Python

Επαναληψη

Έλεγχος ροής - if

Σύνταξη

`if` (λογική συνθήκη):
    εντολή 1
    εντολή 2
    ……..
εντολή (εκτός `if`)
εντολή (εκτός `if`)

Το τέλος της `if` δηλώνετε με το τέλος της στοίχισης των εντολών

if - elif - else

Σύνταξη

```
if (λογική συνθήκη):
    εντολή 1
    εντολή 2
elif (λογική συνθήκη):
    εντολή 3
    εντολή 4
else:
  εντολή 3
  εντολή 4
```

Λογικές εκφράσεις

and

Εκφραση	Αποτέλεσμα
αληθές **and** αληθές	αληθές
αληθές **and** ψευδές	ψευδές
ψευδές **and** αληθές	ψευδές
ψευδές **and** ψευδές	ψευδές

Λογικές εκφράσεις

or

Εκφραση	Αποτέλεσμα
αληθές **or** αληθές	αληθές
αληθές **or** ψευδές	αληθές
ψευδές **or** αληθές	αληθές
ψευδές **or** ψευδές	ψευδές

Λογικές εκφράσεις

`not`

Εκφραση	Αποτέλεσμα
`not` αληθές	ψευδές
`not` ψευδές	αληθές

Λογικές εκφράσεις

Σειρά πράξεων: Προηγείτε το **not** από το **and**

```
>>> print (False and False)
False
>>> print (not False and False)
False
>>> print (not ( False and False))
True
>>>
```

Λογικές εκφράσεις

Σειρά πράξεων: Προηγείτε το **and** από το **or**

```
>>> print (False and False or True)
True
>>> print (False and (False or True))
False
>>> print ((False and False) or True)
True
>>>
```

Ανισότητα - Ισότητα

- `c==40` (ισότητα)
- `c!=40` (διάφορο)
- `c>=40` (μεγαλύτερο ή ίσο)
- `c>40` (γνήσια μεγαλύτερο)
- `not c==40` είναι ίδιο με το `c!=40`

Επαναλήψεις - Loops

while

Σύνταξη

`while` (λογική συνθήκη)**:**
εντολή 1
εντολή 2

Το τέλος της **`while`** δηλώνετε με το τέλος της στοίχισης των εντολών

while

Σύνταξη

```
while (λογική συνθήκη):
εντολή 1
εντολή 2
```

Αν η λογική συνθήκη =`True` εκτελείτε το block εντολών της while. Για να μην γίνεται αυτό συνέχεια πρέπει οι εντολές του block να μπορούν να αλλάξουν τη λογική συνθηκη σε `False`

while

Παράδειγμα

```
while (n<=10):
   εντολή 1
   εντολή 2
   ……
   n=n+1
```

Ακολουθίες

- strings
- λίστες
- tuples

For

Σύνταξη

```
for (μετρητής) in (ακολουθία):
  εντολή 1
  εντολή 2
```

Το τέλος της **for** δηλώνετε με το τέλος της στοίχισης των εντολών

for

- Εντολή `in:` Έλεγχος εγκλεισμού
- `1 in [1,2,3]` ⟶ `True`
- `4 in [1,2,3]` ⟶ `False`
- `4 not in [1,2,3]` ⟶ `True`
- `‘a’ in ‘abc’` ⟶ `True`

for - break

```
for x in range(20):
    if (x%2==0):
        print(x)
        if x==16:
            print('break')
            break
```

→ stop εκτέλεσης `for`

for - continue

```
for x in range(20):
    if (x%2==0):
        print('%d is even'%x)
        continue
    print(x)
```

→ συνέχιση εκτέλεσης αύξηση μετρητή

Λίστες

Λίστες

- `C=[-10,-5,0,10]`
- `C.append(15)` `(C=[-10,-5,0,10,15])`
- `C=C+[20,25]` `(C=[-10,-5,0,10,15,20,25])`
- `C.insert(0,-15)` `(C=[-15,–10,-5,0,10,15,20,25])`

Λίστες

- del C[2] (Διαγραφή 3ου στοιχείου)
  C=[-15,-10,0,10,15,20,25]

- `len(C)` (Μήκος και για αλλες ακολουθιακές δομές)
- `10 in C` (εγκλεισμός)
- `C.pop()` (επιστρέφει το τελευταίο στοιχείο και το αφαιρεί)

Λίστες

- `C=range(-10,30,5)` (λίστα με βήμα 5)
- Η `range` μπορεί να κάνει ακέραιο βήμα

```
c=[]
c_val=10
c_max=12
while c_val<=c_max:
    c.append(c_val)
    c_val+=.2
```

Λίστες

- Θέλουμε να μεταβάλουμε τα στοιχεία μιας δοσμένης λίστας

- C=[0,5,10,15]

- θέλουμε να γίνει [2,7,12,17]

```
for i in C:
   i+=2
```

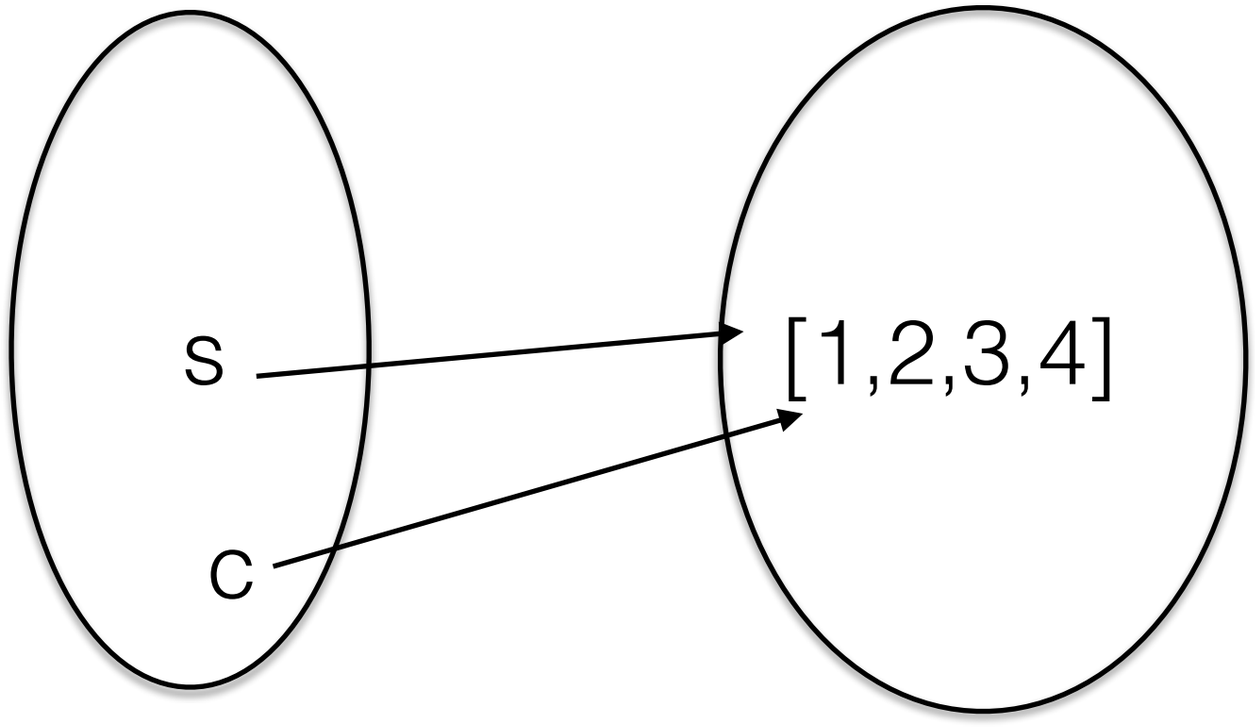
- Λάθος αντιμετώπιση

Λίστες

```
m=len(C)
for i in range(m):
     C[i]+=2
```

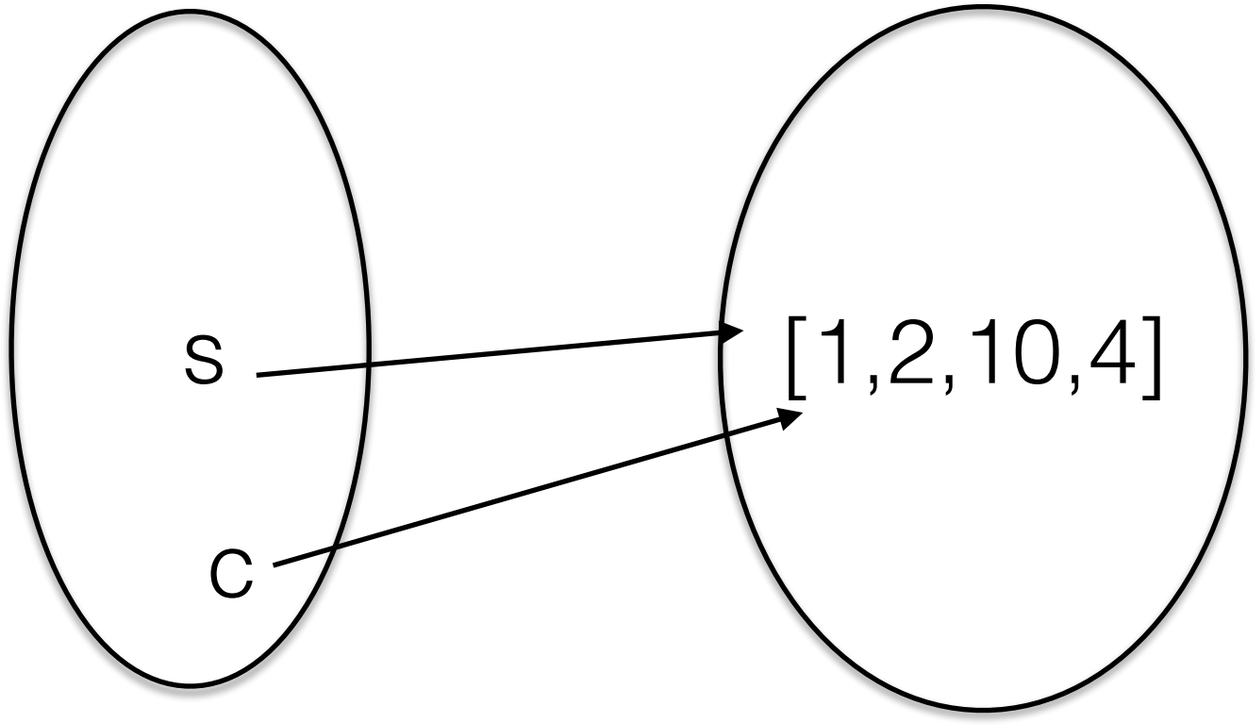
Αλλάζει τη λίστα με τον επιθυμητο τρόπο

Λίστες



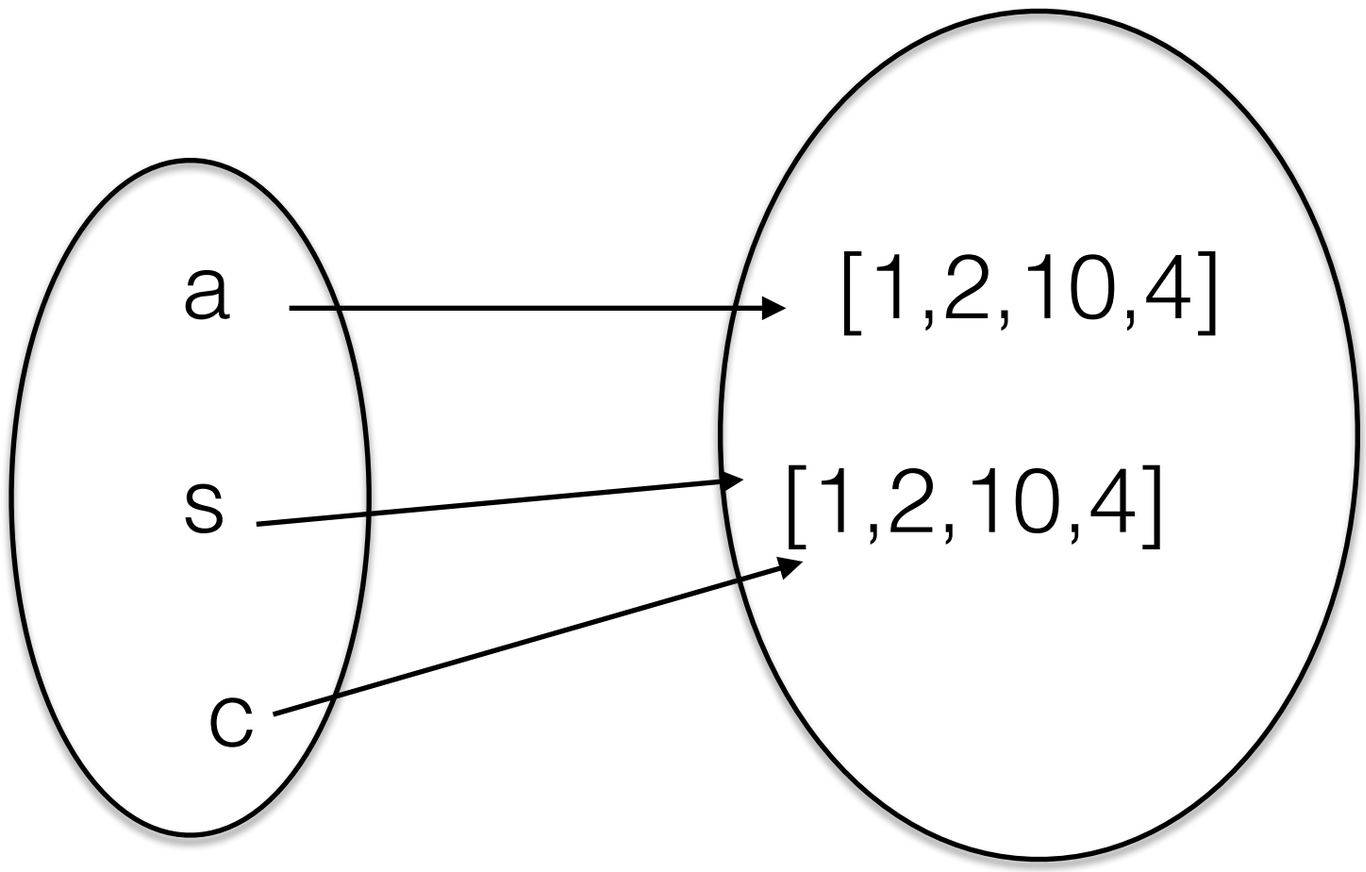

```
s=[1,2,3,4]
c=s
```

Λίστες



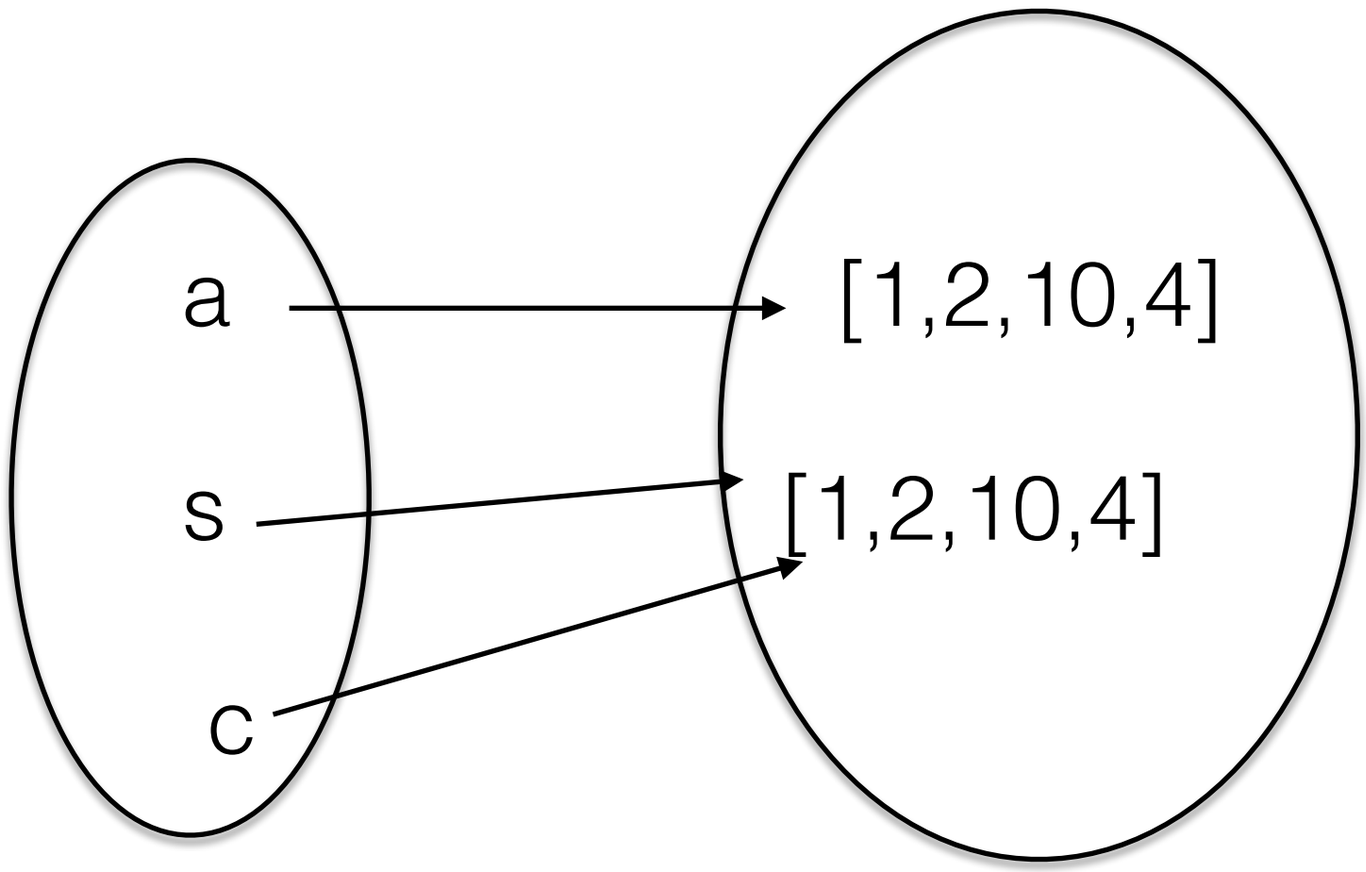

```
c[2]=10
```

Λίστες




```
a=c[:]
```

Λίστες




```
s is c  ⟶  True
a is c  ⟶  False
a==c    ⟶  True
```

Λίστες - comprehension

```
Cdegrees = [-5 + i*0.5 for i in range(n)]
```

Φτιάχνει τη λίστα `[-5, -4.5, -4, -3.5, -3]` για `n=5`

```
Fdegrees = [(9.0/5)*C + 32 for C in Cdegrees]
```

Η λίστα με τις τιμές Farehneit που αντιστοιχούν στις τιμές της `Cdegrees`

```
C_plus_5 = [C+5 for C in Cdegrees]
```

Προσθέτουμε τον αριθμό 5 σε κάθε στοιχείο της `Cdegrees`

tuples

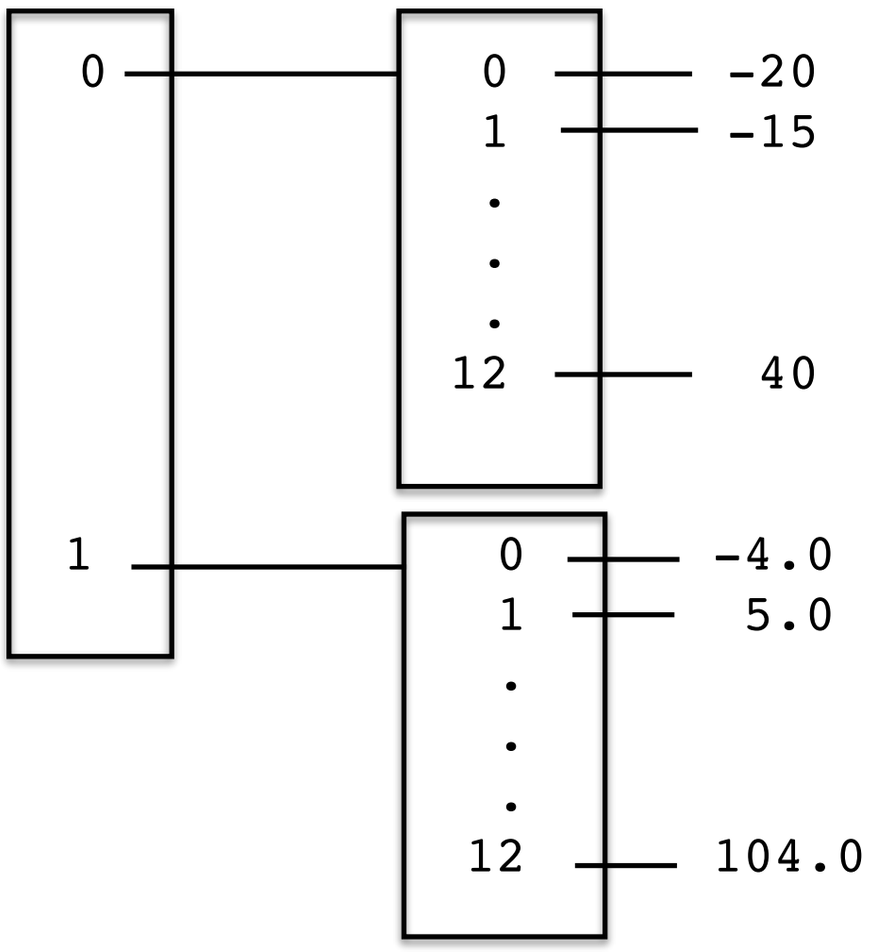
- Δομή παρόμοια με τις λίστες
- Δεν μεταβάλονται
- `c=(2,1,'Hello')`
- Δεν μπορώ να διαγράψω, προσθέσω ή μεταβάλω στοιχείο της c

zip

- `C, F`: Δύο λίστες
- `z=zip(C,F)` (“ένωση”-“ζευγάρωμα” των `C,F`)
- `C=[1,2,3]; F=['a','b','c']`
- `z=list(zip(C,F))`
- `z=[(1,'a'),(2,'b'),(3,'c')]`
- `z` είναι λίστα και κάθε στοιχείο της είναι tuple

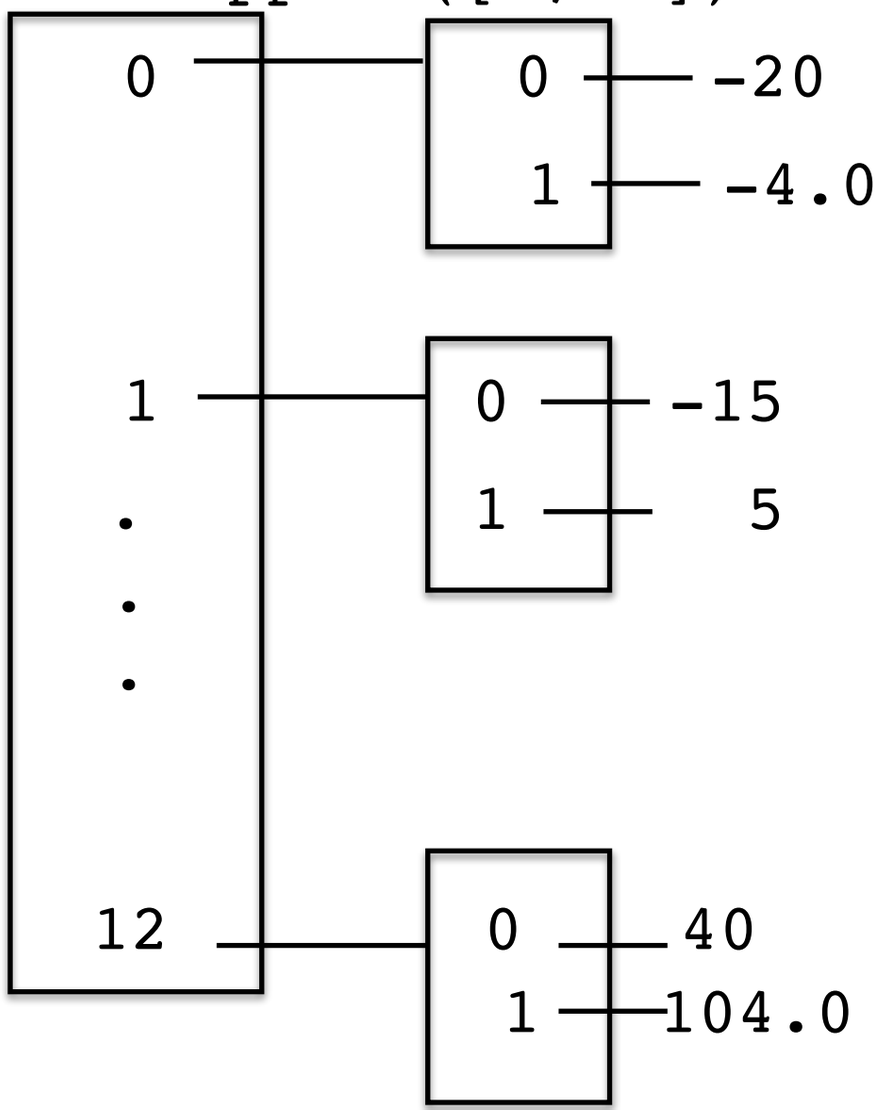
πίνακας

```
Cdegrees = [-20 + i*5 for i in range(n)]
Fdegrees = [(9.0/5)*C + 32 for C in Cdegrees]
table=[Cdegrees, Fdegrees]
```



πίνακας

```
table = []
for C, F in zip(Cdegrees, Fdegrees):
    table.append([C, F])
```



Λίστες - χρήσιμες εντολές

```
a=[1,20,9,40,100,40,'a']
a.remove('a') ⟶ [1,20,9,40,100,40])
a.index(20)   ⟶  1
a.count(40)   ⟶  2
len(a)        ⟶  6
min(a)        ⟶  1
max(a)        ⟶  100
sum(a)        ⟶
a.reverse()   ⟶ [40,100,40,9,20,1]
a.sort()      ⟶ [1,9,20,40,40,100]
```

Συναρτήσεις

καθολική - τοπική μεταβλητή

```
print(sum)⟶   Είναι  built-in συνάρτηση της Python (δεν είναι μεταβλητή)
sum = 500 ⟶   Ξαναορίζουμε την αντιστοιχία του ονόματος sum με έναν ακέραιο
print(sum)⟶   Η sum είναι μια καθολική μεταβλητή
```

```
def myfunc(n):
    sum = n + 1
    print(sum)⟶   Η sum είναι μια τοπική μεταβλητή
    return sum

sum = myfunc(2) + 1 # Νέα τιμή στην καθολική μεταβλητή sum
print(sum)
```

Συναρτήσεις

```
def F(C):
    Fvalue=(9.0/5)*C+32
    print('Inside F: C=%g, Fvalue=%g, r=%g'%(C,Fvalue,r))
    return '%g degrees C is %g degrees F'%(C,Fvalue)
```

```
r=21
print(F(r))
print(C)
print(Fvalue)
```

Σφάλμα - δεν έχει ορισθεί εκτός συνάρτησης

Σφάλμα - δεν έχει ορισθεί εκτός συνάρτησης

καθολική - τοπική μεταβλητή

```
a = 20; b = -2.5          ⟶  καθολικές μεταβλητές
def f1(x):
    a = 21                ⟶  νέα τοπική μεταβλητή
    return a*x + b        ⟶  21*x - 2.5
print a                   ⟶   20

def f2(x):
    global a              ⟶  Η a ορίζεται ως καθολική μεταβλητή
    a = 21                ⟶  νέα τιμή της καθολικής μεταβλητής  a
    return a*x + b        ⟶  21*x - 2.5

f1(3); print(a)           ⟶    20
f2(3); print(a)           ⟶    21
```

Ορίσματα

$$y(t) = v_0 t - \frac{1}{2} g t^2$$

```
def yfunc(t, v0):
    g = 9.81
    return v0*t - 0.5*g*t**2


y = yfunc(0.1, 6)
y = yfunc(0.1, v0=6)
y = yfunc(t=0.1, v0=6)
y = yfunc(v0=6, t=0.1)
```

Ορίσματα

$$y(t) = v_0 t - \frac{1}{2} g t^2$$

```
def yfunc(t, v0, g=9.81):
    return v0*t - 0.5*g*t**2
```

```
y = yfunc(0.1, 6)
y = yfunc(0.1, v0=6)
y = yfunc(t=0.1, v0=6)
y = yfunc(v0=6, t=0.1)
y = yfunc(0.1,6,1.6)
```
Η βολή στη Σελήνη

Επιστροφή πολλών τιμών

$$y(t) = v_0 t - \frac{1}{2} g t^2$$

$$y'(t) = v_0 - gt$$

```
def yfunc(t, v0):
    g=9.81
    y=v0*t - 0.5*g*t**2
    dydt=v0-g*t
    return y,dydt

thesi, taxitita=yfunc(0.1,6)
s=yfunc(0.1,6)
```

⟶ ταυτόχρονη ανάθεση

⟶ Η `s` είναι tuple

docstrings

```
def yfunc(t, v0):
    ''' Calculate position and velocity'''
    g=9.81
    y=v0*t - 0.5*g*t**2
    dydt=v0-g*t
    return y,dydt
```

- Αποτελεί σχόλιο για τη συνάρτηση
- Μπορούμε να τυπώσουμε το **`docstring`** μιας συνάρτησης
    - **`help(yfunc)`**
    - **`print(yfunc.__doc__)`**
- Μπορεί να εκτείνετε σε πολλές σειρές

βιβλιοθήκες - module

- Αρχείο με όνομα π.χ. **`myfunctions.py`**

```
def yfunc(t, v0):
    ''' Calculate position and velocity'''
    g=9.81
    y=v0*t - 0.5*g*t**2
    dydt=v0-g*t
    return y,dydt
```

- Από την python καλούμε import myfunctions

Λεξικό

Λεξικό

- `temps={‘Oslo’:13, ‘Heraklion’:27, ‘London’:15.4}`
- κλειδιά - `temps.keys(): [‘Oslo’, ‘Heraklion’, ‘London’]`
- τιμές - `temps.values(): [13,27,15.4]`
- ```
  for city in temps:
    print(‘The temperature in %s is %g’%(city,temps[city]))
  ```
  ```

Λεξικό

Δεν διατρέχει με τη σειρά που εμφανίζονται:

```
The temperature in Oslo is 13
The temperature in London is 15.4
The temperature in Heraklion is 27
```

- ```
  for city in temps.keys():
    print('The temperature in %s is %g'%(city,temps[city]))
  ```

- ```
  for city in sorted(temps):
    print('The temperature in %s is %g'%(city,temps[city]))
  ```

Λεξικό - πολυώνυμα

$$p(x) = 3x^7 + x^2 - 1$$

- Λίστα συντελεστών

```
p=[-1,0,1,0,0,0,0,3]
```

- Λεξικό συντελεστών

```
p={0:-1, 2:1, 7:3}
```

(`p.keys()`: βαθμός, `p.values()`: συντελεστής)

πολυώνυμα

```
def poly1(data,x):
    #data is a list
    sum_n=0.0
    for power in range(len(data)):
        sum_n+=data[power]*x**power
    return sum_n

def poly2(data,x):
    #data is a dictionary
    sum_n=0.0
    for power in data:
        sum_n+=data[power]*x**power
    return sum_n
```