Python fundamentals

José Antonio de la Torre las Heras

November 30, 2018

Introduction

- Python is a **simple** but **powerful** language
- There are two major versions:
  1. Python 2: Deprecated
  2. Python 3: Actual version
- It is a interpreted language (you don't need a compiler)
- It is weak typed
- It has a completely different syntax (compared with C or C++)

Features

- Easy to start developing
- Tons of libraries
  1. numpy: Matlab like arrays
  2. sklearn: Machine learning library
  3. pandas: big data management
  4. opencv: computer vision library
  5. sympy: symbolic mathematics
  6. ...

Getting started

Python could be executed in two ways

- Interactive: Open a terminal and start typing
- Scripting: Create a file and make it executable

Hello world in an interactive terminal

- Open a terminal
- Type python3 or python3.exe (windows)

Code:

```
1 print("Hello world")
```

Hello world in a script

- Open a editor
- Code:

```
1 print("Hello world")
```

- Save it with hello.py
- Open a terminal
- Go to the folder and type python3 hello.py

Introduction

- Python is not statically typed
- It is not necessary to declare a variable before using it.
- In variables types are not needed

Python is easier

C code:

```
1        int i;
2        char c;
3        i = 0;
4        c = 'a';
```

Python equivalent code:

```
1        i = 0
2        c = 'a'
```

Python is clever

In python a variable can change its type at any time.

```
1  i = 0
2  i = 'a'
```

Basic types

- Numbers:

```
1 myInteger = 0
2 myFloat = 1.1
```

- String and chars:

```
1 myString = "Hi"
2 myString2 = 'Hi'
3 myChar = 'c'
4 myChar2 = "C"
```

Booleans

```
1 isPythonGood = True
2 isWindowsGood = False
```

Python is case sensitive

True is not the same as true. True is correct, true is incorrect

Lists

Lists and arrays are the same in Python.

```
1 myList = []
2 myList.append('a')
3 myList.append('b')
4 myList.append('c')
5 myList.pop[0]
6 print(myList)
7 print(myList[0])
```

Dictionaries

Dictionaries are like arrays but it has key and value. In a dictionary could coexists different types.

```
1 measurements = {
2     "experiment1": 10,
3     "experiment2": 3,
4     "experiment3": "I have not idea what I've done",
5     "am i fired?": True
6 }
```

Dictionaries I

- Remove an item:

```
1 del measurements["am i fired?"]
2 measurements.pop("am i fired?")
```

- Check if item is in a dictionary

```
1 "experiment1" in measurements
```

- Add item

```
1 dictionary["ictp"] = "smr3249"
```

Dictionaries II

- Remove an item:

```
1  del measurements["am i fired?"]
2  measurements.pop("am i fired?")
```

- Check if item is in a dictionary

```
1  "experiment1" in measurements
```

- Add item

```
1  dictionary["ictp"] = "smr3249"
```

Arithmetic Operators

```
1  a = 1 + 1
2  a = 1 - 10
3  a = 1/2
4  a = 2 * 2
5  a = 2 ** 3
6  a = 4 \% 3
7  a = 15 // 4
```

Bitwise operators

```
1        x & 0
2        x | 1
3        x ~ 1
4        x ^       1
5        x >> 2
6        x << 2
```

Identity operators

```
1  a = [1,2,3]
2  b = [1,2,3]
3  a == b
4  a is b
5  a = b
6  b is b
```

Membership operators

```
1  a = [1,2,3]
2  1 in a
3   'b' in a
4  myDict = {'a': 1}
5  'a' in myDict
```

Introduction to blocks

- In python a block is a piece of code that has the same level of indentation and, at most, it has 1 blank line between them
- Each level of indentation has the same level as brackets blocks in C
- Indentation can be done by hard tabs (1 character) or by spaces (typically 4 spaces)
- Only one style can be used at the same time

Conditions I

```
1  a = 1
2  b = 2
3  if a == b:
4      print("Oh...")
5  else:
6      print("Ok")
7
8  a = 1
```

Conditions II

```
1  a = 1
2  b = 1
3  c == True
4  if a == b and c == True:
5      print("Perfect!")
6  else:
7      print("Bad programmer")
8
9  if a == b and c is True:
10     print("Perfect!")
11 else:
```

Conditions III

```
1  a = 1
2  b = 1
3  if a == b and c:
4      print("Perfect!")
5  else:
6      print("Bad programmer")
7
8  if a == b or c is False:
9      print("Perfect!")
10 else:
11     print("Bad programmer")
```

For loops

```
1 for i in range(1, 10):
2         print(i)
```

For loops I

```
1 a = [1,2,3,8]
2 for i in a:
3         print(i)
```

While loops

```
1  counter = 1
2  while counter < 10:
3      counter += 1
4      print(counter)
5
6  print(counter)
```

Introduction to functions

- Python reads the code in the order it is used
- If one function calls another it should be declared before
- Opposite to variables, functions should be defined

Declaration of functions

```
1  def myFunction():
2      print("Hello")
3
4  def myFunction(firstArgument):
5      print("Hello")
6      print(firstArgument)
7
8  def add(a,b):
9          return a + b
```

Invoke a function

```
1  def myFunction():
2      print("Hello")
3  def myFunction1(firstArgument):
4      print("Hello")
5      print(firstArgument)
6  def add(a,b):
7          return a + b
8  myFunction()
9  myFunction1('world')
10 myFunction1(1)
11 add(2,3)
```

Description

In programming a module allows us to organize the code.
In Python, modules are only files with .py extension. In other languages, name of module is determined by other variables but, in python, the name of the file will be the name of the module.

How to create a module

- Create a myModule.py file
- Create some variables and functions inside

How to import a module

- By default Python will search to a set of paths in order to find the module. Normally, those paths are in PYTHON_PATH variable.
- If you create a file and it is not in the PATH, Python will search directly in the same folder where the main file was called.
- Use "import" and the name of module (filename) without .py

Example

```
1 import math
2 math.sqrt(49)
```

Example I

```
1 import math
2 dir(math)
3 help(math)
```

Example II

```
1 from math import sqrt
2 sqrt(49)
```

Packages

Packages are a way to store several modules with a similar purpose in the same folder.
For example, we could have a package called "“game" with several modules inside. like:

- core.py
- player.py
- graphics.py

Packages in python are normal folders with a __init__ .py file inside.

Packages II

```
1 from PyQt4 import QtGui
2 from numpy import
```

Custom modules and packages import

```
1 import numpy as np
2 np.arange(10)
```

Introduction

In Python all is an object. For example type help(1) and you will see the documentation of a Integer class.

Introduction I

- Python allow us to create classes in our scripts.
- class keyword allow us to define a class
- Constructor is created with __init__ method
- All methods inside a class must start with “self” argument
- “self” is passed automatically when you invoke a function

How to use classes

```
1  class Person:
2      def __init__(self, name):
3          self.name = name
4          self.greeting = "Hi"
5
6      def sayHello(self):
7          print(self.greeting)
8          print(self.name)
9  ictpUser = Person('ictp')
10 joseUser = Person('user')
11 ictpUser.sayHello()
```

Count even numbers

- Create a function that gets a list of numbers as arguments and returns number of even numbers
- Call it with different lists and check the result

Super vowels

- Create a function that takes an string and make each vocal capital letter
- Example: hI Ictp I IIkE pythOn

Calculator

- Create a calculator class with different methods to use it (you can use math)
- Use it creating one object