Everything you need to know about Python

to understand that you don't know anything about it…

Omer Shalev
December 2017

Agenda

- Introduction
- Basic programming
- Useful data structures
- Object oriented programming
- Scientific programming

Motivation (spoiler…)

Agenda

- Introduction
- Basic programming
- Useful data structures
- Object oriented programming
- Scientific programming

Why Python?

- Very fast prototyping
- Cross platform
- Rich libraries and capabilities
- Object oriented
- Interactive (console) but also for big projects
- Today: more performance awareness
- Popularity

The Implementation of Python

- Interpreted language
- Multiple implementations
  - CPython (reference and default)
  - Jpython, IronPython, PyPy, …
- Python version
  - Python2 vs Python3
- Automatic memory management
- Dynamic name resolution

References

- Python.org
  - Installation, official documentation, tutorials, …
- PyPI: the Python Package Index
- Python source code: maintained in GitHub
- Programiz
- Hackerrank
- Google's python class

Exercise #1

Hello World!

Agenda

- Introduction
- Basic programming
- Useful data structures
- Object oriented programming
- Scientific programming

Variables and Operators

- Dynamic types
  - No declaration
  - Variable type can change
- The *type( )* function
- Operators’ behavior changes per type
- Casting
- Basic types: *int*, *float*, *str*, *bool*

Conditions

- Block definition by leading colon and indented text
  - Indentation characters must be consistent!

```
if x == 1:
    if y == 2:
        if z == 3:
            print 'x = 1, y = 2, z = 3'
        print 'x = 1, y <> 2, z = 3'
    print 'x = 1, y <> 2, z <> 3'
print 'end'
```

- Keywords: *if, elif, else*
- No switch case in Python

Functions

- No type declarations (for either arguments or return type)
- Keyword: *def, return*
- Can get multiple arguments
- Can return multiple values

Imports

- Including functions or variables from a different Python module
- Keywords: *import*, *from*, *as*
- Import is relative to the paths in *sys.path*
- Dot (.) operator for multi-level hierarchy

Exercise #2

Basic Programming

Agenda

- Introduction
- Basic programming
- Useful data structures
- Object oriented programming
- Scientific programming

Useful Data Structures

	List	**Tuple**	**String**	**Dictionary**
Notation	l = [elem1, elem2]	t = (elem1, elem2)	s = ‘hi‘ # or “hi“	d = { k1 : v1, k2 : v2 }
Read element	l[index]	t[index]	s[index]	d[key]
Mutable	YES	NO	NO	YES
len function	+	+	+	+
in operator	+	+	+ (also for substrings)	+ (for keys)
Slicing (to:from:step)	+	+	+	-
Negative indexing	+	+	+	-
Important methods	*append* *pop*		*split* *join*	*has_key* *keys* *values* *items*

Loops

- For loop:

```
for item in iterable_object:
    do_something(item)
```

  - Examples of iterable objects: list, string, tuple, dictionary, …

- While loop:

```
while boolean_expression:
   do_something()
```

- *continue*: continues to the next iteration
- *break*: breaks the loop execution

Exercise #3

Useful Data Structures

Agenda

- Introduction
- Basic programming
- Useful data structures
- Object oriented programming
- Scientific programming

Class

- Definition of a new object
  - Includes members and methods
- *self* is Python’s equivalence for *this* (C++)
- All class methods get *self* as a first argument (in their definition)
- __*init*__ (initializer method) is implicitly called upon class instantiation
- No destructors in Python
- No private / protected /public members (“We’re all grown-ups”)
- No function overloading, default values are used instead (or *args, **kwargs)

Inheritance

- Super class is specified in parenthesis following the class name

```
class InheritedClass(BaseClass):
```

- Any method / member can be overridden ("We're all grown-ups")
- Calling a method of the super class:

```
super(ThisClass, self).__init__()
```

- Python's equivalence for virtual methods:

```
def my_method(self):
   raise NotImplementedError()
```

- Polymorphism thanks to Python's late binding
- Note: base classes inherit from *object*

Exceptions

```
try:
    pass # code that might throw an exception
except ExceptionClass1 as e:
    pass # do some stuff
except ExceptionClass2 as e:
    pass # do some other stuff
finally:
    pass # this code is executed in any case
```

- Throw exception by

```
raise MyException()
```

- Create a custom exception by inheriting from *Exception*
- Built-in exceptions: *StandardError*, *ArithmeticError*, *BufferError*, …

Exercise #4

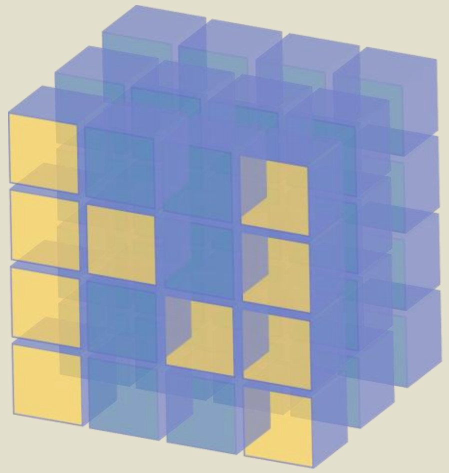
Object Oriented Programming

Agenda

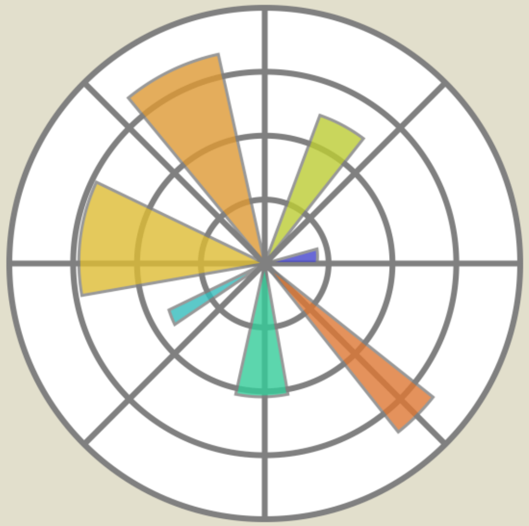
- Introduction
- Basic programming
- Useful data structures
- Object oriented programming
- Scientific programming

NumPy

- Multidimensional matrix manipulation
- High-level mathematical functions
- Basic type: *ndarray* (used by many other libraries)
- Performance speedup



- Matplotlib
  - Plotting library
  - Tight integration with NumPy



OpenCV

- Cross platform open source computer vision library
  - Originally developed by Intel
- Written in C++
  - Exposes APIs in C++, C, Python, Java, etc.
- Tight integration with NumPy (in Python)
- Integration with Deep Learning frameworks (TensorFlow, Torch, Caffe)
- Various hardware accelerations

Exercise #5

Computer Vision Challenge

The Drawbacks of Python

- Performance
  - Slow execution
  - Memory leakage potential
- Prone to runtime errors
- No real concurrent execution in multithreading (CPython)
- Indentation

Thank You!