

week 1.

Εισαγωγή – κλάσεις και αντικείμενα

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Την πρώτη εβδομάδα κάνουμε μια σύντομη ανασκόπηση στην γλώσσα Python, εισάγουμε τα εργαλεία του μαθήματος και κάνουμε πρώτη εισαγωγή στις έννοιες του αντικειμενοστραφούς μοντέλου προγραμματισμού

L1.1 Εισαγωγή στην Python

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Στην πρώτη διάλεξη θα κάνουμε μια γρήγορη επανάληψη στη γλώσσα προγραμματισμού Python, εστιάζοντας στις κύριες δομές δεδομένων και δομές ελέγχου της γλώσσας

w1. Εισαγωγή – κλάσεις και αντικείμενα


L1.1 Εισαγωγή στην Python

L1.2 Python και εργαλεία προγραμματισμού

L1.3 Κλάσεις – αντικείμενα

Μάθημα L1.1

Εισαγωγή στην Python

Προγραμματίζω
με την python 

- V1.1.1 Η γλώσσα - σύνταξη
- V1.1.2 Τύποι δεδομένων
- V1.1.3 Έλεγχος ροής προγράμματος
- V1.1.4 Άσκηση επανάληψης


V1.1.1

Η γλώσσα — εισαγωγή

Νίκος Αβούρης, Πανεπιστήμιο Πατρών



python - ιστορία


Προγραμματίζω
με την python 

- Δημιουργήθηκε το 1989 από τον **Guido Van Rossum** (Ολλανδός, σπούδασε Μαθηματικά και Υπολογιστές, Παν. Άμστερνταμ, Google, Dropbox)
- Python 1.0 → 1994, Python 2.0 → 2000, Python 3.0 → 2008
- σήμερα Python 3.6 (προτεινόμενη έκδοση)

python.org



Η γλώσσα Python ...

Προγραμματίζω
με την python 

- Είναι **αντικειμενοστρεφής** αλλά υποστηρίζει και άλλα στυλ προγραμματισμού
- Είναι διερμηνευόμενη (interpreted)
- Είναι αυστηρή και δυναμική ως προς το σύστημα τύπων δεδομένων (strongly typed - dynamic typing) – δεν απαιτείται δήλωση τύπου μεταβλητών όπως στη Java
- Έχει πολλές χρήσεις (Web, GUI, Scripting, κλπ.)
- Δίνει έμφαση στην παραγωγικότητα και αναγνωσιμότητα

Η γλώσσα python ...

- Περιλαμβάνει διαδραστικό περιβάλλον διερμηνευτή (IDLE)
- Περιλαμβάνει πολλές βιβλιοθήκες (modules)
- Όλα στην Python είναι αντικείμενα, ταυτότητα: **id**
- Ισχυρή ενδοσκόπηση (introspection)
- Υλοποίηση σε όλα τα λειτουργικά (και κινητά)
- CPython, Jython, IronPython, PyPy

χρήσιμες συναρτήσεις

- `id(object)`
- `dir(object)`
- `help(object)`
- σύνταξη

χρήση : και **στοίχιση** για ορισμό ομάδας
εντολών αντί για καλλιγραφικές παρενθέσεις
{ ... }

Στοιχείση!!!

- Στις περισσότερες γλώσσες είναι προαιρετική
- Για τους ανθρώπους πολύ σημαντική (για την python το ίδιο)
- Ομαδοποιούμε και στοιχίζουμε όμοια πράγματα μαζί

σχόλια

```
>>> # σχόλιο μιας γραμμής  
>>> ''' κάθε συμβολοσειρά που δεν εκχωρείται  
σε μεταβλητή θεωρείται σχόλιο. '''
```

V1.1.2


Τύποι δεδομένων

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Όλα είναι αντικείμενα


- Τροποποιήσιμα (list, dictionary)
 - Μη τροποποιήσιμα (string, integer, tuple)
- όλα έχουν id και τιμή

αριθμοί

Προγραμματίζω
με την python 

```
>>> year = 2018
>>> type(year)
<class 'int'>
>>> pi = 3.14159
>>> type(pi)
<class 'float'>
>>> x = 5 + 2j
>>> type(x)
<class 'complex'>
```


δεν υπάρχει μέγιστος αριθμός

Προγραμματίζω
με την python 

```
>>> import sys
>>> sys.maxsize
9223372036854775807
>>> sys.maxsize+1
9223372036854775808
|
```

ο μέγιστος αριθμός με βάση την
αρχιτεκτονική του υπολογιστή πχ
για υπολογιστή 64bit $2^{63} - 1$


ακρίβεια αριθμών

Προγραμματίζω
με την python 

```
>>> q = 0.3
>>> '{:.25f}'.format(q)
'0.2999999999999999999999999999888977698'
>>> print(0.1*3==0.3)
False
```

η κωδικοποίηση IEEE-754 double-precision floating-point έχει
ακρίβεια 15 ψηφίων

Η βιβλιοθήκη **decimal**


Προγραμματίζω
με την python 

για μεγαλύτερη ακρίβεια χρησιμοποιούμε την
κωδικοποίηση Decimal της βιβλιοθήκης decimal

```
>>>import decimal  
>>>decimal.getcontext().prec=100  
>>>two = decimal.Decimal("2")  
>>>print(two**decimal.Decimal('0.5'))
```

```
1.4142135623730950488016887242096980785696718753  
769480731766797379907324784621070388503875343276  
41573
```


Συμβολοσειρές (strings)

Προγραμματίζω
με την python 

```
>>> # 4 διαφορετικοί τρόποι να ορίσουμε συμβολοσειρά:
>>> s1 = 'κάτω στους πόρα κάμπους'
>>> s2 = "κάτω στους πόρα ..."
>>> s3 = ''' συμβολοσειρά πολλών γραμμών:
κάτω στους πόρα
κάμπους που είναι οι ελιές ...'''
>>> s4 = """ το ίδιο με διπλά εισαγωγικά...
κάτω στους πόρα κλπ"""
>>> s3
' συμβολοσειρά πολλών γραμμών:\nκάτω στους πόρα\nκάμπους που είναι
οι ελιές ...'
```

χαρακτήρες διαφυγής String \n, \t, \', \"

String format

Προγραμματίζω
με την python 

<https://pyformat.info/#>

```
>>> "%s %s" % ('hello', 'world')  
'hello world'
```

```
>>> "{:s} {:s}".format('hello', 'world')  
'hello world'
```

{:5d} αριθμοί (ψηφία)

{:1.2f} πραγματικοί αριθμοί (width.precision)

{:10s} συμβολοσειρές (χαρακτήρες)

μέθοδοι τύπου string

s.replace (old, new [, max]) # αντικαθιστά το old με new
s.count (str) # μετράει πόσες φορές υπάρχει το str στο s
s.isalpha() # True αν το s περιέχει μόνο χαρακτήρες
s.isdigit () # True αν το s περιέχει μόνο αριθμούς
s.islower () # True αν το s περιέχει μόνο πεζά γράμματα
s.upper () # Μετατρέπει τα πεζά σε ΚΕΦΑΛΑΙΑ
s.lower () # Μετατρέπει τα ΚΕΦΑΛΑΙΑ σε πεζά
s.capitalize () # Πρώτος χαρακτήρας κεφαλαίο.

μέθοδοι τύπου string


`s.find(str)` # θέση του str στο string s, -1 αν δεν βρεθεί
`s.join(seq)` # συνενώνει τα στοιχεία του seq με το s σαν σύνδεσμο
`s.split(δ)` # διαχωρίζει τα στοιχεία του s με διαχωριστικό δ
`s.strip([chars])` # διώχνει τους chars αν βρίσκονται στην αρχή και το τέλος του s (επίσης `rstrip`, `lstrip` για δεξιό ή αριστερό άκρο)
`s.endswith(str)` # True αν τερματίζει με str
`s.decode(encoding='UTF-8')` # μετατρέπει byte string σε string
`s.encode(encoding='UTF-8')` # # μετατρέπει string σε byte string
`s.format(param)`

None

Όταν ορίζουμε μια μεταβλητή, της οποίας δεν γνωρίζουμε ακόμη την τιμή, δίνουμε την τιμή **None**. Την τιμή αυτή επιστρέφουν και συναρτήσεις που δεν επιστρέφουν συγκεκριμένη τιμή.

```
>>> my_data = None
>>> type(my_data)
<class 'NoneType'>
```

λίστες


Προγραμματίζω
με την python 

```
>>> # λίστες, ο πιο σημαντικός τύπος δεδομένων της python
>>> my_list = []
>>> my_list.append(10)
>>> my_list.append('python')
>>> my_list.extend([3+4j, 3.14])
>>> my_list
[10, 'python', (3+4j), 3.14]
```

τμήμα λίστας

λίστα[από : μέχρι : βήμα]

```
>>> my_list  
[10, 'python', (3+4j), 3.14]  
>>> len(my_list)  
4  
>>> my_list[0]  
10  
>>> my_list[1:3]  
['python', (3+4j)]
```


Προγραμματίζω
με την python 

Άσκηση:

τι επιστρέφει η b;


```
b = my_list[::-1]
```


μέθοδοι λιστών

Προγραμματίζω
με την python 


append(x) προσθήκη στοιχείου στο τέλος `a[len(a):] = [x]`.
extend(L) επέκταση λίστας με τα στοιχεία της L `a[len(a):] = L`.
insert(i, x) εισαγωγή του x στη θέση i
remove(x) διαγραφή της πρώτης εμφάνισης του x στη λίστα, `error` αν δεν υπάρχει το x
pop([I]) διαγραφή του στοιχείου στη θέση I , `pop()` διαγράφει το τελευταίο στοιχείο
index(x) η θέση του στοιχείου x, `error` αν δεν υπάρχει το x
count(x) πόσες φορές εμφανίζεται το x στη λίστα
sort() ταξινόμηση των στοιχείων της λίστας αλλάζοντας την
reverse() αντίστροφη ταξινόμηση των στοιχείων, αλλάζοντας την

Πράξεις σε ακολουθίες <seq>
(λίστες, συμβολοσειρές)

Προγραμματίζω
με την python 

τελεστής	αποτέλεσμα
<seq> + <seq>	συνένωση
<seq> * <int>	επανάληψη
<seq>[i]	δείκτης
len(<seq>)	μήκος ακολουθίας
<seq>[:]	τεμαχισμός
for <var> in <seq>:	επανάληψη
<expr> in <seq>	συμμετοχή (Boolean)

Αντίγραφο λίστας

Προγραμματίζω
με την python 


```
>>> a = [1,2,3]
>>> b = a
>>> a[0]=8
>>> b
[8, 2, 3]
```

```
>>> a = [1,2,3]
>>> b = a[:]
>>> a[0]=8
>>> b
[1, 2, 3]
```

εξηγήστε τα 3 παραδείγματα

```
>>> a = [1,2,3]
>>> b = a.copy()
>>> a[0]=8
>>> b
[1, 2, 3]
```

Λεξικά Dictionaries

Προγραμματίζω
με την python 


Πίνακας συσχετιζόμενων στοιχείων (associative array) μοιάζει με το object της javascript, array της PHP ή τα hashtables της Java, συντακτικά είναι παρόμοιο με τη δομή JSON

Έστω ότι θέλουμε να δημιουργήσουμε μια δομή για τα ενδιαφέροντα σε ειδήσεις ενός χρήστη (areas) για κάθε ενδιαφέρον, μπορεί να οριστούν και λέξεις κλειδιά

```
the_user = {'name': 'maria',  
            'areas': [  
                {'area': 'Ειδήσεις', 'keywords': 'Ερντογάν'},  
                {'area': 'Βόλεϊ', 'keywords': 'ΠΑΟΚ'} ] }
```

Άσκηση, πώς θα ανακτήσουμε τις περιοχές ενδιαφέροντος του χρήστη;

μέθοδοι λεξικών

Προγραμματίζω
με την python 

```
notes={'do': 264, 'do#': 281.6, 're':297, 're#': 316.8, 'mi':330, 'fa':352,  
'fa#':371.25, 'sol':396, 'sol#':422.4, 'la':440, 'la#': 469.33, 'si':495 }
```

```
notes.keys()
```

```
dict_keys(['do', 'do#', 're', 're#', 'mi', 'fa', 'fa#', 'sol', 'sol#', 'la', 'la#', 'si'])
```

```
notes.values()
```


```
dict_values([264, 281.6, 297, 316.8, 330, 352, 371.25, 396, 422.4, 440,  
469.33, 495])
```

```
notes.get('do')
```

```
264
```


```
di.get(key, default_value)
```

del στοιχείο διαγραφή στοιχείου από λίστα ή λεξικό

Προγραμματίζω
με την python 

```
>>> li = [1,2,3]
>>> del li[0]
>>> li
[2, 3]
>>> di = {'a':10, 'b':20}
>>> del di['a']
>>> di
{'b': 20}
```


λογικές μεταβλητές

Προγραμματίζω
με την python 

```
>>> value = None or [] or {} or 0 or 0.0 or () or ' ' or ""  
>>> bool(value)  
False  
>>>
```

όλες οι άλλες τιμές είναι True

help() και dir()

Προγραμματίζω
με την python 

```
>>> help(str.format)
Help on method_descriptor:
```

```
format(...)
    S.format(*args, **kwargs) -> str
```

Return a formatted version of S, using substitutions
from args and kwargs.

The substitutions are identified by braces ('{' and
'}').


```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '
__delitem__', '__dir__', '__doc__', '__eq__', '__format__
```


V1.1.3

Έλεγχος ροής προγράμματος

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

τελεστές

Προγραμματίζω
με την python 

αριθμητικοί

```
a = 10
a += 1
a -= 1
b = a + 1    # 11
c = a - 2    # 8
d = a * 3    # 30
e = a / 2    # 5
f = a % 4    # 2
g = a ** 2   # 100
h = a // 3   # 3
```


λογικοί

```
a and b
a or b
not a
a and not (b or c)
```

συμβολοσειρών

```
>>> wish = 'καλή'
>>> wish += ' χρονιά'
>>> wish
'καλή χρονιά'
>>> wish = ' '.join(['καλή', 'χρονιά'])
>>> wish
'καλή χρονιά'
```

Εκχώρηση πολλαπλών μεταβλητών

Προγραμματίζω
με την python 

Είναι δυνατόν να έχουμε πολλαπλές μεταβλητές στο αριστερό σκέλος μιας εντολής εκχώρησης.

```
>>> x = 10
```

```
>>> a, b, c = x, x**2, x**3
```

Όλες οι εκφράσεις του δεξιού σκέλους υπολογίζονται πρώτα πλήρως και μετά τα αποτελέσματα εκχωρούνται στις μεταβλητές του αριστερού σκέλους. Συνεπώς η αντιμετάθεση τιμών 2 μεταβλητών μπορεί να γίνει ως εξής:

```
a, b = b, a
```

συγκρίσεις


`a == b`

`a != b`

`a > b`


`a >= b`

```
>>> 1 is 1
True
>>> 1 is True
False
>>> 1 == True
True
```

Προγραμματίζω
με την python 

```
>>> a = [1,2,3]
>>> b = a
>>> a is b
True
>>> a = 5
>>> b = 5
>>> a is b
True
>>> a = [1,2,3]
>>> b = [1,2,3]
>>> a is b
False
```

συνθήκη **if-elif-else**


Προγραμματίζω
με την python 

```
if grade >= 5 :  
    result = 'pass'  
else:  
    result = 'fail'
```

σύντομο **if**

```
result = 'pass' if grade >= 5 else 'fail'
```

βρόχος επανάληψης **for** σε
επαναλήψιμα αντικείμενα (iterables)

Προγραμματίζω
με την python 

for item **in** sequence
εντολές...

```
>>> fruta = ['μήλα', 'πορτοκαλια', 'αχλάδια']  
>>> for f in fruta: print(f, end = ' ')
```

```
μήλα πορτοκαλια αχλάδια  
|
```

for σε λεξικό

```
>>> codes = {'Πάτρα': 261, 'Ηράκλειο': 281, 'Θεσσαλονίκη': 231}  
>>> for city, num in codes.items(): print(city, ': ', num)
```

```
Πάτρα : 261  
Ηράκλειο : 281  
Θεσσαλονίκη : 231
```


```
>>> for c in sorted(codes):  
    print(c, codes.get(c))
```

```
Ηράκλειο 281  
Θεσσαλονίκη 231  
Πάτρα 261
```

```
>>> for c in sorted(codes, key= codes.get):  
    print(c, codes.get(c))
```


```
Θεσσαλονίκη 231  
Πάτρα 261  
Ηράκλειο 281
```

βρόχος **while**

Προγραμματίζω
με την python 

```
while συνθήκη :  
    μπλοκ-εντολών-1  
    if συνθήκη :  
        continue # πήγαινε στην αρχή  
    if συνθήκη :  
        break # βγες από το βρόχο  
else:  
    μπλοκ-εντολών        # αν τέλειωσε χωρίς break
```


βρόχος **while**

Προγραμματίζω
με την python 


```
>>> x = 0
>>> while x < 20:
    print(x, end = ' ')
    x += 1
```

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Άσκηση: να υλοποιηθεί με βρόχο for

συνοπτικές λίστες

list comprehension

Προγραμματίζω
με την python 

συντόμευση βρόχου **for**

να βρείτε τους
περιττούς
αριθμούς ως το 20


```
>>> peritoi=[]  
>>> for x in range(20):  
        if x%2: peritoi.append(x)
```

```
>>> print(peritoi)  
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

συνοπτικά:

```
>>> [x for x in range(20) if x%2]  
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

βρόχος επανάληψης **for** σε αρχείο χαρακτήρων

Προγραμματίζω
με την python 


```
for line in open('my_file.txt', 'r') :  
    # εντολές
```

διαχείριση εξαιρέσεων

```
try:
    εντολές # εδώ ελέγχεται το σφάλμα
    ...
except <τύπος σφάλματος -1> :
    εντολές
except <τύπος σφάλματος -2> :
    εντολές
else:
    εντολές αν δεν υπάρχει εξαίρεση
finally :
    εντολές που εκτελούνται σε κάθε περίπτωση
```

διαχείριση εξαιρέσεων:

try/except/else/finally

Προγραμματίζω
με την python 


```
import datetime
import random
day = random.choice(['Εικοστή πέμπτη', 25])
try:
    date = day + ' Μαρτίου'
except TypeError:
    date = datetime.date(1821, 3, day)
else:
    date += ' 1821'
finally:
    print(date)
```

V1.1.4

Άσκηση επανάλληψης

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Άσκηση επανάληψης

Προγραμματίζω
με την python 

Να γράψετε πρόγραμμα python που βρίσκει το
πλήθος εμφάνισης των γραμμάτων σε κείμενο
που βρίσκεται σε αρχείο που δίνει ο χρήστης, πχ
 $\alpha: 100$, $\beta: 20$
κλπ.

L1.2 Python και εργαλεία προγραμματισμού

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Στην δεύτερη διάλεξη θα ολοκληρώσουμε την επανάληψη στη γλώσσα προγραμματισμού Python εστιάζοντας σε συναρτήσεις και θα δούμε το περιβάλλον προγραμματισμού pycharm που θα χρησιμοποιήσουμε στο μάθημα.


w1. Εισαγωγή – κλάσεις και αντικείμενα

L1.1 Εισαγωγή στην Python

L1.2 Python και εργαλεία προγραμματισμού

L1.3 Κλάσεις – αντικείμενα

Μάθημα L1.2

Προγραμματίζω
με την python 

Δομημένη Python και εργαλεία προγραμματισμού


- V1.2.1 Συναρτήσεις και βιβλιοθήκες
- V1.2.2 Εργαλεία – το pycharm

V1.2.1

Συναρτήσεις και βιβλιοθήκες

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

ορισμός συνάρτησης

Προγραμματίζω
με την python 


```
>>> def my_f():  
    ''' it says hi! in Greek'''  
    print('γεια')
```

docstring

```
>>> help(my_f)  
Help on function my_f in module __main__:
```

```
my_f()  
    it says hi! in Greek
```

παράμετροι συνάρτησης


Προγραμματίζω
με την python 

```
>>> # υποχρεωτικές παράμετροι  
>>> def add(x,y):  
    return x+y
```

```
>>> # προαιρετικές με keyword  
>>> def say(phrase='γεια σας'):  
    print(phrase)
```

```
>>> add(5,8)  
13  
>>> say()  
γεια σας  
|
```

παράμετροι απροσδιόριστου πλήθους

Προγραμματίζω
με την python 

```
>>> def func(*args, **kwargs):  
    for a in args: print(a)  
    for k,v in kwargs.items(): print(k,v)
```


```
>>> func(1,2,3, x=10, y =20)
```

```
1  
2  
3  
x 10  
y 20  
,
```

import βιβλιοθήκη

- Με τις βιβλιοθήκες (modules) γίνεται επαναχρησιμοποίηση κώδικα
- Παρέχουν πρόσβαση σε μεταβλητές κλάσεις συναρτήσεις στο ίδιο χώρο διευθύνσεων (namespace)

import παραδείγματα


Προγραμματίζω
με την python 

βρες την ημερομηνία σε 20 μέρες από σήμερα

```
>>> import datetime
>>> print(datetime.date.today()+datetime.timedelta(20))
2018-02-07
```

```
>>> from datetime import date, timedelta
>>> print(date.today()+timedelta(20))
2018-02-07
```


Package Management **pip**

Προγραμματίζω
με την python 

εργαλείο για εγκατάσταση πρόσθετων modules
εγκαθίσταται πλέον με τη γλώσσα
μπορεί να το βρείτε ως `pip3` `pip3.6` κλπ

```
pip install django
```


επίσης είναι δυνατή η εγκατάσταση βιβλιοθηκών από το `pycharm`

V1.2.2

Εργαλεία - pycharm

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

integrated development environments (ide)

Προγραμματίζω
με την python 

<https://wiki.python.org/moin/PythonEditors>

Για απλά προγράμματα (ως 100 γραμμές) αρκεί το IDLE για πιο σύνθετες εφαρμογές θα βοηθούσε η χρήση ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης ή ενός editor

PyCharm


thonny.org

Komodo

Eclipse (PyDev)

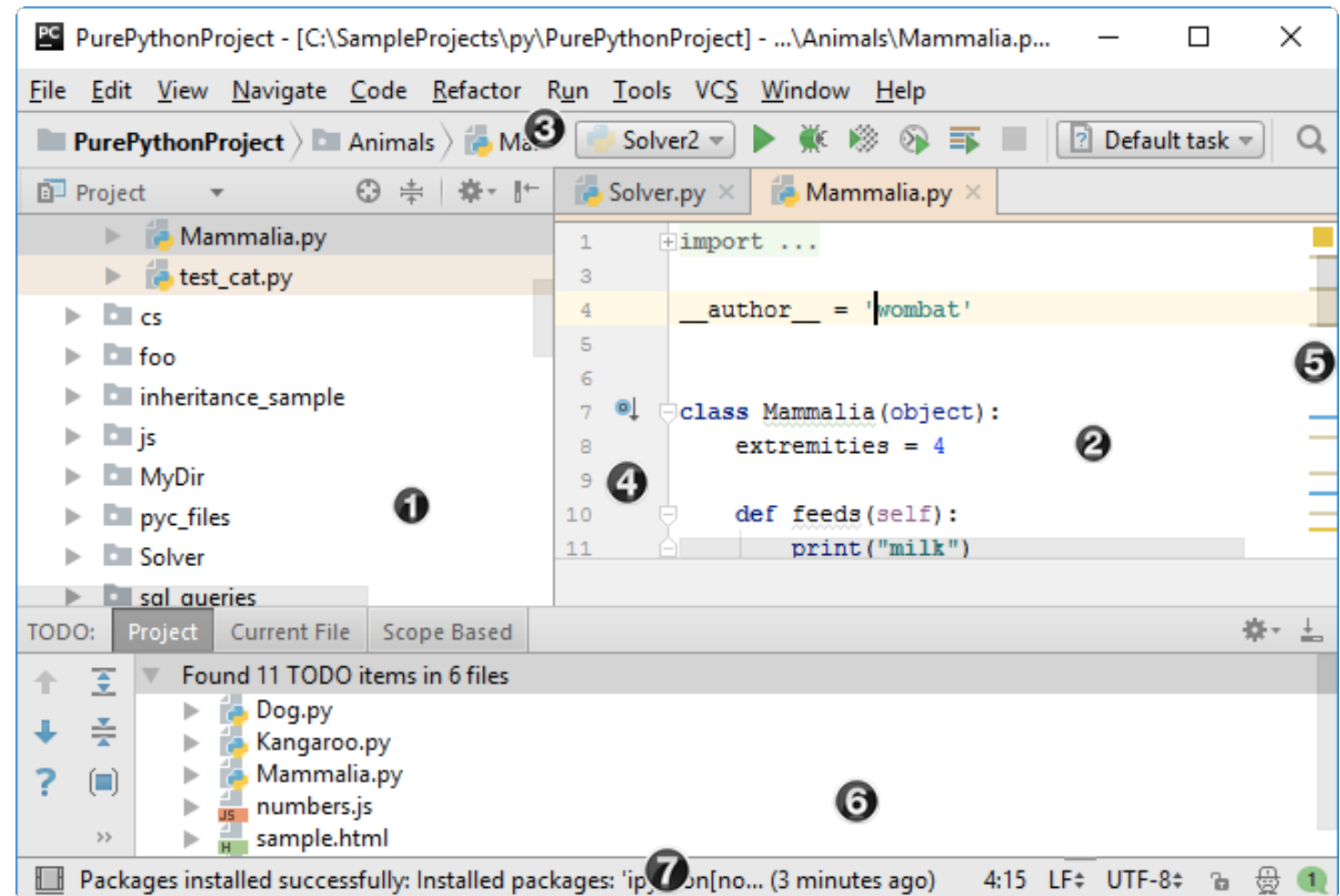
pycharm

<https://www.jetbrains.com/pycharm/>

Προγραμματίζω
με την python 

1. αρχεία του project
2. Επεξεργαστής κώδικα
3. πλοήγηση έλεγχος έκδοσης (πχ git)
4. αριστερό περιθώριο, πλοήγηση στον κώδικα
5. δεξί περιθώριο: σφάλματα
6. χώρος εργαλείων, όπως run, console, terminal
7. status

Μπορούμε να τρέξουμε κώδικα αντιγράφοντας τον στην python console



L1.3 Κλάσεις και αντικείμενα

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Στην Τρίτη διάλεξη θα εισάγουμε την έννοια της κλάσης και του στιγμιότυπου (αντικειμένου), καθώς και μερικές γενικές αρχές αντικειμενοστραφούς προγραμματισμού

w1. Εισαγωγή – κλάσεις και αντικείμενα

week 1


L1.1 Εισαγωγή στην Python

L1.2 Python και εργαλεία προγραμματισμού

L1.3 Κλάσεις – αντικείμενα

Μάθημα L1.3

Κλάσεις και αντικείμενα

Προγραμματίζω
με την python 


- V1.3.1 Κλάσεις και αντικείμενα: Βασικοί ορισμοί
- V1.3.2 Παράδειγμα: Η κλάση Employee
- V1.3.3 Ταξινόμηση αντικειμένων

V1.3.1

Κλάσεις και αντικείμενα: Βασικοί ορισμοί

Νίκος Αβούρης, Πανεπιστήμιο Πατρών


object-oriented programming

Προγραμματίζω
με την python 

Η μέθοδος προγραμματισμού που λέγεται **αντικειμενοστραφής προγραμματισμός** (object-oriented programming) έχει ως κεντρική έννοια την **κλάση** (class) που περιγράφει μια έννοια του προβλήματος που λύνουμε.
Με χρήση της κλάσης παράγονται **αντικείμενα**, ή στιγμιότυπα (instances) της κλάσης.

Οι περισσότερες σύγχρονες γλώσσες όπως η Java και C++ υποστηρίζουν αυτή τη μέθοδο προγραμματισμού.

κλάση – αντικείμενα

Προγραμματίζω
με την python 

κλάση → αντικείμενα (στιγμιότυπα)



Η κλάση **Student**

- name
- age
- origin
- get_age()



s1	s2	s3	s4
'Ορέστης'	'Μαρία'	'Ζωή'	'Κώστας'
22	19	20	21
'Βόλος'	'Σπάρτη'		

```
# ορισμός της κλάσης Student
class Student:
    ''' ένα άτομο που σπουδάζει '''
    def __init__(self, name, age, origin=''):
        self.name = name
        self.age = age
        self.origin = origin
    def get_age(self):
        return self.age
```


```
# ορισμός στιγμιοτύπων της κλάσης Student
s1 = Student('Ορέστης', 22, 'Βόλος')
s2 = Student('Μαρία', 19, 'Σπάρτη')
s3 = Student('Ζωή', 20)
s4 = Student('Κώστας', 21)
```

Βασικές έννοιες

Κλάση (class) : ένας νέος τύπος δεδομένων που ορίζει τη δομή μιας κατηγορίας αντικειμένων. Περιλαμβάνει δεδομένα και τη συμπεριφορά τους (μεθόδους)

Στιγμιότυπα (instances) μιας κλάσης είναι αντικείμενα που δημιουργούνται σύμφωνα με τον ορισμό της κλάσης και έχουν την ίδια δομή με αυτή.

δημιουργός αντικειμένων:

Προγραμματίζω
με την python 

`__init__()`

αναφέρεται στο
αντικείμενο που θα
δημιουργηθεί


`docstring`,
εκχωρείται στο
γνώρισμα `__doc__`

αρχικοποίηση
γνωρισμάτων

```
class Student:
    ''' ένα άτομο που σπουδάζει '''
    def __init__(self, name, age, origin=''):
        self.name = name
        self.age = age
        self.origin = origin
    def get_age(self):
        return self.age
```

τιμές που θα
δοθούν στα
γνωρίσματα


ορισμός μεθόδου

Προγραμματίζω
με την python 

```
class Student:
    ''' ένα άτομο που σπουδάζει '''
    def __init__(self, name, age, origin=''):
        self.name = name
        self.age = age
        self.origin = origin
    def get_age(self):
        return self.age
```

σε όλες τις μεθόδους των
αντικειμένων, όπως και στην
__init__ η πρώτη παράμετρος
είναι το self

δημιουργία και χρήση αντικειμένων

Προγραμματίζω
με την python 

```
s1 = Student(self 'Ορέστης', 22, 'Βόλος')
```

Καλείται η μέθοδος `__init__()` και δημιουργείται ένα αντικείμενο τύπου `Student`. Αυτό ορίζει ένα namespace. Τα γνωρίσματα, μέθοδοι του αντικειμένου καλούνται με τη χρήση **dot notation**


```
print(s1.name)
```

Ορέστης

```
print(s1.get_age())
```

22

Αντικείμενα στην python

Προγραμματίζω
με την python 

- Η python είναι αντικειμενοστραφής γλώσσα χωρίς να επιβάλλει το αντικειμενοστραφές μοντέλο.
- Όλα είναι αντικείμενα πρώτης τάξης (V. Rossum: "One of my goals for Python was to make it so that all objects were *first class*." Έχουμε ήδη συναντήσει τη σημειογραφία τελείας:


πχ

```
my_list.append('z')
```

Η my_list είναι αντικείμενο της κλάσης 'list' που έχει τη μέθοδο append()

```
>>>type(5)  
<class 'int'>
```

δημόσια και ιδιωτικά γνωρίσματα και μέθοδοι

Προγραμματίζω
με την python 

Σε κάποιες γλώσσες προγραμματισμού (java) γίνεται διαχωρισμός ανάμεσα σε γνωρίσματα στα οποία μπορεί μια άλλη κλάση να έχει πρόσβαση (δημόσια) και σε αυτά που δεν μπορεί (ιδιωτικά). Στην python δεν υπάρχει αυτός ο διαχωρισμός.

```
print(s1.age)
```


22

```
print(s1.get_age())
```

22

ιδιωτικά γνωρίσματα

self.__att

Προγραμματίζω
με την python 

Τα γνωρίσματα που αρχίζουν με διπλή κάτω παύλα στην python θεωρούνται ιδιωτικά

```
class My_class():
    def __init__(self):
        self.publ = 'είμαι δημόσιο γνώρισμα'
        self.__priv = 'είμαι ιδιωτικό γνώρισμα'


    def get_priv(self):
        return self.__priv

t = My_class()
print(t.publ)
print(t.get_priv())
print(t.__priv)
```

αποτέλεσμα:

```
είμαι δημόσιο γνώρισμα
είμαι ιδιωτικό γνώρισμα
Traceback (most recent call last): File
"/Users/nma/Desktop/temp1.py", line 12, in
<module>    print(t.__priv)AttributeError:
'Tiny' object has no attribute '__priv'
>>>
```

Κάθε κλάση ορίζει ένα νέο τύπο

Προγραμματίζω
με την python 


Τα αντικείμενα που δημιουργούνται με χρήση της κλάσης είναι αυτού του νέου τύπου

```
>>> type(s1)  
<class '__main__.Student'>
```

Η ίδια η κλάση είναι τύπου **type** (μετα-κλάση)

```
>>> type(Student)  
<class 'type'>
```

Μέθοδοι και γνωρίσματα ενός αντικειμένου


Προγραμματίζω
με την python 

```
>>> dir(s1)
['__class__', '__delattr__', '__dict__', '__dir__',
 '__doc__', '__eq__', '__format__', '__ge__',
 '__getattr__', '__gt__', '__hash__', '__init__',
 '__init_subclass__', '__le__', '__lt__', '__module__',
 '__ne__', '__new__', '__reduce__', '__reduce_ex__',
 '__repr__', '__setattr__', '__sizeof__', '__str__',
 '__subclasshook__', '__weakref__', 'age', 'get_age',
 'name', 'origin']
```

Κάποια γνωρίσματα και μέθοδοι
έχουν κληρονομηθεί από την κλάση


object

αρχίζουν και τελειώνουν με __

Προγραμματίζω
με την python 

```
>>> s1.__dict__
{'name': 'Ορέστης', 'age': 22, 'origin': 'Βόλος'}
>>> s1.__doc__
' ένα άτομο που σπουδάζει '
>>> s1.__class__
<class '__main__.Student'>
>>> s1.__module__
'__main__'
```

Άσκηση

Προγραμματίζω
με την python 


Να γράψετε τη μικρότερη δυνατή κλάση της python, έστω την κλάση Tiny
Στη συνέχεια να βρείτε τα γνωρίσματά της και τις τιμές τους.

V1.3.2

Παράδειγμα: Η κλάση **Employee**

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

παράδειγμα 1

Προγραμματίζω
με την python 

Να δημιουργήσετε μια κλάση Employee που αφορά τους εργαζόμενους μιας επιχείρησης. Για κάθε εργαζόμενο γνωρίζουμε το όνομα και το μισθό του.

Να δημιουργήσετε μια εφαρμογή που ζητάει διαδοχικά τα στοιχεία εργαζομένων από το χρήστη και τα αποθηκεύει σε μια λίστα αντικειμένων τύπου Employee. Όταν ο χρήστης δώσει <enter> το πρόγραμμα σταματάει να ζητάει στοιχεία και τυπώνει τα στοιχεία των εργαζομένων που έχουν ήδη δοθεί.

παράδειγμα 1: η κλάση Employee

Προγραμματίζω
με την python



```
# employee example
```

```
class Employee():
```

```
    ''' Ο εργαζόμενος σε μια επιχείρηση '''
```

```
    def __init__(self, name, salary):
```

```
        self.name = name
```

```
        self.salary = salary
```


V1.3.3

Ταξινόμηση αντικειμένων


Νίκος Αβούρης, Πανεπιστήμιο Πατρών

πώς ταξινομούμε μια λίστα
αντικειμένων `my_list`;

`my_list = sorted(my_list, key = myfunc)`
όπου η συνάρτηση `myfunc` επιστρέφει το
γνώρισμα ταξινόμησης

```
def myfunc(emp):  
    """δέχεται όρισμα αντικείμενο τύπου Employee"""  
    return emp.name
```

Εναλλακτική λύση: ανώνυμη συνάρτηση


Προγραμματίζω
με την python 

Η python επιτρέπει τη δημιουργία **ανώνυμων συναρτήσεων** στο πλαίσιο μιας εντολής με τη λέξη κλειδί **lambda** Η σύνταξη είναι:

lambda όρισμα : έκφραση που επιστρέφει αποτέλεσμα

```
my_list = sorted(my_list, key = lambda x: x.name)
```

άσκηση : ταξινόμηση ενός λεξικού με ανώνυμη συνάρτηση


Προγραμματίζω
με την python 

Με αντίστοιχο τρόπο που είδαμε την ταξινόμηση αντικειμένων μπορεί να γίνει ταξινόμηση στοιχείων ενός λεξικού, όπως το παρακάτω:

```
dd = {1: {'a': 8, 'b': 10}, 8: {'a': 2, 'b': 6},  
3: {'a': 7, 'b': 1}}
```

Ζητείται να ταξινομηθεί ως προς το κλειδί 'b'

παράδειγμα

Προγραμματίζω
με την python 

Στην εφαρμογή της κλάσης Employee, όταν ο χρήστης της εφαρμογής των εργαζομένων της επιχείρησης δώσει <enter> το πρόγραμμα να τυπώνει τα στοιχεία των εργαζομένων που έχουν ήδη δοθεί με αλφαβητική σειρά.