της μορφής (host, port, flowinfo, scopeid).
- η παράμετρος type ελέγχει το πρωτόκολλο μεταφοράς μεταξύ επιπέδων και καθορίζει τον τύπο του socket. Η τιμή SOCK_STREAM για TCP και η τιμή SOCK_DGRAM για UDP.
- η τρίτη παράμετρος ορίζει τον αριθμό πρωτοκόλλου, είναι προαιρετική και τις περισσότερες φορές παραλείπεται.

Κάποιες από τις μεθόδους που το socket αντικείμενο χρησιμοποιεί φαίνονται στον παρακάτω πίνακα:

socket.accept()	Δέχεται μία σύνδεση. Το αντικείμενο socket πρέπει να είναι δεσμευμένο σε μία διεύθυνση και να περιμένει για σύνδεση. Η επιστρεφόμενη τιμή είναι μία πλειάδα (conn, address) όπου conn ένα νέο socket αντικείμενο που χρησιμοποιείται για να στέλνει και να λαμβάνει δεδομένα στη σύνδεση και address η διεύθυνση στην οποία δεσμεύεται το socket στην άλλη πλευρά της σύνδεσης.
socket.bind(address)	Δεσμεύει το socket σε μία διεύθυνση.
socket.close()	Κλείνει το socket. Όλες οι μελλοντικές λειτουργίες που θα αναφέρονται σε αυτό θα αποτυγχάνουν.
socket.connect(address)	Το αντικείμενο socket συνδέεται απομακρυσμένα στη διεύθυνση address.
socket.makefile(mode, bufsize)	Επιστρέφει ένα file object που συνδέεται με αυτό το socket.

socket.recv(bufsize, flags)	Λαμβάνει δεδομένα από το socket. Η επιστρεφόμενη τιμή είναι ένα string που αντιπροσωπεύει τα δεδομένα που λαμβάνονται. Το μέγιστο σύνολο δεδομένων που μπορούν να ληφθούν σε μία φορά καθορίζεται από την παράμετρο bufsize.
socket.send(string, flags)	Στέλνει δεδομένα στο socket. Η προαιρετική παράμετρος flags καθορίζει το μέγιστο μέγεθος δεδομένων που μπορεί να σταλθεί. Οι εφαρμογές είναι υπεύθυνες για τον έλεγχο των δεδομένων που στέλνονται. Αν μόνο ένα μέρος των δεδομένων μεταφερθούν, τότε η εφαρμογή πρέπει να επιχειρήσει το μοίρασμα των δεδομένων που απέμειναν.

Ένα απλό Python πρόγραμμα για να κατανοήσουμε τη λειτουργία του HTTP είναι να θεωρήσουμε ότι το πρόγραμμά μας λειτουργεί ως web browser και πρέπει να συνδεθεί σε έναν web server. Το ακόλουθο παράδειγμα δείχνει πώς να δημιουργήσουμε μία χαμηλού επιπέδου διαδικτυακή σύνδεση με τη βοήθεια ενός socket αντικειμένου. Το αποτέλεσμα είναι μία σειρά από κεφαλίδες που παρέχουν πληροφορίες σχετικά με το αρχείο και το περιεχόμενο του αρχείου.

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
b'HTTP/1.1 200 OK\r\nDate: Wed, 25 Jan 2012 15:56:41 GMT\r\nServer: Apache\r\nLast-
Modified: Fri, 05 Nov 2010 03:37:37 GMT\r\nETag: "180dc16f-a7-4cd37c01"\r\nAccept-R
anges: bytes\r\nContent-Length: 167\r\nConnection: close\r\nContent-Type: text/plai
n\r\n\r\nBut soft what light through yonder window breaks\nIt is the east and Julie
t is the sun\nArise fair sun and kill the envious moon\nWho is already sick and pal
e with grief\n'
>>>
```

Παράδειγμα 10.1

10.1.1.2 urllib module

Το urllib.request module περιέχει συναρτήσεις και κλάσεις που κάνουν εύκολη την ανάκτηση ιστοσελίδων και την προσπέλαση δεδομένων μέσω των URLs. Προσδιορίζουμε ποια σελίδα θέλουμε να ανακτήσουμε και η urllib χειρίζεται όλες τις λεπτομέρειες του HTTP πρωτοκόλλου. Το URL ανοίγει και επιστρέφει ένα αντικείμενο που μοιάζει με αρχείο το οποίο μπορούμε να διαβάσουμε και διαχειριστούμε όπως τα υπόλοιπα αρχεία.

Ο αντίστοιχος κώδικας του Παραδείγματος 10.1 για να διαβάσουμε το αρχείο romeo.txt με χρήση του urllib.request είναι:

```
#ex10.2 urllib.request

import urllib.request
url = "http://www.py4inf.com/code/romeo.txt"
page = urllib.request.urlopen(url)
text = page.read().decode("utf8")
print(text)
```

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)]
win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

>>>
```

Παράδειγμα 10.2

Το παρακάτω παράδειγμα δείχνει πως μπορούμε να υπολογίσουμε τη συχνότητα κάθε λέξης στο αρχείο romeo.txt:

```
#ex10.3 count words in file
import urllib.request
counts = dict()
url = "http://www.py4inf.com/code/romeo.txt"
page = urllib.request.urlopen(url)
text = page.read().decode("utf8")

words = text.split()
for word in words:
    counts[word] = counts.get(word,0) + 1
print(counts)
```

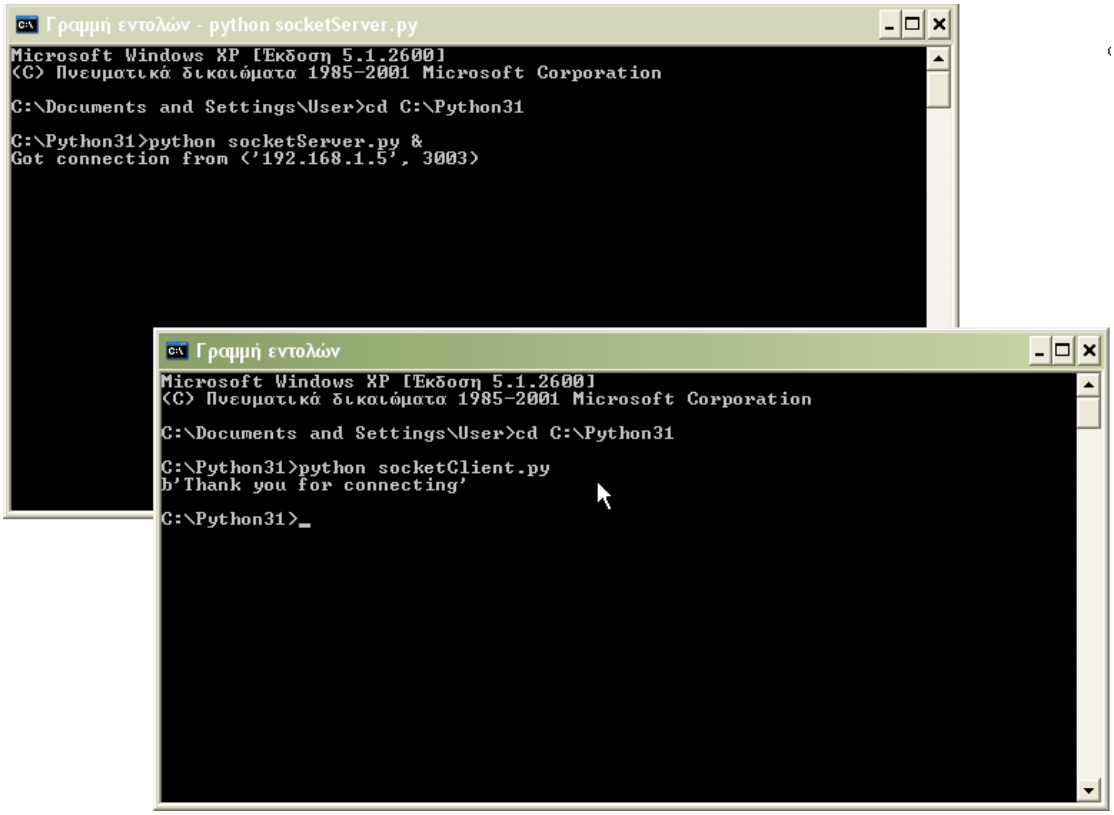
```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
{'and': 3, 'envious': 1, 'already': 1, 'fair': 1, 'is': 3, 'through': 1, 'pale':
 1, 'yonder': 1, 'what': 1, 'sun': 2, 'Who': 1, 'But': 1, 'moon': 1, 'window': 1
, 'sick': 1, 'east': 1, 'breaks': 1, 'grief': 1, 'with': 1, 'light': 1, 'It': 1,
 'Arise': 1, 'kill': 1, 'the': 3, 'soft': 1, 'Juliet': 1}
```

Παράδειγμα 10.3

Ακολουθεί η δημιουργία ενός απλού Server και Client με χρήση του socket module. Τρέχουμε πρώτα τον Server παρασκηνιακά, και μετά τον Client. Τα αποτελέσματα φαίνονται παρακάτω. Ο Client διαβάζει και εμφανίζει τα δεδομένα του web browser:

```
#simple server with socket
import socket                # Import socket module

s = socket.socket()          # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12351                 # Reserve a port for your service.
s.bind((host, port))         # Bind to the port

s.listen(5)                  # Now wait for client connection.
while True:
   c, addr = s.accept()      # Establish connection with client.
   print('Got connection from', addr)
   c.send(bytes('Thank you for connecting','ascii'))
   c.close()                 # Close the connection
```

```
#simple client with socket
import socket                # Import socket module

s = socket.socket()          # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12351                 # Reserve a port for your service.

s.connect((host, port))
print(s.recv(1024))
s.close                      # Close the socket when done
```



Παράδειγμα 10.4 Αλληλεπίδραση SocketServer – SocketClient

10.1.1.3 http module

Όλες οι web εφαρμογές πρέπει να τρέχουν σε έναν web server. Υπάρχουν πολλοί διαθέσιμοι web servers όπως ο Apache, αλλά στην Python δεν χρειάζεται να εγκαταστήσουμε και να ρυθμίσουμε έναν server. Μπορούμε με λίγες μόνο γραμμές κώδικα να δημιουργήσουμε έναν προσωρινό server για να αποκτήσουμε πρόσβαση στα αρχεία ενός φακέλου μέσω του HTTP πρωτοκόλλου.

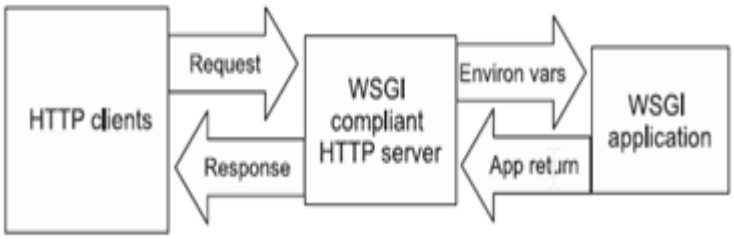
```
#ex10.4 Στιγμιότυπο HTTP server
from http.server import HTTPServer, SimpleHTTPRequestHandler

'''Αυτός ο server θα μοιράσει τα περιεχόμενα του φακέλου στη θύρα 8000 για
όλες τις ενεργές διεπαφές  δικτύου.
Η πρώτη παράμετρος είναι η πλειάδα('', 8000) που θέτει τη διεύθυνση και τη θύρα του
Το κενό string, σημαίνει πως ο server επιτρέπεται να χρησιμοποιήσει όλες τις
διευθύνσεις δικτύου της μηχανής.
Η δεύτερη παράμετρος είναι ο χειριστής αιτημάτων. Είναι μία υποκλάση του
BaseHTTPRequestHandler που έχει γραφτεί για να μοιράζει αρχεία του τρέχοντος
καταλόγου. Συγκεκριμένα προσδιορίζει τις μεθόδους do_GET και do_HEAD, που
αντιστοιχίζουν τα αιτήματα στα περιεχόμενα του τρέχοντα καταλόγου καιεπιστρέφουν
είτε λίστες καταλόγων είτε τα περιεχόμενα ενός αρχείου αν χρειάζονται. '''

server = HTTPServer(('',8000), SimpleHTTPRequestHandler)

print("Starting simple server on port: ", str(server.server_port))
server.serve_forever()
```

Παράδειγμα 10.5 Δημιουργία στιγμιότυπου HTTP server

10.1.1.4 wsgiref module

Η στάνταρ βιβλιοθήκη παρέχει μία απλή εφαρμογή του WSGI server μέσω του module wsgiref.simple_server. Για να δημιουργηθεί ο server πρέπει στη συνάρτηση make_server να περάσουμε τρεις παραμέτρους. Οι δύο πρώτες είναι η διεύθυνση και η θύρα για τον server και η τρίτη είναι μία συνάρτηση, εδώ η hello_world_app η οποία επιστρέφει ένα αντικείμενο application. Η hello_world_app παίρνει με τη σειρά της δύο παραμέτρους : α) ένα λεξικό – environ που περιέχει τις μεταβλητές περιβάλλοντος και ένα αντικείμενο start_response που διαβάζει την κατάσταση του HTTP και τις κεφαλίδες για

ανταπόκριση. Παρακάτω παρουσιάζεται μία βασική «Hello World» WSGI εφαρμογή :




```
#ex10.7.py
from wsgiref.simple_server import make_server

def hello_world_app(environ, start_response):
    status = '200 OK' #κατάσταση του HTTP
    headers = [('Content-type', 'text/plain; charset=utf-8')]
    start_response(status, headers)

    return ["Hello World"]

httpd = make_server('', 8000, hello_world_app)
print('Serving on port 8000...')

httpd.serve_forever()
```

```
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500
win32
Type "copyright", "credits" or "license()" for more informat
>>> ================================ RESTART ================
>>>
Serving on port 8000...
activate.adobe.com - - [29/Jan/2012 02:18:26] "GET / HTTP/1.
activate.adobe.com - - [29/Jan/2012 02:18:26] "GET /favicon.
```

Παράδειγμα 10.6

Αν τρέξουμε αυτό το script και ανοίξουμε έναν browser στην http://127.0.0.1:8000 θα εμφανιστεί το μήνυμα “Hello World”.

Πέρα από τα modules που αναφέρθηκαν παραπάνω υπάρχουν και άλλα σημαντικά modules που μπορούν να χρησιμοποιηθούν για δικτυακό ή internet python προγραμματισμό και παρουσιάζονται στον παρακάτω πίνακα:

Πρωτόκολλο	Λειτουργία	Αριθμός θύρας	Python Modules
HTTP	Web σελίδες	80	http, urllib,
NNTP	Usenet ειδήσεις	119	nntplib
FTP	Μεταφορά αρχείων	20	ftplib, urllib
SMTP	Αποστολή email	25	smtplib

POP3	Μεταφορά email	110	poplib
IMAP4	Μεταφορά email	143	imaplib
Telnet	Γραμμές εντολών	23	telnetlib
Gopher	Μεταφορά Εγγράφων	70	urllib

Πίνακας 10.1

10.2 Python και web programming

Η Python μας παρέχει τη δυνατότητα να δημιουργήσουμε web εφαρμογές με δυναμικό περιεχόμενο. Η ανάπτυξη μίας web εφαρμογής είναι μία σύνθετη διαδικασία. Το πρώτο βήμα που πρέπει να κάνουμε είναι να δημιουργήσουμε έναν web server, διαδικασία την οποία είδαμε στο προηγούμενο κεφάλαιο. Είναι προτιμότερο να χρησιμοποιούμε τον εσωτερικό server της εφαρμογής, επειδή δεν χάνουμε χρόνο με την εγκατάστασή του. Όταν όμως υπάρχει μεγάλη κίνηση στην εφαρμογή μας ένας εξωτερικός server χειρίζεται τον φόρτο καλύτερα. Τα περισσότερα web frameworks σήμερα μπορούν να τρέξουν είτε με τους δικούς τους HTTP servers είτε με έναν εξωτερικό server.

Πολύ σημαντική είναι επίσης η σύνδεση της εφαρμογής μας με μία βάση δεδομένων. Τα αποτελέσματα που προκύπτουν από τα ερωτήματα στις βάσεις είναι που παράγουν το δυναμικό περιεχόμενο μιας εφαρμογής.

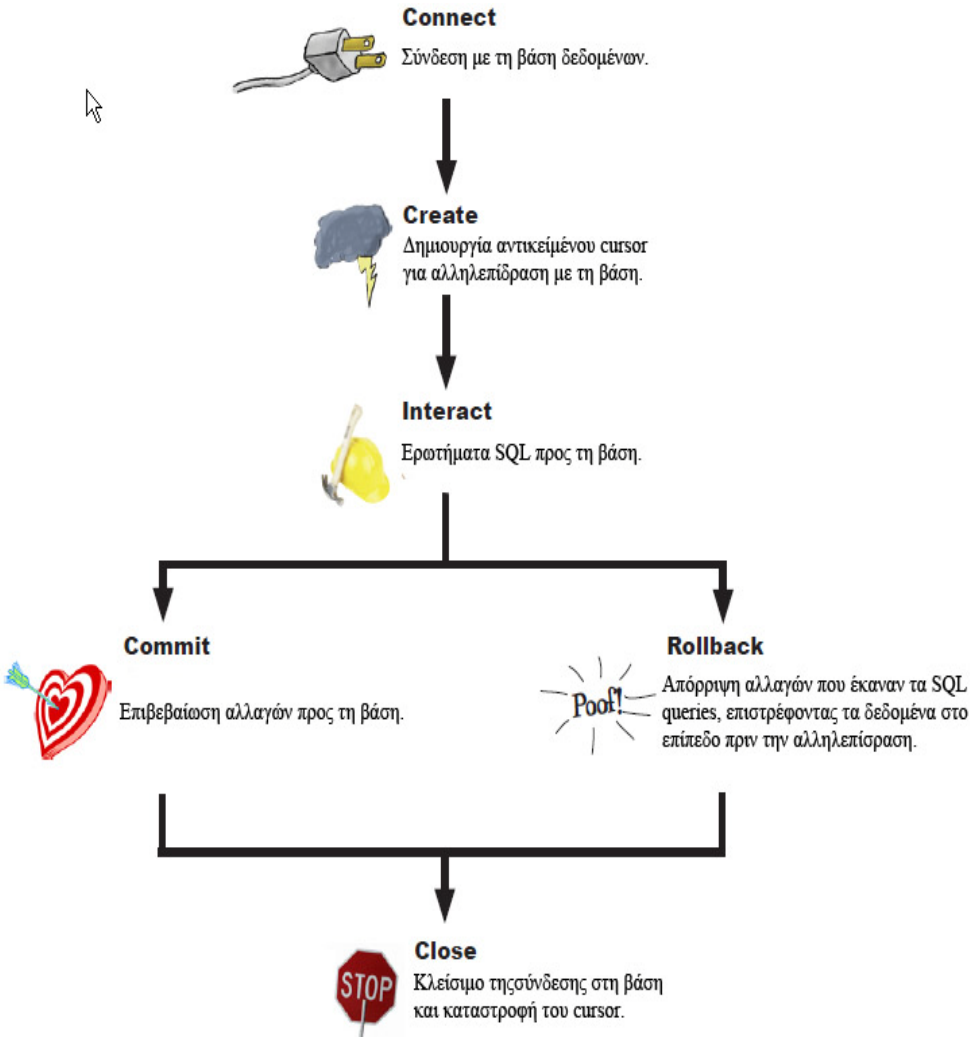
Τέλος μένει να σχεδιάσουμε το περιβάλλον της εφαρμογή μας (HTML σελίδα). Ένας τρόπος είναι με χρήση των CGI scripts μέσω του cgi module. Τα CGI scripts επιτρέπουν τη δημιουργία δυναμικών HTML σελίδων και είναι ιδιαίτερα χρήσιμα στο χειρισμό των φορμών στο Web. Παρ' όλα αυτά αποτελούν ένα ξεπερασμένο τρόπο σχεδίασης και είναι καλύτερο να χρησιμοποιήσουμε κάποιο από τα διαθέσιμα web frameworks που υποστηρίζει η Python όπως τα Django και Zope που υποστηρίζουν παλαιότερες εκδόσεις της Python και τα Pylons, TurboGears και web2py.

10.2.1 Πρόσβαση σε βάση δεδομένων

Η Python μπορεί να προσπελάσει τις πιο δημοφιλείς βάσεις δεδομένων – Oracle, PostgerSQL, MySQL - και το interface που κάθε μία ακολουθεί με χρήση εσωτερικών και εξωτερικών modules που ακολουθούν τη σημειογραφία της βάσης δεδομένων DB-API 2.0.

11.2.1.1 sqlite3 module

Ένας τύπος συστήματος διαχείρισης βάσεων δεδομένων που υποστηρίζει η Python, είναι το SQLite3 μέσω του ενσωματωμένου sqlite3 module και του στάνταρ database API της Python. Το API της Python παρέχει μία σειρά βημάτων που πρέπει να ακολουθήσουμε για να προγραμματίσουμε οποιοδήποτε σύστημα διαχείρισης βάσεων δεδομένων:



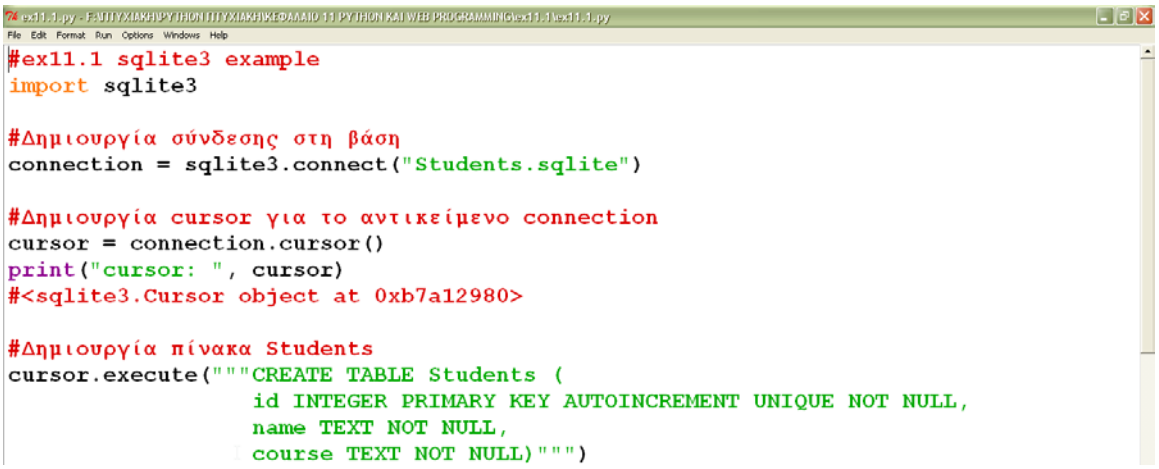
Εικόνα 10.1

Το sqlite3 module αποτελεί μία καλή επιλογή για μικρές εφαρμογές αλλά δεν είναι κατάλληλο για μεγάλες, υψηλής κίνησης εφαρμογές. Παρουσιάζει όμως δύο σημαντικά πλεονεκτήματα: α) είναι μέρος της στάνταρ βιβλιοθήκης και επομένως μπορεί να χρησιμοποιηθεί οπουδήποτε χρειαζόμαστε μία βάση δεδομένων και β)

αποθηκεύει όλες τις εγγραφές του σε ένα τοπικό αρχείο ώστε να μην χρειάζεται μαζί τον client και τον server, όπως συμβαίνει με την MySQL.

Για να χρησιμοποιήσουμε μία sqlite3 βάση δεδομένων, πρέπει πρώτα να δημιουργήσουμε ένα αντικείμενο connection με τη συνάρτηση connect. Η συνάρτηση connect παίρνει σαν παράμετρο το όνομα του αρχείου που θα χρησιμοποιηθεί για να αποθηκεύσει τα δεδομένα. Μπορούμε ακόμη να αποθηκεύσουμε τα δεδομένα στη μνήμη RAM δίνοντας όνομα παραμέτρου ‘:memory’

Αμέσως μετά δημιουργούμε ένα αντικείμενο cursor μέσω του αντικειμένου connection, που μας δίνει τη δυνατότητα να εκτελέσουμε τις εντολές CREATE, INSERT, UPDATE και SELECT προς τη βάση δεδομένων.
Υπάρχουν τρεις μέθοδοι για να εμφανίσουμε τα αποτελέσματα που προκύπτουν από ένα ερώτημα. Η μέθοδος fetchall εμφανίζει όλες τις γραμμές του αποτελέσματος σε μορφή λίστας πλειάδων, η fetchone επιστρέφει μία μόνο γραμμή του αποτελέσματος και η fetchmany επιστρέφει έναν αυθαίρετο αριθμό γραμμών.
Χρησιμοποιήσουμε την μέθοδο commit του αντικειμένου connection για να εφαρμόσουμε τις αλλαγές που θέτουν τα ερωτήματα.Αυτή είναι μία καλή πρακτική πριν κλείσουμε τη σύνδεση με μία βάση, επειδή η μέθοδος close δεν κάνει αυτόματα τις αλλαγές.

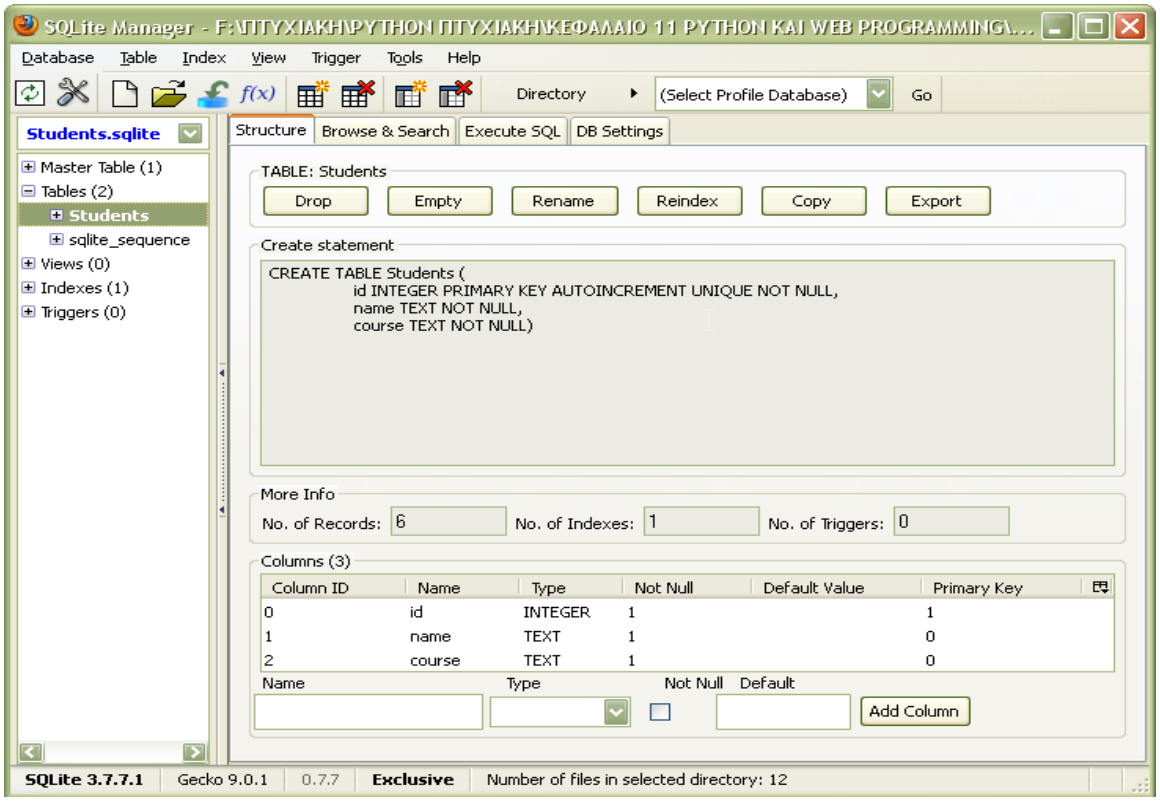


```
#Εισαγωγή τιμών στα πεδία του πίνακα
cursor.execute("INSERT INTO Students (name, course) VALUES (?, ?)",("Alicia","Netwo
cursor.execute("INSERT INTO Students (name, course) VALUES (?, ?)",("Joanna","Netwo
cursor.execute("INSERT INTO Students (name, course) VALUES (?, ?)",("Helen","JavaPr
cursor.execute("INSERT INTO Students (name, course) VALUES (?, ?)",("Alex","WebProg
cursor.execute("INSERT INTO Students (name, course) VALUES (?, ?)",("George","Netwo
cursor.execute("INSERT INTO Students (name, course) VALUES (?, ?)",("Billy","JavaPr

#εκτέλεση ερωτημάτων μέσω του αντικειμένου cursor για ανάκτηση δεδομένων από τη βάσ
result = cursor.execute("select * from Students")
print("All Students: ", result.fetchall())

cursor.execute("update Students set course = ? where course = ?", ("UNIX OS","WebPr
connection.commit()#Επιβεβαίωση αποθήκευσης της αλλαγής στη βάση δεδομένων
result = cursor.execute("select * from Students ")
#προσπέλαση μία προς μία των γραμμών του αρχείου μέσω του cursor
print("Students after update: ")
for row in result:
    print(row)

#Κλείνει η σύνδεση στη βάση
connection.close()
```

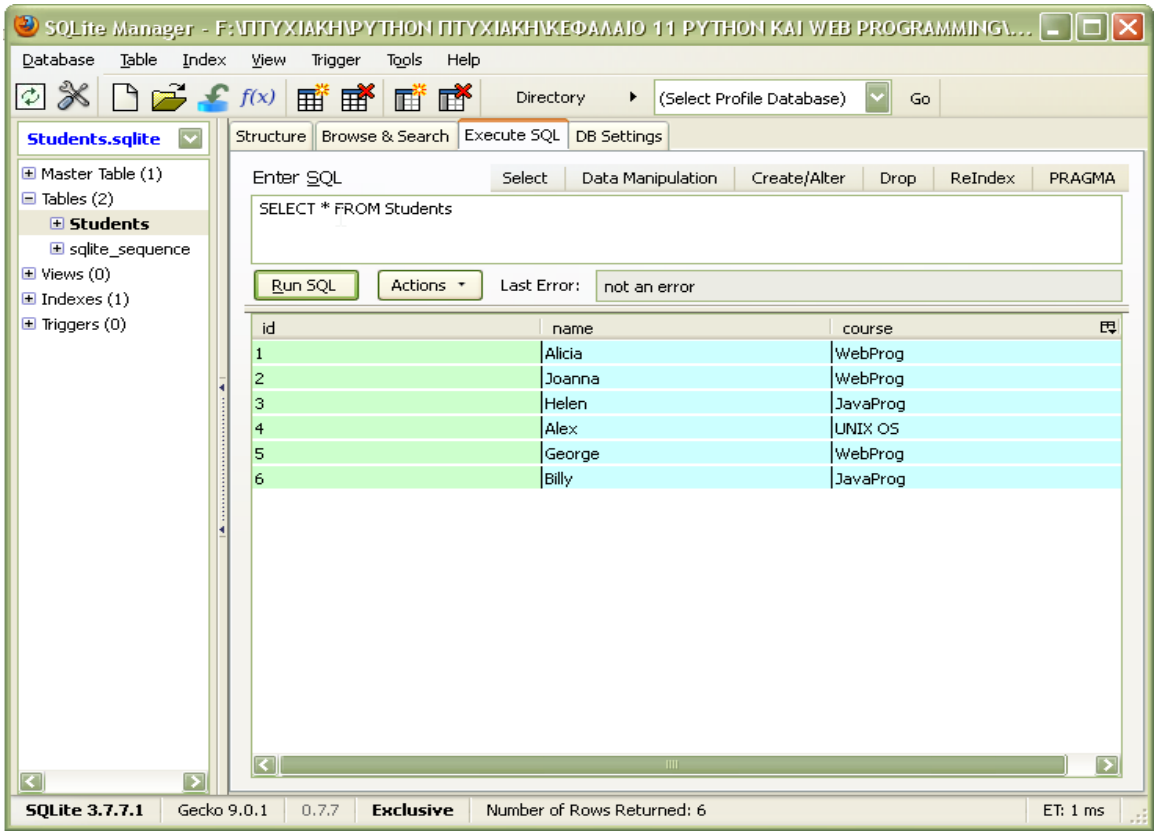
```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.1.3 (r313:86834, Nov 27 2010, 18:30:53) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
cursor:  <sqlite3.Cursor object at 0x0137E9B0>
All Students:  [(1, 'Alicia', 'Networks'), (2, 'Joanna', 'Networks'), (3, 'Helen',
'JavaProg'), (4, 'Alex', 'WebProg'), (5, 'George', 'Networks'), (6, 'Billy', 'JavaP
rog')]
Students after update:
(1, 'Alicia', 'Networks')
(2, 'Joanna', 'Networks')
(3, 'Helen', 'JavaProg')
(4, 'Alex', 'UNIX OS')
(5, 'George', 'Networks')
(6, 'Billy', 'JavaProg')
>>>
```

Παράδειγμα 10.7 sqlite3 module

Για να ελέγξουμε αν οι χειρισμοί που κάναμε στα δεδομένα της βάσης δεδομένων λειτουργούν μπορούμε να χρησιμοποιήσουμε έναν graphical database browser. Ένα τέτοιο παράδειγμα είναι ο SQLite Manager που μπορούμε να εγκαταστήσουμε στον Firefox σαν επέκταση. Μέσω αυτού μπορούμε να ανοίξουμε τη βάση δεδομένων που δημιουργήσαμε με Python κώδικα για να την ελέγξουμε.



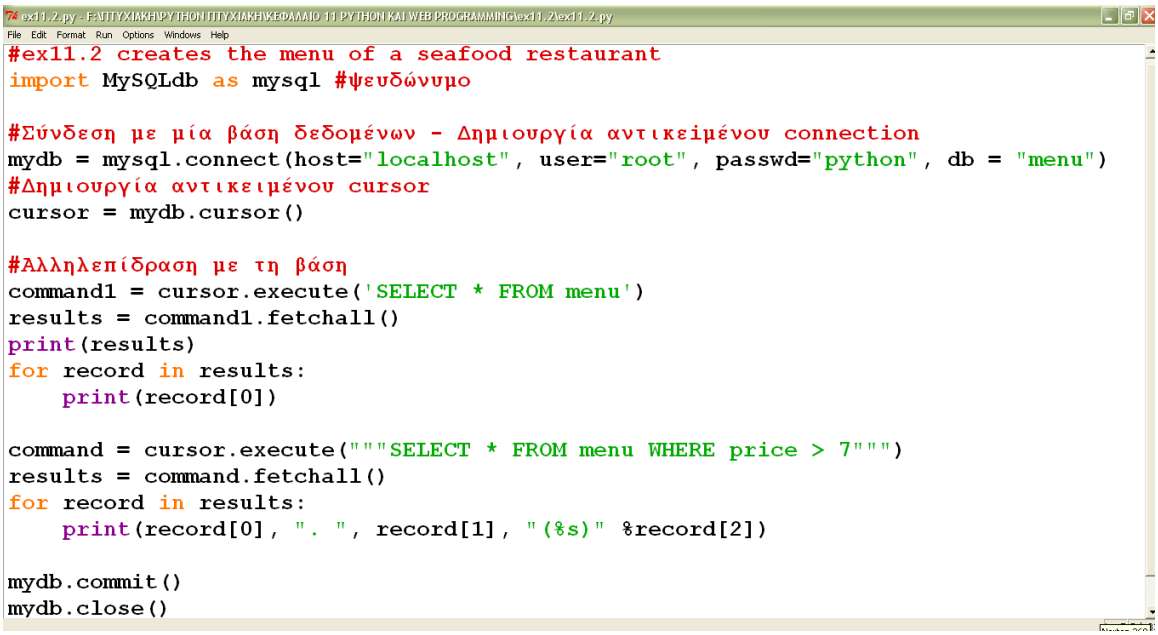
Εικόνα 10.2



Εικόνα 10.3

10.2.1.2 MySQLdb module

Το εξωτερικό module MySQLdb είναι συμβατό με τον MySQL database server και εξαρτάται από το _mysql module που αποτελεί Python αφαίρεση του MySQL C API . Το MySQLdb μας δίνει τη δυνατότητα να δημιουργήσουμε μία SQL βάση δεδομένων ή να συνδεθούμε με μία υπάρχουσα. Ακολουθεί την ίδια φιλοσοφία με το sqlite3, δηλαδή την DB-API 2.0, όπως δείχνει το επόμενο παράδειγμα:



Παράδειγμα 10.8.

10.3 Python και GUI

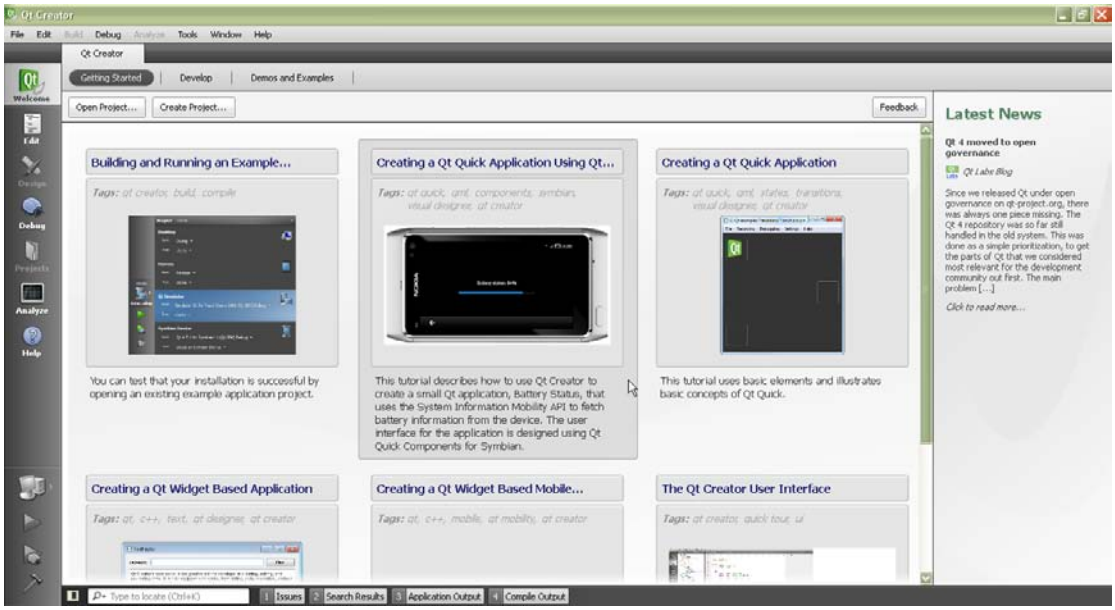
Η Python μας παρέχει τη δυνατότητα να δημιουργήσουμε γραφικά περιβάλλοντα χρήστη υποστηρίζοντας έναν μεγάλο αριθμό από GUI frameworks και toolkits.

Ο πρώτος τρόπος για να δημιουργήσουμε ένα GUI είναι να χρησιμοποιήσουμε την ενσωματωμένη GUI βιβλιοθήκη tkinter της Python. Η βιβλιοθήκη αυτή στηρίζεται στην open-source, cross-platform εργαλειοθήκη Tk. Η Tk παρέχει τα βασικά στοιχεία – widgets (button, menu, canvas, text, frame, label) για να χτίσουμε ένα GUI σε διάφορες γλώσσες προγραμματισμού, και στη Python.

Πέρα όμως από την εσωτερική βιβλιοθήκη η Python υποστηρίζει έναν μεγάλο αριθμό από εξωτερικά GUI frameworks και toolkits όπως το PyGtk που στηρίζεται στην εργαλειοθήκη GTK+ , το WxPython που στηρίζεται στην εργαλειοθήκη wxWidgets και το PyQt που στηρίζεται στην εργαλειοθήκη Qt.

10.3.1 PyQt4 και Qt

Το PyQt είναι ένα εργαλείο για δημιουργία GUI εφαρμογών που βασίζεται στην εργαλειοθήκη Qt. Η Qt δεν είναι μία απλή εργαλειοθήκη. Περιέχει αφαιρέσεις από network sockets, Unicode, κανονικές εκφράσεις, SQL βάσεις δεδομένων, SVG, OpenGL, XML, έναν πλήρως λειτουργικό web browser, ένα σύστημα βοήθειας, ένα framework πολυμέσων και μία πλούσια συλλογή από GUI widgets. Περιλαμβάνει αφαιρέσεις Qt κλάσεων χρησιμοποιώντας έναν signal/slot μηχανισμό που επιτρέπει την επικοινωνία μεταξύ αντικειμένων για reusable software components. Παρέχει επίσης έναν GUI QtDesigner μέσω του οποίου μπορεί να παράγει Python κώδικα.



Εικόνα 10.4 QtDesigner

10.3.2 Συστατικά του PyQt4

Τα δύο σημαντικότερα modules του PyQt4 είναι τα **QtCore** και **QtGui**. Το QtCore δεν περιέχει GUI κλάσεις αλλά τα απλά και βασικά λειτουργικά μέρη του πυρήνα, όπως δομές δεδομένων, timer συναρτήσεις, I/O buffers, threads, mapped files, κανονικές εκφράσεις και τις ρυθμίσεις των χρηστών και των εφαρμογών. Παράδειγμα κλάσεων που περιέχει είναι οι QPoint, QLine και QRect. Το εισάγουμε με την εντολή: from PyQt4.QtCore import *.

Από την άλλη πλευρά το QtGui περιέχει όλες τις κλάσεις για χειρισμό εικόνων, αρχείων, γραφικών στοιχείων (widgets) και κλάσεις που χειρίζονται τα συμβάντα που προκαλούνται από τους χρήστες (πχ mouse clicks και close events) όπως και συστατικά που έχουν να κάνουν με το λεγόμενο “look-and-feel” μιας εφαρμογής όπως δενδρικά μοντέλα για προβολή της ιεραρχίας των καταλόγων, brushes και pens για σχεδιασμό σχημάτων και άλλα. Κάποιες από τις κλάσεις που περιέχει είναι οι QImage, QWidget και QListWidget.

Το εισάγουμε με την πρόταση: from PyQt4.QtGui import *.

Άλλα σημαντικά modules παρουσιάζονται στον παρακάτω πίνακα:

QtHelp	Περιέχει κλάσεις για δημιουργία και προβολή αναζητήσιμη τεκμηρίωση.
QtNetwork	Περιέχει κλάσεις για εγγραφή UDP και TCP clients και servers. Περιλαμβάνουν κλάσεις που υλοποιούν FTP και HTTP clients και υποστηρίζουν DNS lookups.
QtOpenGL	Περιέχει κλάσεις που ενεργοποιούν τη χρήση OpenGL για τη δημιουργία 3D γραφικών σε PyQt εφαρμογές.
QtScript	Περιέχει κλάσεις που ενεργοποιούν τις PyQt εφαρμογές ώστε να γίνουν script με χρήση του Javascript διαρμηνευτή του Qt.
QtSql	Περιέχει κλάσεις που συγχωνεύονται με SQL βάσεις δεδομένων. Περιλαμβάνει μία υλοποίηση του SQLite.
QtSvg	Περιέχει κλάσεις που εμφανίζουν τα περιεχόμενα SVG αρχείων.
QtWebKit	Υλοποιεί μία μηχανή web browser που βασίζεται στην WebKit ανοιχτού κώδικα μηχανή browser.
QtXml	Περιέχει κλάσεις που υλοποιούν τις διεπαφές SAX και DOM στο XML parser του Qt

QtDesigner	Περιέχει κλάσεις που επιτρέπουν το Qt Designer να επεκταθεί με χρήση του PyQt.
phonon	Περιέχει κλάσεις που υλοποιούν έναν cross-platform multimedia framework που ενεργοποιεί τη χρήση του audio και video περιεχομένου στις PyQt εφαρμογές
Qt	Ομαδοποιεί τις κλάσεις που περιέχονται σε όλα τα παραπάνω modules σε ένα μοναδικό module. Έτσι δεν χρειάζεται να ανησυχούμε ποιο module περιέχει μία συγκεκριμένη μέθοδο. Το μειονέκτημα είναι ότι φορτώνει ολόκληρο το Qt framework και για το λόγο αυτό αυξάνει τη μνήμη (memory footprint) μιας εφαρμογής.
uic	Περιέχει κλάσεις για να χειριστούμε τα .ui αρχεία που δημιουργούνται με τον QtDesigner που περιγράφει ολόκληρο ή τμήμα από μία γραφική διεπαφή χρήστη. Περιέχει κλάσεις που φορτώνουν ένα .ui αρχείο και το διαμορφώνουν κατάλληλα και κλάσεις που παράγουν Python κώδικα από ένα .ui αρχείο για μετέπειτα εκτέλεση.

Πίνακας 10.2

Το PyQt 4 περιέχει επίσης έναν αριθμό από λειτουργικά προγράμματα:

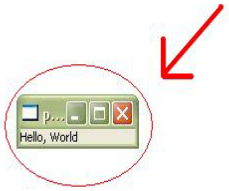
pyuic4	Αντιστοιχεί στη λειτουργία Qt uic. Μετατρέπει τα GUI που δημιουργούνται με χρήση του Qt Designer σε Python κώδικα.
pyrcc4	Ενσωματώνει αυθαίρετες πηγές (πχ εικόνες, αρχεία μετάφρασης) που περιγράφονται από ένα αρχείο σε ένα Python module. Εισάγεται μόνο αν το αντίγραφο του Qt περιλαμβάνει το xml module.
pylupdate4	Εξάγει όλα τα μεταφρασμένα strings από τον κώδικα Python και δημιουργεί ή ενημερώνει τα αρχεία μετάφρασης .ts που με τη σειρά τους χρησιμοποιούνται από το Qt Linguist για να χειριστούν τη μετάφραση αυτών των strings. Εισάγεται μόνο αν το αντίγραφο του Qt περιλαμβάνει το xml module.

Πίνακας 10.3

10.3.3 Παραδείγματα με PyQt4

- Hello World

```
#HelloWorld.py
import sys
from PyQt4 import Qt

#QApplication object
myapp = Qt.QApplication(sys.argv)

#Add a basic widget to this application: label with text Hello World
hello = Qt.QLabel("Hello, World")

#Show the label
hello.show()

#Start the application
myapp.exec_()
```

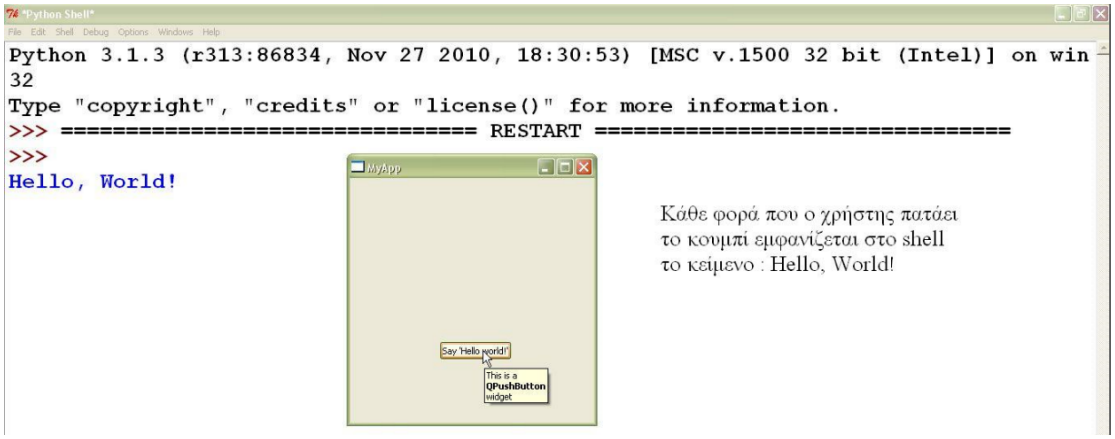


Παράδειγμα 10.9 Hello World

Το αποτέλεσμα δεν είναι και τόσο ικανοποιητικό. Πρέπει να προσθέσουμε περισσότερη λειτουργικότητα στην εφαρμογή:

```
#eventExample.py
import sys
from PyQt4.QtGui import *
from PyQt4.QtCore import *

myapp = QApplication(sys.argv)#αντικείμενο application

window = QWidget()#το widget χωρίς γονέα, καλείται παράθυρο
window.resize(300,300)
window.setWindowTitle("MyApp")
window.show()#εμφάνιση του widget στην οθόνη

#Η συνάρτηση sayHello καλείται όταν πατηθεί το κουμπί
def sayHello():
    print("Hello, World!")
```

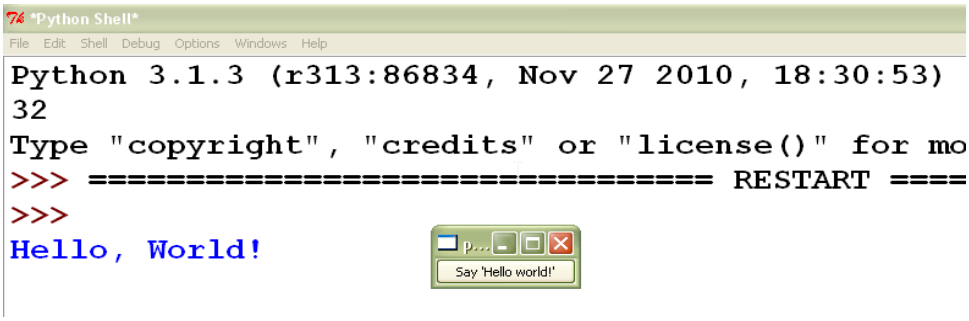
```
#κουμπί με όνομα Button μέσα στο παράθυρο
hellobutton = QPushButton("Say 'Hello world!'", window)
hellobutton.setToolTip('This is a <b>QPushButton</b> widget')
hellobutton.move(110,200)#τοποθέτηση μέσα στο παράθυρο

#Συνδέω την sayHello() με το γεγονός "button has been clicked"
myapp.connect(hellobutton, SIGNAL("clicked()"), sayHello)
hellobutton.show()

sys.exit(myapp.exec())
```



Παράδειγμα 10.10 eventExample

Το ίδιο παράδειγμα περισσότερο δομημένο με χρήση αντικειμενοστραφή προγραμματισμού.

```
#pythonic.py
#pythonic stuff with object-orientation
import sys
from PyQt4 import Qt

class HelloApplication(Qt.QApplication):

    def __init__(self, args):
        """ In the constructor we're doing everything to get our application
            started, which is basically constructing a basic QApplication by
            its __init__ method, then adding our widgets and finally starting
            the exec_loop."""
        Qt.QApplication.__init__(self, args)
        self.addWidgets()
        self.exec_()

    def addWidgets(self):
        """ In this method, we're adding widgets and connecting signals from
            these widgets to methods of our class, the so-called "slots"
        """
        self.hellobutton = Qt.QPushButton("Say 'Hello world!'",None)
        self.connect(self.hellobutton, Qt.SIGNAL("clicked()"), self.slotSayHello)
        self.hellobutton.show()

    def slotSayHello(self):
        """ This is an example slot, a method that gets called when a signal is
            emitted """
        print("Hello, World!")

# Only actually do something if this script is run standalone, so we can test our
# application, but we're also able to import this program without actually running
# any code.
if __name__ == "__main__":
    app = HelloApplication(sys.argv)
```



Παράδειγμα 10.11 Pythonic

10.4 Python και gaming

Η Python μας δίνει τη δυνατότητα να δημιουργήσουμε τα δικά μας παιχνίδια με χρήση της εξωτερικής βιβλιοθήκης Pygame.

10.4.1 Εισαγωγή στη Pygame βιβλιοθήκη

Η Pygame βιβλιοθήκη είναι ένα πακέτο με διαφορετικά modules γραμμένα σε Python και σε C. Η Pygame καλύπτει την SDL (Simple DirectMedia Layer) βιβλιοθήκη πολυμέσων που σχεδιάστηκε για να παρέχει χαμηλού επιπέδου πρόσβαση στον ήχο, στο πληκτρολόγιο, στο ποντίκι, στο χειριστήριο, στο 3D υλικό και στην αποθήκη frames των 2D βίντεο.

Κάποια από τα βασικά χαρακτηριστικά της Pygame είναι ότι :

1. μπορούμε να γράψουμε μόνο pygame προγράμματα και όχι εντολές σε interactive shell
2. **σχεδιάζουμε** γραφικά και κείμενο και η συνάρτηση input() δεν χρησιμοποιείται , δηλαδή δεν υπάρχει text I/O
3. χρησιμοποιεί γεγονότα (events)
4. χρησιμοποιεί τη συνάρτηση print() για εμφάνιση κειμένου στη κονσόλα.
5. χρησιμοποιεί κυρίως πλειάδες αντί για λίστες επειδή το περιεχόμενό τους δεν αλλάζει, οπότε η Python χειρίζεται τα δεδομένα περισσότερα αποδοτικά.

Για να εισάγουμε το πακέτο pygame υπάρχουν δύο τρόποι:

1. import pygame και
2. from pygame.locals import *

Κάποια από τα modules της pygame είναι προαιρετικά και δεν εμφανίζονται πάντα. Παράδειγμα ενός προαιρετικού module είναι το module.font. Όταν εισάγουμε την πρόταση import pygame, γίνεται έλεγχος για την ύπαρξη του font

module. Αν είναι διαθέσιμο τότε μπορούμε να το εισάγουμε με την πρόταση pygame.font, διαφορετικά το pygame.font τίθεται στην τιμή None.

10.4.2 Βασικά modules του pygame

Κάποια από τα βασικά modules της βιβλιοθήκης pygame είναι τα εξής :

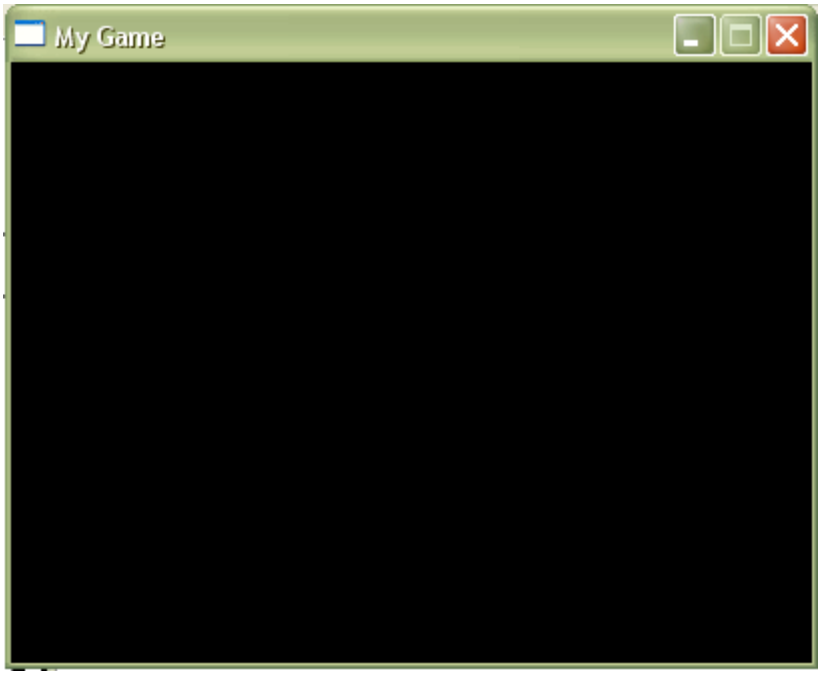
pygame.Color	Αντικείμενο για αναπαράσταση χρωμάτων. Η κλάση Color αντιπροσωπεύει RGBA τιμές χρωμάτων με εύρος τιμών 0 – 255. Επιτρέπει βασικές αριθμητικές λειτουργίες για δημιουργία νέων χρωμάτων, υποστηρίζει μετατροπές σε άλλους χώρους χρωμάτων (HSV ή HSL) και μας επιτρέπει να ρυθμίσουμε μοναδικά κανάλια χρωμάτων.
pygame.camera	για χρήση κάμερας
pygame.cdrom	για έλεγχο cdrom ήχου
pygame.cursors	για cursor resources
pygame.display	για να ελέγχουμε την εμφάνιση των παραθύρων και της οθόνης.
pygame.draw	για να σχεδιάζουμε σχήματα
pygame.event	για αλληλεπίδραση με γεγονότα και ουρές
pygame.font	για φόρτωση και διαμόρφωση γραμματοσειρών
pygame.image	για μεταφορά εικόνων
pygame.joystick	για αλληλεπίδραση με joystick συσκευές
pygame.key	για λειτουργίες σχετικά με το πληκτρολόγιο
pygame.mouse	για λειτουργίες με το ποντίκι
pygame.movie	για playback mpeg video
pygame.surfarray	για πρόσβαση στα δεδομένα των pixel της επιφάνειας με χρήση πινάκων διεπαφής
pygame.time	για διαχείριση χρόνου
pygame.transform	για διαμόρφωση επιφανειών
pygame.sprite	περιέχει υψηλού επιπέδου κλάσεις για να σχεδιάσουμε και να διαχειριστούμε τα αντικείμενα του παιχνιδιού μας.

Πίνακας 10.4 Βασικά pygame modules

10.4.3 Παραδείγματα για κατανόηση

Παράδειγμα 1ο – Βασικό παράθυρο

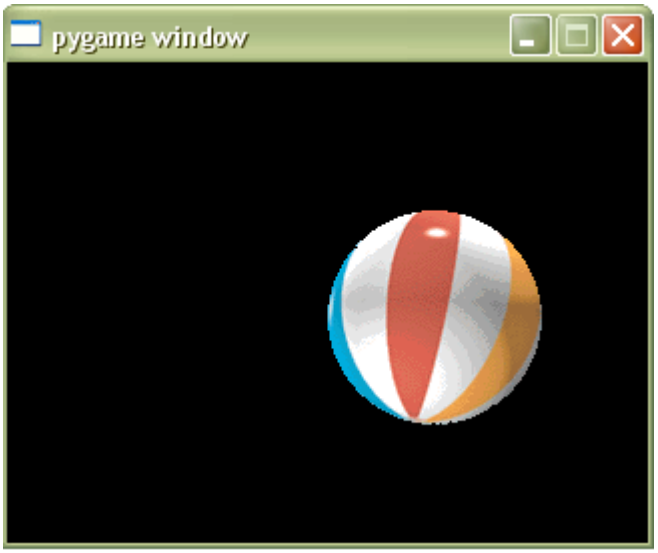
```
#ex16.1 base template
import pygame

#Ορισμός βασικών χρωμάτων
black    = (   0,   0,   0)
white    = ( 255, 255, 255)
green    = (   0, 255,   0)
red      = ( 255,   0,   0)

pygame.init()

# Ορισμός πλάτους και ύψους της οθόνης
size=[400,300]
screen=pygame.display.set_mode(size)

pygame.display.set_caption("My Game")

#Η μεταβλητή done είναι False μέχρι ο χρήστης να πατήσει το κουμπί close
done=False
# Χρήση της μεθόδου Clock για να διαχειριστούμε τη συχνότητα ανανέωσης της οθόνης
clock=pygame.time.Clock()

# -------- Main Program Loop -----------
while done==False:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    # Ορίζω το background της οθόνης
    screen.fill(black)

    # ΚΩΔΙΚΑΣ ΣΧΕΔΙΑΣΜΟΥ

    # Όριο σε 20 frames ανά δευτερόλεπτο
    clock.tick(20)

    # Ανανέωση της οθόνης με ότι σχεδιάσαμε
    pygame.display.flip()

pygame.quit ()
```



Παράδειγμα 2ο – Κινούμενη μπάλα

```
#ex16.2 move a ball
import pygame, sys
pygame.init()# αρχικοποιώ το pygame για να μπορέσω να χρησιμοποιήσω τα modules του

size = width, height = 320, 240 # πλάτος και ύψος της οθόνης
speed = [2, 2] # ταχύτητα
black = (0, 0, 0) # χρώμα

#δημιουργώ παράθυρο γραφικών -αντικείμενο surface- και οτιδήποτε σχεδιάζω
#σε αυτό γίνεται ορατό στην οθόνη
screen=pygame.display.set_mode(size)
pygame.display.set_caption("Happy Ball!!!")

# Χρήση της μεθόδου Clock για να διαχειριστούμε τη συχνότητα ανανέωσης της οθόνης
clock=pygame.time.Clock()

#φορτώνω την εικόνα με τη μπάλα, η συνάρτηση load() επιστρέφει ένα αντικείμενο
#surface που διατηρεί τα χρώματα και τη διαφάνεια του αρχείου
ball = pygame.image.load("ball.gif")
ballrect = ball.get_rect() #δημιουργία τετράγωνης περιοχής - αντικείμενο Rect
done = False
```

```
# -------- Main Program Loop -----------
while done==False:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    screen.fill(black)# Ορίζω το background της οθόνης

    # ΚΩΔΙΚΑΣ ΣΧΕΔΙΑΣΜΟΥ
    ballrect = ballrect.move(speed)#μετακίνηση αντικειμένου
    if ballrect.left < 0 or ballrect.right > width:
        speed[0] = -speed[0]
    if ballrect.top < 0 or ballrect.bottom >height:
        speed[1] = -speed[1]

    clock.tick(20)# Όριο σε 20 frames ανά δευτερόλεπτο

    screen.fill(black)#"σβηνω" την οθόνη γεμίζοντάς τη με μαύρο χρώμα
    screen.blit(ball, ballrect) #αντιγράφω τα pixels από μία Surface σε μία άλλη
    pygame.display.flip() # γίνεται ορατό στην οθόνη ότι έχουμε σχεδιάσει

pygame.quit ()
```

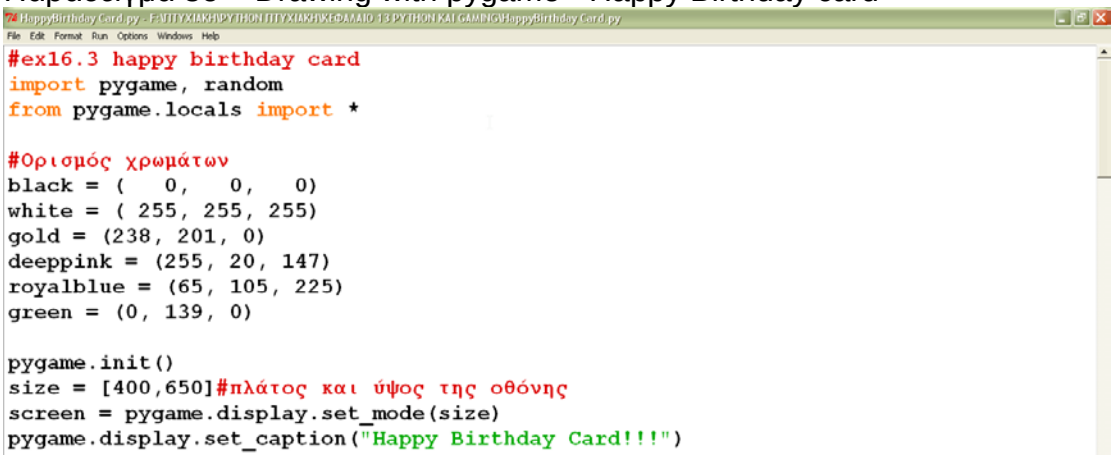


Για να χρησιμοποιήσουμε οτιδήποτε μας παρέχει η βιβλιοθήκη pygame, πρέπει να την αρχικοποιήσουμε με την πρόταση **pygame.init( )**, η οποία αρχικοποιεί τα pygame modules για να μπορέσουμε να τα χρησιμοποιήσουμε. Με την πρόταση pygame.namemodule.init() μπορούμε να αρχικοποιήσουμε το module που μας ενδιαφέρει. Κάθε module έχει επιπλέον και μία συνάρτηση get_init() με την οποία μπορούμε να ελέγξουμε αν ένα module έχει αρχικοποιηθεί οπότε επιστρέφει True. Τα modules που αρχικοποιούνται έχουν και τη μέθοδο quit() που τα «καθαρίζει» αλλά δεν χρειάζεται να την καλούμε αφού η Python το κάνει αυτόματα όταν τερματίζει.

Η μέθοδος **surface.blit()** αντιγράφει τα χρώματα των pixel μίας εικόνας (αντικείμενο surface) σε μία άλλη. Δεν προσθέτει ούτε μετακινεί pixels. Αλλάζει μόνο τα χρώματά των pixel που υπάρχουν ήδη στην οθόνη. Σαν παραμέτρους δίνουμε την εικόνα από την οποία θα αντλήσει τα χρώματα και μία θέση για να τοποθετήσει το αντίγραφο.

Άρα όταν θέλουμε να μετακινήσουμε μία εικόνα, «κάνουμε blit» την εικόνα σε μία νέα θέση, αφού σβήσουμε την παλιά. Διαφορετικά η εικόνα θα εμφανίζεται σε δύο θέσεις ταυτόχρονα στην οθόνη. Με την αλλαγή θέσης της εικόνας και το σβήσιμο της παλιάς με γρήγορη ταχύτητα πετυχαίνουμε το εφέ της κίνησης.

Παράδειγμα 3ο – Drawing with pygame - Happy Birthday card



```
#empty array
star_list=[]
# Loop 50 times and add a star in a random x,y position
for i in range(50):
    x=random.randrange(0,400)
    y=random.randrange(0,400)
    star_list.append([x,y])

#Η μεταβλητή done είναι False μέχρι ο χρήστης να πατήσει το κουμπί close
done=False

# Χρήση της μεθόδου Clock για να διαχειριστούμε τη συχνότητα ανανέωσης της οθόνης
clock=pygame.time.Clock()

# -------- Main Program Loop -----------
while done==False:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True
```

```
    # Ορίζω το background της οθόνης
    screen.fill(gold)

    # ΚΩΔΙΚΑΣ ΣΧΕΔΙΑΣΜΟΥ

    # Process each star in the list
    for i in range(len(star_list)):
        # Draw the star
        pygame.draw.circle(screen,deeppink,star_list[i],2)

        # Move the star down one pixel
        star_list[i][1]+=1

        # If the star has moved off the bottom of the screen
        if star_list[i][1] > 400:
            # Reset it just above the top
            y=random.randrange(-50,-10)
            star_list[i][1]=y
            # Give it a new x position
            x=random.randrange(0,400)
            star_list[i][0]=x
```

```
    basicFont = pygame.font.SysFont(None, 48)
    #σχεδιάζω κείμενο πάνω στο αντικείμενο Surface, επιστρέφει Surface
    text = basicFont.render('Happy Birthday!!!', True, deeppink, gold)
    textRect = text.get_rect()
    textRect.centerx = screen.get_rect().centerx
    textRect.centery = screen.get_rect().centery

    #draw the ballons
    #pygame.draw.circle(Surface, color, pos, radius, width=0): return Rect
    pygame.draw.circle(screen, royalblue, (350, 100), 20, 0)
    pygame.draw.circle(screen, royalblue, (300, 150), 20, 0)

    #pygame.draw.polygon(Surface, color, pointlist, width=0): return Rect
    pygame.draw.polygon(screen, royalblue, ((350, 120), (340, 130), (360, 130)))
    pygame.draw.polygon(screen, royalblue, ((300, 170), (290, 180), (310, 180)))

    #pygame.draw.lines(Surface, color, closed, pointlist, width=1): return Rect
    pygame.draw.line(screen, green, (350, 130), (350, 210), 2)
    pygame.draw.line(screen, green, (300, 180), (300, 260), 2)
```

```
    # pygame.draw.rect(Surface, color, Rect, width=0): return Rect
    pygame.draw.rect(screen, black, (textRect.left - 10, textRect.top - 10, textRec
    #σχεδιάζω το κείμενο πάνω στο αντικείμενο surface
    screen.blit(text, textRect)

    icon = pygame.image.load('birthday_cake.gif')#εικόνα
    iconRect = icon.get_rect().move(50,360)
    screen.blit(icon, iconRect)
    # Όριο σε 20 frames ανά δευτερόλεπτο
    clock.tick(20)
    # Ανανέωση της οθόνης με ότι σχεδιάσαμε
    pygame.display.update()
    pygame.display.flip()

pygame.quit ()
```

"Happy Birthday card"

Παράδειγμα 4ο – Collision game

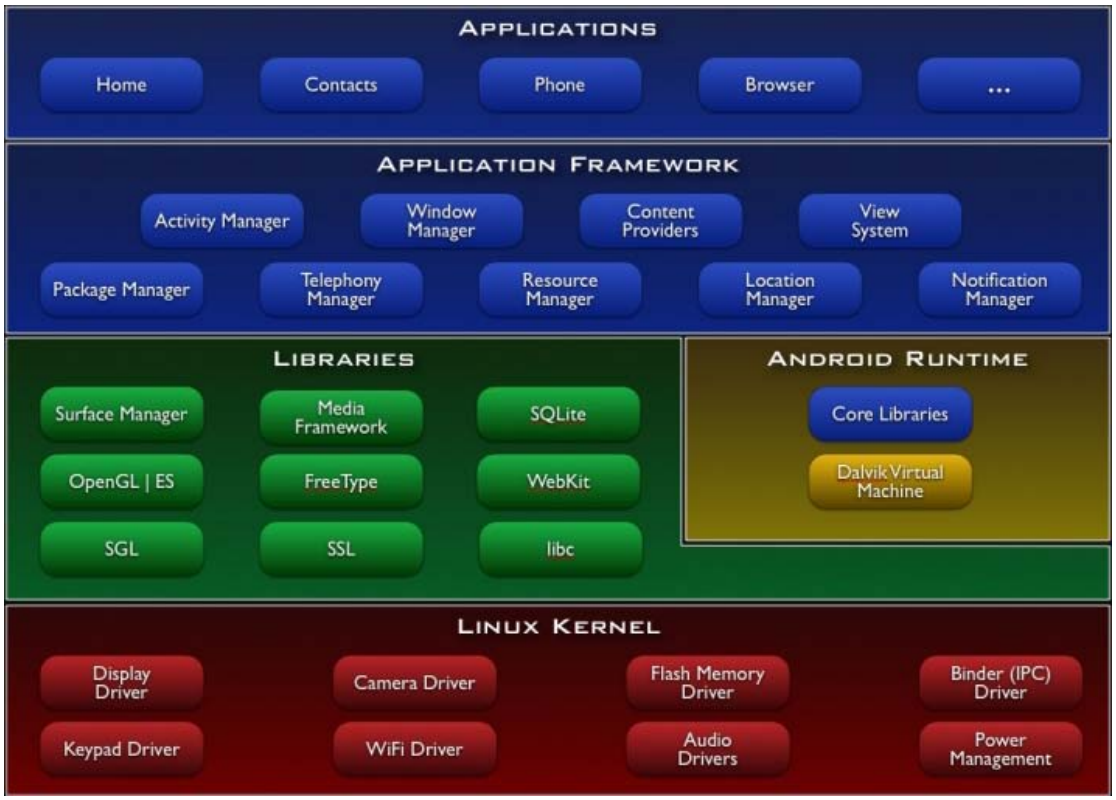
10.5 Python και Android

10.5.1 Λογισμικό Android

Το Android είναι λογισμικό για φορητές συσκευές που περιλαμβάνει λειτουργικό σύστημα, middleware και εφαρμογές. Είναι open source και βασίζεται σε μία τροποποιημένη έκδοση του πυρήνα του Linux.

Χαρακτηριστικά και αρχιτεκτονική του Android

- Πλαίσιο εφαρμογών
- Dalvik virtual machine
- Browser
- Βελτιστοποιημένα γραφικά
- SQLite
- Υποστήριξη πολυμέσων
- GSM Telephony
- Bluetooth, EDGE, 3G, WiFi
- Κάμερα, GPS, πυξίδα και accelerometer
- Πλούσιο περιβάλλον ανάπτυξης εφαρμογών



10.5.2 Python και Android Scripting Environment

Για ανάπτυξη εφαρμογών σε Python υπάρχει η πλατφόρμα Android Scripting Environment (ASE) που επιτρέπει τη δημιουργία και εκτέλεση scripts και διαδραστικών διερμηνευτών κατευθείαν για την πλατφόρμα Android.

Χαρακτηριστικά της πλατφόρμας ASE:
1.Πρόσβαση με τη βοήθεια πολλών διαθέσιμων APIs σε πολλές εφαρμογές της πλατφόρμας.
2.Απλοποιημένα interfaces που βοηθούν στην ανάπτυξη.
3.Δυνατότητα των scripts να τρέχουν διαδραστικά σε ένα terminal, στο background ή μέσω Locale.
4.Υποστηρίξη επιπλέον των γλωσσών προγραμματισμού Perl, JRuby, Lua, BeanShell, JavaScript, Tcl και shell.
5.Εκμετάλλευση των APIs που υπάρχουν για αλληλεπίδραση με τη συσκευή.

Πλεονεκτήματα
1. Επιτρέπει την εκτέλεση scripts από πολλές scripting languages.
2. Αλληλεπίδραση με λειτουργίες της κινητής συσκευής μέσω διάφορων APIs και δημιουργία όμορφων εφαρμογών.
3. Ανάπτυξη ισχυρών εφαρμογών με λίγες γραμμές κώδικα με λιγότερες απαιτήσεις σε resources από τις αντίστοιχες Java εφαρμογές.
4. Τα scripts μπορούν να τρέχουν διαδραστικά σε ένα terminal, στο background ή μέσω Locale.

Μειονεκτήματα
1. Προγραμματισμός στη συσκευή και όχι στο IDE.
2. Δυσκολία στη συγγραφή κώδικα, στο debugging και σε όποιες άλλες ευκολίες προσφέρει ένα IDE.
3. Πιο αργή εκτέλεση των scripts.
4. ΄Υπαρξη bugs.

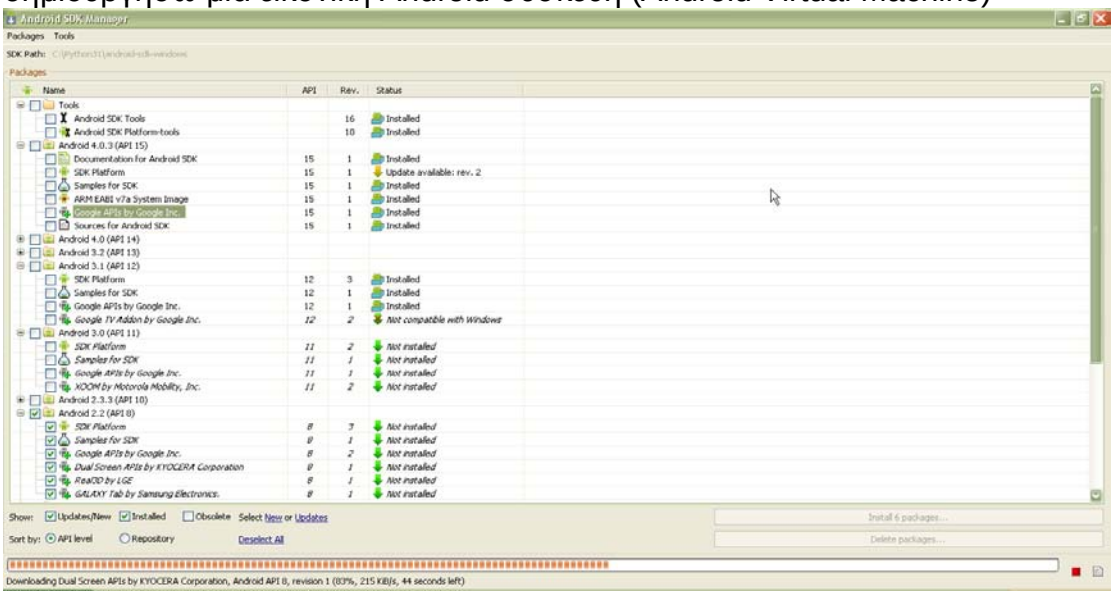
Βιβλιοθήκες της πλατφόρμας Android-ASE
1. System C Library
2. Media Libraries
3. Surface Manager
4. LibWebCore
5. SGL
6. 3D Libraries
7. SQLite

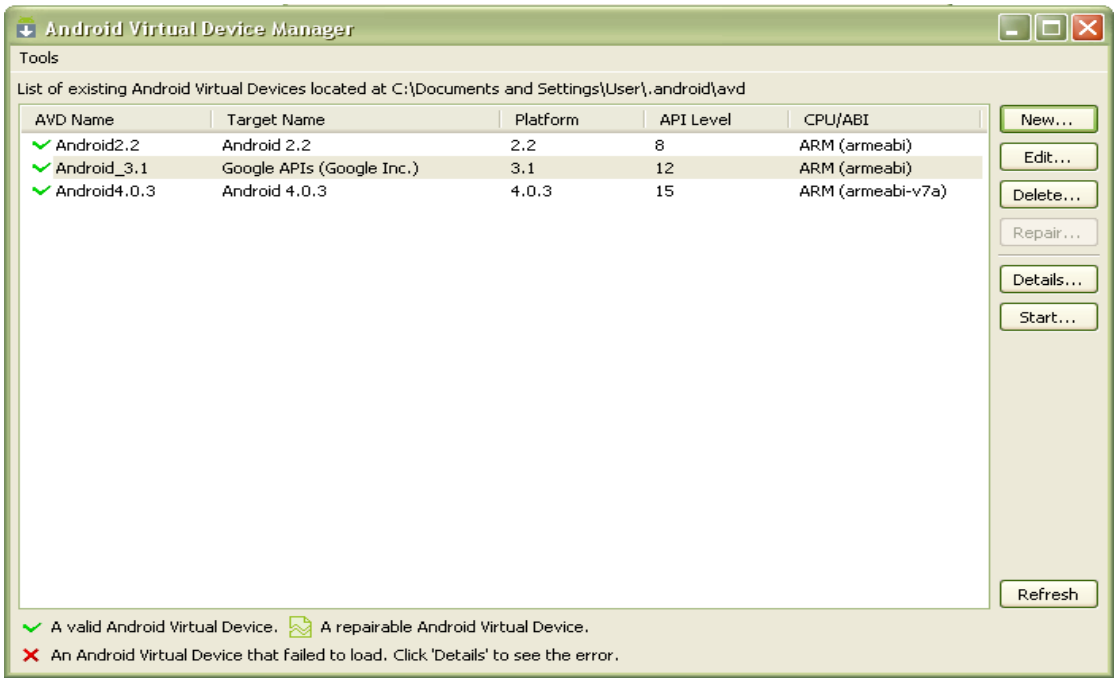
Το project SL4A (Scripting Layer for Android) παρέχει τεχνολογία που μας επιτρέπει να τρέξουμε την Python σε οποιαδήποτε Android συσκευή. Αν και το SL4A είναι συμβατό με την Python 2, δεν υπάρχει πρόβλημα εφόσον ένα μέρος του κώδικα τρέχει στο smartphone σε Python 2 ενώ το υπόλοιπο κομμάτι κώδικα τρέχει στον web server σε Python3.
Η Google παρέχει έναν cross-platform προσομοιωτή για ανάπτυξη κώδικα χωρίς τη παρουσία υλικού, κατευθείαν στον υπολογιστή μας.

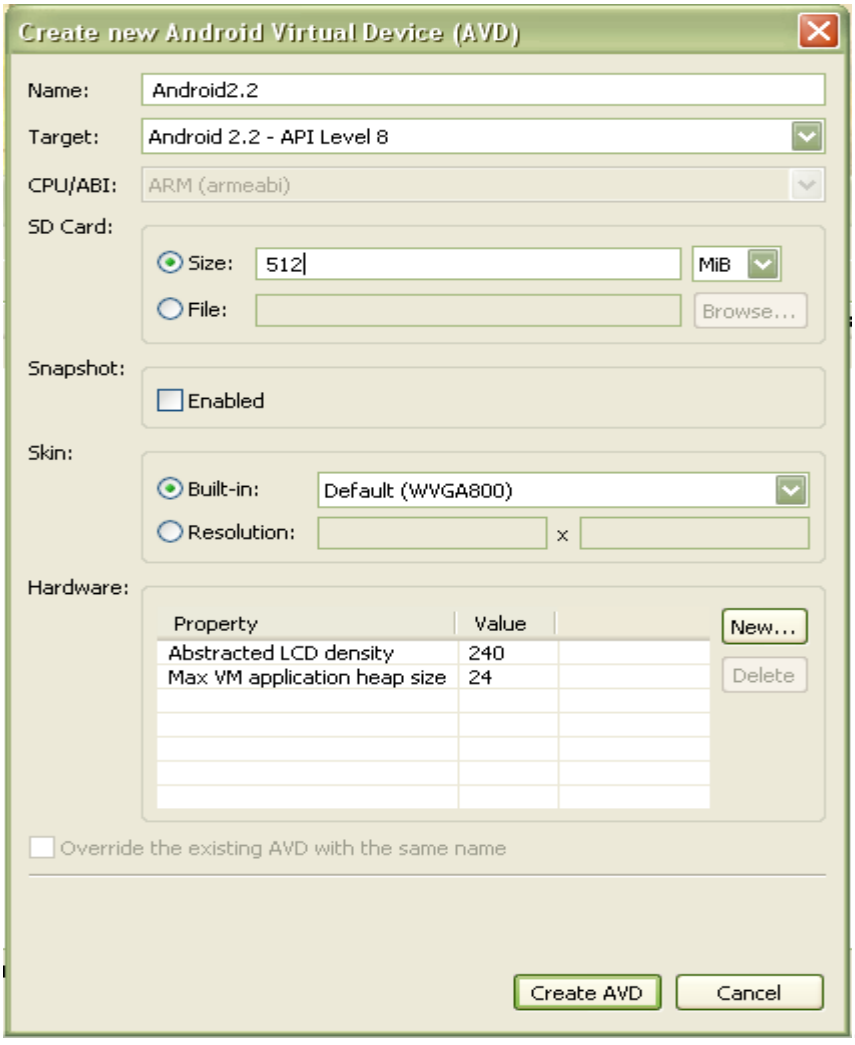
10.5.3 Βήματα εγκατάστασης του προσομοιωτή Android:

1. Εγκαθιστώ το Android SDK ανάλογα με το λειτουργικό σύστημα που έχω από την διεύθυνση http://developer.android.com/sdk/index.html.

2. Τρέχω τον SDK Manager για να προσθέσω μία πλατφόρμα Android ανάλογα με την έκδοση της Python που έχω εγκατεστημένη και τον AVD Manager για να δημιουργήσω μία εικονική Android συσκευή (Android Virtual Machine)

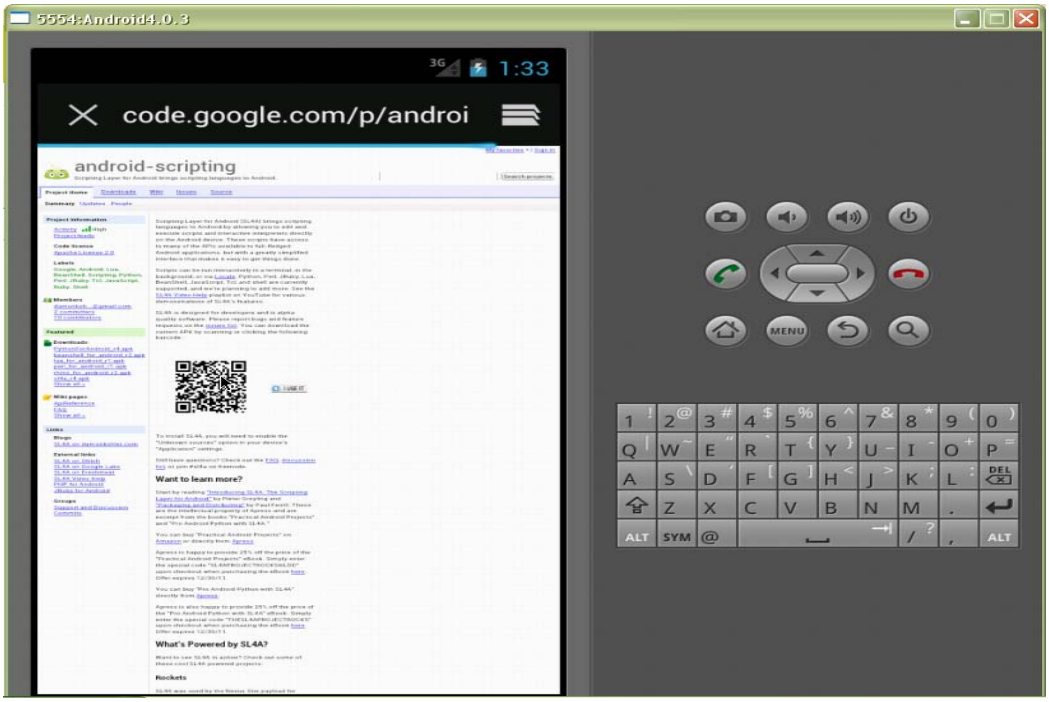


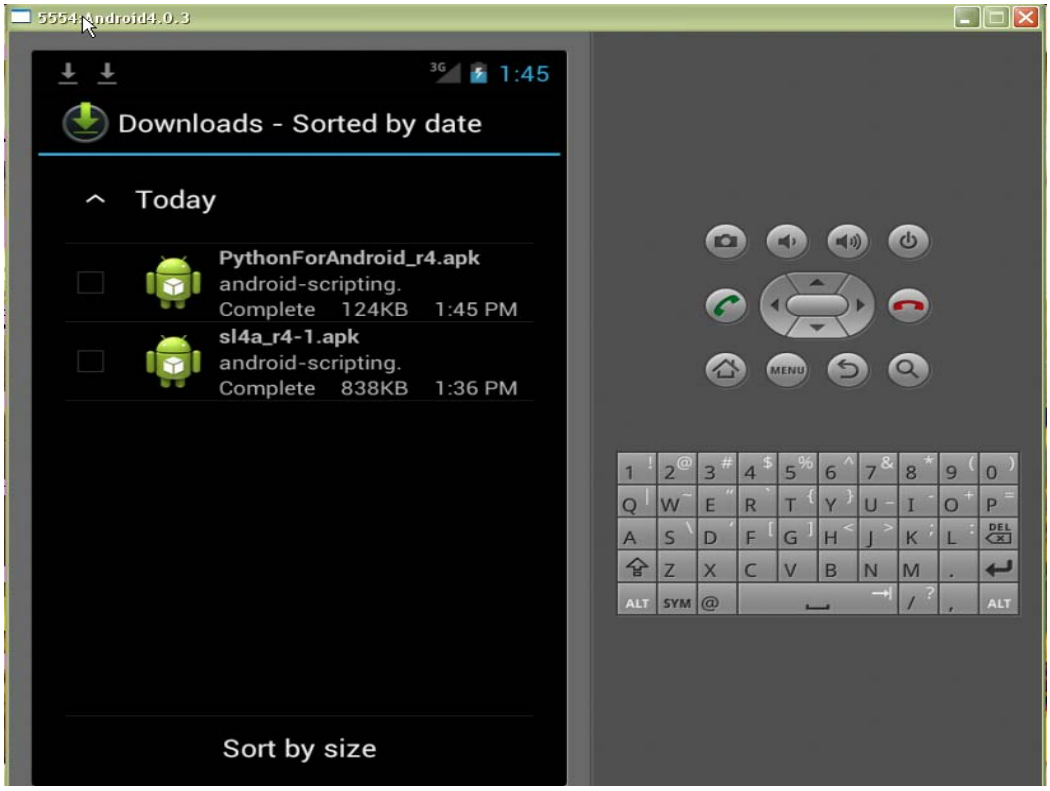




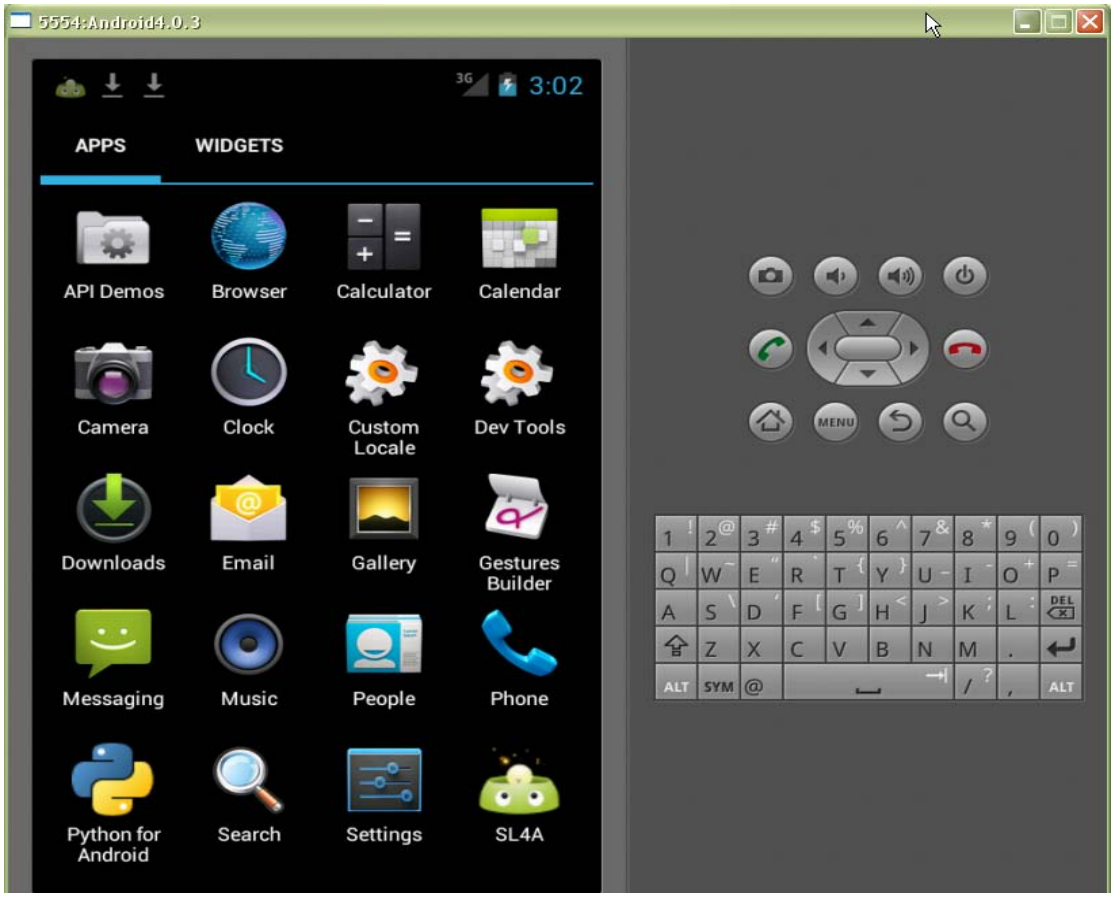
Αφού δημιουργήσω την εικονική μηχανή, την φορτώνω από το κουμπί Start → Launch:

3. Τέλος πρέπει να εγκαταστήσω το ASE. Στον browser του προσομοιωτή πληκτρολογώ τη διεύθυνση http://code.google.com/p/android-scripting/ και εγκαθιστώ το sl4a_r4-1.apk και PythonForAndroid_r4.apk που αποθηκεύονται στα Downloads.

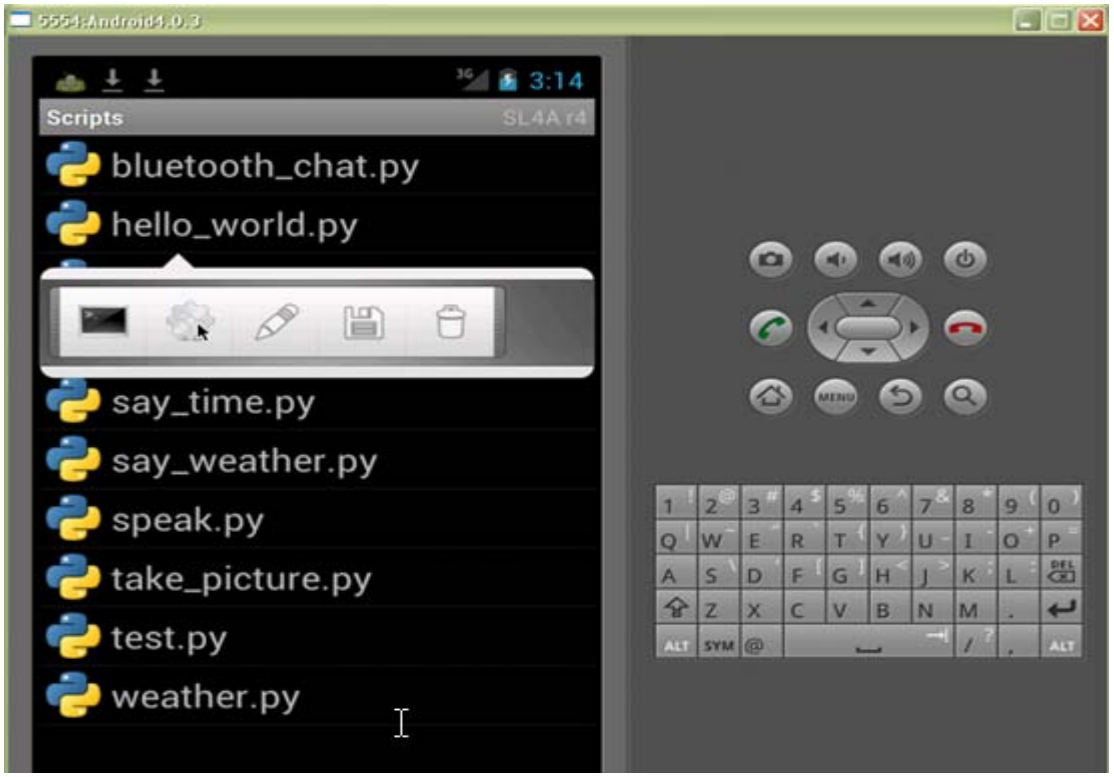


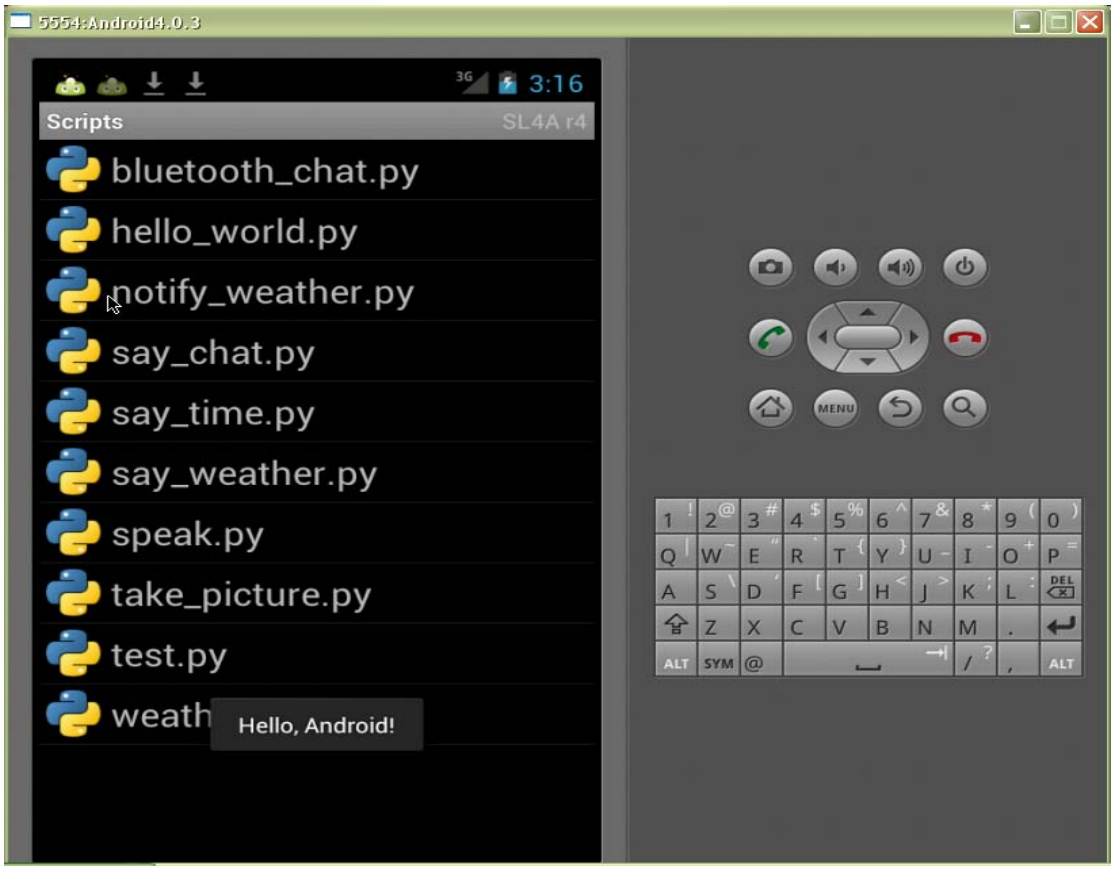


Στην επόμενη εικόνα φαίνονται τα δύο εικονίδια των εφαρμογών sl4a και Python For Android:



Αν πατήσουμε το εικονίδιο του SL4A, εμφανίζεται μία λίστα από διαθέσιμα scripts τα οποία μπορούμε να τρέξουμε για να επιβεβαιώσουμε ότι η εγκατάσταση έγινε σωστά:



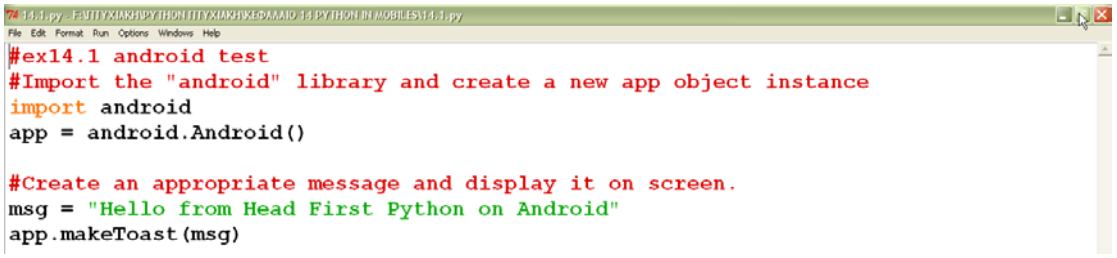


10.5.4 Ενσωμάτωση δικού μας py αρχείου

Μπορούμε να γράψουμε το δικό μας αρχείο .py σε Python και να το ενσωματώσουμε στον προσομοιωτή αντιγράφοντάς το στην εικονική SD κάρτα του. Σε αυτό βοηθάει το πρόγραμμα adb που βρίσκεται στον φάκελο tools. Συγκεκριμένα πληκτρολογώντας την εντολή:

tools/adb push example.py /sdcard/sl4a/scropts

σε ένα τερματικό παράθυρο το πρόγραμμα ενσωματώνεται στη λίστα των scripts του SL4A.



ΒΙΒΛΙΟΓΡΑΦΙΑ

1. The Quick Python Book, 2nd. ed. Manning, 2010 Vernon L. Ceder

2. Programming in Python 3 - A Complete Introduction to the Python Language, Second Edition (2010) Mark Summerfield

3.

4.

5.

6.

7.

8.