

Κύρια συστατικά ενός προγράμματος

Σταθερές

Μεταβλητές

Αριθμητικές πράξεις

Ανάθεση τιμών – ισότητες

Έλεγχος αποτελέσματος – επικοινωνία με το πρόγραμμα

bits and bytes

- ❑ Ο υπολογιστής χρησιμοποιεί τη **κύρια μνήμη** για αποθήκευση δεδομένων
- ❑ Η μνήμη χωρίζεται σε **words** και κάθε word περιέχει τμήμα πληροφορίας
- ❑ Ο αριθμός των words σε μια κεντρική μνήμη εξαρτάται από τον Η/Υ (αρκετές χιλιάδες για ένα μικρο-processor σε εκατοντάδες εκατομμύρια για μεγάλους υπολογιστές)
- ❑ Κάθε word αποτελείται από μικρότερες μονάδες που ονομάζονται **bits**

Το bit μπορεί να έχει μόνο μια από δύο τιμές: **0 ή 1**

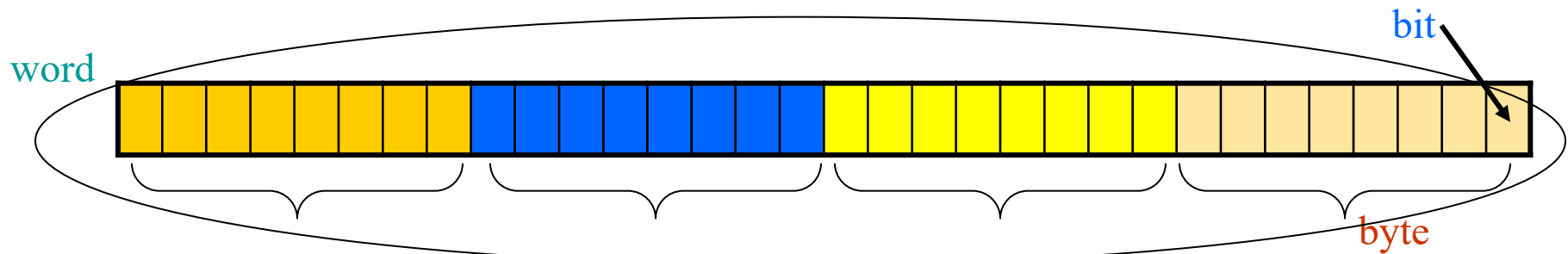
Αν το word περιέχει m bits τότε το word μπορεί να είναι σε οποιαδήποτε από 2^m διαφορετικές καταστάσεις

Η πραγματική σημασία που δίνεται σε κάποιο ιδιαίτερο συνδυασμό από 0 και 1 σε ένα word εξαρτάται από το μοντέλο του υπολογιστή

- ❑ Όλοι οι υπολογιστές (**σχεδόν**) αποθηκεύουν τους θετικούς INTEGERS με τέτοιο τρόπο ώστε κάθε bit που παίρνει τη τιμή 1 στο word αντιπροσωπεύει μια τιμή που είναι $2^{(n-1)}$ όπου n η σχετική θέση του bit στο word μετρώντας από δεξιά.

bits and bytes

- ❑ Οι περισσότεροι υπολογιστές έχουν μνήμη με 32 bits/word με ενδιάμεσο διαχωρισμό του word σε 4 ομάδες από 8 συνεχή bits
- ❑ Ένα τέτοιο γκρουπ αποτελούμενο από 8 bits ονομάζεται **byte**
- ❑ Ένα byte έχει 2^8 ή 256 διαφορετικές καταστάσεις.
Όλοι οι αριθμοί αντιπροσωπεύονται σε δυαδική μορφή



- ❑ Υπάρχει περιορισμένη ακρίβεια και επομένως προσεγγίσεις
- ❑ Το μήκος του word είναι συνήθως 32 bits ή 4 bytes
όπου κάθε byte αποτελείται από 8 bits

Μονάδες μέτρησης για την αποθήκευση αριθμών είναι:

$$1 \text{ byte} = 1\text{B} = 8 \text{ bits} = 8\text{b}$$

$$1 \text{ kbyte} = 2^{10} \text{ bytes} = 1024 \text{ bytes}$$

$$1 \text{ Mbyte} = 2^{10} \text{ kbytes} = 1024 \times 1024 \text{ bytes}$$

Γιατί υπάρχει περιορισμός στους αριθμούς

- ❑ Σχετίζεται με την αναπαράσταση αριθμών στον υπολογιστή
- ❑ Τρία συστήματα αριθμών:
 - Οι αριθμοί μέτρησης 0,1,2,...,32767
Συνδέεται με τον δείκτη των καταλόγων της μηχανής και άρα το εύρος των αριθμών είναι προσδιορισμένο.
 - Οι αριθμοί σταθερής υποδιαστολής
Οι αριθμοί αυτοί έχουν σταθερό μήκος = μήκος του word του Η/Υ ή κάποιο πολλαπλάσιο
Η διαφορά μεταξύ διαδοχικών αριθμών είναι η ίδια
Τέτοιοι είναι οι αριθμοί των υπολογισμών του χεριού (Όλοι οι πίνακες είναι συνήθως σταθερής υποδιαστολής)
 - Οι αριθμοί κινητής υποδιαστολής (floating point numbers)
Σχετίζονται με τη λεγόμενη επιστημονική σήμανση
Το σύστημα αυτό είναι σχεδιασμένο για να περιγράφει και μικρούς αλλά και μεγάλους αριθμούς


Αναπαράσταση αριθμών στον Η/Υ

- Οι αριθμοί κινητής υποδιαστολής (floating point numbers)

$$.31415927 \times 10^1$$

$$.12345678 \times 10^{-1}$$

$$-.12345678 \times 10^4$$



Mantissa: Το πρώτο τμήμα των 8 ψηφίων στους παραπάνω αριθμούς

Εκθέτης: Το τελευταίο ψηφίο (με εύρος τιμών [-38, +38])

Η mantissa και ο εκθέτης αποθηκεύονται στο ίδιο word της μηχανής

Σαν αποτέλεσμα η mantissa ενός αριθμού κινητής υποδιαστολής είναι μικρότερου μήκους του αντίστοιχου αριθμού σταθερής υποδιαστολής

Απεικόνιση αριθμών στους υπολογιστές

- Ένας ακέραιος αριθμός 2^N αντιπροσωπεύεται από N bits

Το 1^ο bit δίνει το πρόσημο

Τα υπόλοιπα $N-1$ bits χρησιμοποιούνται για τον αριθμό

Άρα 32 bits INTEGERS θα πέρνουν τιμές στο διάστημα:

$$-2147483648 < \text{Integer}_{32} < 2147483648 \quad 2^{31}$$

- Η αριθμητική με Integer αριθμούς είναι ακριβής εκτός από **υπερχείλιση** (overflow) και **υποχείλιση** (underflow)

- Οι αριθμοί απλής ακρίβειας (single precision ή floating - point F) χρειάζονται 4 bytes (32 bits) για την αναπαράστασή τους

Τα οποία χρησιμοποιούνται ως εξής:

{	23 bits για mantissa
	8 bits για εκθέτη
	1 bit για το πρόσημο

Η μικρότερη mantissa είναι $2^{-23} \sim 10^{-7}$

Εκθέτης:

2^a όπου $a \in [0, 255]$ (2^8 bits)

← “bias”

Απεικόνιση αριθμών

□ Για αριθμούς διπλής ακρίβειας χρειάζονται 8 bytes δηλαδή 64 bits

και χρησιμοποιούνται ως εξής:

- 52 bits για την mantissa
- 1 bit για το πρόσημο
- 11 bits για τον εκθέτη (**bias=1023**)

Η μικρότερη mantissa είναι $2^{-52} \sim 10^{-16}$

Ο εκθέτης είναι $2^{\alpha-1023}$ με $\alpha \in [0, 2048]$ (2^{11} bits)

Διάστημα

$$2.9 \times 10^{-39} \leq \text{float}_{32} \leq 3.4 \times 10^{38}$$

$$10^{-322} \leq \text{double}_{64} \leq 10^{308}$$

Ακρίβεια

float_{32} : 5-7 δεκαδικά ψηφία

double_{64} : 16 δεκαδικά ψηφία

Αριθμοί κινητής υποδιαστολής

□ Έχουν μια σειρά από ιδιαιτερότητες:

- Ας υποθέσουμε για απλότητα ότι έχουμε μια mantissa με 3 ψηφία και ο εκθέτης αποτελείται από 1 ψηφίο:

Το μηδέν, $0 = .000 \times 10^{-9}$ είναι απομονωμένο από τους υπόλοιπους αριθμούς αφού οι αμέσως επόμενοι θετικοί αριθμοί είναι $.100 \times 10^{-9}$ και $.101 \times 10^{-9}$ βρίσκονται 10^{-12} μακριά!!

Δεν υπάρχει κανένας άλλος αριθμός με mantissa να ξεκινά από 0

- Κάθε δεκάδα έχει ακριβώς τον ίδιο πλήθος αριθμών: **συνολικά 900** πηγαίνοντας από $.100 \times 10^a$ μέχρι $.999 \times 10^a$ με $a = -9, -8, \dots, 0, \dots, 8, 9$

0.999x10 ⁻¹ 0.100x10 ⁰	1x10 ⁻⁴
0.999x10 ⁰ 0.100x10 ¹	1x10 ⁻³
0.999x10 ¹ 0.100x10 ²	1x10 ⁻²

- Σε κάθε δεκάδα οι 900 αριθμοί είναι χωρισμένοι σε ίσα διαστήματα

αλλά η διαφορά μεταξύ 2 δεκάδων είναι άνιση

Υπάρχει γεωμετρικό πήδημα

Αριθμοί κινητής υποδιαστολής

- ❑ Ο αριθμός 0 και -0 (δηλαδή -0.000×10^{-9}) είναι λογικά ίδιοι
Οι περισσότεροι υπολογιστές τους έχουν ίδιους αλλά μερικοί όχι
 - Συνήθως δεν υπάρχει άπειρο που να αντιστοιχεί στο μηδέν
- ❑ Στα μαθηματικά υπάρχει ένα και μοναδικό μηδέν
- ❑ Στο προγραμματισμό υπάρχουν 2 είδη:
 - Αυτό που εμφανίζεται σε εκφράσεις όπως $\alpha \cdot 0 = 0$ που συμπεριφέρεται σαν κανονικό 0
 - Αλλά και το 0 που εμφανίζεται σε σχέσεις της μορφής $\alpha - \alpha = 0$
Αυτό το 0 μπορεί να προέρχεται από $1 - (1 - \epsilon)(1 + \epsilon) = 0$ όταν το $\epsilon < \frac{1}{2}10^{-3}$ όπου το 3 είναι ο αριθμός των δεκαδικών ψηφίων της mantissa
 - Αυτό το τελευταίο 0 εμφανίζεται κατά την αφαίρεση 2 σχεδόν ίσου μεγέθους αριθμών
 - Αποτελεί την πηγή πολλών σφαλμάτων στους υπολογισμούς με ένα ηλεκτρονικό υπολογιστή

Σύνθετες αναθέσεις

- ❑ Είδαμε αναθέσεις τιμών σε μεταβλητές. Υπάρχουν περιπτώσεις που η ανάθεση μπορεί να εμπλέκει περισσότερο σύνθετους τελεστές
- ❑ Σύνθετοι τελεστές ανάθεσης είναι: `+=` `-=` `*=` `/=` `%=` `**=` `//=`

```

>>>
>>> a,b,c = 4,6, 2    Ανάθεση τιμών α=4, b=6, c=2
>>> a += 1            Πρόσθεση 1 στη τιμή του a
>>> a
5
>>> b *=3             Πολ/σμος του b με 3
>>> b
18
>>> c /=2             Διαίρεση του c με 2
>>> b -=4             Αφαίρεση του 4 από το b
>>> b
14
>>> □
...
>>> b %=3             Υπόλοιπο διαίρεσης του b με 3
>>> b
2
>>>
>>> a **=2            Έγψωση της μεταβλητής α στο τετράγωνο
>>> a
25
>>>
>>>
>>> a //=4            Ακέραιο Πηλίκο της διαίρεσης της α με το 4
>>> a
6
>>> □

```


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  
>>> 
```

Το 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**

```
#!/usr/bin/python3
'''
Υπολογισμος χρονου, litrwn venzinis kai kostos venzinis gia ena taksidi
'''

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

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

quit(0)
```

Το σύμβολο `'''` ανοίγει σχόλια και `'''` τα κλείνει

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:

python3 mytrip.py

Το ίδιο μπορεί να συμβεί αν μπούμε πρώτα σε περιβάλλον python δίνοντας την εντολή `python3` στο terminal και κατόπιν την εντολή να δώσουμε:

```
bash-3.2$ python3
Python 3.8.2 (default, Apr  8 2021, 23:19:18)
[Clang 12.0.5 (clang-1205.0.22.9)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import mytrip.py
6.666666666666667
13.333333333333334
53.333333333333336
```

Δεύτερο πρόγραμμα

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

```
#!/usr/bin/python3

# Υπολογισμός της απόστασης δύο σημείων στο χώρο

import numpy as np
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()

print("Good bye!")
```

← Χρειάζεται να κάνουμε import την numpy βιβλιοθήκη

← Χρειάζεται να δηλώσουμε από ποια βιβλιοθήκη θα πάρουμε τη συνάρτηση sqrt(). Εδώ από την βιβλιοθήκη numpy

Τρέχουμε το πρόγραμμα στο terminal είτε με την εντολή: `python3 distance.py` ή αφού ξεκινήσει το περιβάλλον python με την εντολή:

```
bash-3.2$ python3
Python 3.8.2 (default, Apr  8 2021, 23:19:18)
[Clang 12.0.5 (clang-1205.0.22.9)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import distance
13.760813929415658
```


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$ python3
```

```
Python 3.8.2 (default, Apr  8 2021, 23:19:18)
```

```
[Clang 12.0.5 (clang-1205.0.22.9)] 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'
```

string και όχι νούμερο

```
>>> █
```

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

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

Αντικείμενα και μέθοδοι – Objects και methods

Μέχρι τώρα είδαμε αναθέσεις τιμών σε μεταβλητές. Αυτό που κάνει η python είναι να συσχετίζει ένα όνομα με ένα αντικείμενο (**objects**). Σε μεταγενέστερη εντολή μπορούμε να αναφερθούμε στο αντικείμενο με το όνομα της μεταβλητής που το συσχετίσαμε.

Το αντικείμενο όμως μπορεί να έχει περισσότερες ιδιότητες από κάποια τιμή.

Ένα object μπορεί να εμπεριέχει δεδομένα (**data**) αλλά και συναρτήσεις (**methods**)

Η τιμή ενός object αποτελεί ένα από τα δεδομένα του object

Η method ενός object είναι μια συνάρτησή του που ενεργεί στα δεδομένα του object.

Η method μπορεί να δέχεται και περισσότερα δεδομένα

Για να χρησιμοποιήσουμε μία μέθοδο ενός αντικειμένου θα πρέπει να:

- Δώσουμε το όνομα του αντικειμένου
- Ακολουθούμενο από μια τελεία .
- Το όνομα της μεθόδου
- Ένα ζευγάρι παρενθέσεις που περιέχουν τα δεδομένα (ορίσματα της μεθόδου)

Παράδειγμα Objects με methods

Μπορούμε να εξετάσουμε μεθόδους ενός float object *f*, ενός object *s* τύπου string

- **f.is_integer()** Κάθε αντικείμενο τύπου float έχει μια μέθοδο που μας επιτρέπει να εξετάσουμε αν η τιμή του αντικειμένου είναι integer. Η μέθοδος επιστρέφει TRUE αν το δεκαδικό μέρος είναι μηδέν και FALSE διαφορετικά. Δεν απαιτεί κάποια επιπλέον τιμή αλλά πρέπει να υπάρχει το ζευγάρι των παρενθέσεων.
- **s.swapcase()** Κάθε αντικείμενο τύπου string έχει μια μέθοδο που μας επιστρέφει μια νέα string τιμή που η τιμή της είναι μια γραμματοσειρά με τους χαρακτήρες της αρχικής γραμματοσειράς σε αντίθετη κατάσταση (γράμματα από μικρά – κεφαλαία ή κεφαλαία σε μικρά)

```
>>>  
>>> f=3.2  
>>> f.is_integer()  
False  
>>>  
>>>  
>>> s="python"  
>>> s.swapcase()  
'PYTHON'  
>>> □
```

Αντικείμενα τύπου λίστας: list type objects

Αντικείμενα τύπου που ορίζονται με τετραγωνική αγκύλη [] και περιέχουν δεδομένα χωρισμένα με , αποτελούν τύπο **list**.

L = [1, 2, 3] Λίστα με ακεραίους αφού τα δεδομένα είναι ακέραιοι. Η τιμή του αντικειμένου είναι μια ακολουθία 3 ακεραίων αντικειμένων
Μια λίστα μπορεί να περιέχει 1 μόνο αντικείμενο π.χ. [2.7] το οποίο είναι διαφορετικό Από το 2.7 ως float αντικείμενο.
Μια λίστα μπορεί να είναι άδεια π.χ. []

➤ **L.reverse()** Είναι μία μέθοδος του αντικειμένου τύπου list που επιστρέφει τη λίστα σε ανάποδη σειρά και την αντικαθιστά με την νέα list

➤ **L.pop(n)** Είναι μία μέθοδος του αντικειμένου τύπου list που επιστρέφει το αντικείμενο που βρίσκεται στην n-θέση της list (**ξεκινώντας το μέτρημα από τη θέση 0**) **και το αφαιρεί από τη λίστα**

Καλώντας τη μέθοδο pop() χωρίς όρισμα θα αφαιρέσει το **τελευταίο** στοιχείο της list

```
>>>
>>> L=[2,5,8,9,11]
>>> L.reverse()
>>> print(L)
[11, 9, 8, 5, 2]
>>>
>>> L.pop(2)
8
>>> print(L)
[11, 9, 5, 2]
>>>
>>> L.pop()
2
>>> print()

>>> print(L)
[11, 9, 5]
>>> □
```

List type objects

Τα αντικείμενα τύπου list, είναι οι πλέον εύχρηστοι σύνθετοι τύποι δεδομένων.

Μια **λίστα** περιέχει διάφορες **σταθερές** διαχωρισμένες με **,** και περιέχονται σε **[]**.

Μια λίστα μοιάζει με πίνακες δεδομένων με τη διαφορά ότι τα στοιχεία που αποτελούν τη λίστα δεν είναι απαραίτητα του ίδιου τύπου.

Μπορούμε να αποκτήσουμε πρόσβαση στα στοιχεία μιας λίστας χρησιμοποιώντας τον τελεστή **[]** και **[:]** και κάποιον δείκτη η αρίθμηση του οποίου ξεκινά από το 0

Χρησιμοποιώντας αρνητικό δείκτη θέσης, τότε η αρίθμηση ξεκινά από τα δεξιά της λίστας **[-n]**

Ο τελεστής **+** είναι ο τελεστής σύνθεσης δύο ή περισσότερων αντικειμένων τύπου list

Ο τελεστής ***** είναι ο τελεστής επανάληψης

Η μέθοδος **len(listname)** δίνει το πλήθος των στοιχείων της λίστας

List type objects

```
#!/usr/bin/python3
```

```
mylist = ['abc', 234, 1.42, 'mary', 45.]
alist = [123, 'john']
```

```
print(mylist)           #Τυπώνει τη λίστα
print(mylist[0])        #Τυπώνει το 1ο στοιχείο της λίστας
print(mylist[1:3])      #Τυπώνει από το 2ο έως και το 3ο στοιχείο της λίστας
print(mylist[2:])       #Τυπώνει ξεκινώντας από το 3ο στοιχείο της λίστας
print(alist*2)          #Τυπώνει 2 φορές τη λίστα alist
newlist = mylist + alist #Σύνθεση νέας λίστας από τις δύο λίστες
print(newlist)          #Εκτύπωση της νέας λίστας
```

test1.py

Για να τρέξετε το πρόγραμμα:

1) στο terminal:

```
python3 test1.py
```

ή 2) σε περιβάλλον python

```
exec(open("test1.py").read())
```

```
['abc', 234, 1.42, 'mary', 45.]
```

```
abc
```

```
[234, 1.42]
```

```
[1.42, 'mary', 45.0]
```

```
[123, 'john', 123, 'john']
```

```
['abc', 234, 1.42, 'mary', 45.0, 123, 'john']
```

```
#!/usr/bin/python3
```

```
list = ['physics', 'chemistry', 1921, 2021]
```

```
print(list[-1])          #Πρόσβαση σε στοιχείο της λίστας από δεξιά προς τα αριστερά
```

```
print(len(list))         #Πλήθος στοιχείων λίστας
```

```
python3 test1.py
```

2021

4


List type objects – Αλλαγές στη λίστα

Μπορούμε να αλλάξουμε τα στοιχεία μιας λίστας είτε κάνοντας ανάθεση σε κάποια θέση της λίστας ή χρησιμοποιώντας την εντολή **append()**

```
#!/usr/bin/python3
```

```
list = ['physics', 'chemistry', 1921, 2021]
```

```
print("H timi tis listas sti thesi 2 einai: ")
print(list[1])
```

 test1.py

```
list[1] = 1812          #Ανάθεση νέας τιμής σε στοιχείο της λίστας
print("H nea timi tis listas sti thesi 2 einai: ")
print(list[1])
```

```
list.append(1812)       #Πρόσθεση στο τέλος της λίστας ενός νέου στοιχείου με τιμή 1812
print(len(list))        #Έλεγχος του πλήθους των στοιχείων της λίστας
print(list[len(list)-1]) #Εκτύπωση του τελευταίου στοιχείου της λίστας
```

Η θέση είναι το μήκος της λίστας – 1 γιατί η αρίθμηση των στοιχείων ξεκινά από το 0.

```
python3 test1.py
```

```
H timi tis listas sti thesi 2 einai:
chemistry
H nea timi tis listas sti thesi 2 einai:
1812
5
1812_
```

List type objects – Διαγραφή στοιχείου λίστας

Μπορούμε να διαγράψουμε στοιχεία μιας λίστας είτε χρησιμοποιώντας την εντολή **del** αν γνωρίζουμε ποιο στοιχείο θέλουμε να διαγράψουμε ή την μέθοδο **remove()** όταν δεν ξέρουμε την θέση του στοιχείου στη list αλλά την τιμή του

```
bash-3.2$ python3
Python 3.8.2 (default, Apr  8 2021, 23:19:18)
[Clang 12.0.5 (clang-1205.0.22.9)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> L=['physics', 'chemistry', 1921, 2023]
>>> print("To stoixeio tis listas sti thesi 2 einai: ",L[1])
To stoixeio tis listas sti thesi 2 einai:  chemistry
>>> del L[1]           #Διαγραφή στοιχείου της λίστας στη θέση 1
>>> L
['physics', 1921, 2023]
>>> L.remove(1921)    #Διαγραφή στοιχείου του στοιχείου με τιμή 1921
>>> L
['physics', 2023]
>>> █
```


List type objects – Συναρτήσεις λίστας

Υπάρχουν **μέθοδοι** που μπορούμε να χρησιμοποιήσουμε με λίστες όπως:

max(listname): Επιστρέφει το μέγιστο από τα στοιχεία μιας λίστας

min(listname): Επιστρέφει το ελάχιστο από τα στοιχεία μιας λίστας

list(tuple): Μετατρέπει ένα αντικείμενο tuple σε λίστα

comp(list1,list2): Συγκρίνει τα στοιχεία δύο λιστών

len(list1,list2): Επιστρέφει το μέγεθος της λίστας

listname.append(obj): Προσθέτει στο τέλος της λίστας listname το obj

listname.count(obj): Επιστρέφει τη συχνότητα εμφάνισης του obj

listname.extend(tuple): Προσθέτει στο τέλος της λίστας το tuple

listname.index(obj): Επιστρέφει τον μικρότερο δείκτη της θέσης του obj

listname.insert(index,obj): Εισάγει το obj στη θέση index

listname.pop(obj): Διαγράφει και τυπώνει το obj από τη λίστα

listname.remove(obj): Διαγράφει το obj από τη λίστα

listname.reverse(): Αντιστρέφει τη σειρά των δεδομένων στη λίστα

listname.sort([func]): Ταξινομεί τα δεδομένα στη λίστα σύμφωνα με τη func αν δίνεται

Tuple type objects

Τα αντικείμενα τύπου **tuple**, είναι ένα άλλο είδος λίστας δεδομένων που μοιάζει με τα αντικείμενα τύπου list.

Ένα αντικείμενο τύπου **tuple** περιέχει διάφορες **σταθερές** χωρισμένες με **,**
Σε αντίθεση με τις lists, οι τιμές είναι εγκλεισμένες σε ().

Ενώ τα στοιχεία μιας λίστας μπορεί να αλλάξουν ή η list να μεγαλώσει τα **tuples δεν μπορούν να τροποποιηθούν.**

Επομένως μπορούν να χρησιμοποιηθούν μόνο για ανάγνωση των στοιχείων τους αλλά δεν μπορούμε να τα αλλάξουμε

Μπορούμε να αποκτήσουμε πρόσβαση στα στοιχεία ενός tuple χρησιμοποιώντας τον τελεστή **[]** και **[:]** και κάποιον δείκτη ή αρίθμηση του οποίου ξεκινά από το 0

Ο τελεστής **+** είναι ο τελεστής σύνθεσης δύο ή περισσότερων αντικειμένων τύπου tuple

Ο τελεστής ***** είναι ο τελεστής επανάληψης

Ισχύουν οι ίδιες συναρτήσεις και μέθοδοι όπως και στις λίστες.

Δεν μπορούμε να εφαρμόσουμε την διαγραφή ενός στοιχείου (`del tuple[1]`) αλλά μόνο όλου του tuple: `del mytuple`

Υπάρχει μέθοδος μετατροπής λίστας σε tuple: `tuple(listname)`

Tuple type objects

```
#!/usr/bin/python3
```

```
mytuple = ('abc', 786, 2.32, 'john', 35.)
atuple = (123, 'john')
```

```
print(mytuple)           #Τυπώνει το tuple
print(mytuple[0])        #Τυπώνει το 1° στοιχείο του tuple
print(mytuple[1:3])      #Τυπώνει από το 2° έως και το 3° στοιχείο του tuple
print(mytuple[2:])       #Τυπώνει ξεκινώντας από το 3° στοιχείο του tuple
print(atuple*2)          #Τυπώνει 2 φορές του tuple atuple
print(mytuple+atuple)    #Σύνθεση νέου tuple από τα δύο tuples
print(newtuple)          #Εκτύπωση του νέου tuple
```

```
('abcd', 786, 2.32, 'john', 35.0)
abcd
(786, 2.32)
(2.32, 'john', 35.0)
(123, 'john', 123, 'john')
('abcd', 786, 2.32, 'john', 35.0, 123, 'john')
('abcd', 786, 2.32, 'john', 35.0, 123, 'john')
```

```
#!/usr/bin/python3
```

```
mytuple = ('abcd', 786, 2.32, 'john', 35.)
list = ['abcd', 786, 2.32, 'john', 35.]
```

```
mytuple[2] = 1000        #Λανθασμένη χρήση tuple – Δεν μπορούμε να αλλάξουμε στοιχείο tuple
```

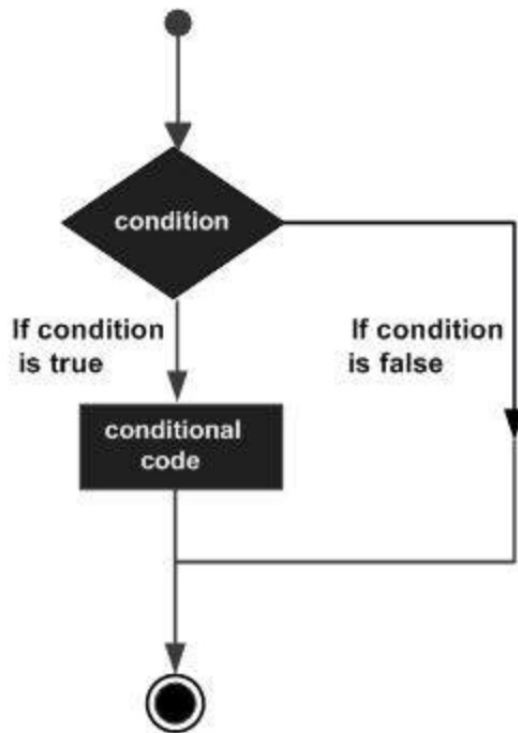
```
list[2] = 1000           #Σωστή χρήση τύπου list
```

Δομές Συνθήκης

Εντολές συνθήκης

Στα περισσότερα προγράμματα χρειάζεται να εξετάσουμε αν ισχύουν συγκεκριμένες συνθήκες για να αλλάξουμε τη ροή του προγράμματος.

Αν η συνθήκη που ελέγχεται είναι TRUE τότε εκτελούνται μια σειρά από εντολές ενώ αν είναι FALSE εκτελούνται κάποιες άλλες εντολές



Εντολή if: Αποτελείται από μια λογική έκφραση και ακολουθείται από μια σειρά εντολών

Εντολή if else ...:

Μια λογική έκφραση που ακολουθείται από μια σειρά εντολών ενώ αν είναι FALSE ακολουθείται από μια σειρά άλλων εντολών

Εντολή πολλαπλών if else if else if ...else ...:

Εξέταση διαφόρων περιπτώσεων

Εντολές συνθήκης

```
var = 100  
if ( var == 100 ) : print("Value of expression is 100")  
print("Good bye")
```

```
Value of expression is 100  
Good bye!
```

Εντολές συνθήκης if else

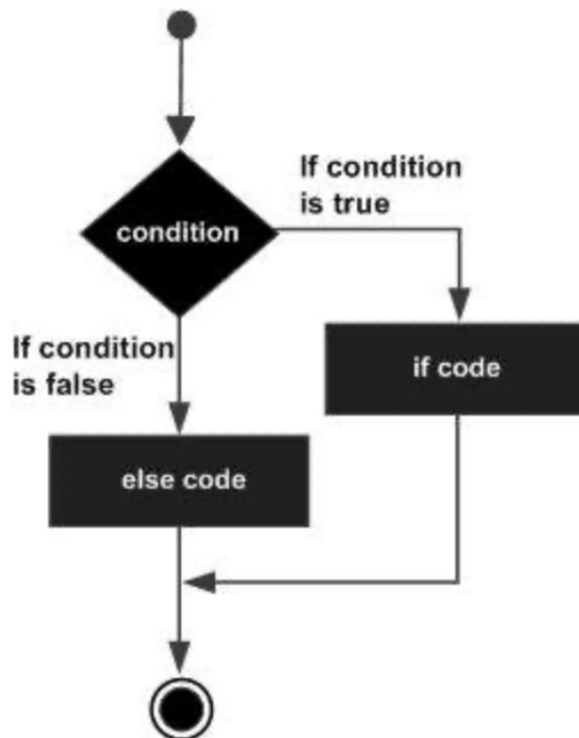
Η σύνταξη είναι:

If λογική έκφραση :

Εντολές

else :

Εντολές



```
var1 = 100
```

```
if var1 :
```

```
    print("1 – Got a true expression value")
```

```
    print(var1)
```

```
else:
```

```
    print("1 – Got a false expression value")
```

```
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

```
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!
```


Τελεστές σύγκρισης

Οι τελεστές σύγκρισης που χρησιμοποιούνται στην PYTHON είναι:

- ==** Αν οι τιμές δύο αντικειμένων είναι ίσες τότε το αποτέλεσμα ($a == b$) είναι TRUE
- !=** Αν οι τιμές δύο αντικειμένων **δεν** είναι ίσες τότε το αποτέλεσμα ($a != b$) είναι TRUE
- <>** Αν οι τιμές δύο αντικειμένων **δεν** είναι ίσες τότε το αποτέλεσμα ($a <> b$) είναι TRUE
- >** Αν η τιμή του αριστερού αντικειμένου είναι μεγαλύτερη του δεξιού τότε το αποτέλεσμα ($a > b$) είναι TRUE
- <** Αν η τιμή του αριστερού αντικειμένου είναι μικρότερη του δεξιού τότε το αποτέλεσμα ($a < b$) είναι TRUE
- >=** Αν η τιμή του αριστερού αντικειμένου είναι μεγαλύτερη ή ίση του δεξιού τότε το αποτέλεσμα ($a >= b$) είναι TRUE
- <=** Αν η τιμή του αριστερού αντικειμένου είναι μικρότερη ή ίση του δεξιού τότε το αποτέλεσμα ($a <= b$) είναι TRUE

```
a=21  
b=10  
c=0
```

```
if ( a == b ) :  
    print("Line 1 - a is equal to b")  
else :  
    print("Line 1 - a is not equal to b")
```

```
if ( a != b ) :  
    print("Line 2 - a is not equal to b")  
else :  
    print("Line 2 - a is equal to b")
```

```
if ( a <> b ) :  
    print("Line 3 - a is not equal to b")  
else :  
    print("Line 3 - a is equal to b")
```

```
if ( a < b ) :  
    print("Line 4 - a is less than b")  
else :  
    print("Line 4 - a is not less than b")
```

```
if ( a > b ) :  
    print("Line 5 - a is greater than b")  
else :  
    print("Line 5 - a is not greater than b")
```

```
a = 5;  
b = 20;  
if ( a <= b ) :  
    print("Line 6 - a is either less than or equal to b")  
else :  
    print("Line 6 - a is neither less than nor equal to b")
```

```
if ( b ==> a ) :  
    print("Line 7 - b is either greater than or equal to a")  
else :  
    print("Line 7 - b is neither greater than nor equal to a")
```

Παράδειγμα

Το αποτέλεσμα του διπλανού κώδικα είναι:

```
Line 1 - a is not equal to b  
Line 2 - a is not equal to b  
Line 3 - a is not equal to b  
Line 4 - a is not less than b  
Line 5 - a is greater than b  
Line 6 - a is either less than or equal to b  
Line 7 - b is either greater than or equal to b
```

Λογικοί Τελεστές

Οι λογικοί τελεστές που χρησιμοποιούνται στην PYTHON είναι:

AND Αν και οι δύο συνθήκες είναι TRUE τότε το αποτέλεσμα (**a and b**) είναι TRUE

OR Αν τουλάχιστον η μία από τις δύο συνθήκες είναι TRUE τότε το αποτέλεσμα (**a or b**) είναι TRUE

NOT Χρησιμοποιείται για να αντιστρέψει το λογικό αποτέλεσμα του αντικειμένου στο οποίο ενεργεί: **not (a and b)** είναι FALSE

```
>>> a, b, c = 5, 2, 20
>>> if (a > b and c > a):
...     print("The 1st condition is true")
... else:
...     print("The 1st condition is false")
...
The 1st condition is true
>>>
```

```
>>> if ( a < b or c > a):
...     print("The 2nd condition is true")
... else:
...     print("The 2nd condition is false")
...
The 2nd condition is true
```

```
>>> if ( not (a > b and c >a) ) :
...     print("The 2nd condition is true")
... else:
...     print("The 3rd condition is false")
...
The 3rd condition is false
```

Η δομή `try/except`

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

```
$ 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

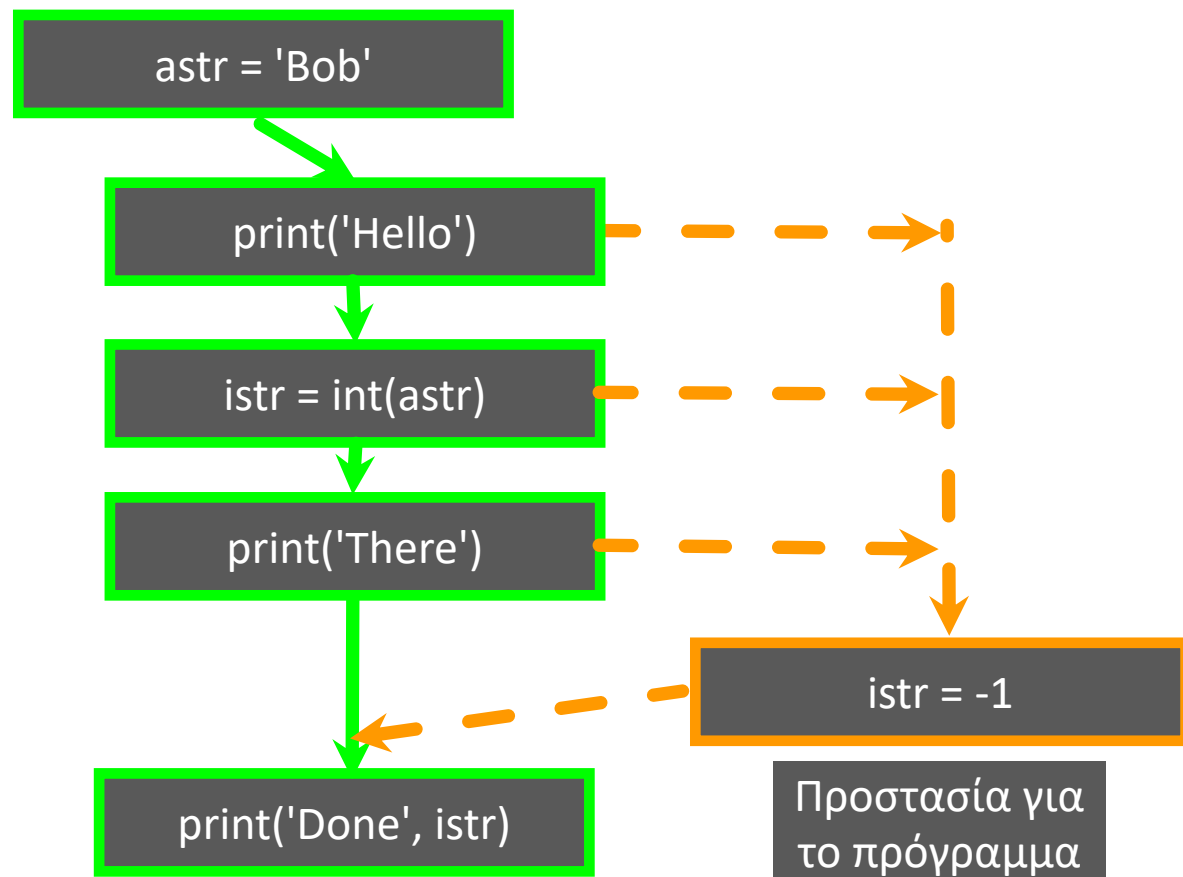
try / except

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

```
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
$
```