



Γιώργος Μανής
Επίκουρος Καθηγητής
Τμήματος Μηχανικών Η/Υ και Πληροφορικής
3 Μαΐου 2015

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»

Δομές Δεδομένων της Python

- # αλφαριθμητικά
- # πλειάδες
- # λίστες
- # λεξικά
- # πίνακες ?

Strings

- # >>>"Hello, world!"
 - 'Hello, world!'
- # s= 'Hello, world!'
 - >>>s
 - 'Hello, world!'
- # >>>x="Hello, "
 - >>>y="world!"
 - >>>x+y
 - 'Hello, world!'

Λίστες

δημιουργία λίστας

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

παράθεση

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

Λίστες

διάβασμα – γράψιμο

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

Λίστες

διαγραφή

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

αντικατάσταση

- `>>> list = [1,2,3,4,5,6]`
- `>>> list[3:]=[7,8,9]`
- `>>> list`
- `[1, 2, 3, 7, 8, 9]`

Πλειάδες

- # Οι πλειάδες δημιουργούνται και δεν τροποποιούνται στη συνέχεια
- # `>>> 1, 2, 3`
 - `(1, 2, 3)`
- # `>>> tuple([1, 2, 3])`
 - `(1, 2, 3)`
- # `>>> tuple('abc')`
 - `('a', 'b', 'c')`

Πλειάδες

>>> x = 1, 2, 3

- >>> x[1]

- 2

- >>> x[0:2]

- (1, 2)

Χρησιμότητα

- Θα μπορούσαμε να χρησιμοποιούσαμε και λίστες αντί τις πλειάδες

- Κάποιες λειτουργίες απαιτούν την χρήση τους αντί αυτή των λιστών για να εξασφαλίζουν ότι δε θα τροποποιηθούν



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'}`
 - `>>> phonebook['Αλίκη']`
 - `'2341'`

Λεξικά

- # >>> d = {}
 - >>> d['name'] = 'Γιώργος'
 - >>> d['age'] = 42
 - >>> d
 - {'age': 42, 'name': 'Γιώργος'}

- # >>> d['age']
 - 42

Δομές Έλεγχου

- # Η δομή if, elif και else
- # Ο βρόχος while
- # Ο βρόχος for
- # Επαναλήψεις σε λεξικά

if, 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')
```

```
>>> x = 1  
>>> eval('x + 1')  
2  
>>> eval('x')  
1
```

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)
```

Ο ιδιωματισμός while True/break

χωρίς break:

```
# 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 *else* στη *for*

```
>>> for n in range(2, 10):  
...     for x in range(2, n):  
...         if n % x == 0:  
...             print n, 'equals', x, '*', n/x  
...             break  
...         else:  
...             # loop fell through without finding a factor  
...             print n, 'is a prime number'  
...  
2 is a prime number  
3 is a prime number  
4 equals 2 * 2  
5 is a prime number  
6 equals 2 * 3  
7 is a prime number  
8 equals 2 * 4  
9 equals 3 * 3
```

Χρήση else στη for

- ✦ Το for μπορεί να ακολουθείται από ένα else
- ✦ το else εκτελείται όταν τελειώσουν οι επαναλήψεις του for
- ✦ αλλά όχι όταν εξερχόμαστε από το βρόχο με break

Χρήση else στη while

```
count = 0
while count < 5:
    print count, " is less than 5"
    count = count + 1
else:
    print count, " is not less than 5"
```

Χρήση else στη while

- # Το else μπορεί να ακολουθείται από ένα else
- # το else εκτελείται όταν γίνει false η συνθήκη της while
- # αλλά όχι όταν εξερχόμαστε από το βρόχο με break

Χρήση else στη for - συμπληρώστε

```
# from math import sqrt
# for n in range(99, 81, -1):
#     root = sqrt(n)
#     if root == int(root):
#         print(n)
#         ???
# 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']
```

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:
- # $A=[0 \text{ for } i \text{ in range}(N)]$

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 $N \times N$

$$C = A \cdot B$$

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

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("")
```

Περισσότεροι Πίνακες

σε βιβλιοθήκη ...

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

- # $\text{fib}(0) = 1$
- # $\text{fib}(1) = 1$
- # $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$

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.

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!
```

```
# >>> 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!
```

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!

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 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
```

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]`

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)  
  
# 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>

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

linearch

```
# def linearch(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 $2^n > \text{αριθμός στοιχείων}$
- Πίνακας 30 στοιχείων χρειάζεται το πολύ 5 βήματα
 - $2^5 > 30$ δηλαδή 5 βήματα

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
```

Άνοιγμα αρχείων

- # `>>>f=open('somefile.txt')`
- # ή `>>>f = open(r'C:\text\somefile.txt')`
- # Το f είναι file object.
- # Αν δεν προσδιορίσω διαφορετικά, το αρχείο έχει ανοιχτεί για ανάγνωση.
- # Αν το αρχείο δεν υπάρχει:
 - Traceback (most recent call last):
 - File "<pyshell#0>", line 1, in ?
 - IOError: [Errno 2] No such file or directory: "C:\\text\\somefile.txt"

File modes

- # 'r' Read mode
 - # 'w' Write mode
 - # 'a' Append mode
 - # 'b' Binary mode (added to other mode)
-
- # Αν δεν προσδιορίσω mode, εννοείται 'r'.
 - # 'rb' σημαίνει ότι ανοίγουμε δυαδικό αρχείο (όχι αρχείο κειμένου).

Μέθοδοι αρχείων

- # `>>> f = open('somefile.txt', 'w')`
 - # `>>> f.write('Hello, ')`
 - # `>>> f.write('World!')`
 - # `>>> f.close()`
- # Η `f.close()` χρειάζεται οπωσδήποτε στην εγγραφή γιατί μπορεί ο υπολογιστής να έχει κρατήσει τα δεδομένα που θέλουμε να γράψουμε σε buffer και να μην τα έχει σώσει στο αρχείο.

Μέθοδοι αρχείων

- # `>>> f = open('somefile.txt', 'r')`
- # `>>> f.read(4)`
 - `'Hell'`
- # `>>> f.read()`
 - `'o, World!'`
- # Το `f.read(4)` διαβάζει 4 χαρακτήρες, ενώ το `f.read()` διαβάζει όλο το υπόλοιπο αρχείο.

Τυχαία πρόσβαση

- # Υπάρχει η μέθοδος `seek(offset[, whence])`.
- # `offset` είναι ο αριθμός χαρακτήρων που θέλουμε να μετακινηθούμε.
- # Το `whence` μπορεί να είναι:
 - 0 (default), όπου μετράμε από την αρχή του αρχείου.
 - 1, όπου μετράμε από την τρέχουσα θέση.
 - 2, όπου μετράμε από το τέλος του αρχείου.

Τυχαία πρόσβαση

```
# >>> f = open('somefile.txt', 'w')  
  
# >>> f.write('01234567890123456789')  
  
# >>> f.seek(5)  
  
# >>> f.write('Hello, World!')  
  
# >>> f.close()  
  
# >>> f = open('somefile.txt')  
  
# >>> f.read()  
    ■ '01234Hello, World!89'
```

Τυχαία πρόσβαση

- # Η μέθοδος `tell()` μας δίνει την τρέχουσα θέση μέσα στο αρχείο.
- # `>>> f = open('somefile.txt')`
- # `>>> f.read(3)`
 - `'012'`
- # `>>> f.read(2)`
 - `'34'`
- # `>>> f.tell()`
 - `5`

readline, readlines, writelines

- # Η μέθοδος `readline()` διαβάζει την επόμενη γραμμή του αρχείου.
- # Το τέλος μίας γραμμής υποδηλώνεται από το `\n`.
- # Η `readline(5)` διαβάζει τους 5 πρώτους χαρακτήρες της γραμμής.
- # Η μέθοδος `readlines()` διαβάζει όλες τις γραμμές του αρχείου και τις επιστρέφει σε μία λίστα.
- # Η μέθοδος `writelines()` παίρνει μία λίστα από αλφαριθμητικά και τα σώζει στο αρχείο. Πρέπει εμείς να προσθέσουμε τα `\n`.

Παραδείγματα

- # Έστω ότι το αρχείο somefile.txt περιέχει:
 - Welcome to this file
 - There is nothing here except
 - This stupid haiku
- # >>> f = open('somefile.txt')
- # >>> f.read(7)
 - 'Welcome'
- # >>> f.read(4)
 - ' to '
- # >>> f.close()

read()

- # >>> f = open('somefile.txt')
- # >>> print(f.read())
 - Welcome to this file
 - There is nothing here except
 - This stupid haiku
- # >>> f.close()

readline()

```
# >>> f = open('somefile.txt')  
  
# >>> for i in range(3):  
#     print(str(i) + ': ' + f.readline(),)  
#     0: Welcome to this file  
#     1: There is nothing here except  
#     2: This stupid haiku  
  
# >>> f.close()
```

write(string)

- # >>> f = open('somefile.txt', 'w')
- # >>> f.write('this\nis no\nhaiku')
- # >>> f.close()
- # Το αρχείο τώρα περιέχει:
 - this
 - is no
 - haiku

writelines(list)

```
# >>> f = open('somefile.txt')  
  
# >>> lines = f.readlines()  
  
# >>> f.close()  
  
# >>> lines[1] = "isn't a\n"  
  
# >>> f = open('somefile.txt', 'w')  
  
# >>> f.writelines(lines)  
  
# >>> f.close()
```

writelines(list)

Τώρα το αρχείο περιέχει:

- this
- isn't a
- haiku

Επαναλήψεις σε αρχεία

- Έστω μια απλή συνάρτηση:
- ```
def process(string):
 print('Processing: ', string)
```
- ```
f = open(filename)
```
- ```
char = f.read(1)
```
- ```
while char:  
    process(char)  
    char = f.read(1)
```
- ```
f.close()
```

# *Me while True/break*

---

```
f = open(filename)

while True:

 char = f.read(1)

 if not char: break

 process(char)

f.close()
```

# Διαβάζοντας μία γραμμή τη φορά

---

```
f = open(filename)

while True:

 line = f.readline()

 if not line: break

 process(line)

f.close()
```

# Διαβάζοντας ολόκληρο το αρχείο

---

- # `f = open(filename)`
- # `for char in f.read():`  
    `process(char)`
- # `f.close()`
- # `f = open(filename)`
- # `for line in f.readlines():`  
    `process(line)`
- # `f.close()`

# Επαναλήψεις με *file objects*

---

- # `f = open(filename)`
- # `for line in f:`  
    `process(line)`
- # `f.close()`
  
- # Εναλλακτικά:
- # `for line in open(filename):`  
    `process(line)`

# ***list(open(filename))***

---

- # Μπορώ να μετατρέψω τα περιεχόμενα του αρχείου σε λίστα χρησιμοποιώντας `list(open(filename))`
- # Αντίστοιχα με το
- # `list(range(5))`

# Παράδειγμα

---

```
>>> f = open('somefile.txt', 'w')

>>> f.write('First line\n')

>>> f.write('Second line\n')

>>> f.write('Third line\n')

>>> f.close()

>>> lines = list(open('somefile.txt'))

>>> lines
 ▪ ['First line\n', 'Second line\n', 'Third line\n']
```

# Παράδειγμα (συνέχεια)

---

```
>>> first, second, third = open('somefile.txt')

>>> first
 ■ 'First line\n'

>>> second
 ■ 'Second line\n'

>>> third
 ■ 'Third line\n'
```

# *numpy*

---

- # βιβλιοθήκη για πίνακες

# *Πίνακες στο numpy με παραδείγματα*

---

```
>>> from numpy import *

>>> a = arange(15).reshape(3, 5)

>>> a
array([[0, 1, 2, 3, 4],
 [5, 6, 7, 8, 9],
 [10, 11, 12, 13, 14]])

>>> a.shape
 (3, 5)

>>> a.ndim
 2
```

# *Πίνακες στο numpy με παραδείγματα*

---

- # >>> a.dtype.name
  - 'int32'
- # >>> a.itemsize
  - 4
- # >>> a.size
  - 15
- # >>> type(a)
  - numpy.ndarray

# *Πίνακες στο numpy με παραδείγματα*

---

```
>>> b = array([6, 7, 8])
```

```
>>> b
```

- `array([6, 7, 8])`

```
>>> type(b)
```

- `numpy.ndarray`

# *Πίνακες στο numpy με παραδείγματα*

---

- # `>>> zeros( (3,4) )`
- # `array([[0., 0., 0., 0.],  
[0., 0., 0., 0.],  
[0., 0., 0., 0.]])`
- # `>>> ones( (2,3,4), dtype=int16 )` # dtype can also be specified
- # `>>> empty( (2,3) )`
- # `array([[ 3.73603959e-262, 6.02658058e-154, 6.55490914e-260],  
[ 5.30498948e-313, 3.14673309e-307, 1.00000000e+000]])`

# *Πίνακες στο numpy με παραδείγματα*

---

- # >>> a = arange(6) # 1d array
- # >>> print a
- # [0 1 2 3 4 5]
- # >>> b = arange(12).reshape(4,3) # 2d array
- # >>> print b
  - [[ 0 1 2]
  - [ 3 4 5]
  - [ 6 7 8]
  - [ 9 10 11]]

# *Πίνακες στο numpy με παραδείγματα*

---

```
>>> c = arange(24).reshape(2,3,4) # 3d array
```

```
>>> print c
```

- `[[[ 0 1 2 3]`
- `[ 4 5 6 7]`
- `[ 8 9 10 11]]`
  
- `[[12 13 14 15]`
- `[16 17 18 19]`
- `[20 21 22 23]]]`

# *Πίνακες στο numpy με παραδείγματα*

---

```
>>> a = array([20,30,40,50])
```

```
>>> b = arange(4)
```

```
>>> b
```

```
array([0, 1, 2, 3])
```

```
>>> c = a-b
```

```
>>> c
```

```
array([20, 29, 38, 47])
```

# *Πίνακες στο numpy με παραδείγματα*

---

- # `>>> b**2`
- # `array([0, 1, 4, 9])`
- # `>>> 10*sin(a)`
- # `array([ 9.12945251, -9.88031624, 7.4511316 , -2.62374854])`
- # `>>> a<35`
- # `array([True, True, False, False])`

# *Πίνακες στο numpy με παραδείγματα*

---

```
>>> A = array([[1,1],
 ... [0,1]])

>>> B = array([[2,0],
 ... [3,4]])

>>> A*B # elementwise product

array([[2, 0],
 [0, 4]])

>>> dot(A,B) # matrix product

array([[5, 4],
 [3, 4]])
```

# Παράλληλος προγραμματισμός

---

```
import multiprocessing as mp
import random
import string

Define an output queue
output = mp.Queue()

define a example function
def rand_string(length, output):
 """ Generates a random string of numbers, lower- and uppercase chars. """
 rand_str = ''.join(random.choice(
 string.ascii_lowercase
 + string.ascii_uppercase
 + string.digits)
 for i in range(length))
 output.put(rand_str)
```

# Παράλληλος προγραμματισμός

---

```
Setup a list of processes that we want to run
processes = [mp.Process(target=rand_string, args=(5, output)) for x in range(4)]

Run processes
for p in processes:
 p.start()

Exit the completed processes
for p in processes:
 p.join()

Get process results from the output queue
results = [output.get() for p in processes]

print(results)
```

# Παράλληλος προγραμματισμός

---

```
def cube(x):
 return x**3
```

```
pool = mp.Pool(processes=4)
results = pool.map(cube, range(1,7))
print(results)
```

```
[1, 8, 27, 64, 125, 216]
```

Εάν χρησιμοποιήσουμε τη `pool.map_async` η λίστα θα περιέχει τις τιμές όπως παρήχθησαν και όχι με τη σειρά που αναλογούν στην λίστα της εισόδου

# Classes and Instantiation

---

```
class ClassName:
 <statement-1>
 .
 .
 .
 <statement-N>
```

```
class MyClass:
 """A simple example class"""
 i = 12345
 def f(self):
 return 'hello world'
```

```
x = MyClass()
```

---

# *Initialization*

---

```
>>> class Complex:
... def __init__(self, realpart, imagpart):
... self.r = realpart
... self.i = imagpart
...
>>> x = Complex(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

# *Classes and Instance Variables*

---

```
class Dog:

 kind = 'canine' # class variable shared by all instances

 def __init__(self, name):
 self.name = name # instance variable unique to each instance

>>> d = Dog('Fido')
>>> e = Dog('Buddy')
>>> d.kind # shared by all dogs
'canine'
>>> e.kind # shared by all dogs
'canine'
>>> d.name # unique to d
'Fido'
>>> e.name # unique to e
'Buddy'
```

# *Classes and Instance Variables*

---

```
class Dog:

 tricks = [] # mistaken use of a class variable

 def __init__(self, name):
 self.name = name

 def add_trick(self, trick):
 self.tricks.append(trick)

>>> d = Dog('Fido')
>>> e = Dog('Buddy')
>>> d.add_trick('roll over')
>>> e.add_trick('play dead')
>>> d.tricks # unexpectedly shared by all dogs
['roll over', 'play dead']
```

# *Inheritance*

---

```
class DerivedClassName(BaseClassName):
 <statement-1>
 .
 .
 .
 <statement-N>
```

```
class DerivedClassName(Base1, Base2, Base3):
 <statement-1>
 .
 .
 .
 <statement-N>
```

# A Clock Example

```
class Clock(object):

 def __init__(self, hours=0, minutes=0, seconds=0):
 self.__hours = hours
 self.__minutes = minutes
 self.__seconds = seconds

 def set(self, hours, minutes, seconds=0):
 self.__hours = hours
 self.__minutes = minutes
 self.__seconds = seconds

 def tick(self):
 """ Time will be advanced by one second """
 if self.__seconds == 59:
 self.__seconds = 0
 if (self.__minutes == 59):
 self.__minutes = 0
 self.__hours = 0 if self.__hours==23 else self.__hours+1
 else:
 self.__minutes += 1;
 else:
 self.__seconds += 1;

 def display(self):
 print("%d:%d:%d" % (self.__hours, self.__minutes, self.__seconds))

 def __str__(self):
 return "%2d:%2d:%2d" % (self.__hours, self.__minutes,
self.__seconds)
```

# The Calendar Class

---

```
class Calendar(object):
 months = (31,28,31,30,31,30,31,31,30,31,30,31)

 def __init__(self, day=1, month=1, year=1900):
 self.__day = day
 self.__month = month
 self.__year = year

 def leapyear(self,y):
 if y % 4:
 # not a leap year
 return 0;
 else:
 if y % 100:
 return 1;
 else:
 if y % 400:
 return 0
 else:
```

...

# *A Clock Calendar Class*

---

```
from clock import Clock
from calendar import Calendar

class CalendarClock(Clock, Calendar):

 def __init__(self, day, month, year, hours=0, minutes=0, seconds=0):
 Calendar.__init__(self, day, month, year)
 Clock.__init__(self, hours, minutes, seconds)

 def __str__(self):
 return Calendar.__str__(self) + ", " + Clock.__str__(self)

if __name__ == "__main__":
 x = CalendarClock(24, 12, 57)
 print(x)
 for i in range(1000):
 x.tick()
 for i in range(1000):
 x.advance()
 print(x)
```

# *Private Variables*

---

- # δεν υπάρχουν
- # υπάρχει η σύμβαση οτιδήποτε ξεκινάει από κάτω παύλα, είτε είναι μέθοδος είτε δεδομένα να το χρησιμοποιούμε σαν μη δημόσιο

# Records - Structs

---

```
class Employee:
 pass

john = Employee() # Create an empty employee record

Fill the fields of the record
john.name = 'John Doe'
john.dept = 'computer lab'
john.salary = 1000
```

# Operating System Interface

---

```
>>> import os
>>> os.getcwd() # Return the current working directory
'C:\\Python26'
>>> os.chdir('/server/accesslogs') # Change current working directory
>>> os.system('mkdir today') # Run the command mkdir in the system shell
0
```

```
>>> import shutil
>>> shutil.copyfile('data.db', 'archive.db')
>>> shutil.move('/build/executables', 'installdir')
```

# *Operating System Interface*

---

```
>>> import glob
>>> glob.glob('*.py')
['primes.py', 'random.py', 'quote.py']
```

```
>>> import sys
>>> print sys.argv
['demo.py', 'one', 'two', 'three']
```

# *Data Compression*

---

```
>>> import zlib
>>> s = 'witch which has which witches wrist watch'
>>> len(s)
41
>>> t = zlib.compress(s)
>>> len(t)
37
>>> zlib.decompress(t)
'witch which has which witches wrist watch'
```

# Modules

---

```
def fib(n):
 if n == 0:
 return 0
 elif n == 1:
 return 1
 else:
 return fib(n-1) + fib(n-2)
def ifib(n):
 a, b = 0, 1
 for i in range(n):
 a, b = b, a + b
 return a
```

```
>>> import fibonacci
>>> fibonacci.fib(30)
832040
>>> fibonacci.ifib(100)
354224848179261915075L
>>> fibonacci.ifib(1000)
4346655768693745643568852767504062
6347752096896232398733224711616429
>>>
```

...

# Modules

---

```
>>> fib = fibonacci.fib
>>> fib(10)
55
>>>
```

δίνουμε ένα μικρότερο,  
πιο βολικό όνομα

# Modules

---

```
>>> import fibonacci
>>> fibonacci.fib(30)
832040
>>> fibonacci.ifib(100)
354224848179261915075L
>>> fibonacci.ifib(1000)
4346655768693745643568852767504062
6347752096896232398733224711616429
>>>
```



```
>>> from fibonacci import fib, ifib
>>> ifib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

```
>>> from fibonacci import *
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

# *Modules as Scripts*

---

```
python fibo.py
```

το εκτελούμε σαν script

```
if __name__ == "__main__":
 import sys
 fib(int(sys.argv[1]))
```

αν έχει κάποιο main ...

```
$ python fibo.py 50
1 1 2 3 5 8 13 21 34
```

τρέχει

```
>>> import fibo
>>>
```

αλλιώς απλά γίνεται import

# Renaming

---

```
>>> import math as mathematics
>>> print mathematics.cos(mathematics.pi)
-1.0
```

```
>>> from math import pi, pow as power, sin as sinus
>>> power(2,3)
8.0
>>> sinus(pi)
1.2246467991473532e-16
```

# *More lists*

---

Από Κελσίου σε Φαρενάιτ

```
>>> Celsius = [39.2, 36.5, 37.3, 37.8]
>>> Fahrenheit = [((float(9)/5)*x + 32) for x in Celsius]
>>> print Fahrenheit
[102.56, 97.700000000000003, 99.140000000000001, 100.03999999999999]
>>>
```

## *More lists*

---

### Τριπλέτες του Πυθαγόρα

```
>>> [(x,y,z) for x in range(1,30) for y in range(x,30) for z in range(y,30)
if x**2 + y**2 == z**2]
[(3, 4, 5), (5, 12, 13), (6, 8, 10), (7, 24, 25), (8, 15, 17), (9, 12, 15),
(10, 24, 26), (12, 16, 20), (15, 20, 25), (20, 21, 29)]
>>>
```

# More lists

---

συνδυασμός από δύο σύνολα

```
>>> colours = ["red", "green", "yellow", "blue"]
>>> things = ["house", "car", "tree"]
>>> coloured_things = [(x,y) for x in colours for y in things]
>>> print coloured_things
[('red', 'house'), ('red', 'car'), ('red', 'tree'), ('green', 'house'),
('green', 'car'), ('green', 'tree'), ('yellow', 'house'), ('yellow',
'car'), ('yellow', 'tree'), ('blue', 'house'), ('blue', 'car'), ('blue',
'tree')]
>>>
```

## More lists

---

πρώτοι αριθμοί

```
>>> noprimess = [j for i in range(2, 8) for j in range(i*2, 100, i)]
>>> primes = [x for x in range(2, 100) if x not in noprimess]
>>> print primes
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97]
>>>
```

```
>>> from math import sqrt
>>> n = 100
>>> sqrt_n = int(sqrt(n))
>>> no_primes = [j for i in range(2, sqrt_n) for j in range(i*2, n, i)]
```

---

**Ευχαριστώ**

---