Basics in Scientific Computing

Master-Module Biological Networks
July 22, 2016

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]:
```

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
```

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
```

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
```

Data structure

- In computer science, a data structure is a particular way of storing and organizing data in a computer to be used efficiently.

- Data structure is one of the key issue in programming, in particular nowadays when we are in the Big Data era.

- Big data is defining the current situation where we need to deal with datasets so huge and complex that it becomes difficult to process using traditional tools and/or data processing applications.

- As a consequence the decision about which is the best structure to represent and/or store your data is crucial.

List

One way to represent a group of data in python is the list.

A list or sequence is a data type that implements a finite **ordered** collection of values.

List in python

```
>>> mylist=[3, 4, 5, 11, 9]
>>> print mylist[2]
5
>>> print mylist[1:3]
[4, 5]
>>> mylist[-1]
9
>>> mylist[1]=‘a’
>>> print mylist
[3, ’a’, 5, 11, 9]
>>> print mylist+[True]
[3, ’a’, 5, 11, 9, True]
>>> mylist.append(True)
[3, ’a’, 5, 11, 9, True]
```

Tuple

Like for a list, a tuple consists of a number of values separated by commas.

Tuple in python

```
>>> mytuple=3, 4, 5, 11,”Text”
>>> print mytuple
(3, 4, 5, 11,”Text”)
>>> len(mytuple)
5
>>> x, y, z = mytuple[:3]
>>> print mytuple+(1,)
(3, 4, 5, 11, ”Text”, 1)
>>> mytuple.append(1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>> mytuple[0]=1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Variables and pointers

For regular variables the reference assignment works as expected

Python variable reference assignment

```
>>> x= 1
>>> y=x
>>> x=2
>>> print x, y
2 1
```

More complex variable types described by pointers

```
>>> mat1= [[1, 0],[0, 1]]
>>> mat2= mat1
>>> mat1[0][0]=0
>>> print mat1, mat2
[[0, 0], [0, 1]] [[0, 0], [0, 1]]
>>> import copy
>>> mat2=copy.deepcopy(mat1)
>>> mat1[0][0]=1
>>> print mat1, mat2
[[1, 0], [0, 1]] [[0, 0], [0, 1]]
```

Files (I)

In python files are represented as objects:

Creating a file object in python

```
f=open(filename,mode)         # object file f is associated to filename
```

modes are: read ('r'), write ('w'), and append ('a').

Most important methods on the object file in write mode

```
>>> f=open('file.txt','w')
>>> f.write("Hello world!")                  # The argument is a string
>>> f.writelines([' Emidio', ' Malay'])      # The argument is a list
>>> f.close()
```

Most important method in on the object file in read mode

```
>>> f=open('file.txt','r')
>>> cont = f.read()
>>> print cont
Hello world! Emidio Malay
```

Files (II)

Use readlines for reading file

```
>>> f=open(‘file.txt’,’r’)
>>> cont = f.readlines()
>>> print cont
[‘Hello world! Emidio Malay’]
```

File object are similar to a stack

```
>>> f=open(‘file.txt’,’r’)
>>> print f.read(5)
‘Hello’
>>> print f.read(50)
‘ world! Emidio Malay’
>>> print f.read(50)
‘’
```

In text file you can have special characters “\t” tab and “\n” newline

String module

To better work with strings python as a string module that can be also imported

Interesting methods on a string object

```
>>> import string
>>> text=“Hello World!\n”
>>> print text.upper()
‘HELLO WORLD!’

>>> text.split()
[‘Hello’, ‘World!’]
>>> text.replace(‘Hello’, ‘Ciao’)
“Ciao World!\n”
>>> text.find(“World”)
5
>>> text.rstrip(‘\n’)
“Ciao World!”
```

Dictionary

Dictionaries are not ordered lists indexed by keys, which can be any immutable type; strings and numbers can always be keys.

Tuples can be used as keys if they contain only strings, numbers, or tuples.

The list used in the previous exercise can be stored as a dictionary

```
>>> fdic= {‘Pietro’:42, ‘Zef’: 42, ‘Tommy’: 43}
>>> print fdic[‘Pietro’]
42
>>> print fdic.get(‘Pietro’,0)
42
>>> print fdic.get(‘Goofy’,0)
0
>>> print fdic.keys()
[‘Pietro’,’Zef’,’Tommy’]
>>> print fdic.values()
[42, 42, 43]
>>> for key,value in fdic.iteritems():
          print key, value
>>> dic= {(0,True):1, (0:False):2, (1,True):3, (1:False):4}
>>> print  dic[(0,True)]
```

Exercise

1. Consider the human proteins TP53 and PRIO and find for both of them the interacting partners in human using IntAct

2. For the interacting gene set extract all the Gene Ontology terms from the goa_human.gaf.gz file and sort them by their occurrence.

3. Look to the function indicated with the code GO:0004674. To which function does it correspond? Is the set of TP53 interactors enriched for this function?