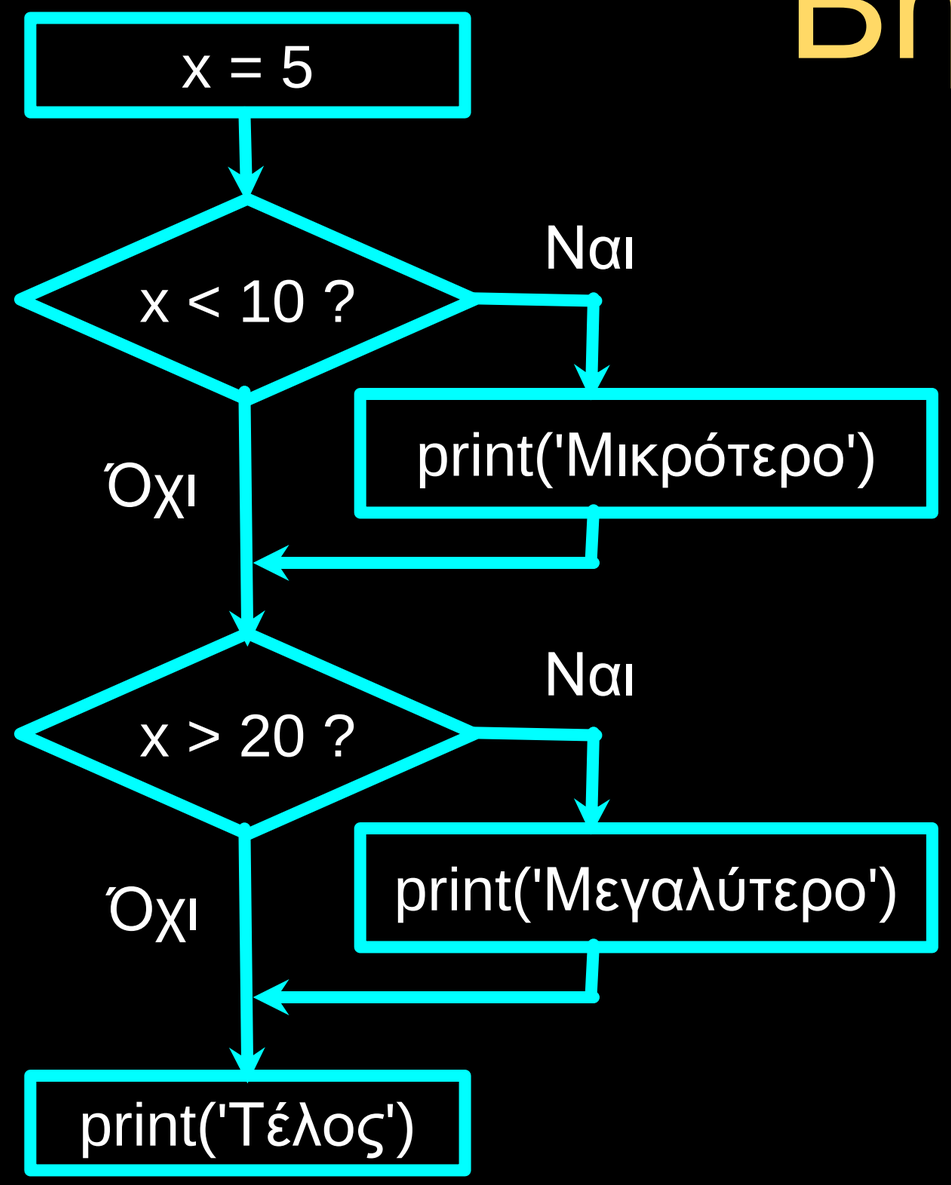
# Υπό όρους Εκτέλεση

3

Έλεγχος ροής προγράμματος

# Βήματα Υπό Όρους – Δομή Επιλογής




Πρόγραμμα:

```
x = 5
if x < 10:
    print('Μικρότερο')
if x > 20:
    print('Μεγαλύτερο')
print('Τέλος')
```

Έξοδος:

Μικρότερο
Τέλος

# Τελεστές σύγκρισης

- Λογικές εκφράσεις θέτουν μια ερώτηση και παράγουν ως αποτέλεσμα ένα Ναι ή Όχι, το οποίο χρησιμοποιούμε για να ελέγξουμε τη ροή του προγράμματος
- Οι Λογικές εκφράσεις με χρήση συγκριτικών τελεστών έχουν ως αποτέλεσμα Αληθής / Ψευδής ή Ναι / Όχι
- Οι συγκριτικοί τελεστές κοιτούν τις μεταβλητές αλλά δεν αλλάζουν τις μεταβλητές

| Python | Σημασία |
|---|---|
| < | Μικρότερο από |
| <= | Μικρότερο από ή ίσο με |
| == | Ίσο με |
| >= | Μεγαλύτερο από ή ίσο με |
| > | Μεγαλύτερο από |
| != | Όχι ίσο / Διάφορο |

Να θυμάστε:
το “=” χρησιμοποιείται για εκχώρηση.

# Τελεστές σύγκρισης

```
x = 5
if x == 5 :
    print('Ισούται με 5')
if x > 4 :
   print('Μεγαλύτερο από 4')
if  x >= 5 :
    print('Μεγαλύτερο ή ίσο από 5')
if x < 6 : print('Μικρότερο από 6')
if x <= 5 :
    print('Μικρότερο ή ίσο από 5')
if x != 6 :
    print('Διάφορο του 6')
```

Ισούται με 5

Μεγαλύτερο από 4

Μεγαλύτερο ή ίσο από 5

Μικρότερο από 6

Μικρότερο ή ίσο από 5

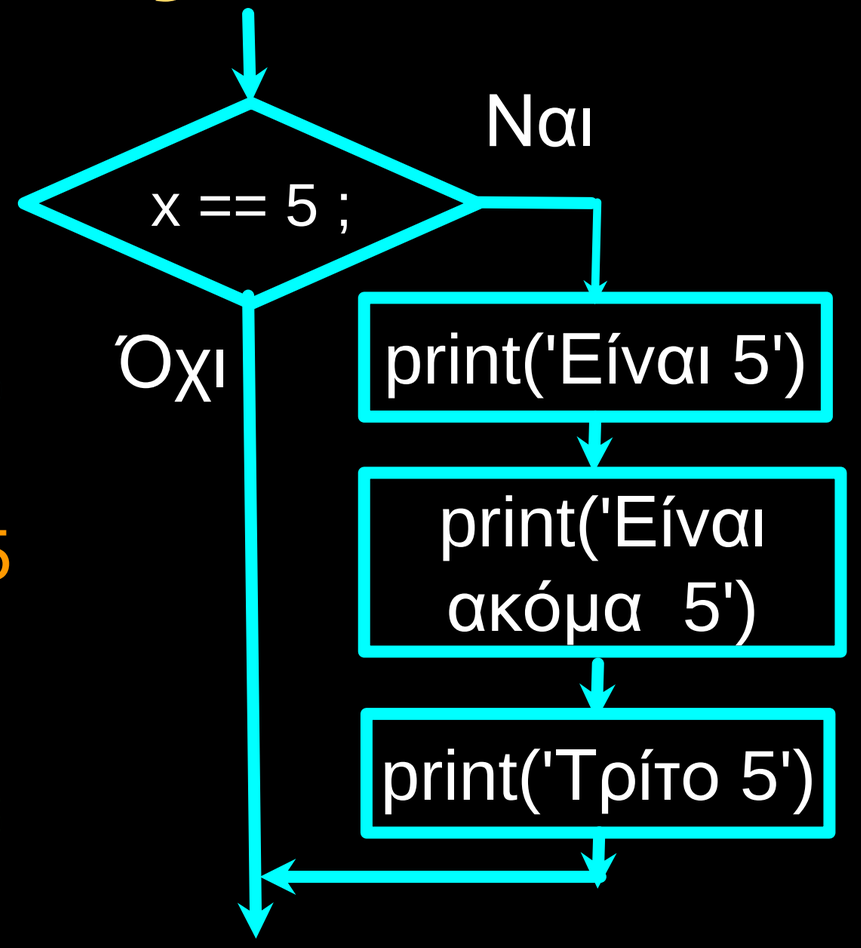
Διάφορο του 6

# Μονόδρομες Αποφάσεις

```
x = 5
print('Αρχικά 5')
if  x == 5 :
    print('Είναι 5')
    print('Είναι ακόμα 5')
    print('Τρίτο 5')
print('Κατόπιν του 5')
print('Πριν το 6')
if x == 6 :
    print('Είναι 6')
    print('Είναι ακόμα 6')
    print('Τρίτο 6')
print('Κατόπιν του 6')
```

Αρχικά 5

Είναι 5
Είναι ακόμα 5
Τρίτο 5
Κατόπιν του 5
Πριν το 6
Είναι 6
Είναι ακόμα 6
Τρίτο 6
Κατόπιν του 6



# Εσοχή

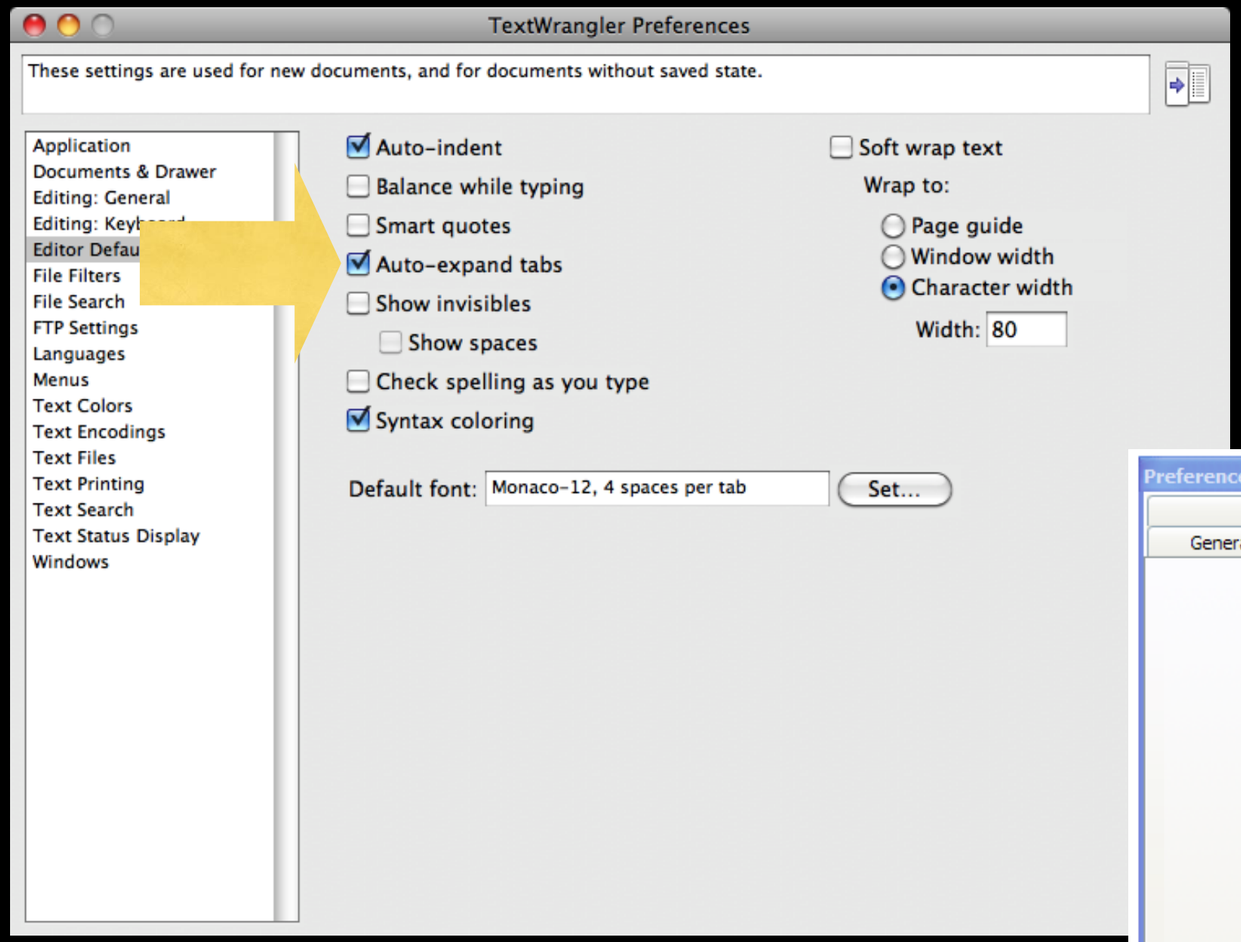
- Αύξηση εσοχής μετά από μια εντολή if ή for (μετά το :)
- Διατήρηση εσοχής για να υποδείξουμε το πεδίο εφαρμογής του μπλοκ (ποιες γραμμές επηρεάζονται από το if/for)
- Μείωση εσοχής πίσω στο επίπεδο του if ή του for για να υποδηλώσουμε το τέλος του μπλοκ
- Κενές γραμμές αγνοούνται – δεν επηρεάζουν τις εσοχές
- Σχόλια σε μια γραμμή, ούτω ή άλλως αγνοούνται σε σχέση με την εσοχή

# Προσοχή: Απενεργοποιήστε τα Tab!!

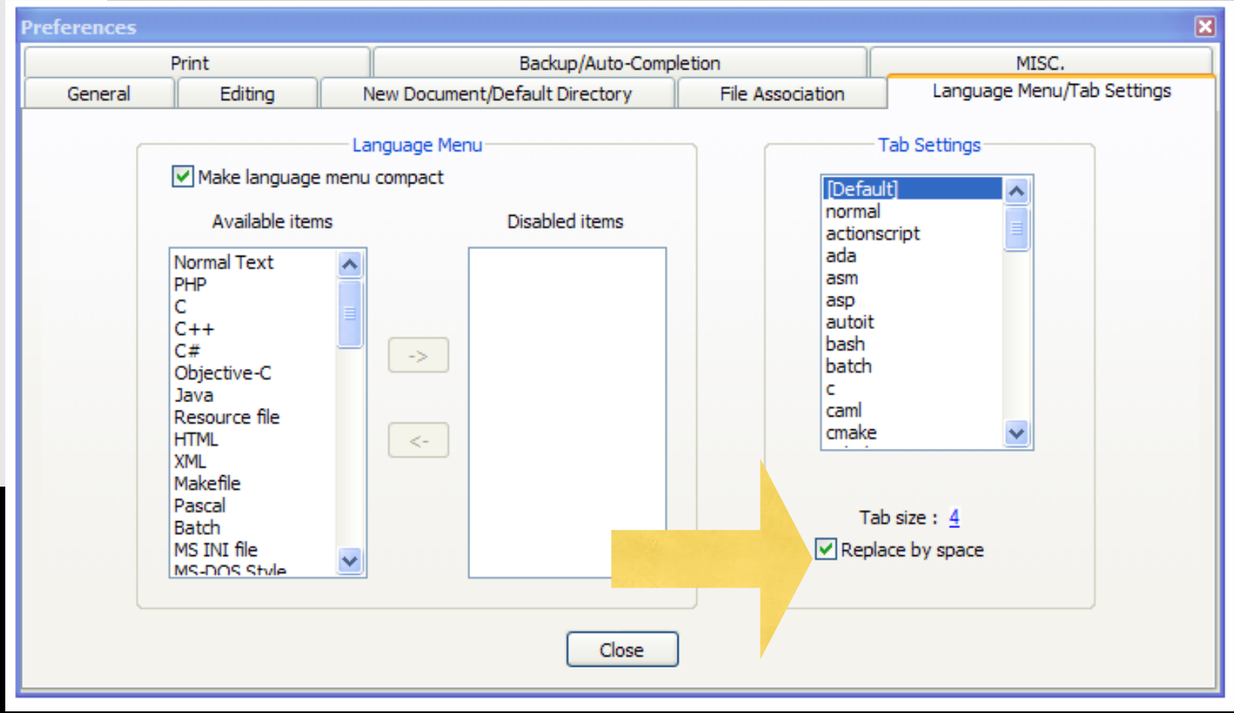
- Οι περισσότεροι επεξεργαστές κειμένου μπορεί να μετατρέψουν τα tab σε κενά - βεβαιωθείτε ότι έχετε ενεργοποιήσει αυτήν τη λειτουργία

  - NotePad++: Settings -> Preferences -> Language Menu/Tab Settings

- Η Python νοιάζεται *πολύ* για το μέγεθος εσοχής μια γραμμή. Αν αναμίξετε tabs και κενά, ενδέχεται να προκύψουν "σφάλματα εσοχής" ακόμη και αν όλα φαίνονται σωστά




Αυτό θα σας γλυτώσει από πολύ περιττό κόπο.



αύξηση / διατήρηση μετά από if ή for

μείωση για να υποδηλώσει το τέλος του μπλοκ

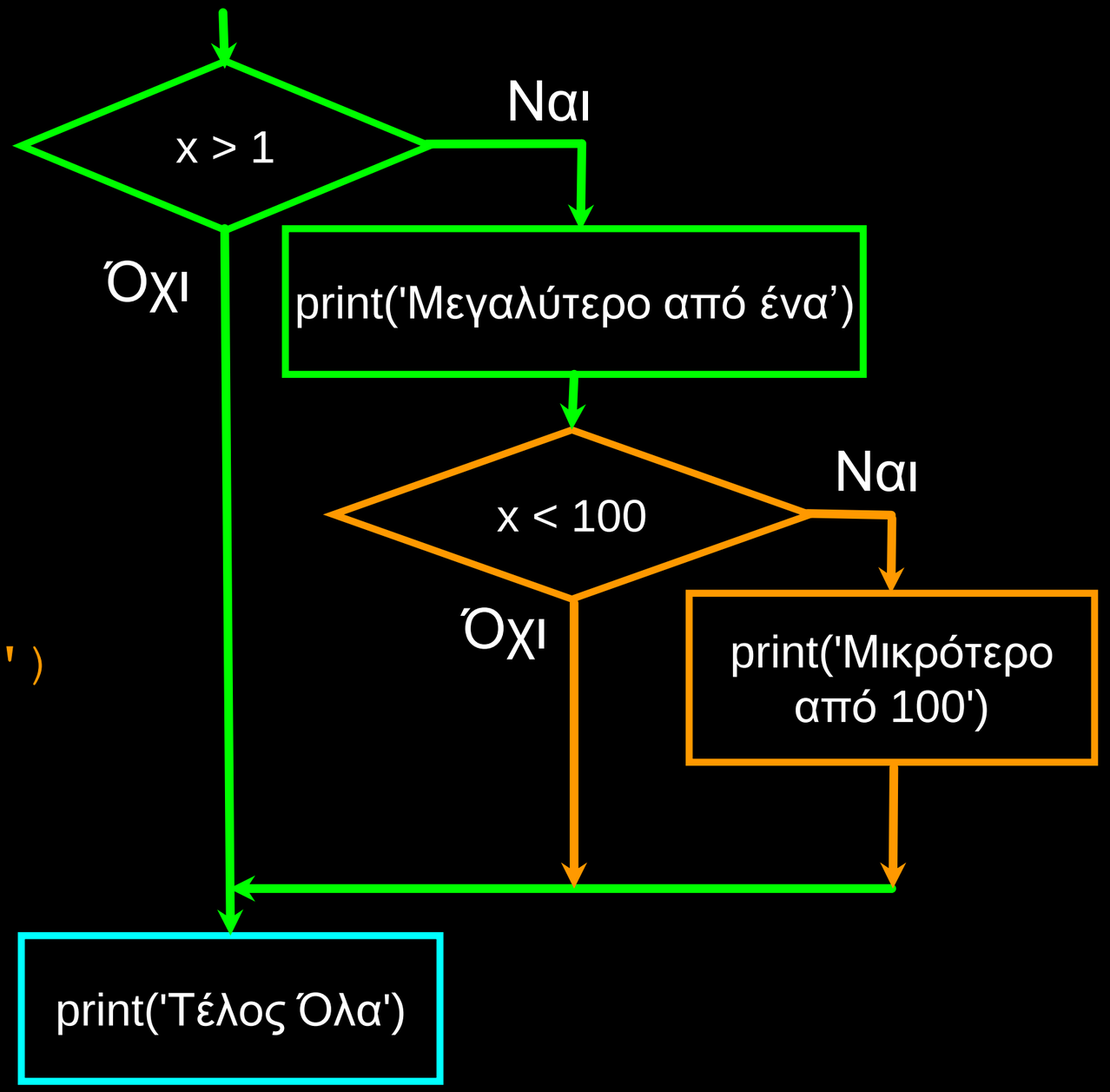
```
x = 5
if x > 2 :
    print('Μεγαλύτερο από 2')
    print('Μεγαλύτερο ακόμη')
print('Τέλος με το 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Μεγαλύτερο από 2')
    print('Τέλος με το i', i)
print('Τέλος Όλα')
```

# Εξετάστε την αρχή/τέλος των Μπλοκ

```
x = 5
if x > 2 :
    print('Μεγαλύτερο από 2')
    print('Μεγαλύτερο ακόμη')
print('Τέλος με το 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Μεγαλύτερο από 2')
    print('Τέλος με το i', i)
print('Τέλος Όλα')
```

# Εμφωλευμένη Επιλογή

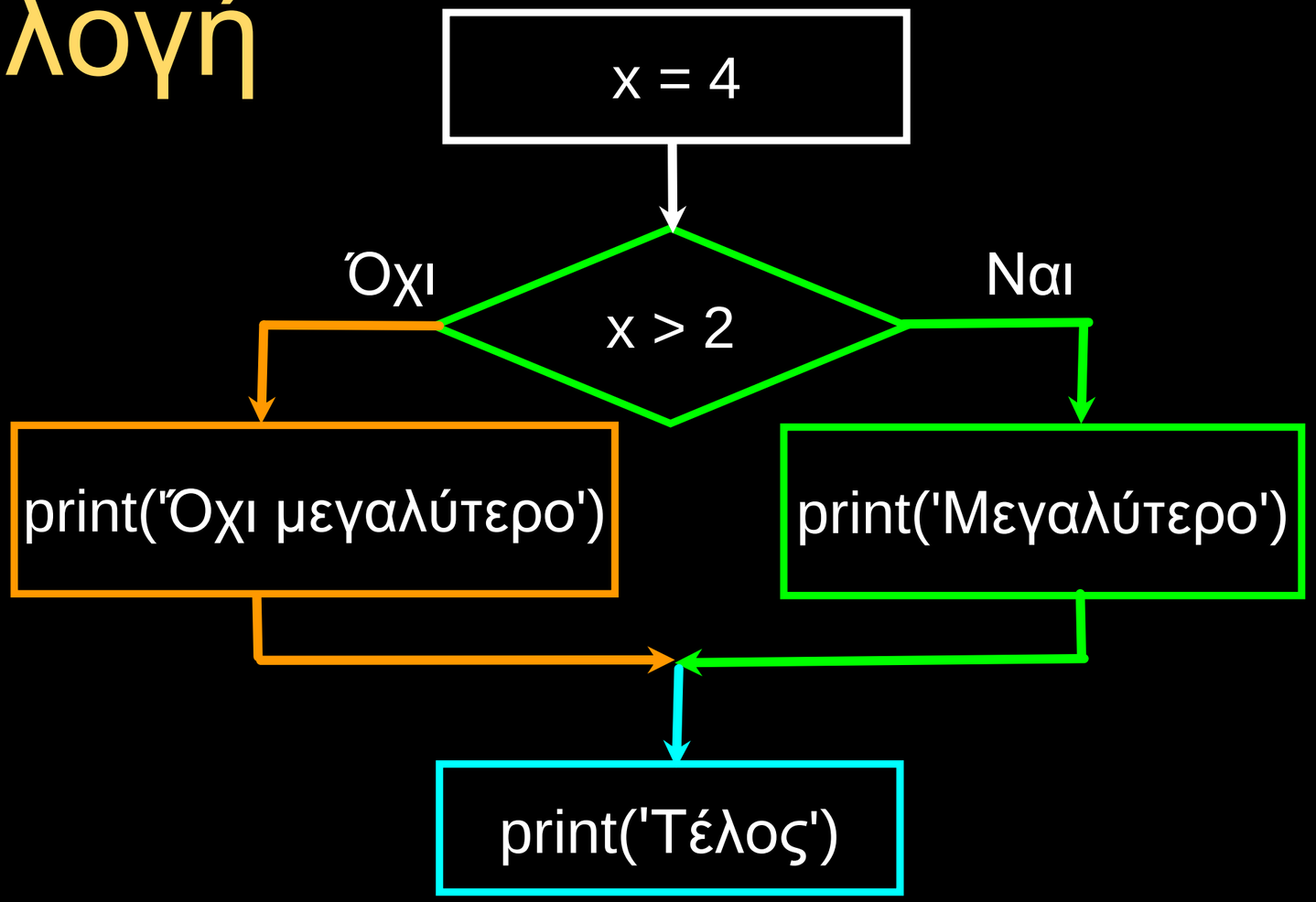
```
x = 42
if x > 1 :
    print('Μεγαλύτερο από ένα')
    if x < 100 :
        print('Μικρότερο από 100')
print('Τέλος Όλα')
```



# Αμφίδρομες αποφάσεις - Σύνθετη Επιλογή

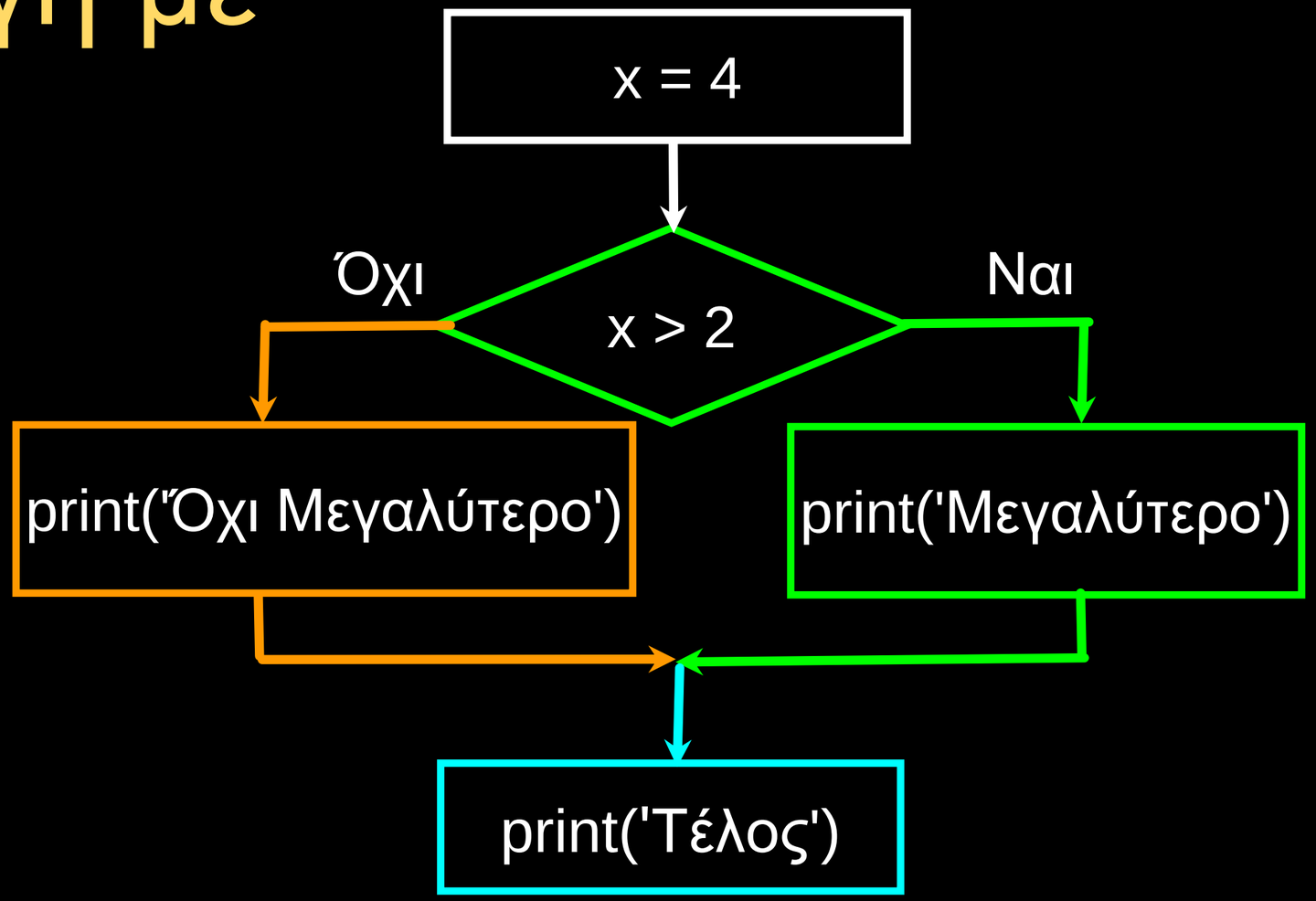
- Κάποιες φορές θέλουμε να κάνουμε κάτι εάν μια λογική έκφραση είναι αληθής και κάτι άλλο εάν η έκφραση είναι ψευδής

- Είναι σαν μια διακλάδωση στο δρόμο - πρέπει να επιλέξουμε το ένα ή το άλλο μονοπάτι αλλά όχι και τα δύο



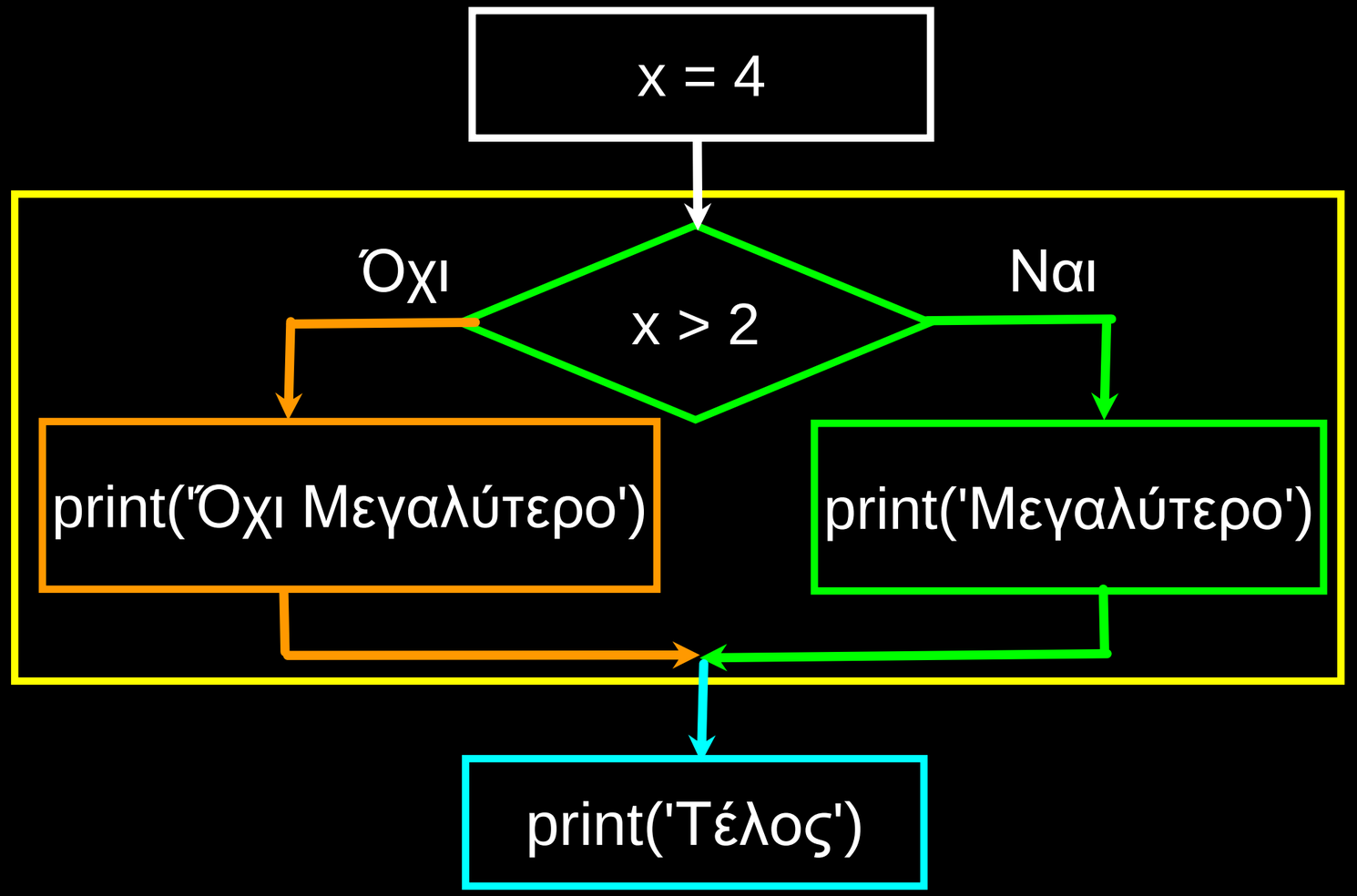
# Σύνθετη Επιλογή με else:

```
x = 4

if x > 2 :
    print('Μεγαλύτερο')
else :
    print('Μικρότερο')

print('Τέλος')
```



# Οπτικοποίηση των Μπλοκ

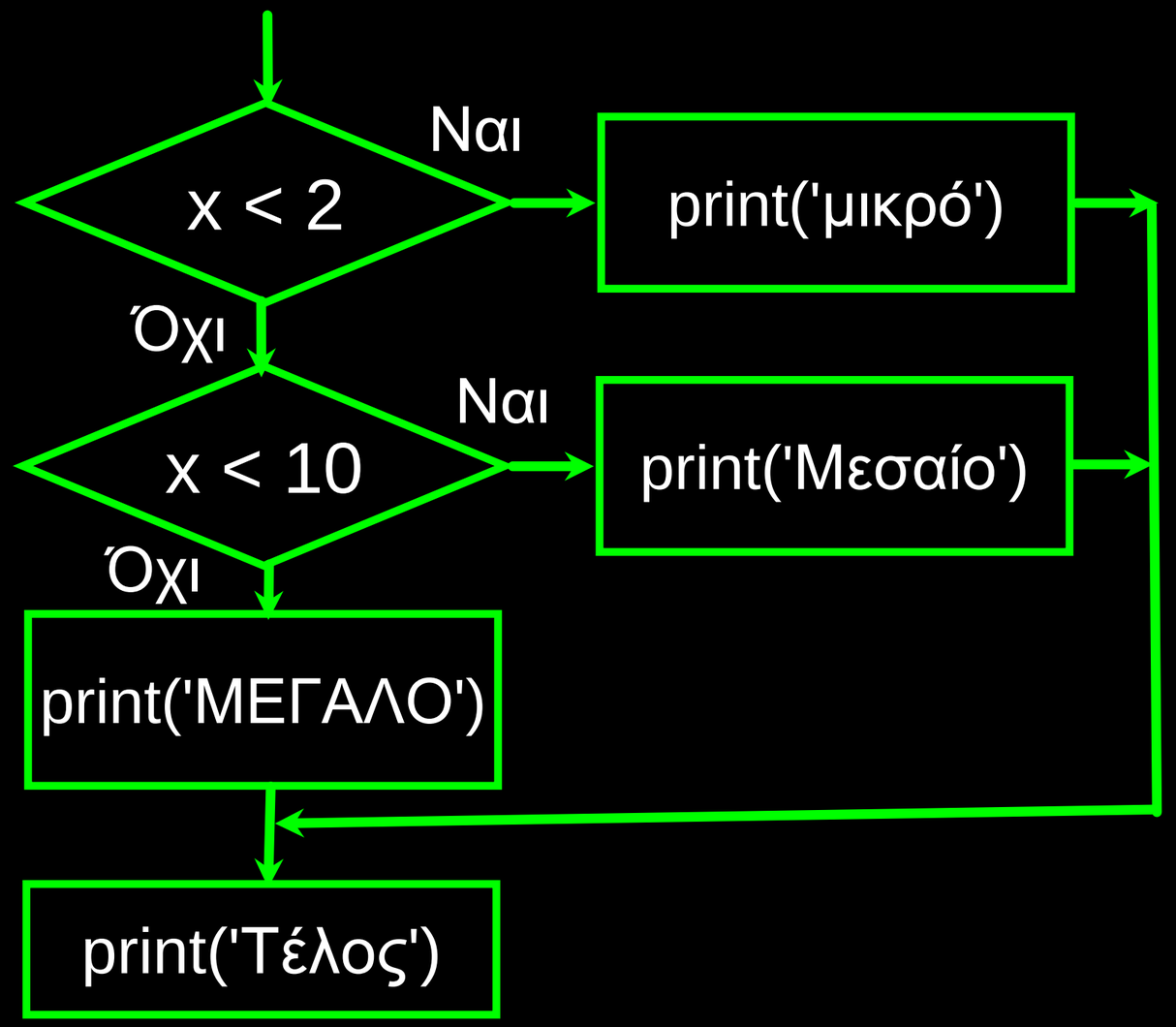
```
x = 4

if x > 2 :
    print('Μεγαλύτερο')
else :
    print('Μικρότερο')

print('Τέλος')
```



# Περισσότερες Δομές Επιλογής...

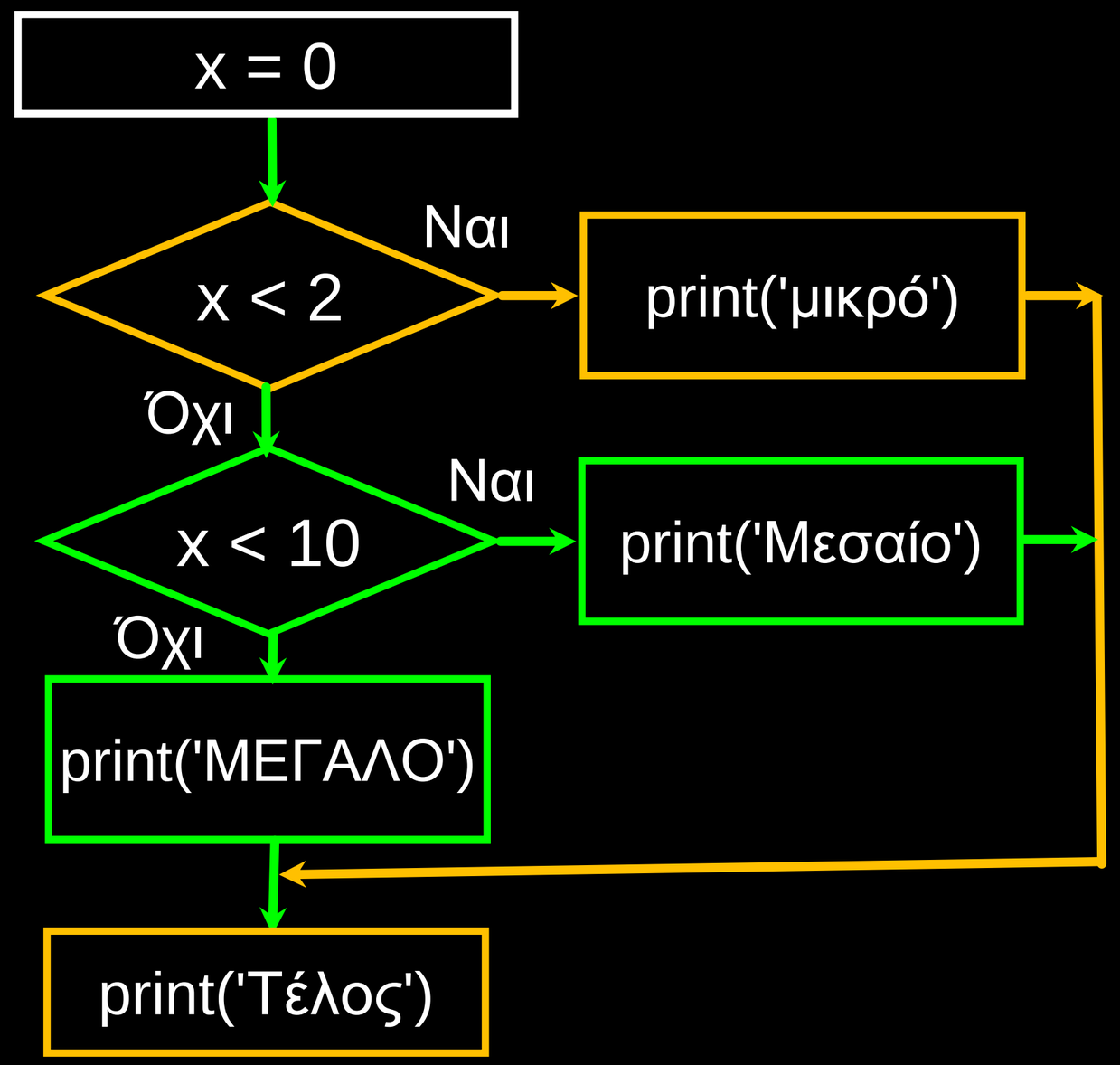
# Πολλαπλών Επιλογών

```
if x < 2 :
    print('μικρό')
elif x < 10 :
    print('Μεσαίο')
else :
    print('ΜΕΓΑΛΟ')
print('Τέλος')
```



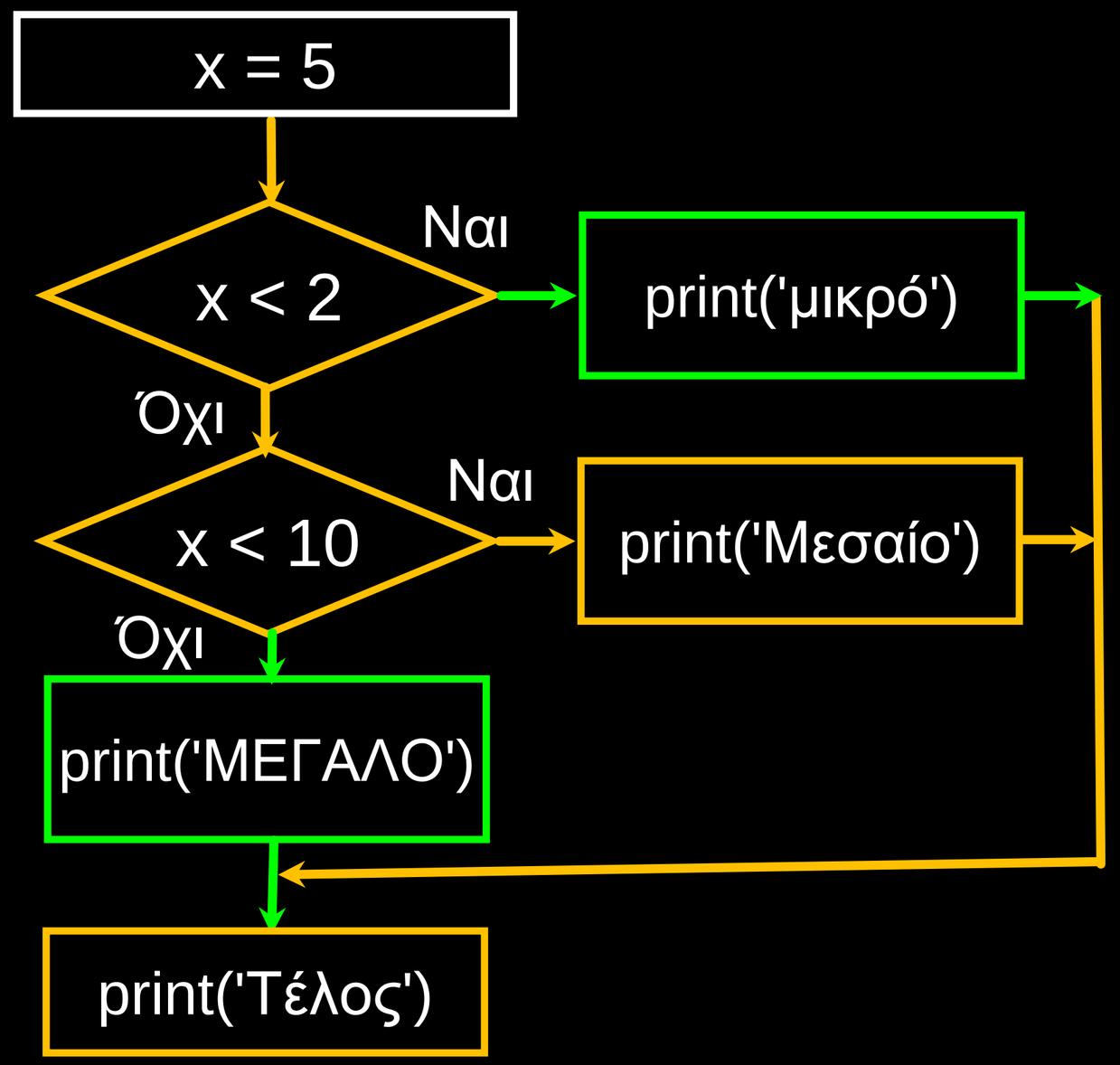
# Πολλαπλών Επιλογών

```
x = 0
if x < 2 :
    print('μικρό')
elif x < 10 :
    print('Μεσαίο')
else :
    print('ΜΕΓΑΛΟ')
print('Τέλος')
```



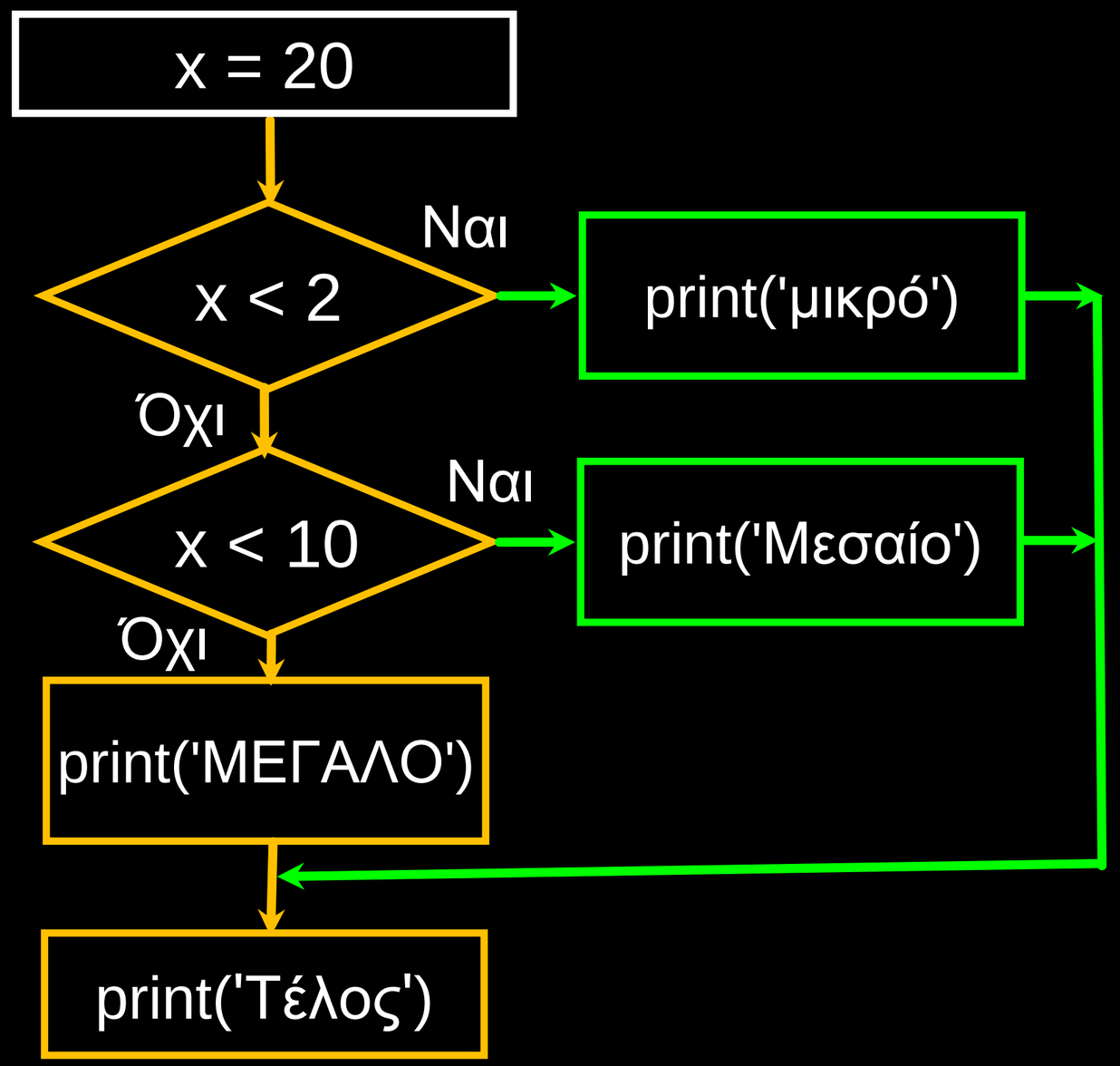
# Πολλαπλών Επιλογών

```
x = 5
if x < 2 :
    print('μικρό')
elif x < 10 :
    print('Μεσαίο')
else :
    print('ΜΕΓΑΛΟ')
print('Τέλος')
```



# Πολλαπλών Επιλογών

```
x = 20
if x < 2 :
    print('μικρό')
elif x < 10 :
    print('Μεσαίο')
else :
    print('ΜΕΓΑΛΟ')
print('Τέλος')
```



# Πολλαπλών Επιλογών

```
# Χωρίς Else
x = 5
if x < 2 :
    print('μικρό')
elif x < 10 :
    print('Μεσαίο')

print('Τέλος')
```

```
if x < 2 :
    print('μικρό')
elif x < 10 :
    print('Μεσαίο')
elif x < 20 :
    print('Μεγάλο')
elif x < 40 :
    print('Μεγαλύτερο')
elif x < 100:
    print('Τεράστιο')
else :
    print('Μεγαλοπρεπές')
```

# Γρίφος Πολλαπλών Επιλογών

Ποιο δεν θα εκτυπωθεί ποτέ ανεξάρτητα από την τιμή του x;

```
if x < 2 :
    print('Κάτω από 2')
elif x >= 2 :
    print('Δύο ή περισσότερο')
else :
    print('Κάτι άλλο')
```

```
if x < 2 :
    print('Κάτω από  2')
elif x < 20 :
    print('Κάτω από  20')
elif x < 10 :
    print('Κάτω από  10')
else :
    print('Κάτι άλλο')
```

# Η δομή try / except

- Περικλείεται ένα επικίνδυνο τμήμα κώδικα σε try και except
- Εάν ο κώδικας στο try λειτουργήσει - το except παραλείπεται
- Εάν ο κώδικας στο try αποτύχει - μεταβαίνει στην ενότητα 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'
```

Τέλος

```
$ cat notry.py
astr = 'Hello Bob'
istr = int(astr)
print('Πρώτο', istr)
astr = '123'
istr = int(astr)
print('Δεύτερο', istr)
```

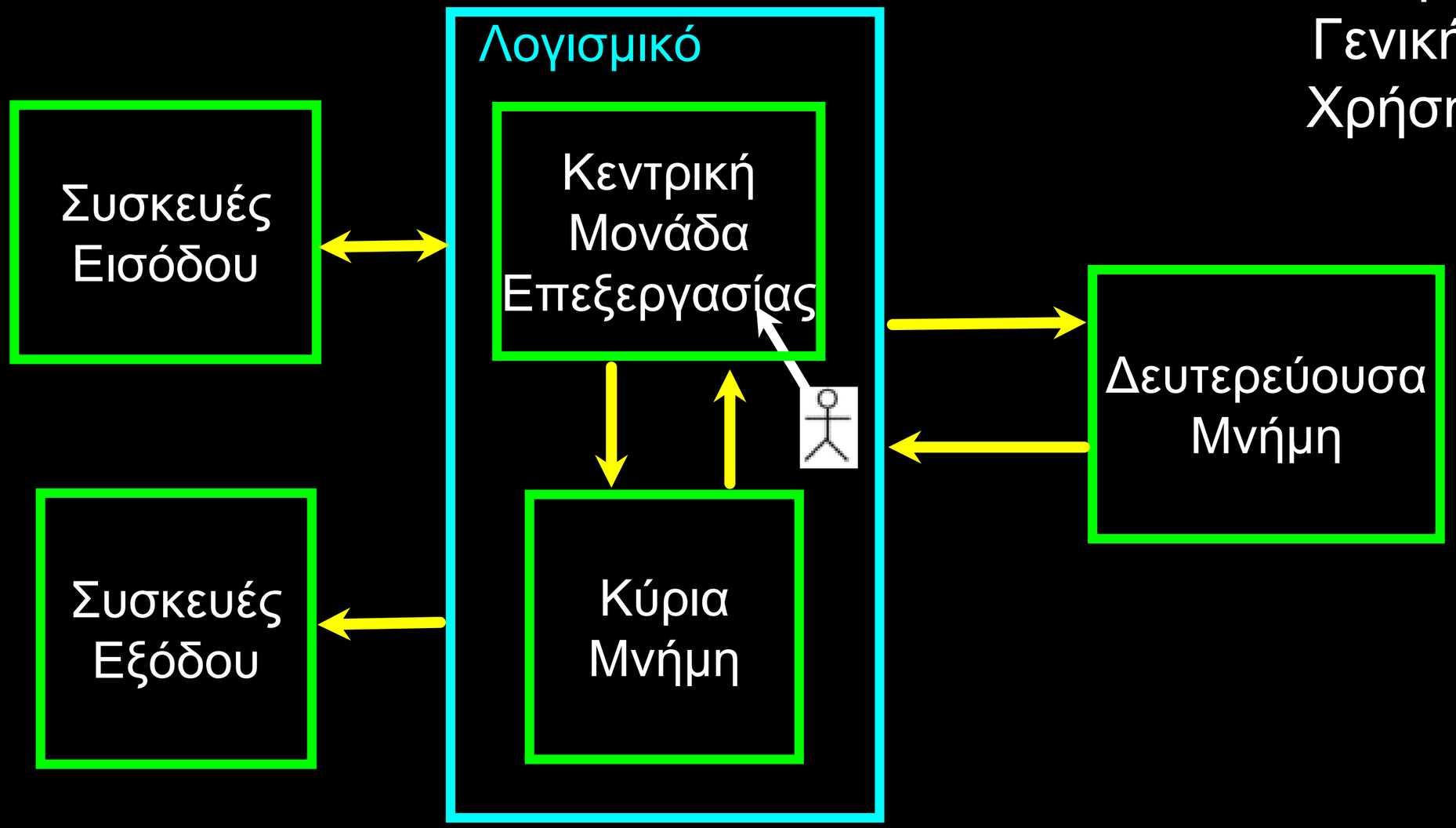
Το πρόγραμμα σταματά εδώ

```
$ cat notry.py
astr = 'Hello Bob'
istr = int(astr)
```

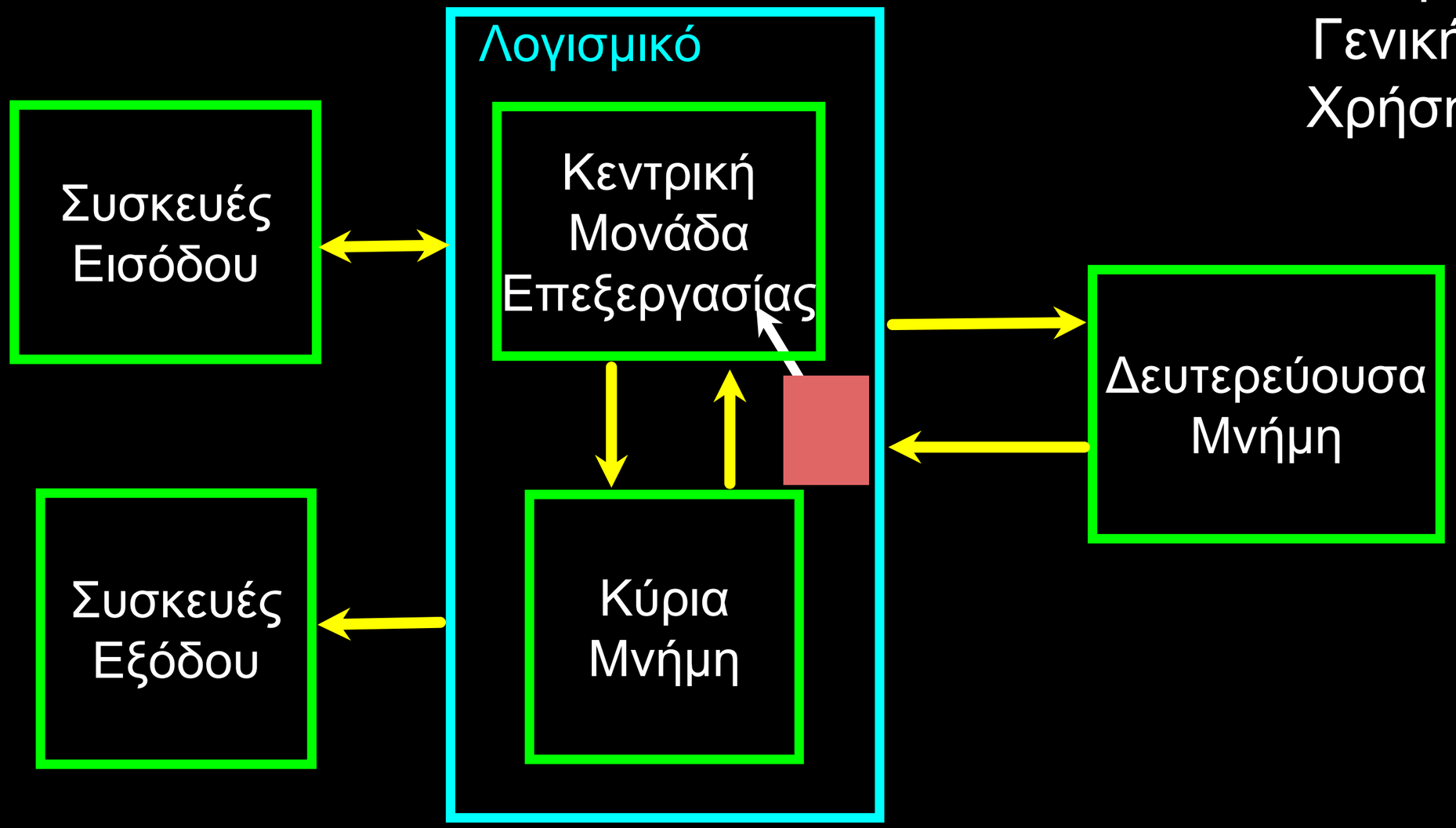
```
$ python3 notry.py
Traceback (most recent call last):
File "notry.py", line 2, in <module>
istr = int(astr)ValueError: invalid literal
for int() with base 10: 'Hello Bob'
```

Τέλος

# Υπολογιστής Γενικής Χρήσης



# Υπολογιστής Γενικής Χρήσης



```
astr = 'Hello Bob'
try:
    istr = int(astr)
except:
    istr = -1

print('Πρώτο', istr)

astr = '123'
try:
    istr = int(astr)
except:
    istr = -1

print('Δεύτερο', istr)
```

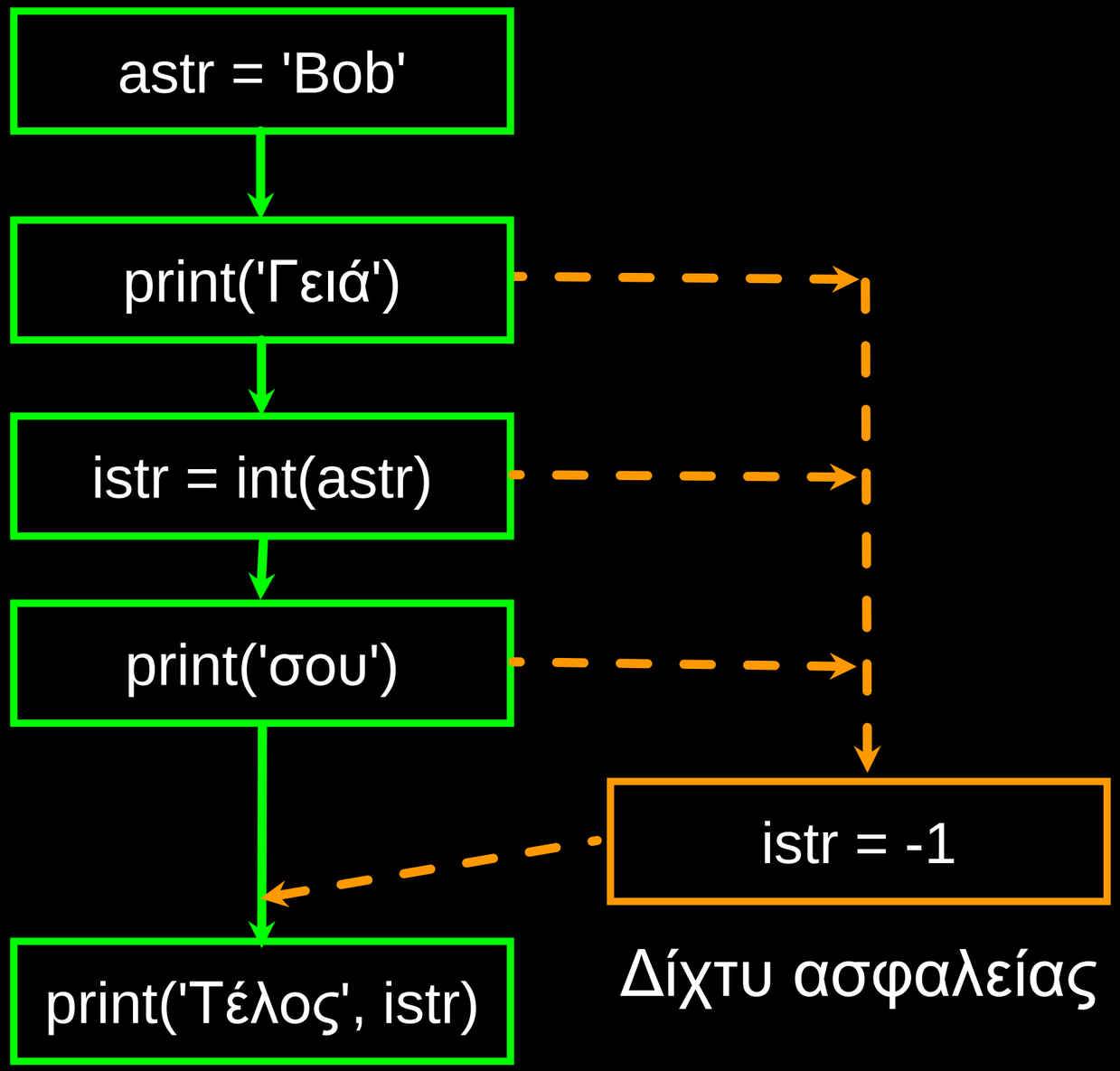
Όταν η πρώτη μετατροπή αποτύχει - απλώς μεταβαίνει στην περίπτωση except: και το πρόγραμμα συνεχίζεται.

```
$ python tryexcept.py
Πρώτο -1
Δεύτερο 123
```

Όταν η δεύτερη μετατροπή επιτύχει - απλώς παραλείπει την περίπτωση except: και το πρόγραμμα συνεχίζεται.

# try / except

```
astr = 'Bob'
try:
    print('Γειά')
    istr = int(astr)
    print('σου')
except:
    istr = -1

print('Τέλος', istr)
```



# Παράδειγμα try / except

```
rawstr = input('Δώστε έναν αριθμό:')
try:
    ival = int(rawstr)
except:
    ival = -1

if ival > 0 :
    print('Ωραία')
else:
    print('Δεν είναι αριθμός')
```

```
$ python3 trynum.py
Δώστε έναν αριθμό:42
Ωραία
$ python3 trynum.py
Δώστε έναν αριθμό :forty-two
Δεν είναι αριθμός
$
```

# Σύνοψη

- Συγκριτικοί τελεστές
  == <= >= > < !=
- Εσοχή
- Απλή Επιλογή
- Σύνθετη Επιλογή:
  if: και else:
- Εμφωλευμένη Επιλογή
- Πολλαπλή Επιλογή με χρήση elif
- try / except για την αποφυγή λαθών

## Άσκηση

Ξαναγράψτε το πρόγραμμά σας για τον υπολογισμό της αμοιβής, ώστε να δώσετε στον υπάλληλο 1,5 φορές την ωριαία χρέωση για τις ώρες εργασίας πέραν των 40 ωρών.

```
Δώστε Ώρες: 45
Δώστε Ποσό/Ώρα: 10

Μισθός: 475.0
```

475 = 40 * 10 + 5 * 15

## Άσκηση

Ξαναγράψτε το πρόγραμμά σας για τον υπολογισμό της αμοιβής χρησιμοποιώντας το try και except ώστε  το πρόγραμμά σας να χειρίζεται μη-αριθμητικά δεδομένα εισόδου χαριτωμένα.

```
Δώστε Ώρες: 20
Δώστε Ποσό/Ώρα: nine
Λάθος, παρακαλώ εισάγεται αριθμό

Δώστε Ώρες: forty
Λάθος, παρακαλώ εισάγεται αριθμό
```