Introduction to Python

CB2-101 – Introduction to Scientific Computing

November 16, 2015

Emidio Capriotti

http://biofold.org/

Biomolecules Folding and Disease

Institute for Mathematical Modeling
of Biological Systems
Department of Biology

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Python

- Python high-level programming language. Its design philosophy emphasizes code readability, and allows programmers to express concepts in few lines of code.

- Python is an object-oriented language supporting imperative and functional programming.

- Object-oriented programming that represents concepts as "objects" that have data fields (attributes) and associated procedures known as methods.

- Python implementation was started at the end of 1989 by *Guido van Rossum*. Python 2.0 was released in 2000, with new features including a full garbage collector and support for Unicode.

The interpreter

The interpreter can be accessed typing python in the shell

```
emidio-imac:data emidio$ python
Python 2.7.6 (default, Nov 23 2013, 23:57:52)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

To exit from the interpreter environment Ctrl+D or exit()

The interactive interpreter: ipython

```
emidio-imac:data emidio$ ipython-2.7
Python 2.7.6 (default, Nov 23 2013, 23:57:52)
Type "copyright", "credits" or "license" for more information.

IPython 0.13.1 -- An enhanced Interactive Python.
?               -> Introduction and overview of IPython's features.
%quickref       -> Quick reference.
help            -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Rules for naming variables

A variable name have to start with a letter start with a number

```
>>> 2name=True
  File "<stdin>", line 1
    2name=True
       ^
SyntaxError: invalid syntax<type 'bool'>
```

A variable can not be one of the 31 python keyword

```
>>> print=“Hello world”
  File "<stdin>", line 1
    print=“Hello world”
         ^
SyntaxError: invalid syntax
```

A variable can not contain illegal characters

```
>>> you*=1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'you' is not defined
```

Basic variable types

The simplest type of variable in programming is the boolean

```
>>> bit=True
>>> type(bit)
<type 'bool'>
```

The simplest numeric variable is integer

```
>>> inum=1
>>> type(inum)
<type 'int'>
```

More complex numeric variable is float

```
>>> fnum=1.5
>>> type(fnum)
<type 'float'>
```

In python character variable does not exist.

```
>>> text=‘Hello World’
>>> type(text)
<type 'str'>
```

String variables

In low level program languages the string is not a basic variable. It is actually a group of concatenated characters.

In programming languages such as fortran the length of a string is fix. In python a string can assume any length and it do not need to be declared.

Python provides built-in functions for dealing with string

```
>>> text=“Hello world!”
>>> print len(text)
12
>>> print text[0]
H
>>> print text[-1]
!
>>> print text[2:5]
llo
```

Convert variable types

- In python variables are not explicitly declared and are defined instantiation.
- Variable types can be converted. This procedure is referred as variable casting.
- In python changing the variable type can be a source of error.

Few example about how to modify variable type

```
>>> fnum=5.6
>>> int(fnum)
5
>>> inum=4
>>> float(inum)
4.0
>>> text=“1024”
>>> int(text)
1024
```

Standard operators

The standard operators for numeric variables are:

```
>>> 2+2
4
>>> 2-2
0
>>> 2*3
6
>>> 7/2
3
>>>7%2
1
>>>2**3
8
```

Some of these operators works also for strings

```
>>> ‘a’+’b’
‘ab’
>>> 2*’a’
‘aa’
```

Function

Function is a set of statements to perform a task.

```
def name(list of variables):
    statements
```

The function consists of two parts: the header and the body.

- The header contains the name and the list of variables
- The body contains the set of statements and is indented

The order of the statements defines the flow of execution.

A function can not be called before it has been defined

Example write a function that write a name.

```
>>> def print_name(name):
…             print “My name is’,name
…
>>> print_name(‘Emidio’)
My name is Emidio
```

Compose functions

Write the a function that calculate the square value of a given *n*

```
>>> def square(x):
…             return x**2
…
>>> square(3)
9
```

The function can composed and we can calculate

```
>>> square(square(3))
81
```

Python as built-in functions such as:

abs(x)	- absolute value
pow(x,n)	- power x**n
max(x,y)	- maximum between x,y
min(x,y)	- minimum between x,y

Import function

Functions and variables can imported from a python script

Define the following polynomial functions in poly.py script

```
k=2

def square(x):
        return x**2

def cubic (x)
        return pow(x,3)
```

Import the functions square and cubic and variable k

```
>>> from poly import square, cubic, k
```

Calculate the value of kx**-x**2 with x=2

```
>>> x=2
>>> k*cubic(x)-square(x)
12
```

if and operators

Basic structure of if in python. Also elif can be used.

```
if    (condition 1):
          do something 1
elif (condition 2):
          do something 2
else:
          do something 3
```

Standard operators are ==, !=, >, >=, <, <= that can be combined with and, or, not

Write a function that check names for length and first characters

```
>>> def check_name(name,name_len,letter):
…             if (len(name>=name_len and name[0]==letter):
…                         return True
…             else:
…                         return False
…
>>> print chech_name('Goofy',5,'G')
True
```

for and while loops

Basic structure of the for and while loop in python.

```
for  i  in  list:
        do something            # Indentation is needed

while (condition):
        do something            # Indentation is needed
```

Build a function that takes a text variable and print all the letters

```
>>> def print_for_letters(text):
…           for i in text:
…                   print i

>>> def print_while_letters(text):
…           i=0
…           while i<len(text):
…                   print text[i]
…                   i+=1
```

Exercise 1

Write a function that takes in input a string and a character and find the position in the string that first match the character

Use the while loop

```
>>> def find(text, character):
…         i=0
…         while i<len(text):
…             if text[i]==character:
…                 return i
…             else:
…                 i+=1
…     return -1

>>> def find('Goofy','o')
1
```

The function find() returns same result

```
>>> 'Goofy'.find('o')
```

Important modules

The module sys

access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

```
>>> import sys
>>> sys.argv[0]
```

The module math

provide standard mathematics functions

```
>>> import math
>>> math.pi
3.141592653589793

>>> math.sqrt(2)
>>> 1.4142135623730951
```

Exercise 2

Write the first python script to address the following problems

1. Count the number of vowels. The script takes in input a string and print the number of vowels
2. Write similar script that takes in input a string and a character and calculates the number matching characters.

Use the function upper() to compare string in uppercase

```
>>> text='abc'
text.upper()
ABC
```

Include sys.argv[2] variable to provide the character.

```
import sys
sys.argv[2]
```