python™

Γιώργος Μανής
Επίκουρος Καθηγητής
Τμήματος Μηχανικών Η/Υ και Πληροφορικής
2 Ιουλίου 2014

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
«ΨΗΦΙΑΚΗ ΣΥΓΚΛΙΣΗ»

ΑΞΟΝΑΣ ΠΡΟΤΕΡΑΙΟΤΗΤΑΣ:	**ΤΠΕ και Βελτίωση της Ποιότητας Ζωής**
ΕΙΔΙΚΟΣ ΣΤΟΧΟΣ	**Βελτίωση της καθημερινής ζωής μέσω ΤΠΕ – Ισότιμη συμμετοχή των πολιτών στην Ψηφιακή Ελλάδα**

Σύμβαση για το Τμήμα Ζ – Αγροτική Ανάπτυξη – Περιβάλλον
του Υποέργου

«Μονάδες Αριστείας ΕΛ/ΛΑΚ»
σε 10 τμήματα στο πλαίσιο της Πράξης
«Ηλεκτρονικές Υπηρεσίες για την Ανάπτυξη και Διάδοση του Ανοιχτού Λογισμικού»

Python

- Η Python δημιουργήθηκε το **1990**
- Ο κύριος στόχος της είναι η **αναγνωσιμότητα** του κώδικά της και η **ευκολία** χρήσης της.
- έχει πολλές **βιβλιοθήκες** που διευκολύνουν ιδιαίτερα αρκετές συνηθισμένες εργασίες
- Και για την **ταχύτητα εκμάθησης** της

Πηγή: Βικιπαίδεια

Python – Ανοικτό Λογισμικό

- Η Python αναπτύσσεται ως **ανοιχτό λογισμικό**
- Η διαχείρισή της γίνεται από τον **μη κερδοσκοπικό οργανισμό** Python Software Foundation
- Ο κώδικας **διανέμεται** με την άδεια Python Software Foundation License η οποία είναι συμβατή με την GPL.

Python – Ανοικτό Λογισμικό

- Ο κώδικας διανέμεται με την άδεια **Python Software Foundation License** η οποία είναι συμβατή με την **GPL**. Οι χρήστες μπορουν:
  - να τρέξουν ένα πρόγραμμα για οποιοδήποτε λόγο.
  - να μελετήσουν τη λειτουργία ενός προγράμματος και να το τροποποιήσουν
  - να διανείμουν αντίγραφα του προγράμματος έτσι ώστε να βοηθήσουν τον πλησίον
  - να βελτιώσουν το πρόγραμμα και να προσφέρουν τις βελτιώσεις στο κοινό, έτσι ώστε να ωφεληθεί ολόκληρη η κοινότητα

python™

Python ☺

- Το **όνομα** της γλώσσας προέρχεται από την ομάδα Άγγλων κωμικών Μόντυ Πάιθον.

Εκδόσεις της Python

- Αρχικά, η Python ήταν γλώσσα σεναρίων που χρησιμοποιούνταν στο λειτουργικό σύστημα Amoeba, ικανή και για κλήσεις συστήματος.
- Η Python 2.0 κυκλοφόρησε το 2000
- Το 2008 κυκλοφόρησε η έκδοση 3.0. Πολλά από τα καινούργια χαρακτηριστικά αυτής της έκδοσης έχουν μεταφερθεί στις εκδόσεις 2.6 και 2.7 που είναι προς τα πίσω συμβατές.
- Η Python 3 είναι ιστορικά η πρώτη γλώσσα προγραμματισμού που σπάει την προς τα πίσω συμβατότητα με προηγούμενες εκδόσεις

Python Διερμηνευτής ή Μεταφραστής?

- Οι γλώσσες **δεν** είναι ούτε διερμηνευτές ούτε μεταφραστές
- Οι γλώσσες απλά **περιγράφουν κανόνες** με τους οποίους μπορούμε να «συνεννοηθούμε» με τον υπολογιστή
- Ο τρόπος **μετάφρασης** κώδικα σε γλώσσα μηχανής μπορεί να διαχωριστεί με βάση τις δύο αυτές τεχνολογίες
- Η χρυσή τομή όμως σε σύγχρονες γλώσσες βρίσκεται κάπου **ανάμεσα στις δύο φιλοσοφίες**, εκμεταλλευόμενες τα πλεονεκτήματα τις κάθε μίας από αυτές

Python Διερμηνευτής ή Μεταφραστής?

- Κάθε υλοποίηση της Python συνοδεύεται και από έναν **διερμηνευτή**
- Όταν ένα κώδικας γίνεται **import** (???) παράγεται bytecode για μία εικονική μηχανή της Python, ο κώδικας αυτός αποθηκεύεται στο δίσκο και δεν ξαναμεταφράζεται, παρά όταν ξαναγίνει import
- Αλλά το bytecode **δεν είναι** γλώσσα μηχανής, είναι ανεξάρτητο από το υλικό στο οποίο τρέχει

Python Διερμηνευτής ή Μεταφραστής?

- Υπάρχουν επίσης υλοποιήσεις της Python που ακολουθούν την πρακτική **just-in-time compilation**
- Η μετάφραση γίνεται σε **χρόνο εκτέλεσης**, όχι νωρίτερα
- Ο κώδικας που θα παραχθεί μπορεί να **αποθηκευτεί στο δίσκο** έτσι ώστε να μην είναι απαραίτητο να μεταφραστεί πάλι όταν εκτελεστεί

Python

- η Python είναι εξαιρετικά **δημοφιλής** γλώσσα προγραμματισμού
- μπορείς να κάνεις **εύκολα και γρήγορα** ανάπτυξη εφαρμογών
- Είναι της φιλοσοφίας του **ανοικτού κώδικα**
- έχει αναπτυχθεί ήδη **πολύ λογισμικό** σε Python
- Υπάρχει μεγάλος αριθμός από διαθέσιμες **βιβλιοθήκες**
- μπορεί κανείς να τη χρησιμοποιήσει ως γλώσσα **διαδικασιακού** ή και **αντικειμενοστρεφούς προγραμματισμού**

Διαδραστικός Διερμηνευτής

- >>> print("Hello, world!“)
- Hello, world!
- >>> 2+2
- 4
- >>> 1/2
- 0.5
- >>> 4/2
- 2.0

Ακέραιοι και Πραγματικοί

Οι αριθμοί 3 και 5 είναι ακέραιοι.

Οι αριθμοί 3.5 και 5.0 είναι αριθμοί κινητής υποδιαστολής.

- >>>1//2
- 0 (ακέραια διαίρεση)
- >>>1.0//2.0
- 0.0
- >>>1.0//2
- 0.0

python™

Διαίρεση και Υπόλοιπο

- >>>10%3
- 1 (υπόλοιπο διαίρεσης)
- >>>2.75/0.5
- 5.5
- >>>2.75//0.5
- 5.0
- >>>2.75%0.5
- 0.25

Ύψωση σε Δύναμη

- >>>2**3
- 8 (Ύψωση σε δύναμη)
- >>>-3**2
- -9
- >>>(-3)**2
- 9

Αριθμητικά Συστήματα

AF στο δεκαεξαδικό σύστημα ισούται με 10*16+15=175 στο δεκαδικό

- >>>0xAF
- 175

10 στο οκταδικό σύστημα ισούται με 1*8+0=8 στο δεκαδικό σύστημα

- >>>0o10
- 8

python™

Μεταβλητές

- >>>x=3 (απόδοση τιμής)
- >>>x*2
- 6
- >>>3*2
- 6
- >>>print(3*2)
- 6

Input

- >>>input(“How old are you? “)
- How old are you? 42
- ’42’
- >>>x=input(“x: “)
- x: 34
- >>>x
- ’34’

Input

- >>>x=int(input(“x: “))
- x: 34
- >>>y=float(input(“y: “))
- y: 42
- >>>print(x*y)
- 1428.0

Συναρτήσεις

- >>>2**3
- 8
- >>>pow(2,3)
- 8
- >>>10+pow(2,3*5)/3.0
- 10932.666666666666
- >>>abs(-10)
- 10
- >>>round(3.8)
- 4

Modules

Το “math” είναι module.

- >>>import math
- >>>math.floor(3.8)
- 3

Μπορώ να εισαγάγω συγκεκριμένες συναρτήσεις από ένα module:

- >>>from math import sqrt
- >>>sqrt(9)
- 3.0

Μιγαδικοί αριθμοί

- >>>sqrt(-1)
- Traceback (most recent call last):

  File "<pyshell#74>", line 1, in <module>

  sqrt(-1)

  ValueError: math domain error

Μιγαδικοί αριθμοί

- >>>import cmath
- >>>cmath.sqrt(-1)
- 1j
- >>>(1+3j)*(9+4j)
- (-3+31j)

python™

Αλφαριθμητικά (strings)

Μονά και διπλά εισαγωγικά είναι το ίδιο

- >>>”Hello, world!”
- ‘Hello, world!’
- >>>”Let’s go!”
- “Let’s go!”
- Εναλλακτικά:
- >>>’Let\’s go!’
- “Let’s go!”

Συνένωση αλφαριθμητικών

- >>> “Hello, ” + “world!”
- ‘Hello, world!’
- Εναλλακτικά:
- >>> “Hello, ” “world!”

- >>>x=“Hello, ”
- >>>y=“world!”
- >>>x+y
- ‘Hello, world!’

Δομές δεδομένων

- Οι βασικές δομές δεδομένων στην Python είναι οι ακολουθίες (sequences).
- Τα κύρια είδη ακολουθιών ειναι οι:
  - Λίστες (lists)
  - Πλειάδες (tuples)
- Η βασική τους διαφορά είναι ότι οι λίστες μπορούν να μεταβληθούν ενώ οι πλειάδες όχι.
- Τα αλφαριθμητικά (strings) είναι επίσης ακολουθίες.

Λίστες

- Έστω ότι θέλουμε να φτιάξουμε μία βάση δεδομένων που να περιλαμβάνει ονόματα και ηλικίες.
- >>>george=[‘Γιώργος Μανής’, 42]
- >>>peter=[‘Πέτρος Παπαδόπουλος’, 50]
- >>>database=[george, peter]
- >>>database
- [[‘Γιώργος Μανής’, 42], [‘Πέτρος Παπαδόπουλος’, 50]]

python™

Δείκτες θέσης (indexing)

- >>>greeting=‘Hello’
- >>>greeting[0]
- ‘H’
- **Οι ακολουθίες στην Python ξεκινούν από το στοιχείο 0.**
- >>>greeting[-1]
- ‘o’
- Αρνητικός δείκτης θέσης μετράει από τα δεξιά.
- >>>‘Hello’[1]
- ‘e’

Δείκτες θέσης (indexing)

- >>> fourth = input('Year: ')[3]
- Year: 2005
- >>> fourth
- '5'

Slicing

- >>>numbers=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- >>>numbers[3:6]
- [4, 5, 6]
- Ο πρώτος αριθμός είναι ο δείκτης θέσης του πρώτου στοιχείου, ενώ δεύτερος αριθμός είναι ο δείκτης θέσης του **επόμενου** στοιχείου από το τελευταίο που θέλουμε να συμπεριλάβουμε.
- >>>numbers[0:1]
- [1]

python™

Slicing

- >>>numbers=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- Αν μετρήσω τα στοιχεία από τα δεξιά:
- >>>numbers[-3:-1]
- [8, 9]
- Έτσι δεν μπορώ να προσπελάσω το τελευταίο στοιχείο:
- >>>numbers[-3:0]
- []

Slicing

- Μπορώ να κάνω το εξής:
- >>>numbers[-3:]
- [8, 9, 10]
- Επίσης:
- >>>numbers[:3]
- [1, 2, 3]
- >>>numbers[:]
- [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Βήμα

- >>>numbers[0:10:1]
- [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- >>>numbers[0:10:2]
- [1, 3, 5, 7, 9]
- >>>numbers[3:6:3]
- [4]
- >>>numbers[::4]
- [1, 5, 9]

Αρνητικό βήμα

- >>>numbers=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- >>>numbers[8:3:-1]
- [9, 8, 7, 6, 5]
- >>>numbers[10:0:-2]
- [10, 8, 6, 4, 2]
- >>>numbers[0:10:-2]
- []
- >>>numbers[::-2]
- [10, 8, 6, 4, 2]
- >>>numbers[5::-2]
- [6, 4, 2]
- >>>numbers[:5:-2]
- [10, 8]

Πρόσθεση ακολουθιών

- >>>[1, 2, 3] + [4, 5, 6]
- [1, 2, 3, 4, 5, 6]
- >>>’Hello, ‘ + ‘world!’
- ‘Hello, world’
- >>>[1, 2, 3] + ‘world!’
- Traceback (most recent call last):
- File "<pyshell#31>", line 1, in <module>
- [1, 2, 3] + 'world!'
- TypeError: can only concatenate list (not "str") to list

python™

Πολλαπλασιασμός

- >>>’python’ * 5
- 'pythonpythonpythonpythonpython‘
- >>>[42] * 10
- [42, 42, 42, 42, 42, 42, 42, 42, 42, 42]

Λίστες

Άδεια λίστα και None

- Άδεια λίστα: []
- None: Χρησιμοποιείται όταν θέλουμε μία λίστα π.χ. με 10 στοιχεία αλλά δεν έχουμε ακόμα αναθέσει τιμές:
- >>>sequence = [None] * 10
- >>>sequence
- [None, None, None, None, None, None, None, None, None, None]

Ιδιότητα του μέλους

- >>>permissions='rw'
- >>>'w' in permissions
- True
- >>>'x' in permissions
- False
- Οι True και False είναι Boolean τιμές:
  - True: Αληθές
  - False: Ψευδές

Ιδιότητα του μέλους

- >>> users = [‘χρήστης1', ‘χρήστης3', ‘χρηστης9']
- >>> input('Enter your user name: ') in users
- Enter your user name: mlh
- True
- >>> subject = '$$$!!! $$$'
- >>> '$$$' in subject
- True

παράθεση

```
>>> x
[1, 5, 4, 7, 8, 9]
>>> y
[7, 8, 9]
>>> x+y
[1, 5, 4, 7, 8, 9, 7, 8, 9]
>>> a=x+y
>>> a
[1, 5, 4, 7, 8, 9, 7, 8, 9]
>>> x
[1, 5, 4, 7, 8, 9]
>>> y
[7, 8, 9]
```

η συνάρτηση list

- δημιουργία λίστας
- >>> list ('Hello')
- ['H', 'e', 'l', 'l', 'o']
- >>>

- >>> x=list('Hello')
- >>> x
- ['H', 'e', 'l', 'l', 'o']
- >>>

min, max, len

- >>> x=[3,5,2,8,7]
- >>> min(x)
- 2
- >>> max(x)
- 8
- >>> len(x)
- 5
- >>>

τροποποίηση στοιχείων λίστας

- >>> x=[1,2,3]
- >>> x
- [1, 2, 3]
- >>> x[2]=-1
- >>> x
- [1, 2, -1]
- >>>

python™

διαγραφή στοιχείων λίστας

- >>> x=[1,2,3]
- >>> del x[1]
- >>> x
- [1, 3]
- >>> del x[0]
- >>> del x[0]
- >>> x
- []
- >>>

αντικατάσταση μέρους λίστας

- >>> list = [1,2,3,4,5,6]
- >>> list[3:]=[7,8,9]
- >>> list
- [1, 2, 3, 7, 8, 9]
- >>> list[1:2]=[10,11,12,13]
- >>> list
- [1, 10, 11, 12, 13, 3, 7, 8, 9]

αντικατάσταση μέρους λίστας

- >>> list[1:1]=[20,21,22]
- >>> list
- [1, 20, 21, 22, 10, 11, 12, 13, 3, 7, 8, 9]
- >>> list[4:]=[]
- >>> list
- [1, 20, 21, 22]
- >>>

append

- >>> lst = [1, 2, 3]
- >>> lst.append(4)
- >>> lst
- [1, 2, 3, 4]

count

- >>> ['to', 'be', 'or', 'not', 'to', 'be'].count('to')
- 2
- >>> x = [[1, 2], 1, 1, [2, 1, [1, 2]]]
- >>> x.count(1)
- 2
- >>> x.count([1, 2])
- 1

index

- >>> a = [1, 5, 4, 7, 8, 9, 7, 8, 9,]
- >>> a.index(7)
- 3
- >>> a.index(9)
- 5

insert

- >>> a = [1, 5, 4, 7, 8, 9, 7, 8, 9, -1]
- >>>
- >>> a.insert(2,10)
- >>> a
- [1, 5, 10, 4, 7, 8, 9, 7, 8, 9, -1]

pop

- >>> a = [1, 5, 10, 4, 7, 8, 9, 7, 8, 9, -1]
- >>> a.pop()
- -1
- >>> a
- [1, 5, 10, 4, 7, 8, 9, 7, 8, 9]
- >>> a.pop(0)
- 1
- >>> a
- [5, 10, 4, 7, 8, 9, 7, 8, 9]
- >>> a.pop(2)
- 4
- >>> a
- [5, 10, 7, 8, 9, 7, 8, 9]

remove

- >>> a
- [5, 10, 7, 8, 9, 7, 8, 9]
- >>>
- >>> a.remove(9)
- >>> a
- [5, 10, 7, 8, 7, 8, 9]

reverse

- >>> x = [1, 2, 3]
- >>> x.reverse()
- >>> x
- [3, 2, 1]

- >>> x = [1, 2, 3]
- >>> list(reversed(x))
- [3, 2, 1]
- >>> x
- [1, 2, 3]

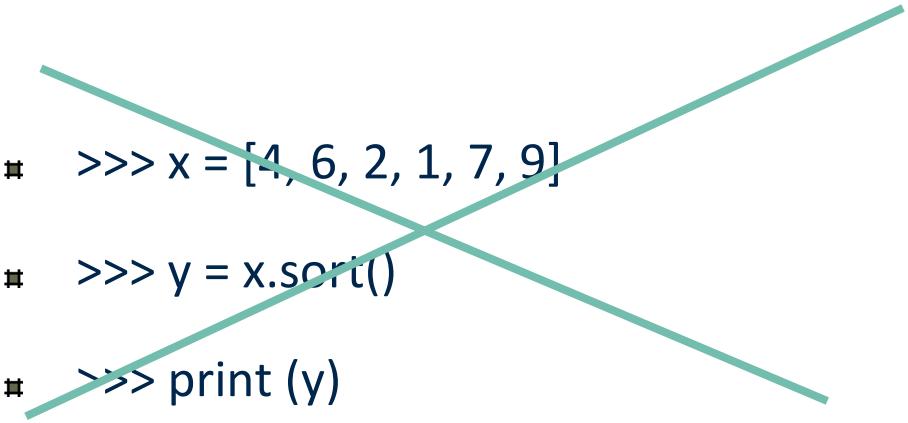
extend

- >>> a = [1, 2, 3]
- >>> b = [4, 5, 6]
- >>> a.extend(b)
- >>> a
- [1, 2, 3, 4, 5, 6]

- >>> a = [1, 2, 3]
- >>> b = [4, 5, 6]
- >>> a + b
- [1, 2, 3, 4, 5, 6]
- >>> a
- [1, 2, 3]

sort

- >>> x = [4, 6, 2, 1, 7, 9]
- >>> x.sort()
- >>> x
- [1, 2, 4, 6, 7, 9]



- >>> x = [4, 6, 2, 1, 7, 9]
- >>> y = x.sort()
- >>> print (y)
- None

sort

- >>> x = [4, 6, 2, 1, 7, 9]
- >>> y = x[:]
- >>> y.sort()
- >>> x
- [4, 6, 2, 1, 7, 9]
- >>> y
- [1, 2, 4, 6, 7, 9]

sort

- >>> y = x
- >>> y.sort()
- >>> x
- [1, 2, 4, 6, 7, 9]
- >>> y
- [1, 2, 4, 6, 7, 9]

sort

- >>> x = [4, 6, 2, 1, 7, 9]
- >>> y = sorted(x)
- >>> x
- [4, 6, 2, 1, 7, 9]
- >>> y
- [1, 2, 4, 6, 7, 9]

Πλειάδες

Πλειάδες

- Οι πλειάδες δημιουργούνται και δεν τροποποιούνται στη συνέχεια

- >>> 1, 2, 3
- (1, 2, 3)

- >>> ()
- ()

Δημιουργία από συνάρτηση

- >>> tuple([1, 2, 3])
- (1, 2, 3)
- >>> tuple('abc')
- ('a', 'b', 'c')
- >>> tuple((1, 2, 3))
- (1, 2, 3)

python™

Λειτουργίες σε μία πλειάδα

- >>> x = 1, 2, 3
- >>> x[1]
- 2
- >>> x[0:2]
- (1, 2)

Χρησιμότητα ?

- Θα μπορούσαμε να χρησιμοποιούσαμε και λίστες αντί τις πλειάδες
- Κάποιες λειτουργίες όμως που θα δούμε αργότερα τις χρησιμοποιούν

Λεξικά

Λεξικά

- χρησιμοποιούν κλειδιά και όχι απλούς δείκτες
- η αναζήτηση βασίζεται στα κλειδιά, τα οποία είναι πλειάδες

- π.χ.
- phonebook =

{‘Αλίκη': '2341', ‘Μπέτυ': '9102', ‘Σεσίλ': '3258'}

Συνάρτηση dict

- >>> items = [('name', ‘Γιώργος'), ('age', 42)]
- >>> d = dict(items)
- >>> d
- {'age': 42, 'name': ‘Γιώργος'}
- >>> d['name']
- ‘Γιώργος‘
- ή
- d = dict(name=‘Γιώργος', age=42)
- >>> d
- {'age': 42, 'name': ‘Γιώργος'}

Βασικές Λειτουργίες

- >>> x = []
- >>> x[42] = ‘τεστ'
- Traceback (most recent call last):
- File "<stdin>", line 1, in ?
- IndexError: list assignment index out of range
- >>> x = {}
- >>> x[42] = ‘τεστ'
- >>> x
- {42: ‘τεστ'}

clear

- >>> d = {}
- >>> d['name'] = ‘Γιώργος'
- >>> d['age'] = 42
- >>> d
- {'age': 42, 'name': ‘Γιώργος'}
- >>> returned_value = d.clear()
- >>> d
- {}
- >>> print (returned_value)
- None

fromkeys

- >>> {}.fromkeys(['name', 'age'])
- {'age': None, 'name': None}

- >>> dict.fromkeys(['name', 'age'])
- {'age': None, 'name': None}

- >>> dict.fromkeys(['name', 'age'], '(unknown)')
- {'age': '(unknown)', 'name': '(unknown)'}

get

- >>> d = {}
- >>> print (d['name'])
- Traceback (most recent call last):
- File "<stdin>", line 1, in ?
- KeyError: 'name‘

- >>> print (d.get('name'))
- None

- >>> d.get('name', 'N/A')
- ‘N/A‘

- >>> d['name'] = 'Eric'
- >>> d.get('name')
- 'Eric'

has_key

- >>> d = {}
- >>> d.has_key('name')
- 0
- >>> d['name'] = 'Eric'
- >>> d.has_key('name')
- 1

items

- >>> d = {'title': 'Python Web Site', 'url': 'http://www.python.org', 'spam': 0}
- >>> d.items()
- [('url', 'http://www.python.org'), ('spam', 0), ('title', 'Python Web Site')]

pop

- >>> d = {'x': 1, 'y': 2}
- >>> d.pop('x')
- 1
- >>> d
- {'y': 2}
- >>> d = {'x': 1, 'y': 2}
- >>> d.pop('x')
- 1
- >>> d
- {'y': 2}

popitem

- >>> d
- {'url': 'http://www.python.org', 'spam': 0, 'title': 'Python Web Site'}
- >>> d.popitem()
- ('url', 'http://www.python.org') **τυχαίο**
- >>> d
- {'spam': 0, 'title': 'Python Web Site'}

update

- >>> d = {
- 'title': 'Python Web Site',
- 'url': 'http://www.python.org',
- 'changed': 'Mar 14 22:09:15 MET 2005'
- }
- >>> x = {'title': 'Python Language Website'}
- >>> d.update(x)
- >>> d
- {'url': 'http://www.python.org', 'changed': 'Mar 14 22:09:15 MET 2005‘, 'title': 'Python Language Website'}

Δομές Έλέγχου

Μπλοκ κώδικα

- Ένα μπλοκ κώδικα είναι μία ομάδα εντολών η οποία θα εκτελεστεί αν μία συνθήκη είναι αληθής ή θα εκτελεστεί πολλές φορές (βρόχος)
- Στην Python, το μπλοκ δημιουργείται γράφοντας το αντίστοιχο μέρος του κώδικα σε εσοχή (βάζοντας διαστήματα μπροστά).
- Προτείνονται τέσσερα διαστήματα.

Μπλοκ κώδικα

- this is a line
- this is another line:
- this is another block
- continuing the same block
- the last line of this block
- phew, there we escaped the inner block

- Η αρχή του μπλοκ ορίζεται με «:»

Συνθήκες

- Είχαμε μιλήσει για Boolean τιμές.
- True, False
- Στην Python, οι παρακάτω τιμές θεωρούνται ισοδύναμες με False:
- False None 0 "" () [] {}
- Οποιαδήποτε άλλη τιμή θεωρείται ισοδύναμη με True.

True και False

- Βασικά, True σημαίνει 1 και False σημαίνει 0.
- >>> True
- True
- >>> False
- False
- >>> True == 1
- True
- >>> False == 0
- True
- >>> True + False + 42
- 43

Τύπος bool

- Οι τιμές True και False ανήκουν στον τύπο bool.
- bool('I think, therefore I am')
- True
- >>> bool(42)
- True
- >>> bool('')
- False
- >>> bool(0)
- False

Η εντολή if

- name = input('What is your name? ')
- if name.endswith('Gumby'):
- print('Hello, Mr. Gumby‘)

elif και else

```
num = eval(input('Enter a number: '))
if num > 0:
    print('The number is positive')
elif num < 0:
    print('The number is negative')
else:
    print('The number is zero')
```

Φωλιασμένα μπλοκ

```
name = input('What is your name? ')
if name.endswith('Gumby'):
    if name.startswith('Mr.'):
        print('Hello, Mr. Gumby')
    elif name.startswith('Mrs.'):
        print('Hello, Mrs. Gumby')
    else:
        print('Hello, Gumby')
else:
    print('Hello, stranger')
```

Τελεστές σύγκρισης

- x == y το x ισούται με το y.
- x < y το x είναι μικρότερο από το y.
- x > y το x είναι μεγαλύτερο από το y.
- x >= y το x είναι μεγαλύτερο ή ίσο από το y.
- x <= y το x είναι μικρότερο ή ίσο από το y.
- x != y το x είναι διάφορο του y.
- x is y x και y είναι το ίδιο αντικείμενο.
- x is not y x και y είναι διαφορετικά αντικείμενα.
- x in y το x είναι μέλος της ακολουθίας y.
- x not in y το x δεν είναι μέλος της ακολουθίας y.

Τελεστές σύγκρισης

- >>> "foo" == "foo"
- True
- >>> "foo" == "bar"
- False
- >>> "foo" = "foo"
- SyntaxError: can't assign to literal

Ο τελεστής is (τελεστής ταυτότητας)

- >>> x = y = [1, 2, 3]
- >>> z = [1, 2, 3]
- >>> x == y
- True
- >>> x == z
- True
- >>> x is y
- True
- >>> x is z
- False

Ο τελεστής in

```
name = input('What is your name? ')
if 's' in name:
    print('Your name contains the letter "s".‘)
else:
    print('Your name does not contain the letter "s".‘)
```

Σύγκριση αλφαριθμητικών και ακολουθιών

- >>> "alpha" < "beta"
- True
- [1, 2] < [2, 1]
- True

Boolean τελεστές

```
number = int(input('Δώσε μου αριθμό ανάμεσα στο 1 και το 10: '))
if number <= 10:
    if number >= 1:
        print 'Σωστά!'
    else:
        print 'Λάθος!'
else:
    print 'Λάθος!'
```

Boolean τελεστές

- Καλύτερα:
- number = int(input(' Δώσε μου αριθμό ανάμεσα στο 1 και το 10 : '))
- if number <= 10 and number >= 1:
- print ‘Σωστά!'
- else:
- print ‘Λάθος!‘

- Ακόμα καλύτερα:
- if 1<=number<=10: …

Boolean τελεστές

- and, or, not όσο πολύπλοκα θέλουμε
- if ((cash > price) or customer_has_good_credit) and not out_of_stock:

    give_goods()

Ο βρόχος while

```
x = 1
while x <= 100:
    print(x)
    x += 1


name = ''
while not name:
    name = input('Please enter your name: ')
print('Hello, %s!' % name)
```

Ο βρόχος for

```
words = ['this', 'is', 'an', 'ex', 'parrot']
for word in words:
    print(word)


numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for number in numbers:
    print(number)
```

range

- for number in range(1,101):
- print(number)

- >>>list(range(0,10))
- [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Το δεύτερο όριο δεν συμπεριλαμβάνεται.

Επαναλήψεις σε λεξικά

```
d = {'x': 1, 'y': 2, 'z': 3}
for key in d:
    print(key, 'corresponds to', d[key])
```

Εναλλακτικά:

```
d = {'x': 1, 'y': 2, 'z': 3}
for key, value in d.items():
    print(key, 'corresponds to', value)
```

Παράλληλες επαναλήψεις

- names = [‘anne', 'beth', 'george', 'damon']
- ages = [12, 45, 32, 102]
- for i in range(len(names)):

    print(names[i], 'is', ages[i], 'years old')

Η συνάρτηση zip

- >>>list(zip(names,ages))
- [('anne', 12), ('beth', 45), ('george', 32), ('damon', 102)]

- >>>list(zip(range(5), range(100000000)))
- [(0, 0), (1, 1), (2, 2), (3, 3), (4, 4)]

Παράλληλες επαναλήψεις με zip

```
names = ['anne', 'beth', 'george', 'damon']
ages = [12, 45, 32, 102]
for name, age in zip(names, ages):
    print(name, 'is', age, 'years old')
```

Αντικατάσταση αλφαριθμητικού

```
for string in strings:
    if 'xxx' in string:
        index = strings.index(string)
        strings[index] = '[censored]’
```

Εναλλακτικά:

```
index = 0
for string in strings:
    if 'xxx' in string:
        strings[index] = '[censored]'
    index += 1
```

reversed και sorted

- >>> sorted([4, 3, 6, 8, 3])
- [3, 3, 4, 6, 8]
- >>> sorted('Hello, world!')
- [' ', '!', ',', 'H', 'd', 'e', 'l', 'l', 'l', 'o', 'o', 'r', 'w']
- >>> list(reversed('Hello, world!'))
- ['!', 'd', 'l', 'r', 'o', 'w', ' ', ',', 'o', 'l', 'l', 'e', 'H']
- >>> ''.join(reversed('Hello, world!'))
- '!dlrow ,olleH'

Η εντολή break

- Θέλω να βρω το μεγαλύτερο τέλειο τετράγωνο μικρότερο του 100.
- from math import sqrt
- for n in range(99, 0, -1):
- root = sqrt(n)
- if root == int(root):
- print(n)
- break

- Η break μας βγάζει έξω από το βρόχο

range με βήμα

- >>>list(range(99, 0, -1))
- [99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

- >>>list(range(0,10,2))
- [0, 2, 4, 6, 8]

Ο ιδιωματισμός while True/break

- word = 'dummy'
- while word:
- word = input('Please enter a word: ')
- # do something with the word:
- print('The word was ' + word)

- Χρειαζόμαστε την αρχική τιμή στη word.

Ο ιδιωματισμός while True/break

- Εναλλακτικά:
- word = input('Please enter a word: ')
- while word:
- # do something with the word:
- print('The word was ' + word)
- word = input('Please enter a word: ')

Ο ιδιωματισμός while True/break

- Με χρήση του while True/break:

- while True:
- word = input('Please enter a word: ')
- if not word: break
- # do something with the word:
- print('The word was ' + word)

Χρήση Boolean σημαίας

```
broke_out = False
for x in seq:
    do_something(x)
    if condition(x):
        broke_out = True
        break
    do_something_else(x)
if not broke_out:
    print "I didn't break out!"
```

Χρήση else στη for

```
from math import sqrt
for n in range(99, 81, -1):
    root = sqrt(n)
    if root == int(root):
        print(n)
        break
else:
    print("Didn't find it!")
```

Περισσότερα για λίστες

- >>> [x*x for x in range(10)]
- [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

- >>> [x*x for x in range(10) if x % 3 == 0]
- [0, 9, 36, 81]

- >>> [(x, y) for x in range(3) for y in range(3)]
- [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]

Περισσότερα για λίστες

- >>> girls = ['alice', 'bernice', 'clarice']
- >>> boys = ['chris', 'arnold', 'bob']
- >>> [b+'+'+g for b in boys for g in girls if b[0] == g[0]]
- ['chris+clarice', 'arnold+alice', 'bob+bernice']

Η εντολή pass

- Δεν κάνει τίποτα!
- if name == 'Ralph':

        print 'Welcome!'

  elif name == 'Enid':

        # Not finished yet...

        pass

  elif name == 'Bill Gates':

        print 'Access Denied'

Παραδείγματα

Ύψωση σε δύναμη

```
x = int(input('Dose ti vasi: '))
y = int(input('Dose ti dynami: '))
value = 1
for i in range(0, y):
    value *= x
print('%d eis tin %d = %d' % (x, y, value))
```

Παραγοντικό

$$n! = 1 \cdot 2 \cdot 3 \cdots n$$

Παραγοντικό ενός αριθμού

```
x = int(input('Dose mou ton arithmo: '))
y = 1
for i in range(1,x+1):
    y *= i
print('To paragontiko tou %d einai %d' % (x,y))
```

python™

Υπολογισμός ψηφίων ακέραιου αριθμού

- Παράδειγμα:

235=2*100+3*10+5

Πρώτο ψηφίο: 235%10=5

235//10=23

Δεύτερο ψηφίο:

23%10=3

23//10=2

Τρίτο ψηφίο: 2%10=2

2//10=0

Υπολογισμός αθροίσματος ψηφίων ακέραιου αριθμού

```
n = int(input('Dose mou ton arithmo: '))
dsum = 0
while n>0:
    dsum += n%10
    n//=10
print('To athroisma ton psifion tou arithmou einai %d' % dsum)
```

Πίνακες

Πίνακες

- Μπορούμε να χρησιμοποιήσουμε λίστες ως πίνακες.
- Μονοδιάστατος πίνακας:
- A=[1, 2, 3, 4, 5]
- Διδιάστατος πίνακας:
  B=[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
- Κάθε γραμμή του πίνακα είναι μία λίστα.

Αρχικοποίηση πινάκων

- Μονοδιάστατος πίνακας
- N=5
- A=[0]*N
- Εναλλακτικά:
- A=[0 for i in range(N)]

Αντί για 0 μπορούμε να βάλουμε και None.

Αρχικοποίηση πινάκων

- Δισδιάστατος πίνακας
- N=3
- pinakas=[[0 for i in range(N)] for i in range(N)]

pinakas

0	pinakas[0][0]	pinakas[0][1]	pinakas[0][2]
1	pinakas[1][0]	pinakas[1][1]	pinakas[1][2]
2	pinakas[2][0]	pinakas[2][1]	pinakas[2][2]
	0	1	2

Προσοχή: Είναι pinakas[i][j] και όχι pinakas[i, j]

Πρόσθεση πινάκων

```
N = 3
A = [[0.0 for j in range(N)] for i in range(N)]
B = [[0.0 for j in range(N)] for i in range(N)]
C = [[0.0 for j in range(N)] for i in range(N)]
for i in range(N):
    for j in range(N):
        A[i][j] = float(input('A[%d][%d]: ' % (i, j)))
    for i in range(N):
        for j in range(N):
            B[i][j] = float(input('B[%d][%d]: ' % (i, j)))
```

Πρόσθεση πινάκων (συνέχεια)

```
for i in range(N):
    for j in range(N):
        C[i][j] = A[i][j] + B[i][j]


for i in range(N):
    for j in range(N):
        print('C[%d][%d]:%7.2f ' % (i, j, C[i][j]), end = '')
    print('')
```

Πολλαπλασιασμός πινάκων NxN

$$C = A \cdot B$$

$$C[i, j] = \sum_{k=0}^{N-1} A[i, k] B[k, j]$$

python™

Πολλαπλασιασμός πινάκων

```
N = 3

A = [[0.0 for j in range(N)] for i in range(N)]

B = [[0.0 for j in range(N)] for i in range(N)]

C = [[0.0 for j in range(N)] for i in range(N)]

for i in range(N):

    for j in range(N):

        A[i][j] = float(input('A[%d][%d]: ' % (i, j)))

for i in range(N):

    for j in range(N):

        B[i][j] = float(input('B[%d][%d]: ' % (i, j)))
```

Πολλαπλασιασμός πινάκων (συνέχεια)

```
for i in range(N):
    for j in range(N):
        thisElement = 0.0
        for k in range(N):
            thisElement += A[i][k]*B[k][j]
        C[i][j] = thisElement
for i in range(N):
    for j in range(N):
        print('C[%d][%d]:%7.2f ' % (i, j, C[i][j]), end = '')
    print('‘)
```

Ανάστροφος πίνακα

```
N = 3

A = [[0.0 for j in range(N)] for i in range(N)]

C = [[0.0 for j in range(N)] for i in range(N)]

for i in range(N):

    for j in range(N):

        A[i][j] = float(input('A[%d][%d]: ' % (i, j)))

for i in range(N):

    for j in range(N):

        C[i][j] = A[j][i]
```

Ανάστροφος πίνακα (συνέχεια)

```
for i in range(N):
    for j in range(N):
        print('C[%d][%d]:%7.2f ' % (i, j, C[i][j]), end = '')
    print('')
```

Συναρτήσεις

Hello world

```
>>> def helloWorld():
        print('Hello World')


>>> helloWorld()
Hello World
>>>
```

Hello world

```
>>> def hello(name):
        return 'Hello, '+name+' !'


>>> hello ('world')
'Hello, world !'
>>> hello ('George')
'Hello, George !'
>>>
```

Fibonacci numbers

- fib(0) = 1
- fib(1) = 1
- fib(n)=fib(n-1)+fib(n-2)

Fibonacci numbers

```
>>> def fibs(num):
        result = [0, 1]
        for i in range(num-2):
                result.append(result[-2] + result[-1])
        return result


>>> fibs(10)
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
>>>
```

Τεκμηρίωση

```
>>> def hello(name):
        'prints hello, a comma and a name following them'
        return 'Hello, '+name+' !'


>>> hello('George')
'Hello, George !'
>>> hello.__doc__
'prints hello, a comma and a name following them'
>>>
```

Τεκμηρίωση

- >>> from math import sqrt
- >>> help(sqrt)
- Help on built-in function sqrt in module math:

- sqrt(...)
- sqrt(x)
-
- Return the square root of x.

Κάτι σαν συναρτήσεις …

```
>>> def OK():
        print ('OK');
        return
        print ('????');

>>> OK()
OK
>>> x=OK()
OK
>>> x
>>>
```

Κάτι σαν συναρτήσεις …

```
>>> def OK():
        print ('OK');
        return
        print ('????');

>>> OK()
OK
????
>>> x=OK()
OK
????
>>> x
>>>
```

Παράμετροι

```
def hello(name):
        return 'Hello, '+name+' !'
```

```
>>> x='George'
>>> hello(x)
'Hello, George !'
>>> x
'George'
```

Παράμετροι

```
def hello(name):
        name='123'
        return 'Hello, '+name+' !'

>>> x='George'
>>> hello(x)
'Hello, 123 !'
>>> x
'George'
>>>
```

Παράμετροι

- >>> def change(n):
- n[0] = 'Mr. Gumby'
- >>> names = ['Mrs. Entity', 'Mrs. Thing']
- >>> change(names)
- >>> names
- ['Mr. Gumby', 'Mrs. Thing']

Παράμετροι

```
def hello_1(greeting, name):
        print '%s, %s!' % (greeting, name)
def hello_2(name, greeting):
        print '%s, %s!' % (name, greeting)
```

```
>>> hello_1('Hello', 'world')
Hello, world!
>>> hello_2('Hello', 'world')
Hello, world!
```

Παράμετροι

- >>> hello_1(greeting='Hello', name='world')
- Hello, world!

- >>> hello_1(name='world', greeting='Hello')
- Hello, world!
- The names *do, however (as you may have gathered):*
- >>> hello_2(greeting='Hello', name='world')
- world, Hello!

Παράμετροι

- Πιο ευανάγνωστος κώδικας
- >>> store('Mr. Brainsample', 10, 20, 13, 5)
- >>> store(patient='Mr. Brainsample',

  hour=10, minute=20, day=13, month=5)

Παράμετροι

- def hello_3(greeting='Hello', name='world'):
  print '%s, %s!' % (greeting, name)
- >>> hello_3()
- Hello, world!
- >>> hello_3('Greetings')
- Greetings, world!
- >>> hello_3('Greetings', 'universe')
- Greetings, universe!
- >>> hello_3(name='Gumby')
- Hello, Gumby!

Παράμετροι

```
>>> def hello(greetings='Hello',name='world'):
        print(greetings+', '+name)


>>> hello()
Hello, world
>>> hello(greetings='Hi')
Hi, world
>>> hello(name='Mike')
Hello, Mike
```

Παράμετροι

- >>> hello(name='Mike',greetings='Hi')
- Hi, Mike
- >>> hello('Hi','Mike')
- Hi, Mike
- >>> hello('Mike','Hi')
- Mike, Hi
- >>> hello('Hi')
- Hi, world
- >>> hello('Mike')
- Mike, world

Μεταβλητός αριθμός παραμέτρων

- def print_params(*params):

  print (params)

- >>> print_params('Testing')
- ('Testing',)

- >>> print_params(1, 2, 3)
- (1, 2, 3)

Μεταβλητός αριθμός παραμέτρων

```
def print_params_2(title, *params):
        print (title)
        print (params)

print_params_2('Params:', 1, 2, 3)
Params:
(1, 2, 3)

>>> print_params_2('Nothing:')
Nothing:
()
```

Μεταβλητός αριθμός παραμέτρων

- def print_params_3(**params):

  print (params)

- >>> print_params_3(x=1, y=2, z=3)
- {'z': 3, 'x': 1, 'y': 2}

Μεταβλητός αριθμός παραμέτρων

```
def print_params_4(x, y, z=3, *pospar, **keypar):
        print (x, y, z)
        print (pospar)
        print (keypar)
>>> print_params_4(1, 2, 3, 5, 6, 7, foo=1, bar=2)
1 2 3
(5, 6, 7)
{'foo': 1, 'bar': 2}
>>> print_params_4(1, 2)
1 2 3
()
{}
```

Κατανομή στις παραμέτρους

- def add(x, y): return x + y
- params = (1, 2)
- >>> add(*params)
- 3

Κατανομή στις παραμέτρους

- >>> params = {'name': 'Sir Robin', 'greeting': 'Well met'}
- >>> hello_3(**params)
- Well met, Sir Robin!

Κατανομή στις παραμέτρους

- >>> def with_stars(**kwds):

  print (kwds['name'], 'is', kwds['age'], 'years old‘)

- >>> def without_stars(kwds):

  print (kwds['name'], 'is', kwds['age'], 'years old‘)

- >>> args = {'name': 'Mr. Gumby', 'age': 42}
- >>> with_stars(**args)
- Mr. Gumby is 42 years old
- >>> without_stars(args)
- Mr. Gumby is 42 years old

python™

Παραδείγματα

- def story(**kwds):
- return 'Once upon a time, there was a ' \
- '%(job)s called %(name)s.' % kwds

- >>> print story(job='king', name='Gumby')
- Once upon a time, there was a king called Gumby.
- >>> print story(name='Sir Robin', job='brave knight')
- Once upon a time, there was a brave knight called Sir Robin.

Παραδείγματα

- def story(**kwds):
- return 'Once upon a time, there was a ' \
- '%(job)s called %(name)s.' % kwds

- >>> params = {'job': 'language', 'name': 'Python'}
- >>> print story(**params)
- Once upon a time, there was a language called Python.
- >>> del params['job']
- >>> print story(job='stroke of genius', **params)
- Once upon a time, there was a stroke of genius called Python.

Παραδείγματα

```
def power(x, y, *others):
        if others:
                print ('Received redundant parameters:', others)
        return pow(x, y)
```

```
>>> power(2,3)
8
>>> power(3,2)
9
>>> power(y=3,x=2)
8
```

Παραδείγματα

```
def power(x, y, *others):
        if others:
                print ('Received redundant parameters:', others)
        return pow(x, y)
```

```
>>> params = (5,) * 2
>>> power(*params)
3125
>>> power(3, 3, 'Hello, world')
Received redundant parameters: ('Hello, world',)
27
```

Εμβέλεια μεταβλητών

- >>> def foo(): x = 42
- ...
- >>> x = 1
- >>> foo()
- >>> x
- 1

Εμβέλεια μεταβλητών

- >>> def output(x): print x
- ...
- >>> x = 1
- >>> y = 2
- >>> output(y)
- 2

Εμβέλεια μεταβλητών

- >>> def combine(parameter):

  print (parameter + external)

- ...
- >>> external = 'berry'
- >>> combine('Shrub')
- Shrubberry

Εμβέλεια μεταβλητών

- >>> def combine(parameter)

  print (parameter + globals()['parameter'])

- ...
- >>> parameter = 'berry'
- >>> combine('Shrub')
- Shrubberry

Εμβέλεια μεταβλητών

- >>> x = 1
- >>> def change_global():

  global x

  x = x + 1
- >>> change_global()
- >>> x
- 2

Ταξινόμηση

Ταξινόμηση

- Η Python έχει ενσωματωμένη μέθοδο για ταξινόμηση λιστών:
- >>> seq = [34, 67, 8, 123, 4, 100, 95]
- >>>seq.sort()
- >>>seq
- [4, 8, 34, 67, 95, 100, 123]

python™

Η συνάρτηση sorted

- >>>sorted([4, 3, 6, 8, 3])
- [3, 3, 4, 6, 8]

Ταξινόμηση με επιλογή (selection sort)

```
def SortIntegerArray(array):
    n=len(array)
    for lh in range(0,n):
        rh=FindSmallestInteger(array, lh, n-1)
        SwapIntegerElements(array, lh, rh)

def FindSmallestInteger(array, low, high):
    spos=low
    for i in range(low, high+1):
        if array[i]<array[spos]:
            spos=i
    return(spos)
```

Ταξινόμηση με επιλογή (selection sort)

- def SwapIntegerElements(array, p1, p2):
- array[p1], array[p2] = array[p2], array[p1]

bubblesort

```
def bubblesort(numbers):
    array_size=len(numbers)
    for i in range(array_size-1, -1, -1):
        for j in range(1,i+1):
            if numbers[j-1]>numbers[j]:
                numbers[j-1], numbers[j]=numbers[j], numbers[j-1]
```

Insertionsort

```
def InsertionSort(numbers):
    array_size=len(numbers)
    for i in range(1, array_size):
        index=numbers[i]
        j=i
        while (j>0) and numbers[j-1]>index:
            numbers[j]=numbers[j-1]
            j=j-1
        numbers[j]=index
```

Αναζήτηση στην Python

- >>>x=[232, 2, 21, 1, 2]
- >>>21 in x
- True
- >>>3 in x
- False

- Δεν μου λέει σε ποια θέση της λίστας βρίσκεται το στοιχείο που ψάχνω.

Η μέθοδος index

- >>>[2, 3, 4, 2, 1].index(2)
- 0
- >>>[2, 3, 4, 2, 1].index(5)
- Traceback (most recent call last):
- File "<pyshell#24>", line 1, in <module>
- [2, 3, 4, 2, 1].index(5)
- ValueError: 5 is not in list

Γραμμική Αναζήτηση (Linear Search)

- Ψάχνουμε την θέση μιας τιμής κλειδί
- Γραμμική Αναζήτηση (Linear search)
  - Απλούστερη δυνατή
  - Σύγκρινε σειριακά κάθε στοιχείο του πίνακα με την τιμή-κλειδί
  - Χρήσιμο για μικρούς και ΜΗ ταξινομημένους πίνακες

linearsearch

```
def linearsearch(a, key):
    array_size=len(a)
    for i in range(array_size):
        if key==a[i]:
            return i
    return -1
```

Δυαδική Αναζήτηση (Binary Search)

- Δυαδική Αναζήτηση
  - Σε ταξινομημένους πίνακες μόνο
  - Συγκρίνει το **middle** στοιχείο με το ζητούμενο **key**
    - Αν είναι ίσα βρέθηκε
    - Αν **key < middle**, ψάχνει στο 1ο μισό του πίνακα
    - Αν **key > middle**, ψάχνει στο 2ο μισό του πίνακα
    - Επανάληψη
  - Πολύ γρήγορη
    - Στη χειρότερη περίπτωση n βήματα, για

      2^n > αριθμό στοιχείων
- Πίνακας 30 στοιχείων χρειάζεται το πολύ 5 βήματα
  - $2^5 > 30$ δηλαδή 5 βήματα

python™

binarysearch

```
def binarysearch(a, key, low, high):
    while low<=high:
        middle=(low+high)//2
        if key==a[middle]:
            return middle
        elif key<middle:
            high=middle-1
        else:
            low=middle+1
    return -1
```