Εισαγωγή στην επιστήμη των υπολογιστών

Σημειώσεις για προγραμματισμό σε Python

Κωνσταντίνος Πατσάκης

Πειραιάς
Οκτώβριος 2014

Πρόλογος

Το παρόν έγγραφο αποτελεί μέρος των σημειώσεων για τη διδασκαλία του μαθήματος ``Εισαγωγή στην επιστήμη των υπολογιστών'' του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς. Σκοπός των σημειώσεων είναι οι φοιτητές να εξοικειωθούν με τη γλώσσα Python μέσω παραδειγμάτων. Όλα τα παραδείγματα αφορούν την Python 2.7. Οι σημειώσεις θα ενημερώνονται και θα βελτιώνονται συνεχώς από το διδάσκοντα. Το έγγραφο διανέμεται δωρεάν με άδεια Attribution-NonCommercial-ShareAlike 4.0 International της Creative Commons. Για λεπτομέρειες σε σχέση με το τι επιτρέπει αυτή η άδεια χρήσης μπορείτε να επισκεφθείτε τη σελίδα:

http://creativecommons.org/licenses/by-nc-sa/4.0/

Περιεχόμενα

1 Εγκατάσταση και χρήση της Python

1.1 Εγκατάσταση

Για Debian/Ubuntu:

```
apt-get install python
```

1.1.1 Για Windows

Πηγαίνετε στη διεύθυνση: https://www.python.org/downloads/windows/, κατεβάστε το πρόγραμμα εγκατάστασης για το λειτουργικό σας σύστημα και εγκαταστήστε το.

Για να μπορείτε να δουλεύετε από το command line χωρίς προβλήματα πρέπει να προσθέσετε τον φάκελο εγκατάστασης της Python στο Path. Για να γίνει αυτό κάντε δεξί κλικ στον υπολογιστή σας, επιλέξτε ιδιότητες και στη συνέχεια Ρυθμίσεις για προχωριμένους. Τέλος επιλέξτε ``Μεταβλητές Περιβάλλοντος''. Εκεί επιλέξτε την μεταβλητή PATH και προσθέστε στο τέλος το ``;C:Python27''.

Για την αγγλική έκδοση ακολουθήστε τη διαδρομή ``My Computer→Properties→Advanced→Environment→Variables''.

1.1.2 MacOS

Πηγαίνετε στη διεύθυνση: https://www.python.org/downloads/mac-osx/, κατεβάστε το πρόγραμμα εγκατάστασης για το λειτουργικό σας σύστημα και εγκαταστήστε το.

1.2 Χρήση

Ανοίξτε έναν επεξεργαστή κειμένου και γράψτε το ακόλουθο κείμενο:

```
print "Hello world"
```

Αποθηκεύστε το αρχείο ως ``hello.py'' σε ένα φάκελο της αρεσκίας σας. Ανοίξτε ένα τερματικό και πηγαίνετε στο φάκελο που αποθηκεύσατε το αρχείο ``hello.py'' και εκτελέστε την εντολή:

```
python hello.py
```

Ανάλογα το περιβάλλον το οποίο επιλέξετε να χρησιμοποιήσετε για να γράφετε τον κώδικά σας σε Python, μπορείτε να έχετε επιπλέον λειτουργίες και βοήθεια.

Στην έκδοση για Windows και για MacOS η Python έρχεται μαζί με ένα επιπλέον περιβάλλον προγραμματισμού σε Python, το IDLE, το οποίο σας επιτρέπει να γράφετε τον κώδικα σας και να τον εκτελείτε κατευθείαν μέσα από αυτό.

Εναλλακτικοί επεξεργαστές κειμένου που μπορείτε να χρησιμοποιήσετε και διανέμονται δωρεάν είναι οι ακόλουθοι:

- Notepad++ http://notepad-plus-plus.org/
- UltraEdit http://www.ultraedit.com/
- Sublime Text http://www.sublimetext.com/
- jEdit http://www.jedit.org/
- Geany http://www.geany.org/
- GEdit https://wiki.gnome.org/Apps/Gedit
- PSPad http://www.pspad.com/

Η πλειοψηφία αυτών των κειμενογράφων έρχονται με αρκετά πρόσθετες λειτουργίες, πέρα του syntax highlighting, που θα σας διευκολύνουν να γράψετε το κώδικά σας, όπως code navigation, code completion.

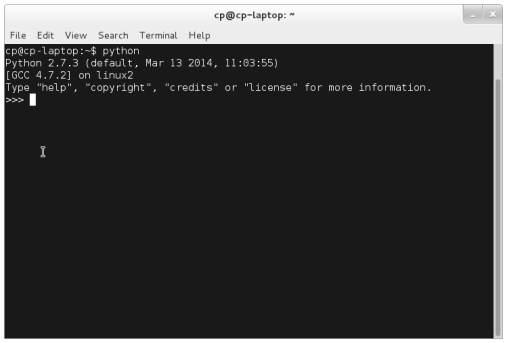
Αν θέλετε ένα πιο πλήρες περιβάλλον προγραμματισμού σε Python, ώστε να έχετε refactoring, unit testing, debugger, stack viewer κ.ά., μπορείτε να χρησιμοποιήσετε τα ακόλουθα:

- Netbeans https://netbeans.org/
- Komodo IDE http://komodoide.com/

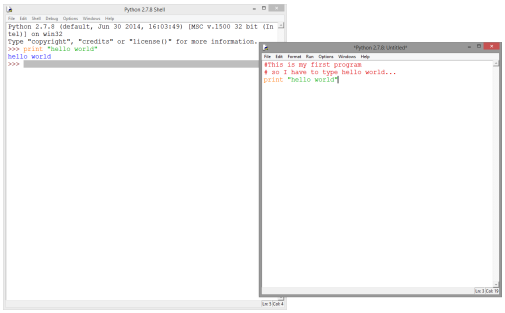
- KDevelop https://www.kdevelop.org/
- Spyder https://pythonhosted.org/spyder/
- Pyshield http://dashingsoft.com/wp/?page_id=9
- Wing IDE http://wingware.com/

Πέρα από τον να εκτελείτε προγράμματα της Python, μπορείτε να αλληλεπιδράσετε με τη γλώσσα ανοίγοντας ένα τερματικό και εκτελώντας την εντολή:

```
python
```



(α΄) Περιβάλλον της Python στο Linux.



(β΄) Περιβάλλον της Python σε Windows.

Σχήμα 1.1: Περιβάλλον της Python σε Linux και Windows

Εκτελώντας το παραπάνω, θα εμφανιστεί ένα περιβάλλον όπως αυτό στις Εικόνες 1.1α΄ και 1.1β΄, ανάλογα το λειτουργικό σας σύστημα. Δοκιμάστε τώρα να εκτελέσετε την εντολή:

```
print "Hello world"
```

Η Python κάνει διάκριση πεζών-κεφαλαίων (είναι case sensitive)! Αυτό σημαίνει ότι οι εντολές και οι μεταβλητές θα πρέπει να γράφονται όπως ακριβώς δίνονται στα παραδείγματα, διαφορετικά η Python δε θα τις αναγνωρίζει.

2 Εισαγωγή

Για την εμφάνιση δεδομένων χρησιμοποιούμε τη συνάρτηση ``print''. Προσέξτε τα ακόλουθα παραδείγματα όλα εκ των οποίων επιστρέφουν το ίδιο αποτέλεσμα

```
print "Hello World"
print """Hello World"""
print 'Hello World'
```

Δοκιμάστε να εκτυπώσετε δύο αλφαριθμητικά και προσέξτε τη διαφορά

```
print "Hello","World"
print "Hello"+"World"
```

Στην πρώτη περίπτωση το ένα αλφαριθμητικό εκτυπώνεται μετά το άλλο με κενό, ενώ στη δεύτερη τα ενώνει.

Στην Python, κάθε εντολή δίνεται σε μία γραμμή. Αν θέλω να εκτελέσω πολλές εντολές σε μία γραμμή πρέπει να προσθέσω το χαρακτήρα ``;'' στο τέλος κάθε εντολής.

```
print "Hello World"; print """Hello World"""
print 'Hello World'
```

Για να προσθέσετε σχόλια στον κώδικά σας υπάρχουν δύο τρόποι με το σύμβολο ``#'' και με τη χρήση τριπλών διπλών εισαγωγικών.

```
print "Hello World";
#This is a comment
print 'Hello World'
"""This is a comment as well"""
print 'Hello World' #comment...
```

Προφανώς ότι βρίσκεται μέσα σε σχόλια **δεν** εκτελείται.

Προσοχή: Τα τριπλά διπλά εισαγωγικά μπορούν να μας δώσουν σχόλια που εκτίνονται σε πολλές γραμμές, σε αντίθεση με τα σχόλια που γίνονται με το χαρακτήρα ``#''.

Προσοχή: Τα τριπλά διπλά εισαγωγικά μπορούν να χρησιμοποιηθούν και ως αλφαριθμητικά, συνεπώς πρέπει να προσέχουμε που τα εισάγουμε!

3 Πράξεις

Η Python έχει τους συνήθεις τελεστές (+-*/) για να κάνει πράξεις.

```
print 5+10
print 5-10
print 5*10
print 10/2
```

Η προτεραιότητα των πράξεων στην Python είναι η γνωστή και παρακάμπτεται μόνο από τις παρενθέσεις.

Προσοχή: Στην Python οι αγκύλες έχουν άλλη χρήση, συνεπώς **δεν** τις χρησιμοποιούμε σε μαθηματικές πράξεις.

```
print 5-10*20+(30+246)/3
```

Προσοχή: Ο τελεστής της δύναμης στην Python είναι το ``**'' και όχι το ``^''.

```
print 3**10
print 2**8
```

Προσέξτε το ακόλουθο αποτέλεσμα:

```
print 2/3
```

Ο λόγος που το αποτέλεσμα είναι 0 είναι ότι η Python πιστεύει ότι θέλουμε να κάνουμε πράξεις με ακεραίους, οπότε πράγματι η ακέραια διαίρεση επιστρέφει 0. Αν θέλουμε να έχουμε δεκαδικό, τότε πρέπει να χρησιμοποιήσουμε έναν από τους ακόλουθους τρόπους:

```
print 2/3.0
print 2.0/3
print (2*1.0)/3
```

Προσοχή: Ο ακόλουθος κώδικας δεν επιστρέφει το σωστό δεκαδικό αποτέλεσμα!

```
print (2/3)*1.0
```

Η Python έχει τη δυνατότητα να κάνει πράξεις με πολύ μεγάλους αριθμούς:

```
print 2**1024+2**2048+3
```

Επιπλέον μπορεί να εκτελεί λογικές πράξεις (θα αναλυθεί στη συνέχεια)

```
print 2>1
print 5+5>=11
print 5==5
print 1>=0
print -1<=-10
```

Προσέξτε τα διαφορετικά αποτελέσματα:

```
print 5+100100
print "5+100100"
print "5"+"100100"
```

Στην πρώτη περίπτωση η Python αναγνωρίζει ότι είναι αριθμοί και επιστρέφει το άθροισμα κανονικά. Στη δεύτερη περίπτωση, καταλαβαίνει ότι έχει ένα αλφαριθμητικό και το εμφανίζει. Τέλος, στην τελευταία περίπτωση καταλαβαίνει ότι έχει δύο αλφαριθμητικά και τα συνενώνει.

4 Μεταβλητές

Η Python κάνει δυναμική δέσμευση μνήμης συνεπώς **δε** χρειάζεται να δηλώνετε τις μεταβλητές πριν τη χρήση τους. Επιπλέον δε χρειάζεται να δηλώσετε τον τύπο τους, η Python θα επιλέξει τον κατάλληλο.

```
text="abcd"
num=3
print text,num
```

Επιπλέον, μία μεταβλητή στην Python μπορεί να αλλάξει τον τύπο της με μία απλή ανάθεση τιμής.

```
tmp="abcd"
print tmp
tmp=1234
print tmp
```

Με τις μεταβλητές μπορώ να κάνω πράξεις και τα αποτελέσματά τους να τα αναθέτω σε άλλες μεταβλητές

```
age1=40
age2=32
diff=age1-age2
print "The first age is", age1
print "the second age is",age2
print "and their difference is", diff
```

```
prod1=22.5
prod2=12.35
prod3=9.31
```

```
avg=(prod1+prod2+prod3)/3
print "The average price is", avg
```

5 Εμφάνιση δεδομένων

Χρησιμοποιώντας τον χαρακτήρα % μπορούμε να αντικαταστήσουμε μέρη του αλφαριθμητικού μας. Αν έχω μία αντικατάσταση μπορώ να κάνω το ακόλουθο:

```
myvarr=314159265359
print "Let's substitute an integer variable %d without breaking the quot
```

Στην περίπτωση που έχω πολλές μεταβλητές, θα πρέπει όλες οι μεταβλητές που θα αντικατασταθούν να μπουν μέσα σε παρένθεση.

```
var1=1
var2=2
var3=3
var4=10
print "I know my numbers %d,%d,%d up to %d" %(var1,var2,var3,var4)
```

Προσέξτε ότι οι αντικαταστάσεις που έγιναν αφορούσαν αριθμητικά δεδομένα και πιο συγκεκριμένα ακέραιους.

```
num1=138323
num2=3.14159265359
txt="abcdef"
print "Mixing numbers %d, %.3f with strings %s!" %(num1,num2,txt)
```

Συνεπώς:

	Μεταβλητή
%d	ακέραιος
%f	δεκαδικός
%s	αλφαριθμητικό

Προσοχή: Ο αριθμός μετά την τελεία στο %f δηλώνει το πλήθος των δεκαδικών ψηφίων που θα εμφανιστούν.

6 Λογικές Συνθήκες

Στην Python χρησιμοποιούμε την ακόλουθη δομή:

```
if (λογική συνθήκη):
  εντολές
elif (λογική συνθήκη):
  εντολές
else:
  εντολές
```

Οι λογικές συνθήκες μπορεί να είναι απλές προτάσεις ελέγχου π.χ. var_a<10 ή πιο σύνθετες πχ:

```
  age>18 and gender=="male" and
    (country=="Greece" or country=="Italy")
```

Προσοχή: Όταν γίνεται έλεγχος για ισότητα βάζουμε διπλό ίσον ==. Το απλό ίσο σημαίνει ανάθεση τιμής.

7 Επαναλήψεις

Για να επαναλάβω κάποιες διαδικασίες, μπορώ να χρησιμοποιήσω δύο εντολές της Python, την for και τη while. Στην πρώτη περίπτωση ορίζω το σύνολο των επαναλήψεων, ενώ στη δεύτερη η επανάληψη γίνεται όσο ισχύει μία λογική συνθήκη.

```
for i in range(10):
  print i
```

Προσοχή: Ο παραπάνω κώδικας θα κάνει 10 επαναλήψεις, ότι δηλώσαμε στο range, αλλά η μέτρηση ξεκινάει από το 0 και συνεπώς τελειώνει στο 9.

Αν θα ήθελα η αρίθμηση να αρχίζει από το 5 και όχι από το 0, πρέπει να αλλάξω την range. Έτσι θα έχω:

```
for i in range(5,10):
  print i
```

Αν επιθυμούμε η αρίθμηση να μη γίνεται με βήμα 1 αλλά ανά 10, τότε:

```
for i in range(5,100,10):
  print i
```

Επιπλέον μπορούμε να μετράμε και αντίστοφα, αρκεί να γίνει το βήμα αρνητικό και να αλλάξουμε τα όρια της range:

```
for i in range(100,5,10):
  print i
```

Μία επανάληψη μπορεί να εμπεριέχεται μέσα σε μία άλλη επανάληψη. Για παράδειγμα, ο ακόλουθος κώδικας εμφανίζει όλα τα δυνατά αποτελέσματα από τη ρήψη δύο ζαριών.

```
for i in range(1,7):
  for j in range(1,7):
    print i,j
```

Προσοχή: Αφού οι ενδείξεις ενός ζαριού ξεκινούν από το 1, πρέπει να δηλώσουμε στη range να ξεκινήσει από το 1. Επιπλέον επειδή οι ενδείξεις φτάνουν μέχρι και το 6, δηλώνουμε ότι η range φτάνει το 7.

8 Αρχεία

Τα αρχεία μπορούμε τα ανοίγουμε στην Python με την εντολή open, όπου πρέπει να δηλώσουμε το τι θέλουμε να κάνουμε με αυτά, να τα διαβάσουμε ή να γράψουμε δεδομένα σε αυτά.

Το ακόλουθο πρόγραμμα ανοίγει το αρχείο myfile.txt, που βρίσκεται στον ίδιο φάκελο, και εκτυπώνει κάθε του γραμμή.

```
file_in=open("myfile.txt","r")
for line in file_in:
  print line
file_in.close()
```

Το ακόλουθο πρόγραμμα ανοίγει το αρχείο myfile.txt και θέτει τα περιεχόμενά του να είναι ``new text''.

```
file_out=open("myfile.txt","w")
file_out.write("new text")
file_out.close()
```

Προσοχή: Αν το αρχείο δεν υπάρχει το δημιούργεί, ενώ αν το αρχείο υπάρχει, διαγράφει τα περιεχόμενά του και γράφει αυτά που του ζητάμε.

Αν θέλουμε να προσθέσουμε δεδομένα στο τέλος του αρχείου, θα πρέπει να ανοίξουμε το αρχείο με ``a'' αντί για ``w''.

9 Λίστες

Η Python έχει μία δομή η οποία ονομάζεται λίστα η οποία κρατάει ένα σύνολο τιμών. Ο ακόλουθος κώδικας καταχωρεί μία λίστα η οποία ονομάζεται a και τις αναθέτει τις τιμές 2,5,6,8 και 1.

```
a=[2,5,6,8,1]
```

Για να επιστρέψουμε την τιμή της λίστας σε κάποια θέση πρέπει να χρησιμοποιήσουμε τη θέση μέσα σε [].

```
a=[2,5,6,8,1]
print a[1]
```

Προσοχή: Η αρίθμηση των στοιχείων της λίστας ξεκινάει από το 0!

Μπορούμε να καλέσουμε στοιχεία, μετρώντας τη σειρά τους από το τέλος της λίστας χρησιμοποιώντας αρνητικούς αριθμούς. Ο ακόλουθος κώδικας θα εκτυπώσει την τελευταία και προτελευταία τιμή:

```
a=[2,5,6,8,1]
print a[-1],a[-2]
```

Προσοχή: Αν ζητήσουμε στοιχείο σε θέση μεγαλύτερη από το μήκος της λίστας η Python θα επιστρέψει πρόβλημα!

Για να αλλάξουμε την τιμή ενός στοιχείου, απλά ζητάμε το στοιχείο στη θέση που επιθυμούμε να κάνουμε την αλλαγή και κάνουμε την αντίστοιχη ανάθεση.

```
a=[2,5,6,8,1]
a[3]=11
print a
```

Για να προσθέσουμε ένα στοιχείο στο τέλος της λίστας χρησιμοποιούμε την append:

```
a=['a','b','c','d','e']
a.append('f')
print a
```

Αν θέλουμε να προσθέσουμε ένα στοιχείο σε συγκεκριμένη θέση της λίστας χρησιμοποιούμε την insert, όπου στη σύνταξή της δηλώνουμε τη θέση και την τιμή.

```
a=[2,5,6,8,1]
a.insert(1,100)
print a
```

Με την remove αφαιρούμε το πρώτο στοιχείο της λίστας που περιέχει την τιμή που του δηλώσαμε. Έτσι ο κώδικας:

```
a=[2,5,1,6,8,1]
a.remove(1)
print a
```

θα επιστρέψει τη λίστα [2,5,6,8,1].

Με την count μετράμε πόσα στοιχεία έχουν την τιμή που ζητήσαμε. Έτσι ο κώδικας:

```
a=[1,2,5,1,6,8,1]
print a.count(1)
```

θα επιστρέψει την τιμή 3.

Για να αντιστρέψουμε τα στοιχεία μίας λίστας χρησιμοποιούμε τη reverse:

```
a=[1,2,3,4,5]
a.reverse()
print a
```

Μπορούμε να φτιάξουμε λίστες οι οποίες είναι μέρος μίας άλλης λίστας. Πιο συγκεκριμένα αν a είναι η λίστα [1,2,3,4,5,6,7,8,9,10], τότε:

Κλήση	Αποτέλεσμα
a[3:8]	[4, 5, 6, 7, 8]
a[:8]	[1, 2, 3, 4, 5, 6, 7, 8]
a[:-1]	[1, 2, 3, 4, 5, 6, 7, 8, 9]
a[4:]	[5, 6, 7, 8, 9, 10]
a[::2]	[1, 3, 5, 7, 9]
a[1::2]	[2, 4, 6, 8, 10]

Προσέξτε το πως ζητάμε τα ``μέρη'' της λίστας, η σύνταξη είναι [αρχή, τέλος, βήμα]

Για να διαγράψουμε στοιχεία μίας λίστας μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις del και pop

```
a=[1,2,3,4,5]
del a[1]
print a
```

Ο παραπάνω κώδικας διαγράφει το στοιχείο της λίστας a στη θέση 1 (δεύτερη θέση).

Η pop αφαιρεί το τελευταίο στοιχείο της λίστας και το αφαιρεί.

```
a=[1,2,3,4,5]
b=a.pop()
print b,a
```

Ο κώδικας παραπάνω θα θέσει στο b την τιμή 5, ενώ η λίστα a θα είναι πλέον [1,2,3,4]. Η pop επιπλέον μπορεί να διαγράψει και ένα στοιχείο το οποίο βρίσκεται σε άλλη θέση.

```
a=[1,2,3,4,5]
b=a.pop(2)
print b,a
```

Ο κώδικας παραπάνω θα θέσει στο b την τιμή 3, ενώ η λίστα a θα είναι πλέον [1,2,4,5].

Στις λίστες μπορώ να εφαρμόσω διάφορες συναρτήσεις. Πιο συγκεκριμένα, η len επιστρέφει το μήκος της λίστας, η max το μέγιστο στοιχείο, η min το ελάχιστο στοιχείο, ενώ τέλος η sum το άθροισμα των στοιχείων.

```
a=[11,62,30,45,15]
print len(a), max(a), min(a), sum(a)
```

Ο παραπάνω κώδικας θα επιστρέψει τις τιμές: 5 62 11 163.

Μπορούμε να ενώσουμε δύο λίστες:

```
a=[1,2,3,4,5]
b=[6,7,8]
c=a+b
print c
```

Η απλά να ενώσουμε προσθέσουμε σε μία λίστα στοιχεία μέσω της extend:

```
a=[1,2,3,4,5]
a.extend([6,7,8])
print a
```

Για να αντιγράψουμε τα στοιχεία μίας λίστας σε μία άλλη χρησιμοποιούμε τη list.

```
a=[1,2,3,4,5]
b=list(a)
c=a
a.append(6)
print a,b,c
```

Προσοχή: Με το να θέσουμε μία λίστα ίση με μία άλλη δεν αντιγράφονται τα στοιχεία, αλλά οι δύο λίστες δείχνουν στην ίδια λίστα για αυτό και τον προηγούμενο κώδικα η c περιέχει και το στοιχείο 6, ενώ η b δεν το περιέχει.

Για να ελένξουμε αν δύο λίστες περιέχουν τα ίδια στοιχεία χρησιμοποιούμε την cmp.

```
a=[1,2,3,4,5]
b=[1,2,3,4,5]
c=[1,2,3]
print cmp(a,b)
print cmp(a,c)
print cmp(c,a)
```

Προσοχή: Η cmp δεν επιστρέφει True/False, αλλά ένα αριθμό. Το 0 σημαίνει ότι οι λίστες έχουν τα ίδια στοιχεία, -1 ή 1 ότι τα στοιχεία δεν είναι ίσα.

Η Python διαθέτει τη συνάρτηση sort και την sorted για να βάζει στη σειρά τα στοιχεία μίας λίστας. Προσέξτε τη διαφορά στη χρήση τους:

```
a=[27,35,14,95,11]
b=[45,8,99,11,3,0]
a.sort()
print a
c=sorted(b)
print b
print c
```

Θα μπορούσαμε επιπλέον να αντιστρέψουμε τη σειρά:

```
a=[27,35,14,95,11]
b=[45,8,99,11,3,0]
a.sort(reverse=True)
print a
c=sorted(b,reverse=True)
print b
print c
```

```
fin=open("a.txt","r")
lst=[]
for line in fin:
        line=line.strip()
        lst.append( line)
headers= lst[0].split(",")
values= lst[-1].split(",")
tmp="<html><title>Meteo</title><body><h1>Meteo</h1>"
for i in range(len(headers)):
        tmp+= "%s : %s<br>" %(str(headers[i]),str(values[i]))
tmp+="</body></html>"
fout=open("out.html","w")
fout.write(tmp)
fout.close()
```