

# Υπολογιστικές Μέθοδοι στη Φυσική

**ΦΥΣ 145**

Άνοιξη 2021

Διδάσκων:

**Φώτης Πτωχός**

e-mail: [fotis@ucy.ac.cy](mailto:fotis@ucy.ac.cy)

Τηλ: 22.89.2837

Γραφείο: B235 – ΘΕΕ02

# Γενικές Πληροφορίες

## ▪ Ώρες/Αίθουσα διδασκαλίας:

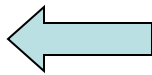
- Δευτέρα 11:30 – 13:00
- Αίθουσα: 012 ΦΥΣΙΚΟΥ

Διαλέξεις

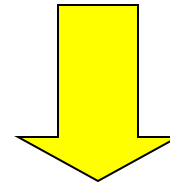


## Εργαστήρια

- Δευτέρα 14:30 – 18:00
- Αίθουσα: 012 Φυσικής



ΥΠΟΧΡΕΩΤΙΚΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗ



Πάνω από μια απουσία στα εργαστήρια χωρίς σημαντικό λόγο ισοδυναμεί με αυτόματη αποτυχία στο μάθημα

# Απορίες

- Διακόπτεται για απορίες κατά την διάρκεια των διαλέξεων.
- Ρωτάτε κατά την διάρκεια των εργαστηρίων
- Στείλτε μου e-mail, θα μπορούσαμε να κανονίσουμε συναντήσεις διαδικτυακά για να λύσετε απορίες
- Πείτε μου αν κάτι στο μάθημα δεν δουλεύει για σας και πως μπορεί να αλλάξει
- Ο Δρ. Mousa θα βοηθήσει στα εργαστήρια ώστε να χωριστείτε σε ομάδες και να μπορέσουμε να απαντήσουμε στις ερωτήσεις σας κατά τη διάρκεια των εργαστηρίων
- Δύο μεταπτυχιακοί συνάδελφοί σας θα είναι παρόντες κατά τη διάρκεια των εργαστηρίων και θα βοηθήσουν σε οποιαδήποτε προβλήματα

# Βιβλιογραφία

Δεν θα ακολουθήσω κάποιο συγκεκριμένο βιβλίο.

Οι διαλέξεις/σημειώσεις θα είναι στο web.

Οι ασκήσεις/οδηγίες των εργαστηρίων όπως και οι **εργασίες** που θα δουλεύετε σπίτι θα τις βρίσκετε επίσης στο web.

(<http://www2.ucy.ac.cy/~fotis/phy145/phy145.html>)

Χρήσιμη βιβλιογραφία/παραδείγματα μπορούν να βρεθούν στο web.  
Προσπαθήστε και μόνοι σας τις πρώτες δύο βδομάδες να ασχοληθείτε και να εξοικειωθείτε με

Λειτουργικό σύστημα: **Linux**

Γλώσσα προγραμματισμού: **Python v3**

Λογισμικό Γραφικών αναπαραστάσεων: **μέσα στην Python**

Όλοι έχετε accounts στο τμήμα της Φυσικής και μπορείτε να χρησιμοποιείται τους υπολογιστές του τμήματος για πρακτική εξάσκηση πέρα από τις ώρες εργαστηρίου.

- Στην ιστοσελίδα του μαθήματος υπάρχουν οδηγίες για εγκατάσταση του λειτουργικού centos (linux) σε περιβάλλον Windows μέσω της χρήσης του virtual box

## Βιβλιογραφία

- Για Linux θα μπορούσατε να χρησιμοποιήσετε τα ακόλουθα:
  - [Linux in a nutshell](#) από Ellen Siever, Stephen Spainhour, Stephen Figgins and Jessica P. Hekman , ed. O'Reilly
  - [Running Linux](#) από Matt Welsh, Matthias Kalle Dalheimer, and Lar Kaufman, ed. O'Reilly
- Πολλά μπορούν να βρεθούν επίσης χρησιμοποιώντας την μηχανή αναζήτησης στο web [google](#)
- Για το κύριο μέρος του μαθήματος: **Python και Φυσική**
  - **Programming in Python 3: “A complete introduction to the python language”** από O'Reilly .
  - **Learning Python** by David Ascher and Mark Lutz
  - **Python Crash course** από E. Matthes
  - **Learning Scientific programming with Python**, C. Hill, Cambridge University Press
  - **A student's guide to python for physical modeling**, από J. Kinder και P. Nelson, Princeton University Press
  - **Essential Python for Physicists**, από G. Moruzzi, Springer

# Βαθμολογία

- Η βαθμολογία θα βασιστεί στα ακόλουθα:
  - **15%** εργασίες στο σπίτι
  - **15%** quiz την ώρα των εργαστηρίων
  - **30%** 1 εξέταση προόδου (Δευτέρα 1 Μάρτη, 9:30–11:30 & 12:00 – 14:00)
  - **40%** τελική εξέταση (Μάη)



## Εργασίες στο σπίτι

- Εργασίες για το σπίτι θα περιέχουν 4–5 ασκήσεις
- Μπορείτε να συνεργάζεστε με συναδέλφους σας αλλά θα πρέπει να είναι .... δικιές σας και όχι αντιγραφή λύσεων άλλων συναδέλφων σας.

## Περιεχόμενο του Μαθήματος

- Σύντομη επανάληψη των βασικών εντολών της γλώσσας Python, βασικές εντολές λειτουργικού συστήματος, εισαγωγή και βασικές λειτουργίες του [Editor/Emacs](#). Λογισμικό πακέτο γραφικών παραστάσεων δεδομένων.
- Χρήση γραφικών παραστάσεων σε 2 και 3 διαστάσεις.
- Επίλυση προβλημάτων φυσικής (επίλυση διαφορικών εξισώσεων, και ολοκλήρωση) με χρήση αλγορίθμων μεγάλης ακρίβειας.
- Εφαρμογή των μεθόδων σε κινούμενα σχήματα και κατασκευές
- Ανάλυση πειραματικών δεδομένων.
- Χρήση τυχαίων αριθμών για προσομοίωση διαφόρων φαινομένων
- Εισαγωγή στις ιδέες της μηχανικής μάθησης (machine learning)

# Βασικές εντολές PYTHON

## ■ PYTHON:

- Ξεκίνησε σαν γλώσσα προγραμματισμού στις αρχές της δεκαετίας του 90. Βρήκε ευρεία εφαρμογή εξαιτίας της απλότητάς της. Τα τελευταία 10 χρόνια χρησιμοποιείται η έκδοση Python v3
- Χρειάζεται να μάθουμε την γλώσσα προγραμματισμού Python ώστε να μπορέσουμε να περάσουμε στον υπολογιστή τις οδηγίες/εντολές που θέλουμε.
- Η γλώσσα χρησιμοποιείται σε μια σειρά εφαρμογών: Ανάπτυξη internet και web, επιστημονική χρήση.
- Όταν κάνουμε λάθος σε κάποια οδηγία ο υπολογιστής μας υποδεικνύει το λάθος
- Πρέπει να είμαστε επίμονοι για να μάθουμε τη γλώσσα. Άλλωστε είναι ευκολότερο για μας να μάθουμε τη γλώσσα από τον υπολογιστή να μάθει τη γλώσσα μας/



# Πρόγραμμα

- ❑ Πρόγραμμα είναι μια σειρά από οδηγίες (εντολές προγράμματος) συνταγμένες με τέτοιο τρόπο ώστε να επιτρέψει έναν ηλεκτρονικό υπολογιστή να λύσει κάποιο πρόβλημα. Το πρόβλημα προς λύση είναι σπασμένο σε πολύ μικρότερα κομμάτια. Τα κομμάτια αυτά θα πρέπει να σχηματίζουν μια καλώς ορισμένη δομή, όπου το πιο περίπλοκο και μεγάλο πρόβλημα στην κορυφή και το πιο απλό και εύκολα επιλύσιμο στη βάση.
  - Από κει και ο ορισμός: top down programming: προγραμματισμός από πάνω προς τα κάτω
- ❑ Για να πετύχετε τη λύση ενός προβλήματος με την χρήση μιας γλώσσας προγραμματισμού καλό είναι να ακολουθήσετε τα παρακάτω βήματα:
  - ❑ Εκφράστε το πρόβλημα στα Αγγλικά
  - ❑ Εξετάστε το πρόβλημα και σπάστε το σε μικρότερα προβλήματα
  - ❑ Εξετάστε τα επιμέρους τμήματα και επεξεργαστείτε τα σε μικρότερα
  - ❑ Σχεδιάστε γραφικά το σχέδιο δράσης
  - ❑ Δοκιμάστε το πρόγραμμα

# Χρήση Python ως υπολογιστική

Η python μπορεί να χρησιμοποιηθεί για υπολογισμό αλγεβρικών πράξεων:

Πληκτρολογώντας την εντολή **python3** εισέρχεστε σε περιβάλλον της python

```
bash-3.2$ python3
Python 3.7.6 (default, Dec 22 2019, 01:09:06)
[Clang 11.0.0 (clang-1100.0.33.12)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

---

Μπορούμε να κάνουμε ανάθεση τιμών σε διάφορες μεταβλητές:

```
>>> x = 100
>>> █
```

---

Και να δούμε την τιμή αυτή χρησιμοποιώντας την εντολή **print()**:

```
>>> print(x)
100
>>> █
```

---

Μπορούμε να κάνουμε διάφορες αριθμητικές πράξεις:

```
>>> x+10  Πρόσθεση
110
>>> x-10  Αφαίρεση
90
>>> x*10  Πολλαπλασιασμός
1000
>>> x/10  Διαίρεση
10.0
>>> x**2  Εκθέτης **
10000
>>> x%3   Υπόλοιπο διαίρεσης %
1
>>> x//49 Ακέραιο μέρος διαίρεσης και αποκοπή δεκαδικών ψηφίων //
2
>>> █
```

---

## Σημασία του $a=a+1$

Για να επικοινωνήσουμε με τον Η/Υ χρειάζεται να χρησιμοποιήσουμε μια γλώσσα κατανοητή στον Η/Υ. Κάποιες από τις εντολές μπορεί να μοιάζουν περίεργες

```
>>>  
>>> a = 100  
>>> a = a+1  
>>> print(a)  
101  
>>> 
```

← Μαθηματικά δεν έχει νόημα

Ωστόσο σημαίνει: (1) Ανέθεσε σε μια μεταβλητή  $a$  την τιμή 100 και (2) πρόσθεσε στην μεταβλητή  $a$  μία μονάδα και το αποτέλεσμα ανέθεσέ στην ίδια μεταβλητή

Οι πράξεις στον Η/Υ γίνονται πάντοτε στο δεξί μέλος της εξίσωσης.


Το  $=$  σημαίνει ανάθεση του αποτελέσματος του δεξιού μέλους σε μια μεταβλητή που κρατά μια θέση στη μνήμη του υπολογιστή αφού πρώτα διαγράψει ότι υπήρχε στη θέση αυτή της μνήμης

- Το σύμβολο  $=$  δίνει οδηγία στη Python να αλλάξει την κατάσταση της.
- **Προσοχή:**  $a+1 = a$  δεν αποτελεί μια σωστή εντολή για οποιαδήποτε γλώσσα προγραμματισμού παρόλο που μαθηματικά είναι σωστή


## Ανάγκη για έλεγχο ισότητας

Πολλές φορές χρειάζεται να εξετάσουμε αν μια μεταβλητή έχει συγκεκριμένη τιμή

Για αποφυγή συγχύσεων μεταξύ ανάθεσης και ελέγχου ισότητας οι περισσότερες γλώσσες χρησιμοποιούν το σύμβολο της **διπλής ισότητας == για έλεγχο ισότητας**

```
>>>  
>>> a=1  
>>> a==0  
False  
>>> 
```

Θα πρέπει να αποφεύγονται μικτές αναθέσεις της μορφής:

```
>>>  
>>> b = (a==1)  
>>> print(b)  
True  
>>> 
```

Έλεγχος αν `a == 1` και ανάθεση της λογικής πράξης (**true ή false**) στη νέα μεταβλητή `b`

## Ανάγκη για έλεγχο ισότητας

Πολλές φορές χρειάζεται να εξετάσουμε αν μια μεταβλητή έχει συγκεκριμένη τιμή

Για αποφυγή συγχύσεων μεταξύ ανάθεσης και ελέγχου ισότητας οι περισσότερες γλώσσες χρησιμοποιούν το σύμβολο της **διπλής ισότητας == για έλεγχο ισότητας**

```
>>>
>>> a=1
>>> a==0
False
>>> ■
```

Θα πρέπει να αποφεύγονται μικτές αναθέσεις της μορφής:

```
>>>
>>> b = (a==1)
>>> print(b)
True
>>> ■
```

Έλεγχος αν  $a == 1$  και ανάθεση της λογικής πράξης (**true ή false**) στη νέα μεταβλητή  $b$

Ιδιαίτερη προσοχή για τις περιπτώσεις που ξεκινώντας το περιβάλλον Python υπάρχουν εντολές της μορφής  $b = a^2 - a$

Είναι μαθηματικά σωστή αλλά δεν έχει γίνει ανάθεση τιμής στην μεταβλητή  $a$

```
>>> b = a**2 - a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>> ■
```



## Συστατικά του περιβάλλοντος python

**PYTHON:** η γλώσσα προγραμματισμού. Πως θα περιγράψουμε έναν αλγόριθμο στον ηλεκτρονικό υπολογιστή

**python** (ή **python3**): μια εφαρμογή που επιτρέπει να χρησιμοποιούμε τη γλώσσα άμεσα στο terminal

**NumPy:** βιβλιοθήκη που δίνει αριθμητικούς πίνακες και μαθηματικές συναρτήσεις

**PyPlot:** βιβλιοθήκη που δίνει διάφορα εργαλεία για απεικόνιση

**SciPy:** βιβλιοθήκη που δίνει διάφορα εργαλεία για επιστημονικές υπολογιστικές μεθόδους

## Ξεκινώντας την python

**python:** Πληκτρολογώντας python σε terminal ξεκινάτε το περιβάλλον python και μπορείτε να αρχίσετε να πληκτρολογείτε εντολές


Υπάρχουν περιπτώσεις που η python φαίνεται να μην αντιδρά στις εντολές που δίνονται. Αυτό οφείλεται κυρίως στο γεγονός ότι κάποια (, [, { δεν έχει την αντίστοιχη ), ], } για να κλείσει και η Python διαβάζει περισσότερες εντολές μέχρι να βρει την ), ], } που θα κλείνει αυτή που είναι ακόμη ανοικτή

Αν δεν βρίσκεται ποιο block (, [, { είναι ανοικτό μπορείτε να βάλετε <ESC>

Η ανάθεση μιας τιμής σε μεταβλητή δεν εμφανίζεται στην οθόνη. Για να δείτε την τιμή θα πρέπει να δώσετε την εντολή:

`print (variable)` ή `print (variable1, variable2,...,variableN)`

Μπορείτε να κάνετε επίσης πολλαπλές αναθέσεις με μια μόνο εντολή:

```
>>> a =1; b=2; c=3
>>> A, B, C = 5, 10, 15
>>> print(a,b,c)
1 2 3
>>> print(A,B,C)
5 10 15
>>> 
```

## Help για τις εντολές

Μπορεί να μην ξέρετε αν υπάρχει κάποια εντολή στην python. Μπορείτε να δείτε αν υπάρχει η εντολή δίνοντας την εντολή:

```
help(round)
```

```
-----
```

```
round(number, ndigits=None)
```

```
Round a number to a given precision in decimal digits.
```

```
The return value is an integer if ndigits is omitted or None. Otherwise  
the return value has the same type as the number. ndigits may be negative.
```

```
(END)
```

Αν πληκτρολογήσετε το γράμμα q βγαίνετε από το help menu



## Python script (οδηγία)

Να δίνουμε εντολές απευθείας στην Python είναι χρήσιμο για να δοκιμάσουμε κάτι ή για 2-3 εντολές μόνο

Όταν οι οδηγίες που να δώσουμε είναι πολλές, τότε χρησιμοποιούμε έναν επεξεργαστή για να γράψουμε τις εντολές σε ένα αρχείο και λέμε στην Python να τρέξει τις εντολές δίνοντας το αρχείο

Δίνουμε επομένως στην Python την οδηγία ή την συνταγή για να τρέξει και να μας πει κάποιο αποτέλεσμα υπολογισμών

Συνήθως το όνομα του αρχείου που περιέχει τις οδηγίες, έχει το ακρονύμιο **".py"**

# Βήματα ενός προγράμματος ή ροή ενός προγράμματος

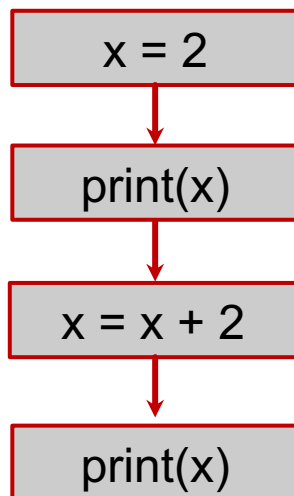
Όπως οι συνταγές, έτσι και το πρόγραμμα είναι ένα πλήθος από οδηγίες που πρέπει να εκτελεστούν με συγκεκριμένη σειρά για το επιθυμητό αποτέλεσμα.

Κάποια από τα βήματα αυτά είναι **υπό συνθήκη** και μπορεί να προσπεραστούν

Κάποια από τα βήματα αυτά είναι **επαναλήψιμα** και εκτελούνται πολλές φορές

Κάποια από τα βήματα/οδηγίες εμφανίζονται πολλές φορές σε διάφορα τμήματα του προγράμματος. Στις περιπτώσεις αυτές μπορούμε να κρατήσουμε τις οδηγίες συγκεντρωμένες κάπου να τις καλούμε όποτε χρειάζεται

Όταν το πρόγραμμα «τρέχει» τότε ρέει από τη μία οδηγία στην άλλη. Η δουλειά μας είναι να δώσουμε τη διαδρομή που θα ακολουθήσει το πρόγραμμα.



Program

```
x = 2
```

```
print(x)
```

```
x = x + 2
```

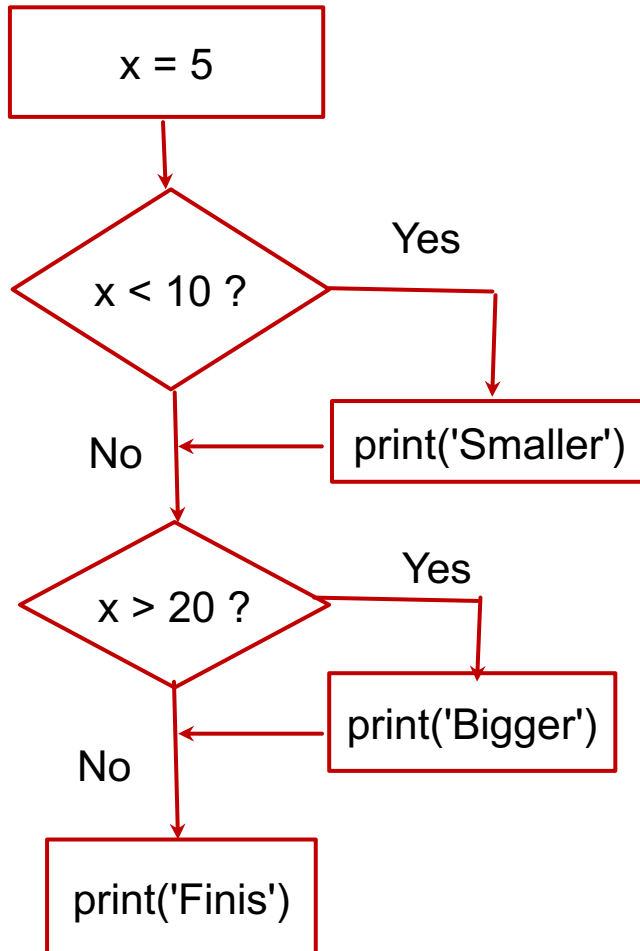
```
print(x)
```

Output:

2

4

## Βήματα υπό συνθήκη



Program:

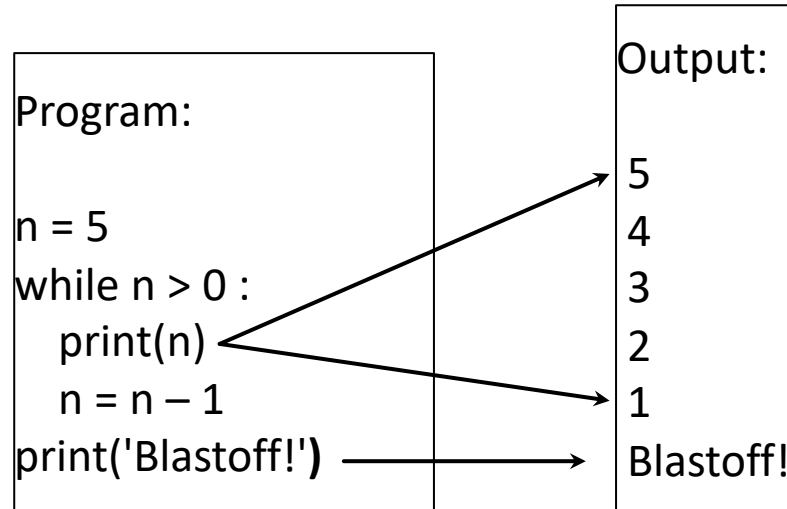
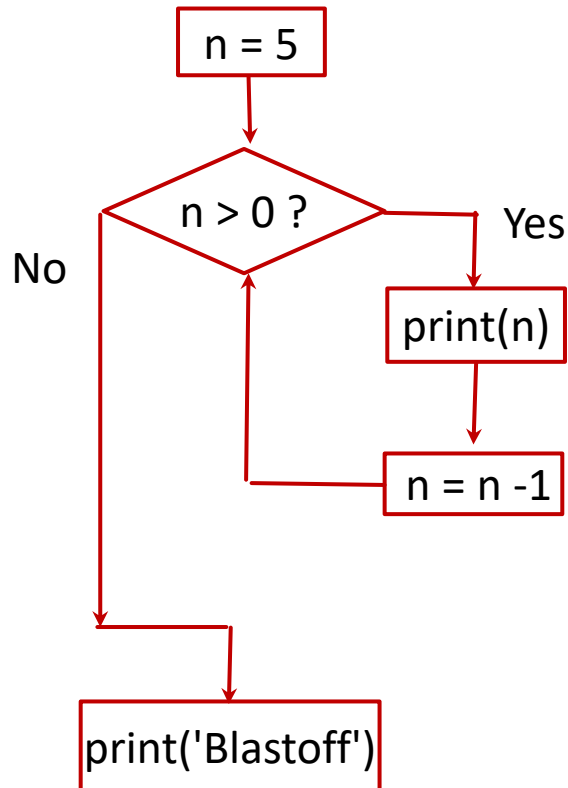
```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')
print('Finis')
```

Output:

Smaller

Finis

## Επαναλήψιμα βήματα



Οι μεταβλητές αλλάζουν κάθε φορά που εκτελείται το loop (ο βρόχος)

# Πρόγραμμα

```
name = input('Enter file:')  
handle = open(name, 'r')  
  
counts = dict()  
for line in handle:  
    words = line.split()  
    for word in words:  
        counts[word] = counts.get(word,0) + 1  
  
bigcount = None  
bigword = None  
for word,count in counts.items():  
    if bigcount is None or count > bigcount:  
        bigword = word  
        bigcount = count  
  
print(bigword, bigcount)
```

Διαδοχικά βήματα

Βήματα υπό συνθήκη

Βήματα επαναλήψιμα

# Πρόγραμμα

```
name = input('Enter file:')  
handle = open(name, 'r')  
  
counts = dict()  
for line in handle:  
    words = line.split()  
    for word in words:  
        counts[word] = counts.get(word,0) + 1  
  
bigcount = None  
bigword = None  
for word,count in counts.items():  
    if bigcount is None or count > bigcount:  
        bigword = word  
        bigcount = count  
  
print(bigword, bigcount)
```

Ενα μικρό αρχείο το οποίο δείχνει πως μετρούμε λέξεις με Python

Μια «λέξη» που χρησιμοποιείται για να διαβάσουμε δεδομένα από έναν χρήστη

Μια «πρόταση» για το πως αλλάζουμε τον πλήθος των μετρούμενων λέξεων

Μια «παράγραφος» για το πως θα βρεθεί το μεγαλύτερο στοιχείο σε μια λίστα αριθμών

# Κύρια συστατικά ενός προγράμματος

Σταθερές

Μεταβλητές

Αριθμητικές πράξεις

Ανάθεση τιμών – ισότητες

Έλεγχος αποτελέσματος – επικοινωνία με το πρόγραμμα

# Σταθερές (**constants**)

- ❑ Είναι οι ποσότητες το μέγεθος των οποίων δεν αλλάζει κατά την εκτέλεση ενός προγράμματος.
- ❑ Μπορεί να είναι αριθμητικές ή να περιέχουν χαρακτήρες
- ❑ Οι σταθερές χαρακτήρων «**strings**» ορίζονται με `' '` ή διπλό `" "`
  - **INTEGER** (ακέραια σταθερά)
  - **FLOAT** (πραγματική σταθερά)
  - **DOUBLE PRECISION** (σταθερά διπλής ακρίβειας)
  - **STRING** (σταθερά χαρακτήρων)
  - **LOGICAL** (λογική σταθερά) (αργότερα)

```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Hello world')
Hello world
```



## Μεταβλητές

Οι μεταβλητές είναι ονοματισμένες θέσεις στη μνήμη του υπολογιστή που χρήστης μπορεί να αποθηκεύσει δεδομένα και κατόπιν χρησιμοποιώντας το όνομα της μεταβλητής να αποκτήσει πρόσβαση στα δεδομένα

Τα ονόματα των μεταβλητών επιλέγονται από τον προγραμματιστή

Οι μεταβλητές αποθηκεύουν τα δεδομένα με μια εντολή ανάθεσης. Με μια εντολή ανάθεσης μπορούμε να αλλάξουμε το περιεχόμενο μιας μεταβλητής

x = 12.2

x

~~12.2 100~~

y = 14

y

14

x = 100


## Ονομασία μεταβλητών

- ❑ Ονόματα μεταβλητών μπορούν να αρχίζουν από **γράμμα** ή **\_** και ακολουθεί μια σειρά χαρακτήρων ή αριθμών
- ❑ Μπορεί να υπάρχει **\_**
- Δεν μπορεί να υπάρχει κενός χαρακτήρας στην ακολουθία των χαρακτήρων
- Δεν μπορεί να υπάρχει οποιοδήποτε σύμβολο όπως **\$ # % ^ & \* !**
- ❑ Τα ονόματα των μεταβλητών είναι διαφορετικά για κεφαλαία ή μικρά γράμματα π.χ. AB είναι διαφορετική από ab
- Υπάρχουν συγκεκριμένα ονόματα που αντιστοιχούν σε συναρτήσεις ή τελεστές της γλώσσας και θα πρέπει να αποφεύγονται:

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

## Python Modules/Συναρτήσεις

Υπάρχουν συναρτήσεις όπως η `sqrt` που δεν βρίσκονται στην python αν αν πληκτρολογήσουμε το όνομά τους

```
>>> sqrt(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqrt' is not defined
>>> 
```


Πληθώρα συναρτήσεων που γράφηκαν από άλλους χρήστες

Για να αποκτήσουμε πρόσβαση σε αυτές δίνουμε την εντολή **import**  
**import numpy**      **(Numerical Python)**

Για να δείτε όλες τις συναρτήσεις που περιέχονται στο module numpy  
**dir(numpy)**      Λίστα όλων των συναρτήσεων

**numpy.lookfor('sqrt')**      Ψάξιμο για συγκεκριμένη συνάρτηση

**numpy.sqrt(2)**      Χρήση της συνάρτησης sqrt


```
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.sqrt(2)
1.4142135623730951
>>> 
```

## Python Modules/Συναρτήσεις

Ένας άλλος τρόπος για εισαγωγή των συναρτήσεων χωρίς να πληκτρολογούμε συνέχεια το όνομα του module numpy είναι

**from numpy import \***


**sqrt(2)**

```
Type "help", "copyright", "credits" or "license" for more information.  
>>> from numpy import *  
>>> sqrt(2.)  
1.4142135623730951  
>>> 
```

---

Δημιουργείται πρόβλημα αν υπάρχει και άλλο module με το ίδιο όνομα συνάρτησης

Το module **math** έχει συνάρτηση sqrt επίσης.

```
Type "help", "copyright", "credits" or "license" for more information.  
>>> import numpy  
>>> import math  
>>> math.sqrt(2)  
1.4142135623730951  
>>> numpy.sqrt(2)  
1.4142135623730951  
>>> 
```

To module **math** έχει συνάρτηση sqrt επίσης.

---

Μπορούμε να χρησιμοποιήσουμε nickname για τα modules.


**import numpy as np**    np είναι τώρα ένα nickname για το module numpy.

**np.sqrt(2)**

# Python Modules/Συναρτήσεις

Ένας ακόμα τρόπος για εισαγωγή των συναρτήσεων

**from numpy import sqrt, exp** Προσθήκη μόνο συγκεκριμένων συναρτήσεων


```
Type "help", "copyright", "credits" or "license" for more information.  
>>> from numpy import sqrt, exp  
>>> sqrt(2)  
1.4142135623730951  
>>> exp(0)  
1.0  
>>> 
```

---

Προσθήκη συνάρτησης με nickname:

**from numpy.random import random as rng**

Έχουμε ένα module random μέσα στο module numpy. Το module random έχει μια συνάρτηση που λέγεται random και της δίνουμε το nickname rng

```
>>> from numpy.random import random as rng  
>>> rng()  
0.3360322594431543  
>>> 
```

---

Είναι σαν να έχουμε δώσει την εντολή: `numpy.random.random`

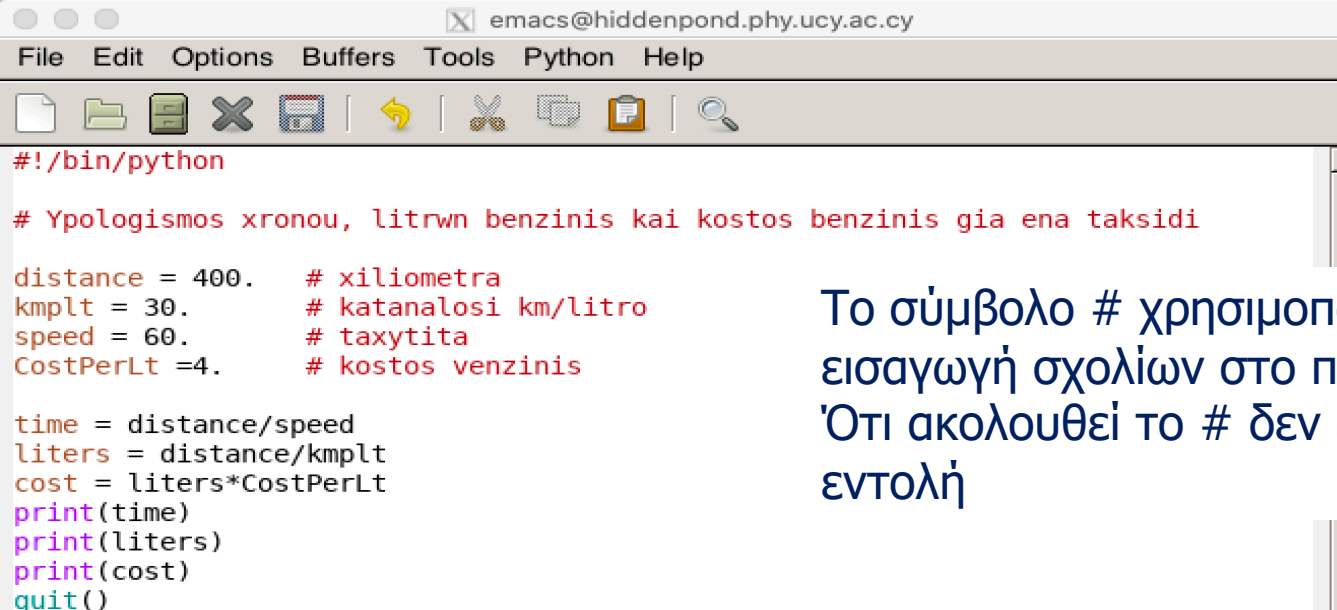
Είναι καλό σε κάθε python session να δίνετε τις εντολές:

**import numpy as np**

**import matplotlib.pyplot as plot**

# Πρώτο πρόγραμμα python

Χρησιμοποιώντας τον editor **emacs** μπορούμε να γράψουμε το πρώτο πρόγραμμα **mytrip.py**

A screenshot of the Emacs editor window. The title bar shows 'emacs@hiddenpond.phy.ucy.ac.cy'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Python', and 'Help'. The toolbar contains icons for file operations and editing. The code in the buffer is as follows:

```
#!/bin/python

# Υπολογισμος χρονου, litrwn benzinis kai kostos benzinis gia ena taksidi

distance = 400.    # xiliometra
kmplt = 30.        # katanalosi km/litro
speed = 60.        # taxytita
CostPerLt = 4.     # kostos venzinis

time = distance/speed
liters = distance/kmplt
cost = liters*CostPerLt
print(time)
print(liters)
print(cost)
quit()
```

Το σύμβολο # χρησιμοποιείται για εισαγωγή σχολίων στο πρόγραμμα. Ότι ακολουθεί το # δεν είναι εκτελέσιμη εντολή

Μπορούμε να τρέξουμε το πρόγραμμα δίνοντας στο terminal την εντολή:

## **python mytrip.py**

Το ίδιο μπορεί να συμβεί αν μπούμε πρώτα σε περιβάλλον pythhon δίνοντας την εντολή python στο terminal και κατόπιν την εντολή να δώσουμε:

```
bash-3.2$ python
Python 3.7.6 (default, Dec 22 2019, 01:09:06)
[Clang 11.0.0 (clang-1100.0.33.12)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import mytrip
6.666666666666667
13.333333333333334
53.333333333333336
```

## Δεύτερο πρόγραμμα

Απόσταση δύο σημείων στο χώρο: **distance.py**

```
#!/bin/python

# Υπολογισμός της απόστασης δύο σημείων στο χώρο

import numpy as np  ← Χρειάζεται να κάνουμε import την numpy βιβλιοθήκη

x1, y1, z1 = 10.0, 5.9, 4.0
x2, y2, z2 = -3.0, 1.5, 3.0

dR = (x1 - x2)**2 + (y1-y2)**2 + (z1-z2)**2

dR = np.sqrt(dR)

print(dR)

quit()
```

Τρέχουμε το πρόγραμμα είτε με την εντολή: `python distance.py`  
ή αφού ξεκινήσει το περιβάλλον `python` με την εντολή:

```
python
import distance
```

## Επικοινωνία με το προγράμματος - πληκτρολογίου

Σε αρκετές περιπτώσεις θέλουμε να είμαστε σε θέση να περάσουμε δεδομένα σε ένα πρόγραμμα από το πληκτρολόγιο

Η python έχει μία συνάρτηση που ονομάζεται **input** για να δεχθεί input από τον χρήστη και να την αναθέσει σε μια μεταβλητή. Η μορφή της εντολής είναι:

**strname = input("prompt to user")**

Όταν εκτελεστεί η εντολή input, εμφανίζει στην οθόνη του υπολογιστή το text που περικλείεται στα " " και περιμένει input από τον χρήστη

Ο χρήστης πληκτρολογεί μια γραμματοσειρά και πατά το return

Η συνάρτηση input αναθέτει την γραμματοσειρά που πληκτρολόγησε ο χρήστης στην μεταβλητή που βρίσκεται στα αριστερά του τελεστή της =

```
bash-3.2$ python
Python 3.7.6 (default, Dec 22 2019, 01:09:06)
[Clang 11.0.0 (clang-1100.0.33.12)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> distance = input("what is the distance travelled ? ")
what is the distance travelled ? 500
>>> distance
'500'
```

Ανάθεση της **γραμματοσειράς** 500 στη sting μεταβλητή distance

**string** και όχι νούμερο

Μπορούμε να μετατρέψουμε την μεταβλητή σε αριθμό χρησιμοποιώντας τις συναρτήσεις **eval(strname)** - ακέραιο distance = eval(distance)  
**float(strname)** - κινητής υποδιαστολής distance = float(distance)



## Κανόνες προτεραιότητας πράξεων

Όταν χρησιμοποιούμε πολλούς τελεστές πράξεων, η Python θα πρέπει να ξέρει ποιες πράξεις εκτελούνται πρώτα

$$x = 1 + 2 * 3 - 4 / 5 ** 6$$

Η προτεραιότητα καθορίζεται ως:

Πρώτα εκτελούνται οι παρενθέσεις

Ακολουθούν τα εκθετικά (δυνάμεις)

Πολλαπλασιασμοί, διαιρέσεις και υπόλοιπα


Προσθέσεις και αφαιρέσεις

Αριστερά προς τα δεξιά

# Κανόνες προτεραιότητας πράξεων

```
>>> x = 1 + 2 ** 3 / 4 * 5  
>>> print(x)  
11.0  
>>>
```

Παρένθεση  
Δυνάμεις  
Πολ/σμός  
Πρόσθεση  
Αριστερά-δεξιά



1 + 2 \*\* 3 / 4 \* 5

1 + 8 / 4 \* 5

1 + 2 \* 5

1 + 10

11

## Είδος μεταβλητής

Η Python ξέρει για διαφορές μεταξύ μιας μεταβλητής string και μιας ακεραίου

Για παράδειγμα ο τελεστής + σημαίνει προσθεση για αριθμητικές μεταβλητές και σύμπτυξη (concatenation) για μεταβλητές χαρακτήρων

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

Μερικές πράξεις απαγορεύονται:

Δεν μπορούμε να προσθέσουμε  
χαρακτήρες και νούμερα

Μπορούμε να μάθουμε τι είδους  
είναι μια μεταβλητή χρησιμοποιώντας  
την συνάρτηση **type()**

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last): File
"<stdin>", line 1, in <module>TypeError:
Can't convert 'int' object to str implicitly
>>> type(eee)
<class'str'>
>>> type('hello')
<class'str'>
>>> type(1)
<class'int'>
>>>
```

# Αριθμητικές μεταβλητές

Δύο βασικοί τύποι αριθμητικών μεταβλητών:

Ακέραιες: (int)

Δεκαδικοί: (float)

Όταν χρησιμοποιείται ένας ακέραιος και ένας δεκαδικός σε μια μαθηματική έκφραση, τότε ο ακέραιος μετρέπεται σε float

Αυτό μπορεί να αλλάξει με την χρήση των συναρτήσεων **int()** και **float()**

```
>>> xx = 1
>>> type(xx)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class'float'>
>>>
```

```
>>> print(float(99) + 100)
199.0
>>> i = 42
>>> type(i)
<class'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class'float'>
```

## Διαίρεση ακεραίων

Η διαίρεση δύο ακεραίων οδηγεί πάντοτε σε δεκαδικό αποτέλεσμα (**όχι σε Python2**):

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

## Μετατροπή χαρακτήρων (strings)

Χρησιμοποιώντας τις συναρτήσεις **int()** και **float()** μπορούμε να μετατρέψουμε String variables σε ακέραιους ή πραγματικούς

Αν η string μεταβλητή δεν περιέχει νούμερα θα μας γυρίσει σφάλμα

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last): File "<stdin>", line 1, in
<module>
TypeError: Can't convert 'int' object to str implicitly
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last): File "<stdin>", line 1, in
<module>
ValueError: invalid literal for int() with base 10: 'x'
```

## Εισαγωγή δεδομένων από τον χρήστη

Μπορούμε να δώσουμε οδηγία στην Python να σταματήσει την εξέλιξη ενός προγράμματος και να διαβάσει δεδομένα από το πληκτρολόγιο χρησιμοποιώντας την συνάρτηση **input()**

Η συνάρτηση **input()** επιστρέφει μια μεταβλητή τύπου string

```
nam = input('Who are you? ')
print('Welcome', nam)
```

```
Who are you? John
Welcome John
```

Για να διαβάσετε ένα νούμερο θα πρέπει να μετατραπεί σε int ή float

```
inp = input('What is your age?')
age = int(inp) + 1
print('Age is ', age)
```

# Σχόλια στην Python

Ότι ακολουθεί το σύμβολο **#** αποτελεί σχόλιο στην Python και δεν είναι εκτελέσιμο

Γιατί χρειάζεται να έχουμε σχόλια σε κάποιο πρόγραμμα?

- Για να περιγράψουμε τι θα συμβεί σε μια ακολουθία εντολών του προγράμματος
- Για να περιγράψουμε οποιαδήποτε βοηθητική πληροφορία
- Για να μην εκτελέσουμε προσωρινά κάποια εντολή του προγράμματος

```
# Get the name of the file and open it
name = input('Enter file:')
inpfile = open(name, 'r')

# Count word frequency
for line in infile:
    numbers = line.split()
    for number in numbers:
        counter[numbers] += counts.get(number,0) + 1

# All done
print(counter)
```



## Σχόλια στην Python

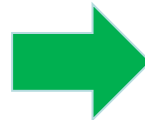
Σε αρκετές περιπτώσεις χρειάζεται να περιγράψουμε τη λειτουργία ενός τμήματος κώδικα σε περισσότερες από 2 γραμμές.

Σε άλλες περιπτώσεις θέλουμε να μην εκτελέσουμε προσωρινά ένα τμήμα του προγράμματος που περιέχει αρκετές εντολές, για να εξακριβώσουμε κάποιο λάθος του προγράμματος ή να δοκιμάσουμε κάτι άλλο.

Για μεγάλο αριθμό γραμμών ως σχόλια ή για να έχουμε μια σειρά εντολών που δεν εκτελείται χρησιμοποιούμε `'''` στην αρχή και `'''` στο τέλος του τμήματος που θέλουμε να μετατρέψουμε σε σχόλιο ή μη εκτελέσιμες εντολές

```
# This is a very simple  
# script to demonstrate  
# the use of long comments
```

```
name = input('Enter file:')  
inpfiler = open(name, 'r')  
print(inpfiler)
```



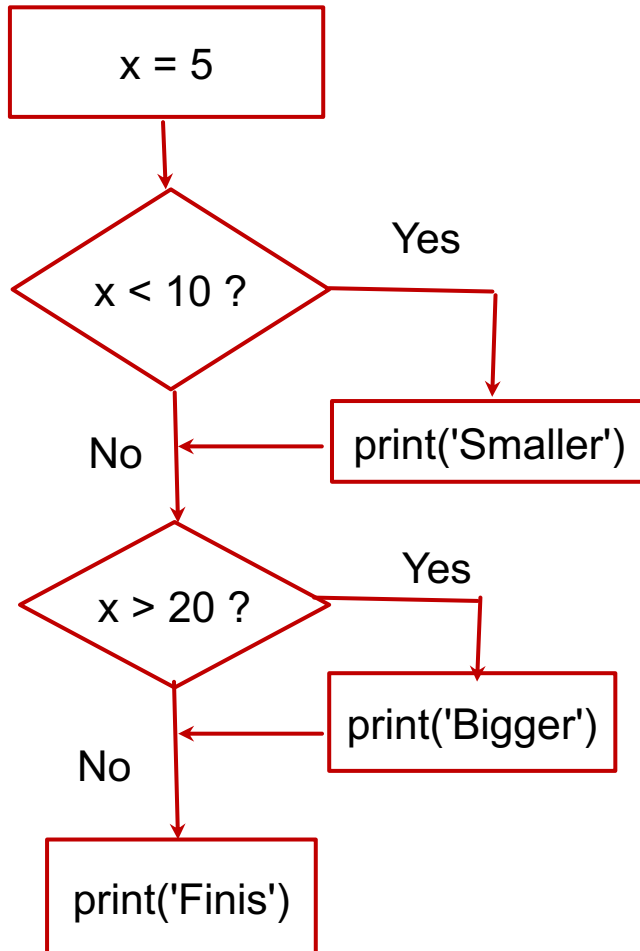
```
'''
```

```
This is a very simple  
script to demonstrate  
the use of long comments
```

```
'''
```

```
name = input('Enter file:')  
inpfiler = open(name, 'r')  
print(inpfiler)
```

## Εκτέλεση εντολών υπό συνθήκη



Program:

```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')
print('Finis')
```

Output:

Smaller

Finis

## Τελεστές σύγκρισης

Οι λογικές εκφράσεις (**Boolean**) θέτουν μια ερώτηση και έχουν σαν αποτέλεσμα ένα **YES** (ή **True**) ή **NO** (**False**) το οποίο χρησιμοποιείται για να ελέγξουμε την ροή εκτέλεσης των εντολών

Οι boolean εκφράσεις χρησιμοποιούν τελεστές σύγκρισης για δώσουν αποτέλεσμα **YES** (ή **True**) ή **NO** (**False**)

Σε αντίθεση με τους κανονικούς τελεστές πράξεων, οι τελεστές σύγκρισης ελέγχουν τις μεταβλητές αλλά δεν αλλάζουν τη τιμή τους

Python	Σημασία
<	Less than
<=	Less than or Equal to
==	Equal to
>=	Greater than or Equal to
>	Greater than
!=	Not equal

**Προσοχή:** ο τελεστής **=** χρησιμοποιείται για ανάθεση τιμών σε μεταβλητές

## Τελεστές σύγκρισης

<pre>x = 5</pre>	
<pre>if x == 5 :</pre>	
<pre>    print('Equals 5')</pre>	Equals 5
<pre>if x &gt; 4 :</pre>	
<pre>    print('Greater than 4')</pre>	Greater than 4
<pre>if x &gt;= 5 :</pre>	
<pre>    print('Greater than or Equals 5')</pre>	Greater than or Equals 5
<pre>if x &lt; 6 : print('Less than 6')</pre>	
<pre>if x &lt;= 5 :</pre>	
<pre>    print('Less than or Equals 5')</pre>	Less than or Equals 5
<pre>if x != 6 :</pre>	
<pre>    print('Not equal 6')</pre>	Not equal 6

# Μονόδρομες αποφάσεις

```
x = 5
print('Before 5')
if x == 5 :
    print('Is 5')
    print('Is Still 5')
    print('Third 5')
print('Afterwards 5')
print('Before 6')
if x == 6 :
    print('Is 6')
    print('Is Still 6')
    print('Third 6')
print('Afterwards 6')
```

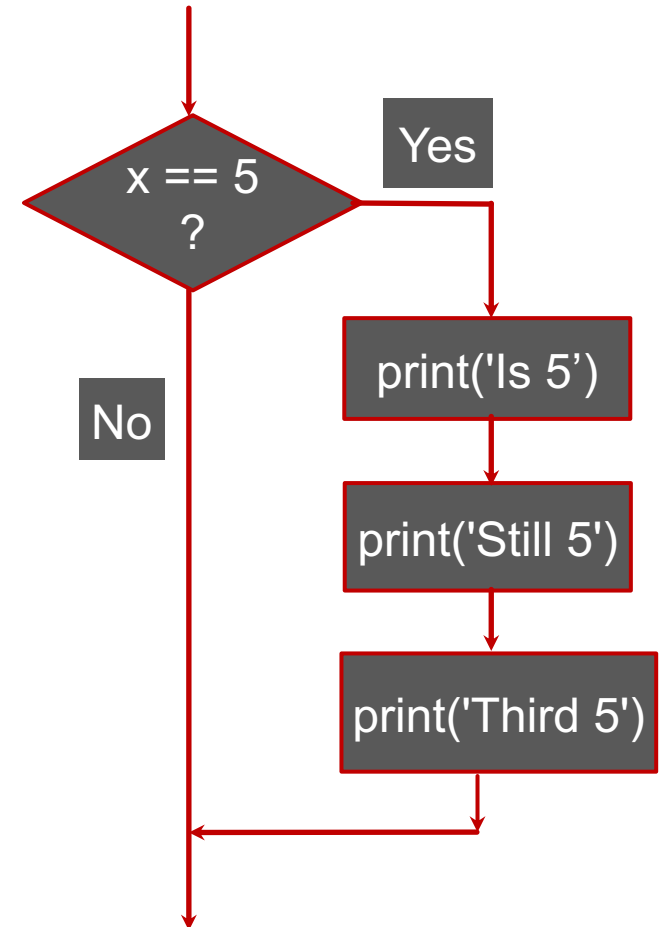
Before 5

Is 5  
Is Still 5  
Third 5

Afterwards 5  
Before 6

Δεν εισέρχεται  
στο τμήμα αυτό

Afterwards 6



## Στοίχιση (**indentation**)

Θα πρέπει να υπάρχει στοίχιση προς το εσωτερικό ως προς την αρχή της γραμμής για να δηλώσουμε ότι οι εντολές που ακολουθούν θα πρέπει να εκτελεστούν ανάλογα με το αποτέλεσμα εξέτασης της συνθήκης

Η σύμβαση αυτή ακολουθείται τόσο για το `if` όσο και για επαναλήψιμα (***for***) βήματα, **γίνεται μετά το :**

Θα πρέπει να διατηρηθεί η ίδια στοίχιση για όλες τις εντολές που ανήκουν στο ίδιο `if` ή `for` block εντολών

Η στοίχιση μετατοπίζεται αριστερά κάτω από το `if` ή `for` για να δηλωθεί το τέλος του συγκεκριμένου `if` ή `for` block

Κενές γραμμές δεν επηρεάζουν την στοίχιση

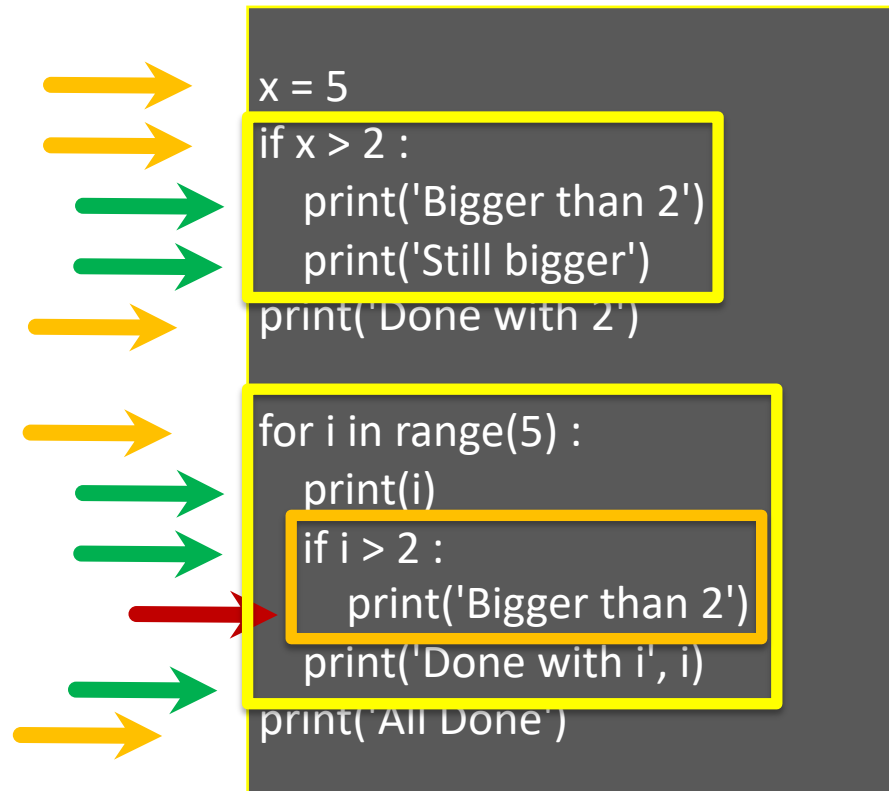
Σχόλια δεν επηρεάζουν την στοίχιση

Tabs και spaces είναι διαφορετικά ανάλογα με τον επεξεργαστή κειμένου που χρησιμοποιείται και δεν θα πρέπει να αναμειγνύονται γιατί οδηγούν σε λάθος ερμηνεία για το που αρχίζει και τελειώνει ένα `if` ή `for` loop.

**Ο επεξεργαστής emacs μετατρέπει τα tabs σε spaces**

## Στοίχιση

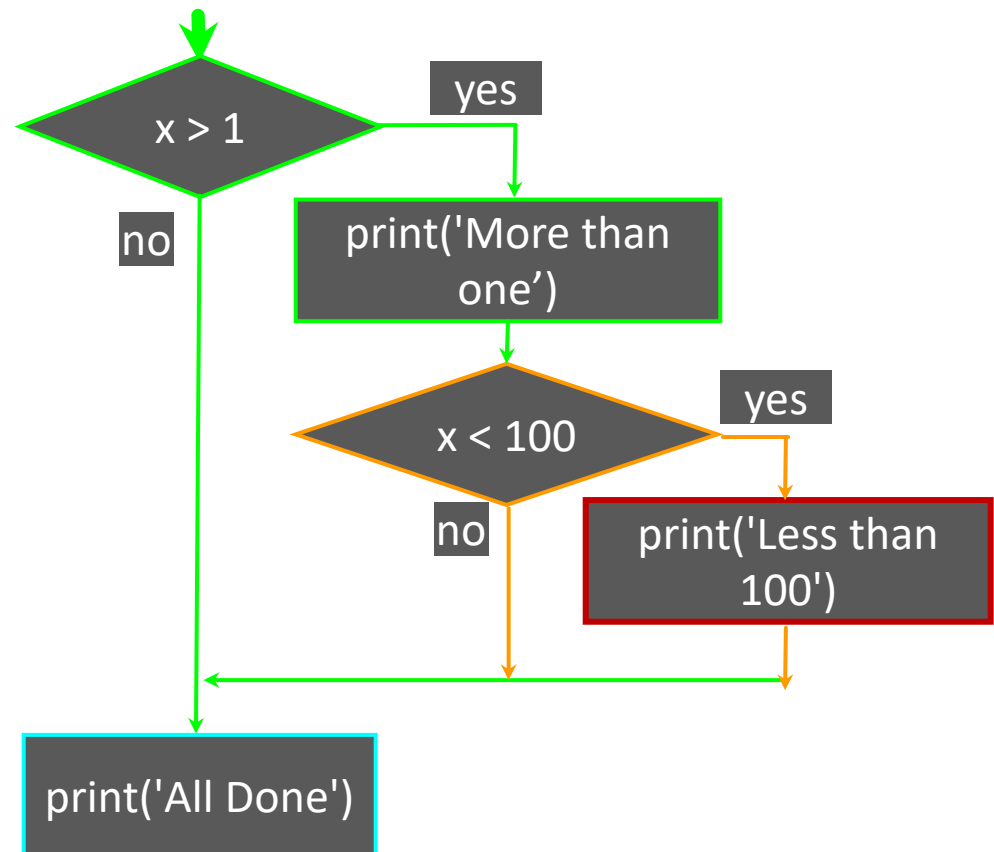
Αυξάνετε την στοίχιση μετά από την αρχή ενός if ή for block και διατηρείτε την  
Ελαττώστε τη στοίχιση για να δηλώσετε το τέλος ενός if ή for block



## Φωλιασμένες/εσωτερικές (**nested**) συνθήκες

Σε πολλές περιπτώσεις χρειάζεται να εξετάσουμε διάφορες συνθήκες μέσα σε συνθήκες που έχουν εξεταστεί

```
x = 43  
if x > 1 :  
    print('Bigger than 1')  
    if x < 100 :  
        print('Less than 100')  
print('All done ')
```





# Εντολές συνθήκης διπλής διαδρομής

Αρκετές φορές χρειάζεται να εκτελέσουμε μια σειρά από εντολές όταν κάποια συνθήκη είναι αληθής (True) και μια διαφορετική σειρά από εντολές όταν η συνθήκη δεν ισχύει (False).

**if <συνθήκη> :**

Αποτελείται από μια λογική έκφραση και ακολουθείται από μια σειρά εντολών

**else :**

Αν η συνθήκη είναι FALSE μια σειρά άλλων εντολών

Είναι σα να συναντάμε στον δρόμο κάποια διακλάδωση και θα πρέπει να αποφασίσουμε ποιο τμήμα θα ακολουθήσουμε

```
x = 4
```

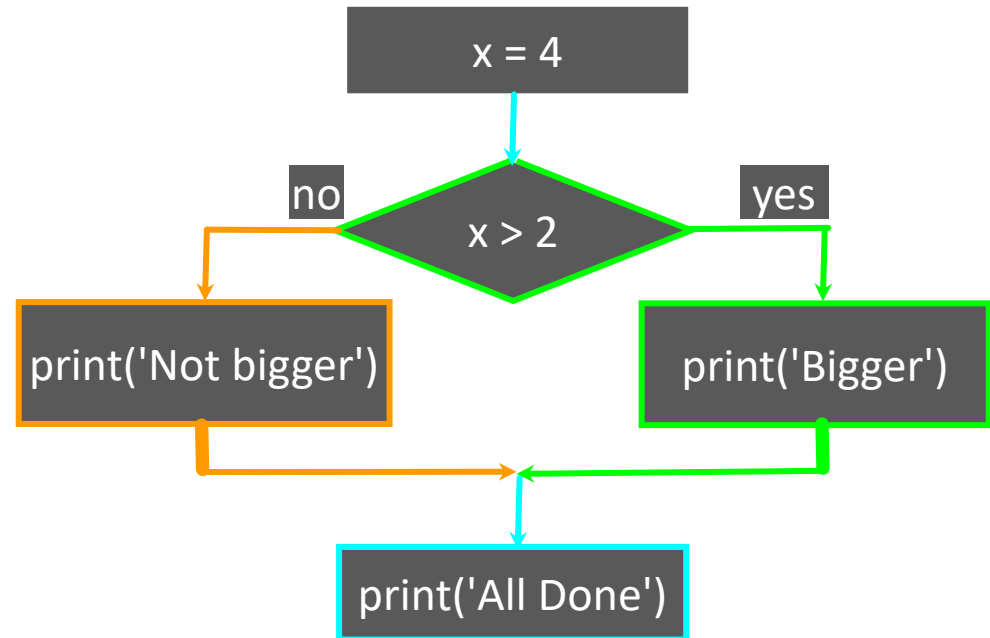
```
if x > 2 :
```

```
    print('Bigger')
```

```
else :
```

```
    print('Smaller')
```

```
print('All done')
```



# Εντολές συνθήκης πολλαπλών διαδρομών

Σε πολλές περιπτώσεις εξετάζουμε διάφορες συνθήκες και ανάλογα με το αποτέλεσμα της εξέτασης αυτής ακολουθούμε κάποια σειρά εντολών

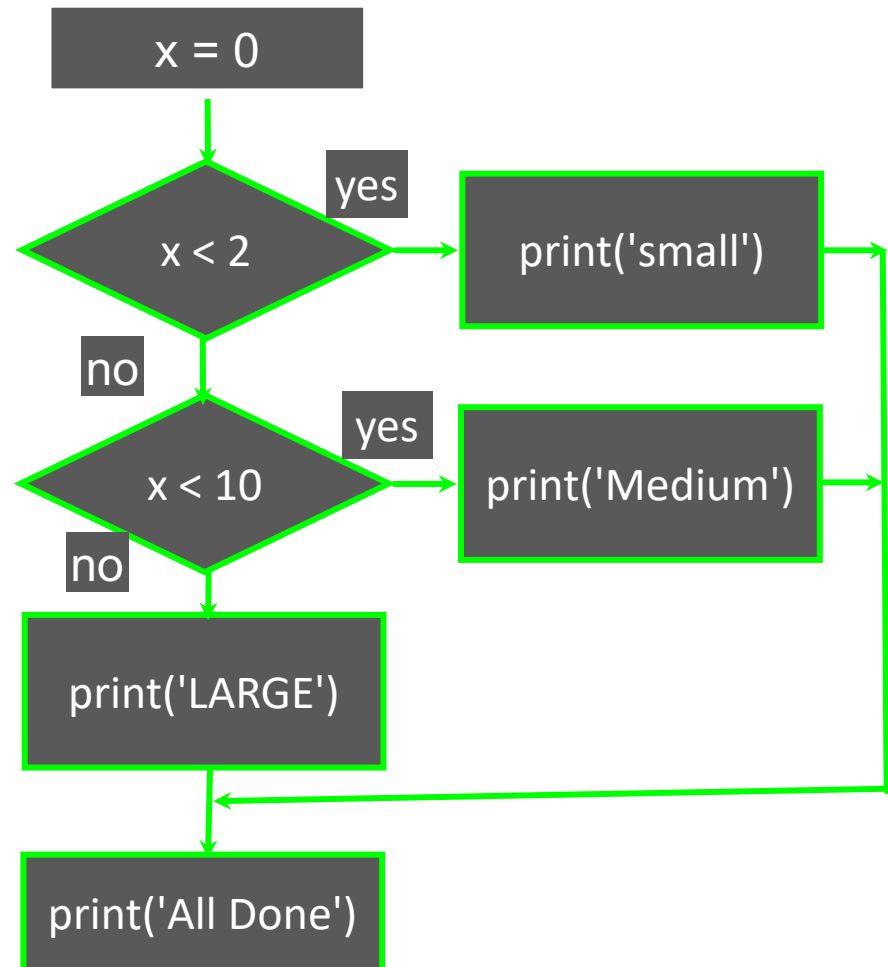
**if <συνθήκη> :**

**elif <συνθήκη> :**

**elif <συνθήκη> :**

**else :**

```
x = 0
if x < 2 :
    print('small')
elif x < 10:
    print('Medium')
else :
    print('Large')
print('All done')
```



# Εντολές συνθήκης if .... Else

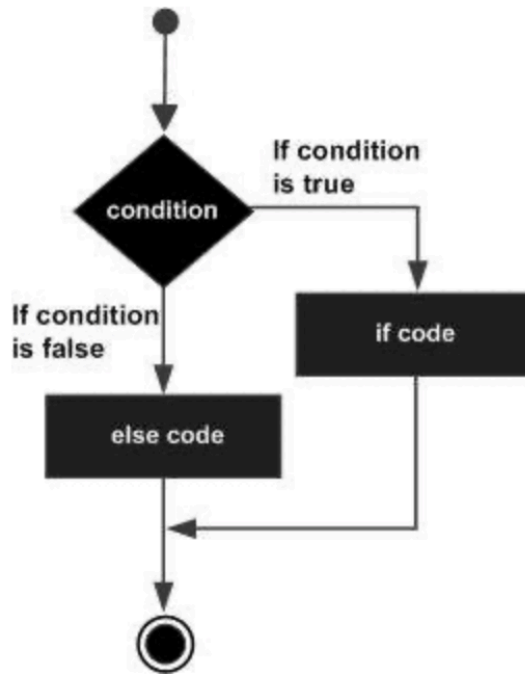
Η σύνταξη είναι:

If λογική εκφραση :

Εντολές

else :

Εντολές



```
#!/usr/bin/python
```

```
var1 = 100
```

```
if var1:
```

```
    print "1 - Got a true expression value"
```

```
    print var1
```

```
else:
```

```
    print "1 - Got a false expression value"
```

```
    print var1
```

```
var2 = 0
```

```
if var2:
```

```
    print "2 - Got a true expression value"
```

```
    print var2
```

```
else:
```

```
    print "2 - Got a false expression value"
```

```
    print var2
```

```
print "Good bye!"
```

```
1 - Got a true expression value
```

```
100
```

```
2 - Got a false expression value
```

```
0
```

```
Good bye!
```

## Εντολές συνθήκης **if .... elif ... elif ... else**

Η σύνταξη είναι:

If λογική έκφραση :

Εντολές

elif λογική έκφραση1:

Εντολές

elif λογική έκφραση2:

Εντολές

else:

Εντολές

```
#!/usr/bin/python
```

```
var = 100
if var == 200:
    print "1 - Got a true expression value"
    print var
elif var == 150:
    print "2 - Got a true expression value"
    print var
elif var == 100:
    print "3 - Got a true expression value"
    print var
else:
    print "4 - Got a false expression value"
    print var

print "Good bye!"
```

```
3 - Got a true expression value
100
Good bye!
```

## Η δομή try/except

Η δομή με try/except χρησιμοποιείται για να περικλείουμε επικίνδυνα τμήματα του κώδικα που δεν ξέρουμε τι ακριβώς γίνεται

Αν το τμήμα του κώδικα μέσα στο try δουλεύει τότε το τμήμα που περικλείεται στο except δεν εκτελείται

Αν το τμήμα του κώδικα μέσα στο try δεν δουλεύει τότε εκτελείται το τμήμα που περικλείεται στο except

```
$ python3 notry.py
```

```
Traceback (most recent call last): File "notry.py", line 2,  
in <module>   istr = int(astr)ValueError: invalid literal for  
int() with base 10: 'Hello Bob'
```

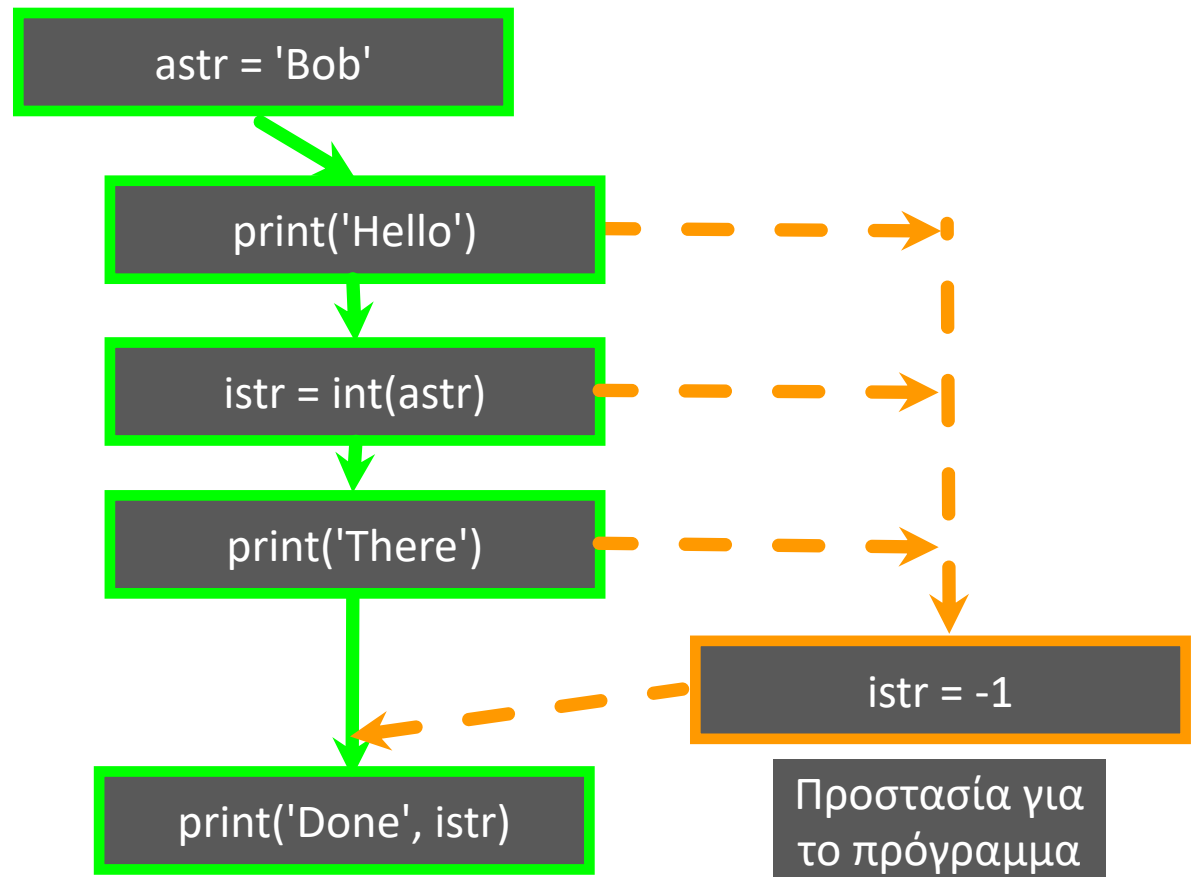
```
astr = 'Hello Bob'  
istr = int(astr)  
print('First', istr)  
astr = '123'  
istr = int(astr)  
print('Second', istr)
```

πρόβλημα

error

# try / except

```
astr = 'Bob'  
try:  
    print('Hello')  
    istr = int(astr)  
    print('There')  
except:  
    istr = -1  
  
print('Done', istr)
```



## try / except

```
rawstr = input('Enter a number:')
try:
    ival = int(rawstr)
except:
    ival = -1

if ival > 0 :
    print('Nice work')
else:
    print('Not a number')
```

```
$ python3 trynum.py
Enter a number:42
Nice work
$ python3 trynum.py
Enter a number:forty-two
Not a number
$
```