Εισαγωγή στον Προγραμματισμό

Μέρος 9ο: Εισαγωγή στην Python

Εξάμηνο Σπουδών: 3ο

Κωδικός Μαθήματος: 343

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Μιχάλης Α. Μπέκος
bekos@uoi.gr

Μέρος 1°:
Εισαγωγή

Επισκόπηση

- Σύνοψη θεμάτων που θα καλύψουμε:
  - Ιστορικό
  - Συντακτικό
  - Τύποι / Τελεστές / Ροή ελέγχου
  - Συναρτήσεις
  - Κλάσεις
  - Εργαλεία

Τι είναι η Python;

- Μια ευανάγνωστη, δυναμική, ευχάριστη, ευέλικτη και ισχυρή γλώσσα.
- Εξαιρετικά δημοφιλής (Google, PBS, NASA, Library of Congress, the ONION...)
- Πολλαπλών εφαρμογών (Web, GUI, Scripting, Scientific, Numeric κ.λπ.)
- Αντικειμενοστραφής (τα πάντα είναι ένα αντικείμενο)
- Εστιασμένη στην αναγνωσιμότητα και την παραγωγικότητα
- Ανεξάρτητη πλατφόρμας (Cross Platform)
- Δεν έχει μεταγλωττιστή, αλλά διερμηνέα (μετάφραση και εκτέλεση εντολή προς εντολή)
- CPython, Jython, IronPython, PyPy

Εκδόσεις

- Η Python δημιουργήθηκε το 1989 από τον Guido Van Rossum
- Η Python 1.0 κυκλοφόρησε το 1994
- Η Python 2.0 κυκλοφόρησε το 2000
- Η Python 3.0 κυκλοφόρησε το 2008
- Οι διαφορές μεταξύ των εκδόσεων 2.x και 3.x (τουλάχιστον στο επίπεδο αυτού του μαθήματος) δεν είναι πολύ μεγάλες.
- Δεν είναι όμως συμβατές μεταξύ τους.

Python με το Visual Studio Code

- Το Visual Studio Code το έχουμε ήδη χρησιμοποιήσει για την ανάπτυξη προγραμμάτων C++
- Μπορούμε να το χρησιμοποιήσουμε, ωστόσο, και για την ανάπτυξη προγραμμάτων Python
- Γενικά, υποστηρίζει πολλές γλώσσες προγραμματισμού (C++, Html, PHP, Java, Python).
- Για την μεταγλώττιση και εκτέλεση προγραμμάτων Python θα χρειαστείτε:
  - Να εγκαταστήσετε αρχικά την Python
    https://www.python.org/downloads/
  - Να εγκαταστήσετε το περιβάλλον Visual Studio Code
    https://code.visualstudio.com/
  - Να εγκαταστήσετε τα πρόσθετα “Python extension for Visual Studio Code” και “Code Runner”
    https://code.visualstudio.com/docs/languages/python

Άλλα IDEs + Jupyter Notebook

- PyCharm
  https://www.jetbrains.com/
- Eclipse + PyDev
  https://www.pydev.org/
- **Jupyter Notebook:**
  https://jupyter.org/install
  Διαδραστικό περιβάλλον για την εκτέλεση κώδικα σε φυλλομετρητή ιστοσελίδων (browser).
- Google colab:
  https://colab.google/
  Online υπηρεσία Jupyter Notebook που δεν απαιτεί εγκατάσταση.
- Vim
  https://www.vim.org/
- Notepad++
  https://notepad-plus-plus.org/

```
$ pip install jupyterlab
$ jupyter-lab
$ pip install notebook
$ jupyter notebook
```

Μέρος 2°:

Συντακτικό

Hello World

```python
# Display the hello world message
print("Hello World!")
```

Part09/hello_world.py

```
$ python hello_world.py
Hello World!
$ python
Python 3.10.8 (tagsv3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>> quit()
$
```

Εσοχές κώδικα [Indentations]

- Οι περισσότερες γλώσσες προγραμματισμού δεν ενδιαφέρονται για τις εσοχές στον κώδικα
- Οι περισσότεροι άνθρωποι ενδιαφέρονται (για λόγους αναγνωσιμότητας του κώδικα)
- Π.χ. στην C++ τα παρακάτω τμήματα κώδικα είναι ισοδύναμα:

```
# C++ (unformated) code
int x, y;
cin >> x >> y;

if (x!=y)
if (x>y)
cout << "First is greater" << endl;
else
cout << "Second is greater" << endl;
```

Προσοχή: Η else ανήκει στη 2η εντολή if

```
# C++ (formated) code
int x, y;
cin >> x >> y;
if (x!=y) {
  if (x>y) {
    cout << "First is greater" << endl;
  }
  else {
    cout << "Second is greater" << endl;
  }
}
```

Εδώ είναι πιο ξεκάθαρη η αντιστοίχηση

Εσοχές κώδικα [Indentations]

- Οι περισσότερες γλώσσες προγραμματισμού δεν ενδιαφέρονται για τις εσοχές στον κώδικα
- Οι περισσότεροι άνθρωποι ενδιαφέρονται (για λόγους αναγνωσιμότητας του κώδικα)
- Η Python υιοθετεί τις εσοχές κώδικα στο συντακτικό της:

```python
# Python (indentated) code
x = int(input())
y = int(input())

if x!=y:
  if x>y:
    print("First is greater")
  else:
    print("Second is greater")
```

Part09/indentation.py

Σχόλια

- Τα σχόλια στην Python ξεκινούν με #
- Η Python δεν διαθέτει σύνταξη για σχόλια πολλαπλών γραμμών.
- Για την εισαγωγή ενός πολυγραμμικού σχολίου εισάγετε τη # σε κάθε γραμμή.
- Δεδομένου ότι η Python θα αγνοήσει τις συμβολοσειρές που δεν έχουν εκχωρηθεί σε μια μεταβλητή, μπορείτε να προσθέσετε στον κώδικά σας μια συμβολοσειρά πολλών γραμμών (τριπλά εισαγωγικά) και να τοποθετήσετε το σχόλιό σας μέσα σε αυτήν:

```
#This is a traditional comment

"""
Any string not assigned to a variable,
is considered a comment
"""

"This is a single line comment as well"
```

Part09/comments.py

Μέρος 3°:
Τύποι

Μεταβλητές

- Η Python δεν έχει εντολή για τη δήλωση μιας μεταβλητής.
- Μια μεταβλητή δημιουργείται τη στιγμή που της αναθέτετε για πρώτη φορά μια τιμή.
- Οι μεταβλητές δεν χρειάζεται να δηλωθούν με κάποιον συγκεκριμένο τύπο.
  - Π.χ., μπορούν να αλλάξουν τύπο μετά τον ορισμό τους.
  - Αν θέλετε να καθορίσετε τον τύπο δεδομένων, αυτό μπορεί να γίνει με casting
- Τα ονόματα των μεταβλητών είναι case-sensitive
- Η εκτύπωση τους γίνεται μέσω της συνάρτησης `print()`

```
# The type of a variable can change
a = 4       # a is of type int
a = "Sally" # a is now of type str

# This will create two variables
b = 4
B = "Sally" # B will not overwrite b

# Print the variables
print(a, b, B)

# Specify input with casting
c = str(input())    # c will string
d = int(input())    # d will int
e = float(input())  # e will float

# Assign values to multiple variables
x, y, z = "Python ", "is ", "awesome"
print(x + y + z)

# Assign a value to multiple variables
u = v = w = "Orange"
print(u)
print(v)
print(w)
```

Part09/variables.py

Αριθμητικοί τύποι

- Υπάρχουν τρεις αριθμητικοί τύποι στην Python:
  - `int` ακέραιοι
  - `float` αριθμοί κινητής υποδιαστολής
  - `complex` μιγαδικοί αριθμοί
- Η Python δεν διαθέτει συνάρτηση για την παραγωγή τυχαίων αριθμών.
  - Με μια απλή επέκταση της όμως η παραγωγή τυχαίων αριθμών γίνεται εφικτή.

```
# Define three numbers of different types
x = 1    # int
y = 2.8  # float
z = 1j   # complex

# Convert from int to float:
a = float(x)

# Convert from float to int:
b = int(y)

# Convert from int to complex:
c = complex(x)

# Print the numbers
print(x, y, z)
print(a, b, c)

# Print the type of each variable
print(type(a), type(b), type(c))

# Support for random numbers
import random
print(random.randrange(1, 10))
```

Part09/numbering.py

Συμβολοσειρές

- Οι συμβολοσειρές στην Python περικλείονται είτε σε απλά είτε σε διπλά εισαγωγικά.
  - Το `'hello'` είναι το ίδιο με το `"hello"`.
- Αν είναι πολαπλών γραμμών χρησιμοποιούνται τρία εισαγωγικά.

```python
# Printing a string
print("Hello World!")
print('Hello World!')

# Assign a string to a variable
str = "Hello World!"
print(str)
print(len(str)) #length of str

# Character at position 0
print(str[0])

# Loop through the letters in a string
for x in str:
  print(x)

# Slices
print(str[2:6]) #from 2 to 6 (excluded)
print(str[2:])  #slice from 2 on

# Upper and lower case concatenated
print(str.upper() + " " + str.lower())

# Modifications
print(str.replace("H", "J"))
```

Part09/strings.py

Λίστες

- Αποθηκεύουν ≥ 1 στοιχεία σε μία μεταβλητή.
- Δημιουργούνται με τη χρήση `[]`
- Είναι διατεταγμένες $\Rightarrow$ μπορούν να έχουν στοιχεία με την ίδια τιμή
- Είναι τροποποιήσιμες $\Rightarrow$ υποστηρίζουν προσθήκες, ενημερώσεις, διαγραφές
- Για να αλλάξετε την τιμή ενός συγκεκριμένου στοιχείου, ανατρέξτε στον κατάλληλο δείκτη:
  - Υποστηρίζονται και πολαπλές αλλαγές
- Η παράθεση στοιχείων γίνεται μέσων των μεθόδων `append()`, `extend()` και του τελεστή `+=`.
- Η εισαγωγή νέου στοιχείου, χωρίς αντικατάσταση άλλων, γίνεται με τη μέθοδο `insert()`.

```python
# Lists can be heterogeneous
favorites = []

# Appending
favorites.append(42)

# Extending
favorites.extend(["Python", True])

# Equivalent to
favorites = [42, "Python", True]

# Updates
favorites[1:3] = ["C++", False]
favorites.insert(1, "Henry")
favorites += ["Lab", -1]

print("Length:", len(favorites))
for x in favorites:
  print(x)

# Slices
print(favorites[0])    # 1st
print(favorites[1:3])  # 2rd, 3rd
print(favorites[0:])   # all
```

Part09/lists.py

Σύνολα

- Αποθηκεύουν ≥ 1 στοιχεία σε μία μεταβλητή.
- Δημιουργούνται με τη χρήση {}
- Είναι μη διατεταγμένα $\Rightarrow$ δεν μπορούν να έχουν στοιχεία με την ίδια τιμή
- Είναι μη τροποποιήσιμα $\Rightarrow$ δεν μπορούν να τροποποιηθούν μετά τη δημιουργία τους
  - Προσθήκες και διαγραφές υποστηρίζονται

```python
# Set items can be of any data type
numbers  = {1, 5, 7, 9, 3}
booleans = {True, False, False}

# Print the sets and their lengths
print(numbers, "->", len(numbers))
print(booleans, "->", len(booleans))

# A set can contain different data types
items = {"abc", 34, True, 40, "male"}
items = items.union(numbers)

# Duplicates are not allowed
fruits = {"apple", "banana", "apple"}

# Additions / removals
fruits.add("watermelon")
fruits.remove("banana")

# Loop through the items with a for loop
for fruit in fruits: print(fruit)

# Sets can be appended to lists
mylist = ["orange", "pear"]
mylist += fruits
```

Part09/sets.py

Λεξικά

- Χρησιμοποιούνται για την αποθήκευση ζευγών κλεδί-τιμή.
- Τα λεξικά συντάσσονται με αγκύλες {}
  - `car = { "model": "Ford Mustang", "year": 1964}`
- Τα λεξικά δεν μπορούν να έχουν δύο στοιχεία με το ίδιο κλειδί.
- Η πρόσβαση στην τιμή γίνεται με αναφορά στο κλειδί του μέσα σε αγκύλες [] ή μέσω της `get()`.
- Η πρόσβαση στα κλειδιά, στα τιμές και στα ζεύγη κλειδιών-τιμών γίνεται μέσω των μεθόδων:
  - `keys(), values(), items()`

```python
# Instrantiate a dictionary with { }
car = {}

# Set by key
car['model'] = 'Ford Mustang'
car['year'] = 1964
car['electric'] = False
car['accesories'] = ['mirrors','ashtray']

# Get by key
model = car.get("model")
print(model) #equiv: print(car["model"])

# Updates / removals
car['accesories'] += ['automatic']
car.pop("electric")

# keys / values / items
print(car.keys())
print(car.values())
print(car.items())

# print using a for loop
for x, y in car.items():
  print(x, ": ",  y)
```

Part09/dictionaries.py

Λογικοί τύποι

- Οι λογικοί τύποι παριστούν μία από δύο τιμές:
  - `True`
  - `False`
- Όταν εκτελείτε μια συνθήκη σε μια εντολή if, η Python επιστρέφει `True` ή `False`
- Η συνάρτηση `bool()` επιτρέπει την αποτίμηση οποιαδήποτε έκφρασης σε `True` ή `False`
  - Όλες οι συμβολοσειρές είναι True, εκτός της κενής
  - Όλοι οι αριθμοί είναι True, εκτός του `0`.
  - Όλες οι λίστες, σύνολα και λεξικά είναι True, εκτός των κενών.
- **Σημείωση:** `Null` → `None`
  - Π.χ., `optional_data = None`

```python
# Simple examples of comparisons
print(10 > 9)
print(10 == 9)
print(10 < 9)

# If statement
a, b = 200, 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")

# This is a boolean variable
is_python = True

# Everything can be casted to boolean
is_python = bool("Python")

# All below are equivalent to False
f = bool(0 or "" or {} or [] or None)

# Everything else is equivalent to True
t = bool(1 and "a" and {'b'} or ['c'])

print(f,t)
```

Part09/booleans.py

Μέρος 4°:
Τελεστές

Τελεστές

- Οι τελεστές χρησιμοποιούνται για την εκτέλεση πράξεων σε μεταβλητές και τιμές.

- Η Python υποστηρίζει τους ακόλουθους τελεστές:
  - Αριθμητικοί τελεστές `+, -, *, /, %, **`
  - Τελεστές ανάθεσης `=, +=, -=, /=, %=, **=`
  - Τελεστές σύγκρισης `==, !=, <, >, <=, >=`
  - Λογικοί τελεστές `and, or, not`
  - Τελεστές ταυτότητας `is, is not`
  - Τελεστές συμμετοχής `in, not in`

```
# Arithmetic and assignment operators
a = 10      # 10
a += 1      # 11
a -= 1      # 10

b = a + 1   # 11
c = a - 1   # 9

d = a * 2   # 20
e = a / 2   # 5
f = a % 3   # 1  (modulo)
g = a** 2   # 100 (power)

print(a,b,c,d,e,f,g)

if a == b:
  print("a and b are equal")
else:
  if a < b:
    print ("a is smaller than b")
  else:
    print("b is smaller than a")

if a != b and a !=c and b!= c:
  print ("a,b,c are mutually different")
```

Part09/operators.py

Τελεστές

- Οι τελεστές χρησιμοποιούνται για την εκτέλεση πράξεων σε μεταβλητές και τιμές.

- Η Python υποστηρίζει τους ακόλουθους τελεστές:
  - Αριθμητικοί τελεστές `+, -, *, /, %, **`
  - Τελεστές ανάθεσης `=, +=, -=, /=, %=, **=`
  - Τελεστές σύγκρισης `==, !=, <, >, <=, >=`
  - Λογικοί τελεστές `and, or, not`
  - Τελεστές ταυτότητας `is, is not`
  - Τελεστές συμμετοχής `in, not in`

```
# Identity and Membership Operators
x = ["apple", "pineapple"]
y = ["apple", "pineapple"]
z = x

# True since z is the same object as x
print(x is z)

# False since x is not the same object as
# y, even if they have the same content
print(x is y)

# Likewise this will be True
print(x is not y)

# True since x is contentwise equal to y
# Note the difference between is and ==
print(x == y)

# True since "pineapple" is in the list
print("pineapple" in x)

# True since "banana" is not in the list
print("banana" not in x)
```

Part09/operators.py

Χειρισμός συμβολοσειρών

- Ο χειρισμός συμβολοσειρών μπορεί να γίνει
  - είτε μέσω τελεστών
  - είτε με την κλήση κατάλληλων μεθόδων.

```
# Cats Dogs Rabbits
animals  = "Cats " + "Dogs "
animals += "Rabbits"

# Fred, Wilma
flintstones = ','.join(['Fred', 'Wilma'])

# Sept 11, 2001
date = '%s %d, %d' % ('Sept', 11, 2001)

# Apple Banana Orange
fruits = '%(first)s %(second)s %(third)s'
% {
    'first' : 'Apple',
    'second': 'Banana',
    'third' : 'Orange'
}
print(animals)
print(flintstones)
print(date)
print(fruits)

# Yet another way to format strings
txt = "My name is {}, and I am {}"
print(txt.format('John', 22))
```

Part09/string_manipulate.py

Μέρος 5°:

Έλεγχος Ροής Προγράμματος

If ... Else

- Η Python υποστηρίζει τις συνήθεις λογικές συνθήκες από τα μαθηματικά:
  - Ισότητα: `a == b`
  - Ανισότητα: `a != b`
  - Μεγαλύτερο ή ίσο: `a >= b`
  - Μικρότερο ή ίσο: `a <= b`
  - Γνησίως μεγαλύτερο: `a > b`
  - Γνησίως μικρότερο: `a < b`
- Αυτές οι συνθήκες χρησιμοποιούνται με διάφορους τρόπους, π.χ. σε εντολές if και βρόχους.
- Μια εντολή if χωρίς εσοχή θα προκαλέσει σφάλμα.
- Υποστηρίζονται εντολές if μέσα σε εντολές if (εμφωλευμένες).

```
# Conditionals
grade = 82
if grade >= 90:
  if grade == 100:
    print('A+')
  else:
    print("A")
elif grade >= 80:
  print("B")
elif grade >= 70:
  print("C")
else:
  print("F")

# Short Hand If
a, b = 2, 1
if a > b: print("a is greater than b")

# Short Hand If ... Else
print("a>b") if a > b else print("a<b")

# More than one conditions
if a > 0 and b > 0:
    print("a and b are positive")
```

Part09/if_else.py

For loops

- Με τον βρόχο `for` μπορούμε να εκτελέσουμε ένα σύνολο εντολών, μία φορά για κάθε στοιχείο
  - μιας λίστας, συνόλου ή λεξικού
  - μιας συμβολοσειράς κ.λ.π.
- Δεν απαιτείται ο ορισμός μιας μεταβλητής δείκτη.
- Η εντολή `break` τερματίζει την επαναληπτική διαδικασία.
- Η εντολή `continue` διακόπτει την τρέχουσα επανάληψη και συνεχίζει στην επόμενη.
- Η συνάρτηση `range()` επιστρέφει μια ακολουθία αριθμών, που ξεκινά από το 0, αυξάνει κατά 1, και καταλήγει σε έναν καθορισμένο αριθμό.

```
# A simple iteration over a list
fruits = ["apple", "cherry", "banana"]
for fruit in fruits:
  if fruit == "cherry": #exclude cherry
    continue
  print(fruit)

# Illustration of the range function
for number in range(10):   #0-9
  print(number)

# Loop through the letters in a word
for char in fruits[0]:
  print(char)

# Nested loops
colors = ["red", "blue", "green"]
for x in colors:
  for y in fruits:
    print(x, y)

# Iteration over dictionaries
countries={'GR':'Greece','DE':'Germany'}
for key, value in countries.items():
  print ('%s: %s' % (key, value))
```

Part09/for_loop.py

While loops

- Με τον βρόχο `while` μπορούμε να εκτελέσουμε ένα σύνολο εντολών όσο μια συνθήκη είναι αληθής.
- Η εντολή `break` τερματίζει την επαναληπτική διαδικασία.
- Η εντολή `continue` διακόπτει την τρέχουσα επανάληψη και συνεχίζει στην επόμενη.

```
# A simple iteration over a list
x = 0
while x < 10:
  print(x)
  x = x+1

print()  # An empty line

# While loop with break statement
x = 0
while x < 10:
  print(x)
  if x == 5:
    break
  x += 1

print()  # An empty line

# While loop with continue statement
x = 0
while x < 10:
  x += 1
  if x % 2 == 0:
    continue
  print(x)
```

Part09/while_loop.py

Μέρος 6°:
Συναρτήσεις

Συναρτήσεις

- Μπλοκ κώδικα που εκτελούνται όταν καλούνται.
  - Στην Python μια συνάρτηση ορίζεται με τη λέξη-κλειδί `def`
- Μια συνάρτηση μπορεί να δέχεται δεδομένα ως παραμέτρους (ορίσματα).
  - Εάν δεν γνωρίζετε το πλήθος των ορισμάτων, προσθέστε ένα `*` πριν από το όνομα της παραμέτρου στον ορισμό της συνάρτησης.
  - Υποστηρίζονται προεπιλεγμένες τιμές παραμέτρων
- Μια συνάρτηση μπορεί να επιστρέψει δεδομένα ως αποτέλεσμα.

```python
# Functions are defined with keyword def
def hellow_world():
  print("Hello World (from a function)")

hellow_world()  #function call

# A function with arguments that returns
def make_name(fname, lname):
  return fname + " " + lname

# print(make_name("Albert")) # error
print(make_name("Albert", "Einstein"))

# Function with arbitrary no of arguments
def print_kids(*kids):
  for kid in kids: print(kid)

print_kids("Emil", "Tobias", "Linus")

# Default Parameter Value
def my_country(country = "Greece"):
  print("I am from " + country)

my_country("France") # standard call
my_country()         # default value
```

Part09/functions.py

Παραδείγματα σε C++ και Python

```cpp
long power(long base, long exponent) {
  if (exponent==0) return 1;
  return base * power(base, exponent-1);
}
```

Part02/Recursion/power.cpp

```python
def power(base, exponent):
  if exponent==0: return 1
  return base * power(base, exponent-1)
```

Part09/examples/power.py

```cpp
long factorial(long n) {
  if (n<=1) return 1;
  return n * factorial(n-1);
}
```

Part02/Recursion/factorial.cpp

```python
def factorial(n):
  if n<=1: return 1
  return n * factorial(n-1)
```

Part09/examples/factorial.py

```cpp
long fibonacci(long n) {
  if (n==0) return 0;
  if (n==1) return 1;
  return fibonacci(n-1) + fibonacci(n-2);
}
```

Part02/Recursion/fibonacci.cpp

```python
def fibonacci(n):
  if n==0: return 0
  if n==1: return 1
  return fibonacci(n-1) + fibonacci(n-2)
```

Part09/examples/fibonacci.py

```cpp
bool isPrime(int n, int i) {
    if (n == 1) return false;
    if (n == 2 || i > n/2) return true;
    if (n % i == 0) return false;
    return isPrime(n, i + 1);
}
```

Part02/Recursion/prime_numbers.cpp

```python
def isPrime(n, i=2):
  if n == 1: return False
  if n == 2 or i > n/2: return True
  if n % i == 0: return False
  return isPrime(n, i + 1)
```

Part09/examples/prime_numbers.py

Μέρος 7°:
Κλάσεις

Κλάσεις και αντικείμενα στην Python

- Δεν υπάρχουν πραγματικά ιδιωτικά πεδία/μέθοδοι
- Τα ιδιωτικά μέλη αρχίζουν (αλλά δεν τελειώνουν) με διπλή υπογράμμιση.
- Ειδικές μέθοδοι της κλάσης αρχίζουν και τελειώνουν με διπλή υπογράμμιση.
  - `__init__` Κατασκευαστής
  - `__str__` Επιστροφή συμβολοσειράς
  - `__doc__` Τεκμηρίωση
- self: Η πρώτη παράμετρος κάθε συνάρτησης της κλάσης. Παρέχει πρόσβαση στα μέλη της κλάσης

```python
# Class declaration
class Person:
  """A class to represent a person"""
  def __init__(self, fname, lname, age):
    self.fname = fname
    self.lname = lname
    self.age = age

  def __str__(self):
    """Returns a string represenation"""
    s="%s (%i)" % (self.name(), self.age)
    return s

  def name(self):
    """Returns the name"""
    return self.fname + " " + self.lname


# Create an object
player = Person("Steph", "Curry", 34)
print(player)  # prints details

# Print class/method documentation
print("Person:",Person.__doc__)
print("Name func:",Person.name.__doc__)
```

Part09/classes/Person.py

Κλάσεις και αντικείμενα στην Python

- Για τη δημιουργία μιας κλάσης, χρησιμοποιούμε τη λέξη-κλειδί `class`
  - ```
    class Person:
        """A class to represent a person"""
        ...
    ```
- Κάθε κλάση χρησιμοποιείται για τη δημιουργία αντικειμένων:
  - `player = Person("Steph", "Curry", 34)`
- Η συνάρτηση `__init__()` καλείται αυτόματα κάθε φορά που δημιουργείται ένα νέο αντικείμενο.

```python
# Class declaration
class Person:
  """A class to represent a person"""
  def __init__(self, fname, lname, age):
    self.fname = fname
    self.lname = lname
    self.age = age

  def __str__(self):
    """Returns a string represenation"""
    s="%s (%i)" % (self.name(), self.age)
    return s

  def name(self):
    """Returns the name"""
    return self.fname + " " + self.lname


# Create an object
player = Person("Steph", "Curry", 34)
print(player)  # prints details

# Print class/method documentation
print("Person:",Person.__doc__)
print("Name func:",Person.name.__doc__)
```

Part09/classes/Person.py
```

Κλάσεις και αντικείμενα στην Python

- Τα αντικείμενα διαθέτουν μεθόδους.
- Οι μέθοδοι είναι συναρτήσεις που ανήκουν στο αντικείμενο.
- Η πρώτη παράμετρος (`self`) μιας μεθόδου είναι μια αναφορά στην τρέχουσα κλάση.
  - Παρέχει πρόσβαση στα μέλη της κλάσης.
  - Όταν η μέθοδος καλείται, η παράμετρος αυτή αγνοείται.
  - ```
    player = Person("Steph", "Curry", 34)
    print(player.name())
    ```

```
# Class declaration
class Person:
  """A class to represent a person"""
  def __init__(self, fname, lname, age):
    self.fname = fname
    self.lname = lname
    self.age = age

  def __str__(self):
    """Returns a string represenation"""
    s="%s (%i)" % (self.name(), self.age)
    return s

  def name(self):
    """Returns the name"""
    return self.fname + " " + self.lname


# Create an object
player = Person("Steph", "Curry", 34)
print(player)  # prints details

# Print class/method documentation
print("Person:",Person.__doc__)
print("Name func:",Person.name.__doc__)
```

Part09/classes/Person.py
```

Κλάσεις και αντικείμενα στην Python

- Η συνάρτηση `__str__()` καθορίζει τι επιστρέφεται όταν ένα αντικείμενο μιας κλάσης αναπαρίσταται ως συμβολοσειρά.
  - `player = Person("Steph", "Curry", 34)`
    `print(player)`
- Εάν η συνάρτηση `__str__()` δεν έχει οριστεί, η αναπαράσταση δεν είναι η επιθυμητή

```
# Class declaration
class Person:
  """A class to represent a person"""
  def __init__(self, fname, lname, age):
    self.fname = fname
    self.lname = lname
    self.age = age

  def __str__(self):
    """Returns a string represenation"""
    s="%s (%i)" % (self.name(), self.age)
    return s

  def name(self):
    """Returns the name"""
    return self.fname + " " + self.lname


# Create an object
player = Person("Steph", "Curry", 34)
print(player)  # prints details

# Print class/method documentation
print("Person:",Person.__doc__)
print("Name func:",Person.name.__doc__)
```

Part09/classes/Person.py

Κλάσεις και αντικείμενα στην Python

- Μπορείτε να τροποποιήσετε ένα αντικείμενο, αλλάζοντας την τιμή των πεδίων του.
  - ```
    player = Person("Steph", "Curry", 34)
    player.fname = "Michael"
    player.lname = "Jordan"
    print(player)
    ```
- Μπορείτε να διαγράψετε ιδιότητες αντικειμένων χρησιμοποιώντας τη λέξη-κλειδί `del`
  - ```
    del player.age
    ```
- Μπορείτε ακόμη και να διαγράψετε αντικείμενα που έχετε δημιουργήσει με τη λέξη-κλειδί `del`
  - ```
    del player
    ```

```
# Class declaration
class Person:
  """A class to represent a person"""
  def __init__(self, fname, lname, age):
    self.fname = fname
    self.lname = lname
    self.age = age

  def __str__(self):
    """Returns a string represenation"""
    s="%s (%i)" % (self.name(), self.age)
    return s

  def name(self):
    """Returns the name"""
    return self.fname + " " + self.lname


# Create an object
player = Person("Steph", "Curry", 34)
print(player)  # prints details

# Print class/method documentation
print("Person:",Person.__doc__)
print("Name func:",Person.name.__doc__)
```

Part09/classes/Person.py
```

Μέρος 8°:
Εργαλεία

Modules

- Επεκτείνουν την Python πέρα από τις βασικές εντολές/συναρτήσεις της
  - Όμοια με τις βιβλιοθήκες της C++
- Η συνάρτηση import επιτρέπει τη χρήση κλάσεων και συναρτήσεων από modules

```
# Float input assigned to x
x = float(input("x="));

# Import math
import math

# Use functions of math
print("Floor of x is", math.floor(x))
print("Sqrt of x is", math.sqrt(x))

# Alternative way to import math
import math as m

# We can call math function using m
print("Ceil of x is", m.floor(x))

# Import fractions
import fractions
f1 = fractions.Fraction(3,4)
f2 = fractions.Fraction(1,2)
print(f1, '+' , f2, '=', f1+f2)

# Support for random numbers
import random
print(random.randrange(1, 10))
```

Part09/modules.py

Διαχείρηση πακέτων

- **NumPy:** https://numpy.org/install/
  Πακέτα για επιστημονικούς υπολογισμούς.

```
$ pip install numpy
```

- **SciPy:** https://scipy.org/install/
  Συναρτήσεις για επιστημονικούς υπολογισμούς.

```
$ pip install scipy
```

- **Flask:** https://flask.palletsprojects.com/
  Web framework.

```
$ pip install Flask
```

```python
# Import NumPy with alias as np
import numpy as np

# Create ndarray from a list
arr1 = np.array([10,20,30])
print(arr1)

# Convert back to list
lst = arr1.tolist()
print(lst)

# Create multi dimensional array
arr2 = np.array([[10,20,30],[40,50,60]])
print("Size:", arr2.size)          #6
print("Shape:", arr2.shape)        #(2,3)
print("Dimension: ", arr2.ndim)    #2
print(arr2)

# Array indexing
print(arr2[0,2])                   #30
print(arr2[1,1])                   #50

# Obtain array by reshaping integers
arr3 = np.arange(35).reshape(5,7)
print(arr3)
```

Part09/num_py.py