

Εισαγωγή στον Προγραμματισμό
Μάθημα 4:
Αλγόριθμοι και Γλώσσες Προγραμματισμού

Δεκέμβριος 2015

Χ. Αλεξανδράκη

Αλγόριθμος (τι είναι)

- Στα μαθηματικά και την επιστήμη ΗΥ
 - Ο αλγόριθμος είναι η λογική διαδικασία που πρέπει να ακολουθηθεί για την επίλυση ενός προβλήματος, όπου
 - Η διαδικασία αναλύεται σε ένα σύνολο από **βήματα**
 - Ο αριθμός το βημάτων είναι **πεπερασμένος!**

Παράδειγμα (αλγοριθμικών) προβλημάτων

- Δεδομένου ενός συνόλου από αριθμούς:
 - Βρες το μέγιστο
 - Βρες το ελάχιστο
 - Βρες το μέσο όρο
 - Βρες τον κοινό παρανομαστή
 - Ταξινόμησέ τους σε αύξουσα (ή φθίνουσα σειρά)

Σχεδιασμός αλγορίθμων

- Γενικά υπάρχουν δύο τρόποι για να ορίσουμε έναν αλγόριθμο:
 - Ψευδο-κώδικας (Pseudo-code)
 - Η καταγραφή των βημάτων σε ανθρώπινη γλώσσα
 - Διαγράμματα ροής (Flow charts)
 - Η αναπαράσταση των βημάτων με σύμβολα

Συστατικά Αλγορίθμου

1. Αρχή & Τέλος

ΑΡΧΗ

ΤΕΛΟΣ

2. Είσοδος/Εξοδος Δεδομένων

Διάβασε N

3. Εκτέλεση Πράξεων/ Εντολές

$n = n * i$

4. Έλεγχος Συνθήκης

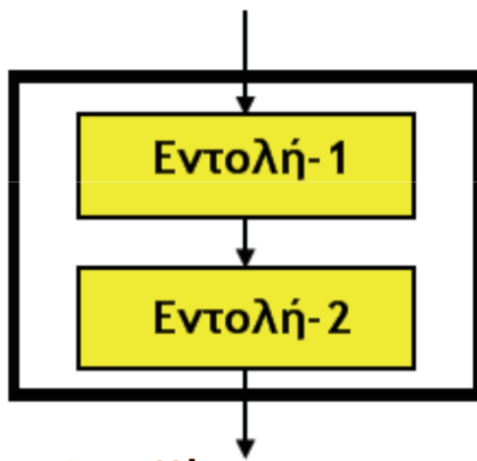
$i \leq N$

Δομές Ελέγχου (1/2)

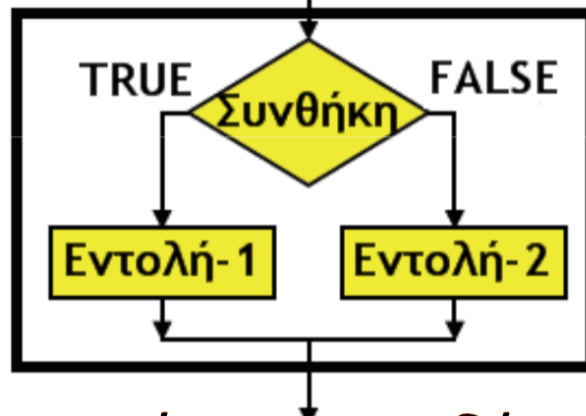
- Βάσει του **Θεωρήματος Δομημένου Προγραμματισμού** ή **Θεωρήματος Böhm-Jacopini**, ο αλγόριθμος οποιουδήποτε επιλύσιμου προβλήματος μπορεί να αναλυθεί σε συνδυασμό μικρότερων βημάτων με τρεις δομές ελέγχου (*control structures*):
 - **Ακολουθία (sequence)**
 - Εκτέλεση βήματος το ένα μετά το άλλο
 - **Απόφαση (selection)**
 - Εκτέλεση βήματος κατ' επιλογήν, δηλαδή εφόσον ισχύει κάποια συνθήκη (if-else)
 - **Επανάληψη (iteration)**
 - Εκτέλεση βήματος κατ' επανάληψη (for, while)

Δομές Ελέγχου (2/2)

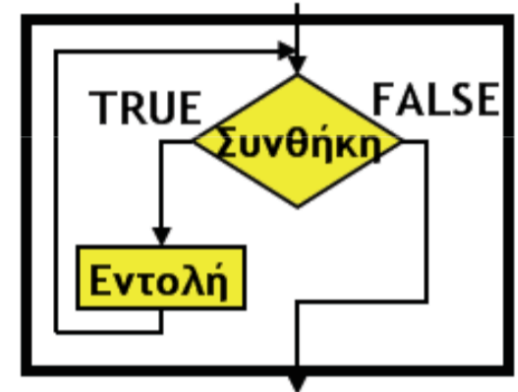
Ακολουθία



Απόφαση

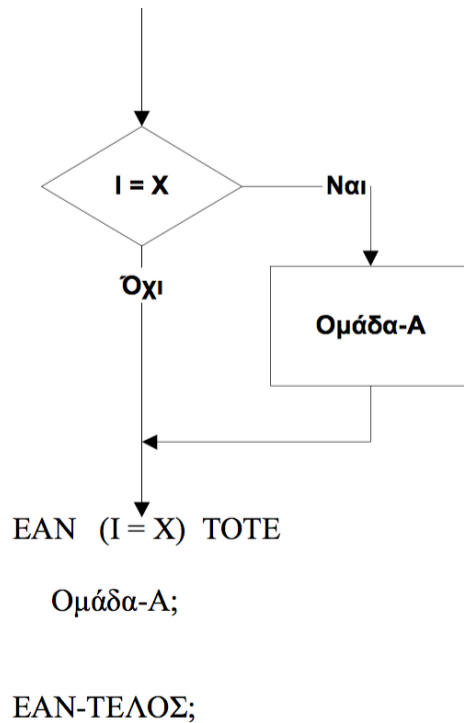


Επανάληψη

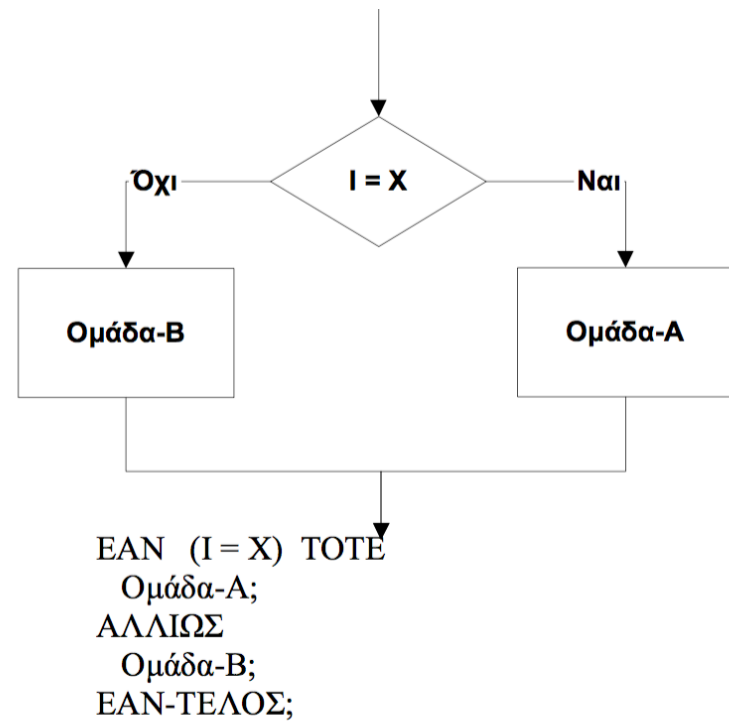


Απόφαση

ΔΟΜΗ IF



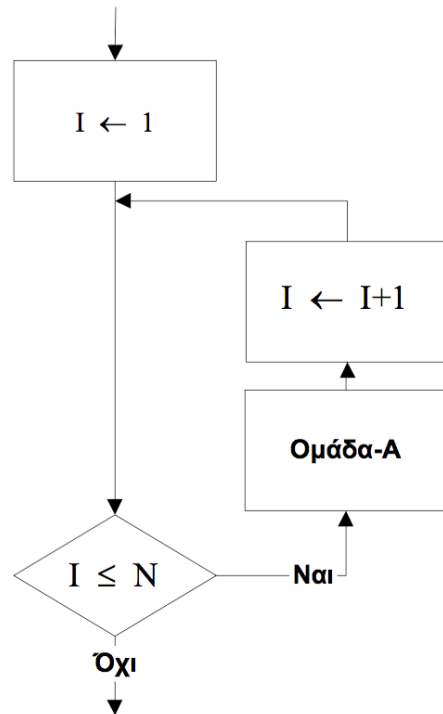
ΔΟΜΗ IF-ELSE



Επανάληψη

ΔΟΜΗ FOR

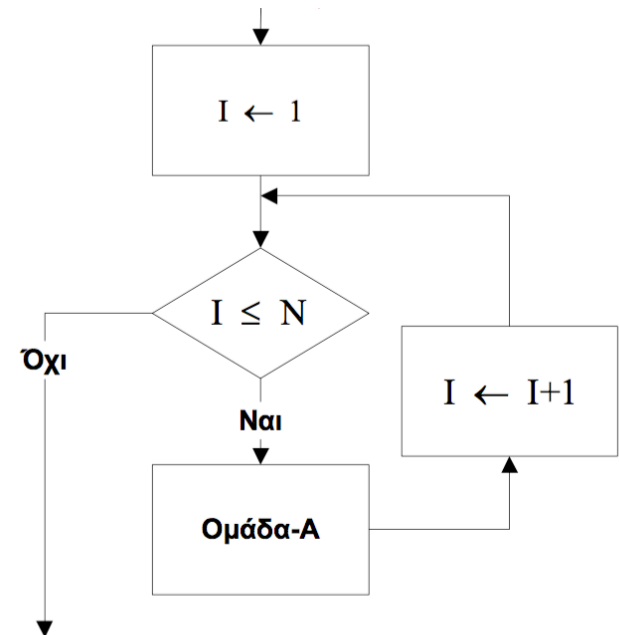
Χρησιμοποιείται όταν γνωρίζουμε τον αριθμό των επαναλήψεων



ΓΙΑ $I := 1$ ΕΩΣ N ΕΠΑΝΑΛΑΒΕ
Ομάδα-A;
ΓΙΑ-ΤΕΛΟΣ;

ΔΟΜΗ WHILE

Χρησιμοποιείται όταν ο αριθμός των επαναλήψεων εξαρτάται από μια συνθήκη



$I := 1$;
ΕΝΟΣΩ $(I \leq N)$ ΕΠΑΝΕΛΑΒΕ
Ομάδα-A;
 $I := I + 1$;
ΕΝΟΣΩ-ΤΕΛΟΣ;

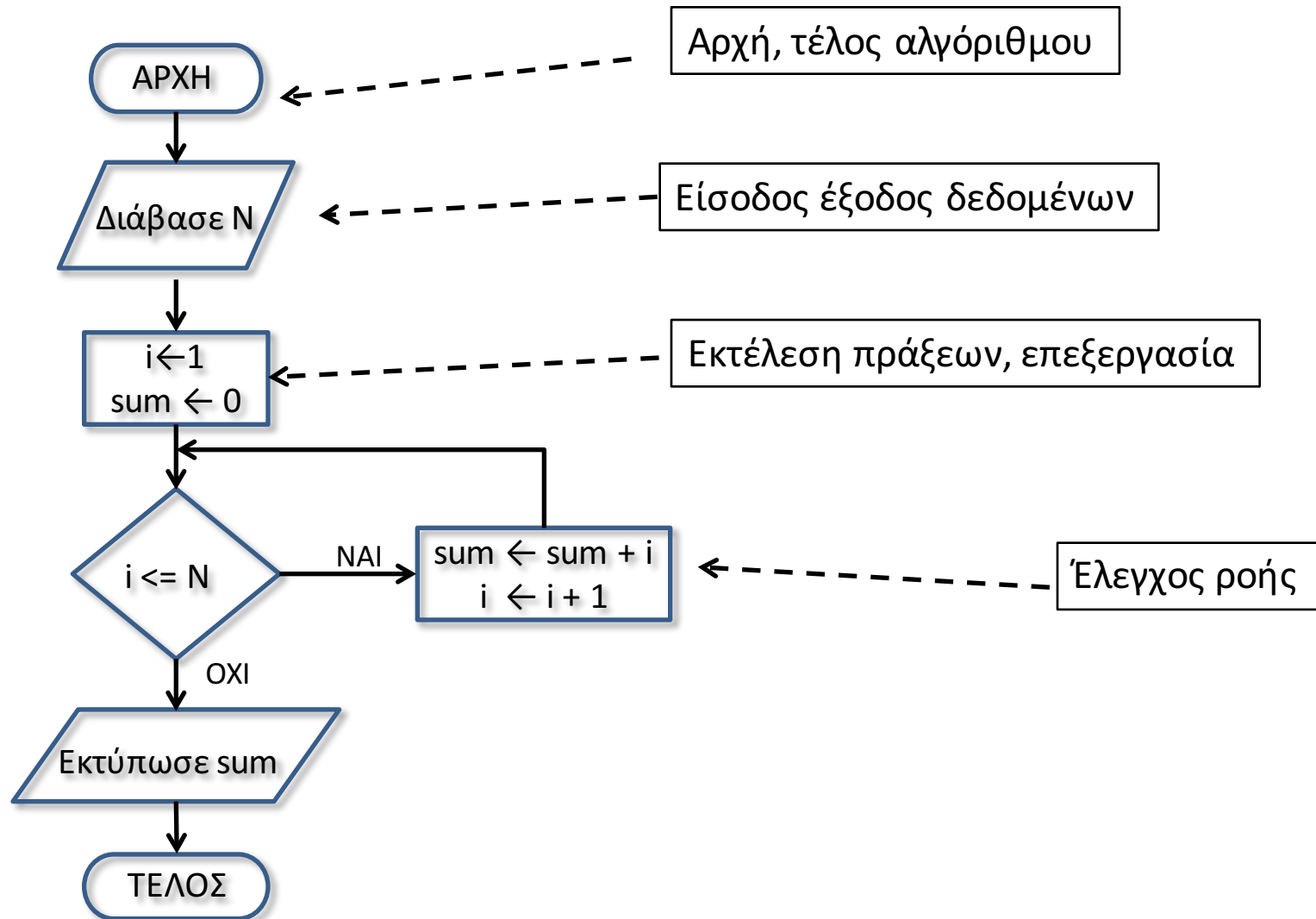
Παράδειγμα Αλγορίθμου

- Να βρεθεί και να εκτυπωθεί το άθροισμα των N ακεραίων από το 1 μέχρι το N , δεδομένου του N

Επίλυση με ψευδο-κώδικα

- Αλγόριθμος ΆθροισμαN
 - Διάβασε N
 - $Sum \leftarrow 0$
 - Για i από 1 έως N
 - $Sum \leftarrow Sum + i$
 - Τέλος Επανάληψης
 - Εκτύπωσε Sum
- Τέλος ΆθροισμαN

Επίλυση με διάγραμμα ροής



Πολυπλοκότητα Αλγορίθμων

- Η **πολυπλοκότητα** ενός αλγορίθμου ορίζεται ως ο χρόνος (επεξεργασία) και ο χώρος (μνήμη) που απαιτείται ως συνάρτηση του πλήθους των δεδομένων εισόδου (n)!
 - Πολυπλοκότητα $O(n^*n)$ σημαίνει ότι για διπλάσιο πλήθος δεδομένων εισόδου ο χρόνος εκτέλεσης του προγράμματος θα τετραπλασιαστεί...
 - Η μνήμη που απαιτείται υπολογίζεται σε bytes, ανάλογα με το πλήθος και το μέγεθος των μεταβλητών που χρειάζεται να χρησιμοποιήσει ο αλγόριθμος

Βελτιστοποίηση (Optimisation)

- Για τα περισσότερα προβλήματα υπάρχουν περισσότεροι από ένας τρόποι επίλυσης (αλγόριθμοι)
- Βελτιστοποίηση ονομάζεται η διαδικασία αναζήτησης του αλγορίθμου ο οποίος για δεδομένο πρόβλημα, απαιτεί:
 - Τον ελάχιστο χρόνο επεξεργασίας
 - Την ελάχιστη μνήμη δεδομένων

Υλοποίηση (Implementation)

- Η διαδικασία καταγραφής αλγορίθμου σε κάποια γλώσσα προγραμματισμού

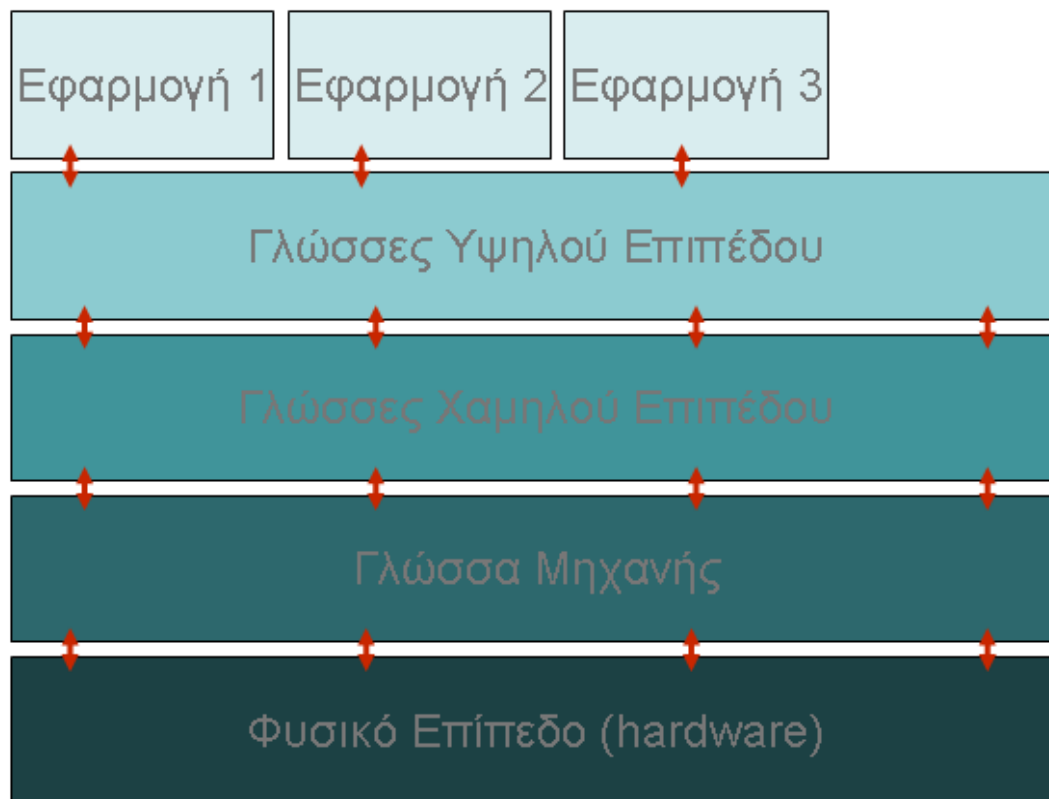
Γλώσσες Προγραμματισμού

- Τρόποι κατηγοριοποίησης γλωσσών προγραμματισμού:
 - Ως προς τη δυνατότητα **αφαιρετικότητας** (abstraction):
 - Χαμηλού Επιπέδου και Υψηλού Επιπέδου
 - Ως προς τον **τρόπο εκτέλεσης**:
 - Μεταγλωττιζόμενες και Διερμηνευόμενες
 - Ως προς το ακολουθούμενο **παράδειγμα**:
 - Procedural, Functional, Object Oriented, Multi-paradigm

Γλώσσα Μηχανής

- Περιλαμβάνει εντολές γραμμένες σε μορφή ακολουθιών bit άμεσα εκτελέσιμες από την Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ).
- Ονομάζεται γλώσσα μηχανής επειδή μέσω αυτής, και καμίας άλλης, επιτυγχάνεται «επικοινωνία» με τον υπολογιστή.
- Προγράμματα που γράφονται σε άλλες γλώσσες προγραμματισμού, για να γίνουν εκτελέσιμα πρέπει να «μεταφραστούν» σε γλώσσα μηχανής.
- Τα πρώτα προγράμματα γράφονταν σε δυαδικό κώδικα κι επομένως απαιτούσαν από τον προγραμματιστή να έχει πολύ καλή γνώση της αρχιτεκτονικής του ΗΥ
- Η πρώτη εξέλιξη από το δυαδικό κώδικα ήταν ο προγραμματισμός με δεκαεξαδικό κώδικα
 - Κάθε σύμβολο δεκαεξαδικού αντιστοιχεί σε μια τετράδα δυαδικού κώδικα, κι επομένως ένα byte αναπαρίσταται με δύο δεκαεξαδικά σύμβολα.
 - Π.χ. 1001 1101 \Leftrightarrow 9D

Γλώσσες Υψηλού και Χαμηλού Επιπέδου



Γλώσσες Χαμηλού επιπέδου

- Είναι εκείνες στις οποίες δεν υπάρχει κανένα επίπεδο αφαιρετικότητας, δηλαδή δεν υπάρχει η έννοια της σημασιολογίας:
 - Υπάρχει **ένα-προς-ένα** αντιστοιχία ανάμεσα:
 - Στις εντολές της γλώσσας και στην αντίστοιχη εντολή σε γλώσσα μηχανής
- Παράδειγμα:
 - Δεκαεξαδικός κώδικας
 - Assembly

```
fib:
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

@@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

@@:
    push ebx
    mov ebx, 1
    mov ecx, 1

@@:
    lea eax, [ebx+ecx]
    cmp edx, 3
    jbe @f
    mov ebx, ecx
    mov ecx, eax
    dec edx
    jmp @b

@@:
    pop ebx
    ret
```

Γλώσσες Υψηλού Επιπέδου

- Παρέχουν υψηλό βαθμό αφαίρεσης από την αρχιτεκτονική του ΗΥ
- Αποτελούνται από εντολές εύκολα κατανοητές στον προγραμματιστή, καθώς μοιάζουν με *-περιορισμένη- φυσική γλώσσα*.
- Για την εκτέλεση του προγράμματος απαιτείται η μετατροπή του κώδικα σε γλώσσα μηχανής

Μεταγλωττιζόμενες, Διερμηνευόμενες και Υβριδικές Γλώσσες

- Μεταγλωττιζόμενες είναι οι γλώσσες οι οποίες μέσω ενός ειδικού προγράμματος (το μεταφραστή/ compiler) μετατρέπουν τον πηγαίο κώδικα σε γλώσσα μηχανής, ο οποίος και αποτελεί το εκτελέσιμο του προγράμματος
 - Πλεονέκτημα: **ταχύτητα**
- Διερμηνευόμενες είναι αυτές στις οποίες το πρόγραμμα εκτελείται βήμα-βήμα. Ένα ειδικό πρόγραμμα (ο διερμηνέας/interpreter) μετατρέπει σταδιακά την κάθε εντολή σε γλώσσα μηχανής και εν συνεχεία την εκτελεί
 - Πλεονέκτημα: **φορητότητα**
- Υβριδικές είναι οι γλώσσες στις οποίες ο compiler δεν μεταφράζει απευθείας σε γλώσσα μηχανής αλλά σε μια ενδιάμεση μορφή που λέγεται bytecode. Ο bytecode εκτελείται από έναν interpreter κατά την εκτέλεση ενός προγράμματος
 - Πλεονέκτημα: **ταχύτητα** και **φορητότητα**

Διαδικαστικός ή δομημένος Προγραμματισμός

- **Διαδικαστικός (procedural) ή δομημένος προγραμματισμός (structured programming)** είναι μία προσέγγιση στον προγραμματισμό, η οποία βασίζεται στην έννοια της **κλήσης διαδικασίας**. Η διαδικασία, γνωστή επίσης και ως *ρουτίνα*, *υπορουτίνα*, *μέθοδος* ή *συνάρτηση*, είναι απλά ένα αυτοτελές σύνολο εντολών προς εκτέλεση.
- Ο δομημένος προγραμματισμός βασίζεται στην αρχή του *διαίρει και βασίλευε*, καθώς διασπά το βασικό πρόβλημα σε μικρότερα υποπροβλήματα. Κάθε εργασία με πολύπλοκη περιγραφή διαιρείται σε μικρότερες, έως ότου οι εργασίες να είναι αρκετά μικρές, περιεκτικές και εύκολες προς κατανόηση
- Κάθε διαδικασία μπορεί να καλεί άλλες διαδικασίες.
- Παράδειγμα γλωσσών δομημένου προγραμματισμού:
 - C, Fortran, Pascal, Basic

Παράδειγμα δομημένου προγραμματισμού

```
#include <stdio.h>
```

```
int sum(int k, int l) {  
    return k+l;  
}
```

```
int main() {  
    int x,y;  
    puts("Δύο ακέραιοι με κενό ενδιάμεσα:");  
    scanf("%d %d",&x,&y);  
    printf("Sum is %d\n", sum(x,y));  
    return 0;  
}
```

Παράδειγμα: Μέγιστος Κοινός Διαιρέτης

```
int mkd(int x, int y) {
    int mkd = 0;
    int z = 0;
    while (mkd == 0 ) {
        if (z==0) {
            mkd = y;
            return mkd;
        }
        x = y;
        y = z;
    }
}

int main() {
    int x,y;
    puts("Dose akeraious xorismenous me keno:");
    scanf("%d %d",&x,&y);
    printf("MKD is %d\n", sum(x,y));
    return 0;
}
```


Αντικειμενοστραφής Προγραμματισμός

- Εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990,
 - αντικαθιστώντας σε μεγάλο βαθμό τον παραδοσιακό δομημένο προγραμματισμό
- ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά.
- Αυτή η δομή δεδομένων καλείται **αντικείμενο** και αποτελεί πραγματικό στιγμιότυπο στη μνήμη του ΗΥ

Παράδειγμα Αντικειμενοστραφούς Προγραμματισμού

```
class Point {  
    int x;  
    int y;  
  
    void moveHorizontal(int x0) {  
        x = x + x0;  
    }  
  
    public void main(String[] args) {  
  
        Point p1 = new Point(2, 5);  
        p1.moveHorizontal(8);  
    }  
  
}
```

Python

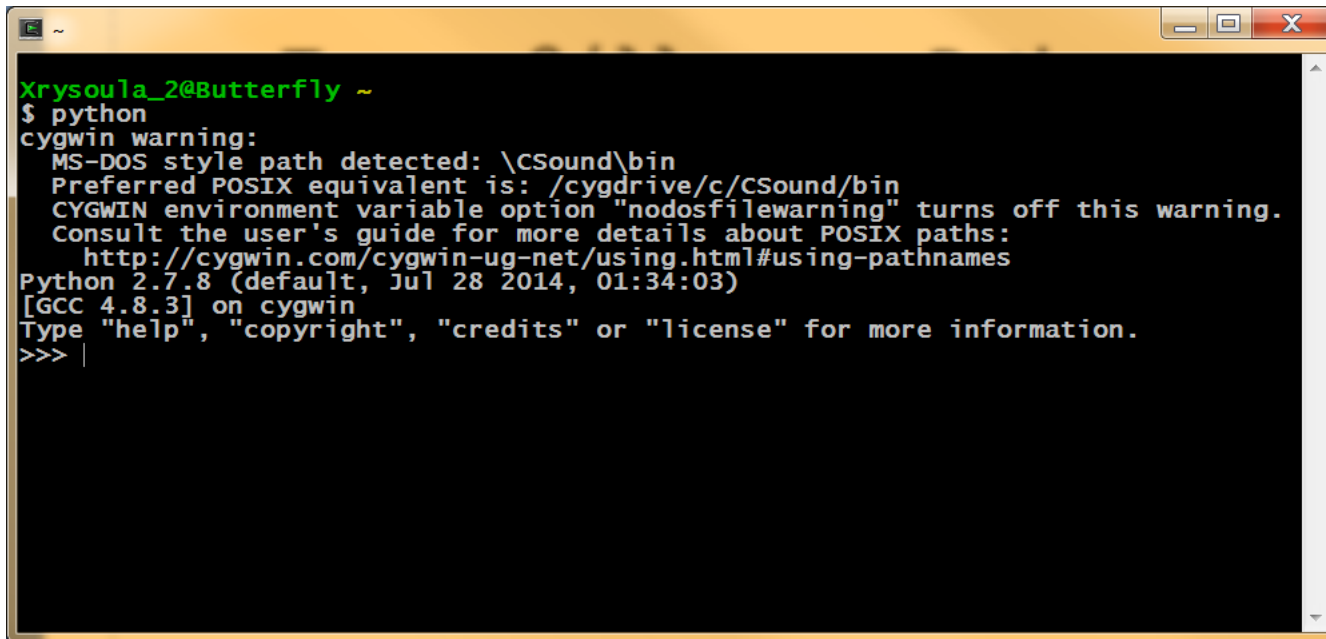
- Είναι ίσως η απλούστερη γλώσσα για να ξεκινήσει κάποιος προγραμματισμό
- Ξεκίνησε να αναπτύσσεται στις αρχές τις δεκαετίας του 1990 από τον μαθηματικό **Guido van Rossum**
- Χαρακτηριστικά
 - Γενικής χρήσης (general- purpose)
 - Υψηλού επιπέδου
 - Είναι ως επί το πλείστον διερμηνευόμενη αλλά μπορεί να εκτελεστεί και ως υβριδική για ταχύτερη εκτέλεση
 - Υφίσταται με διάφορες μορφές, δηλαδή υλοποιεί πολλαπλά προγραμματιστικά παραδείγματα:
 - Procedural, Functional, Object Oriented

Πλεονεκτήματα Python

- Μπορεί κανείς να την χρησιμοποιήσει διαδραστικά (interactively) με τον ίδιο τρόπο που χρησιμοποιεί μια αριθμομηχανή. Κι αυτός είναι ο πιο εύκολος τρόπος να αρχίσει κανείς να μαθαίνει προγραμματισμό
- Αν και είναι γλώσσα υψηλού επιπέδου, διαθέτει πολύ απλή σύνταξη. Έτσι επιτρέπει στον προγραμματιστή να ασχοληθεί με την επίλυση του προβλήματος κι όχι με τις ιδιαιτερότητές της.
- Μπορεί να συνδυαστεί με άλλες δημοφιλείς γλώσσες όπως C, C++ και Java
- Αποτελεί **ελεύθερο και ανοιχτού κώδικα** λογισμικό και διατίθεται δωρεάν από το επίσημο site <http://www.python.org>
- Υπάρχουν εκδόσεις για κάθε λειτουργικό σύστημα (ακόμα και για κινητά τηλέφωνα).

Το περιβάλλον της Python

- Γράφω τη λέξη python στο terminal, οπότε το linux prompt (\$), αλλάζει στο python prompt (>>>)
- Εδώ πλέον εισάγω εντολές της python και όχι του linux cli



```
Xrysoula_2@Butterfly ~  
$ python  
cygwin warning:  
MS-DOS style path detected: \CSound\bin  
Preferred POSIX equivalent is: /cygdrive/c/CSound/bin  
CYGWIN environment variable option "nodosfilewarning" turns off this warning.  
Consult the user's guide for more details about POSIX paths:  
  http://cygwin.com/cygwin-ug-net/using.html#using-pathnames  
Python 2.7.8 (default, Jul 28 2014, 01:34:03)  
[GCC 4.8.3] on cygwin  
Type "help", "copyright", "credits" or "license" for more information.  
>>> |
```

Χρήση της Python ως Αριθμομηχανή

```
>>> print 5+3
```

```
8
```

```
>>> print (7-2)*4
```

```
20
```

- Η γλώσσα επιτρέπει τη χρήση παρενθέσεων. Όταν σε μια αριθμητική πράξη υπάρχουν πολλαπλές παρενθέσεις ενσωματωμένες η μία στην άλλη, πρώτα εκτελείται η εσωτερικότερη, όπως δηλαδή θα γινόταν και αν κάναμε στις πράξεις στο χαρτί, π.χ.

```
>>> (12+(12/3))/2
```

```
8
```

```
>>>
```

- Επίσης μπορούμε να τυπώνουμε πολλά αποτελέσματα με την ίδια εντολή, αρκεί να χωρίζουμε με κόμματα:

```
>>> print 7+2, 7-2, 7*2, 7/2
```

```
9 5 14 3
```

- Μια μικρή προσοχή χρειάζεται στην πράξη της διαίρεσης. Η Python εκτελεί ακέραια διαίρεση, εκτός αν δηλώσουμε ότι θέλουμε να κάνει διαίρεση πραγματικών αριθμών. Ο πιο απλός τρόπος να το δηλώσουμε αυτό είναι να γράψουμε τον διαιρετέο ή τον διαιρέτη (ή και τους δυο) ως πραγματικούς. Πχ.:

```
>>> print 7/2, 7/2.0, 7.0/2, 7.0/2.0, 7./2, 7/2., 7./2.
```

```
3 3.5 3.5 3.5 3.5 3.5 3.5
```

- Για να υψώσω αριθμό σε δύναμη, μπορώ να χρησιμοποιώ το σύμβολο **, π.χ.

```
>>> 2**10
```

```
1024
```

```
>>>
```

Μεταβλητές (Variables)

- Στην αριθμομηχανή μπορώ να αποθηκεύσω έναν αριθμό με το πλήκτρο MR (memory register) για μετέπειτα χρήση.
- Στον προγραμματισμό, μπορώ να αποθηκεύω όσα δεδομένα θέλω με τη χρήση μεταβλητών, π.χ.

```
>>> a = 5
>>> b = a**2
>>> print a, b
5 25
>>>
```

Εκχώρηση τιμής σε μεταβλητή

- Εκχώρηση (Assignment)
 - Είναι η διαδικασία κατά την οποία δίνω τιμή σε μια μεταβλητή
 - Έχει πολύ συγκεκριμένη σύνταξη:
 - `a = 3`
 - Και όχι το ανάποδο
- Η τιμή μιας μεταβλητής μπορεί να αλλάζει μέσω της εκχώρησης, όσες φορές θέλω κατά τη διάρκεια εκτέλεσης του προγράμματος, π.χ.

```
>>> a = 5
>>> b = a**2
>>> print a, b
5 25
>>> a = b - 2.3
>>> a
22.7
>>>
```