Advanced Functions and Modules

CB2-101 – Introduction to Scientific Computing

November 19, 2015

Emidio Capriotti

http://biofold.org/

Biomolecules Folding and Disease

Institute for Mathematical Modeling
of Biological Systems
Department of Biology

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Errors and Exceptions

In python errors can be divided in Syntax Errors and Exceptions.

- Syntax Errors are generate by the interpreter when it is not able to correctly parse the python script.

- Although the code is syntactically correct, errors can happen when the script is executed. These errors, detected during execution, are called Exceptions and are not unconditionally fatal.

- Python has Built-in Exceptions that allow to hand selected exceptions in the program.

Handeling Exceptions

Exceptions are used when particular values the variables assume can generate errors. To handle Exception python implement the try/except statements

Basic syntax of try/except statements

```
try:
        do something
except:
        do something else
```

Example with ZeroDivisionError

```
>>> def inverse(x):
....      return 1/float(x)
....
>>> inverse(2)
0.5
>>> inverse(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in inverse
ZeroDivisionError: float division by zero
```

Else can be used after verifying possible exceptions.

sys and os

Important modules for interacting with operating systems and to modify settings and exe commands are sys and os.

Important methods and variables in sys module

```
>>> import sys
>>> print sys.path
['', '/Library/Python/2.7/site-packages/seqmagick-0.6.0_dev-py2.7.egg', ….
>>> sys.stderr.write('ERROR: File not found\n.')
ERROR: File not found.
$> python -c  "import sys; print sys.argv" arg1 arg2 arg3
```

Important methods and variables in sys module

```
>>> import os
>>> print os.environ
{'LOGNAME': 'emidio', 'USER': 'emidio', ….
>>> print os.getpid()
96708
>>> print os.getcwd()
/home/cb2user
>>> print os.path.isdir('/home/cb2user')
True
>>> print os.path.isfile('/home/cb2user/.bashrc')
True
```

system and subprocess

The subprocess modules allows to run programs on your machine.

Run subprocess in a python script

```
>>> import os
>>> os.system(‘pwd’)
/home/cb2user
0

>>> from commands import getstatusoutput
>>> getstatusoutput(‘pwd’)
(0, '/home/cb2user')
>>> getstatusoutput(‘ppwd’)
(32512, 'sh: ppwd: command not found’)

>>> import subprocess
>>> proc = subprocess.Popen([“cmd”,”flag1”,”arg1”,…] \
                stdout=subprocess.PIPE, stderr=subprocess.PIPE)
>>> stdout, stderr = proc.communicate()
```

sets

Comparison of different sets is a common problem in different scientific fields.

Managing sets in python using sets

```
>>> set1=set([1,2,3,4,5])
>>> set2=set([2,3,4,7,8])
>>> set1.intersection(set2)
set([2, 3, 4, 5])
>>> list(set1.intersection(set2))
[2, 3, 4, 5]
>>> list(set1.union(set2))
[1, 2, 3, 4, 5, 7, 8]
>>> list(set1.difference(set2))
[1]
>>> list(set1.symmetric_difference(set2))
[8, 1, 7]
>>> set1.add(6)
set([1,2, 3, 4, 5, 6])
>>> new_set=set1.copy()
>>> print new_set
set([1,2, 3, 4, 5, 6])
```

itertools

Itertools is a module to optimize the generation of combinations of elements in a set.

How calculate possible combinations of elements?

```
>>> import itertools
>>> list(itertools.product([‘A’,’B’,’C’],repeat=2))
[('A', 'A'), ('A', 'B'), ('A', 'C'), ('B', 'A'), ('B', 'B'), ('B', 'C'), ('C', 'A'), ('C', 'B'), ('C', 'C')]
>>> list(itertools.permutations([‘A’,’B’,’C’], 2))
[('A', 'B'), ('A', 'C'), ('B', 'A'), ('B', 'C'), ('C', 'A'), ('C', 'B')]
>>> list(itertools.combinations([‘A’,’B’,’C’], 2))
[('A', 'B'), ('A', 'C'), ('B', ‘C')]
>>> list(itertools.combinations_with_replacement([‘A’,’B’,’C’], 2))
[('A', 'A'), ('A', 'B'), ('A', 'C'), ('B', 'B'), ('B', 'C'), ('C', 'C')]
```

random

To generate random variables and shuffle dataset python has the a random module.

Random generates random number and shuffle sets

```
>>> import random
>>> random.random()
>>> 0.254942802653793
>>> random.randint(1,100)
>>> 25
>>> v=[1,2,3,4,5,6,7,8,9,10]
>>> random.shuffle(v)
>>> print v
[5, 1, 3, 2, 7, 10, 6, 9, 4, 8]
>>> random.gauss(20, 1)
18.685381502066623
```

Exercise 1

1. Write a script that generates n values that are normally distributed around the average value x with a standard deviation std.

2. Select average and sigma values on your choice. Using the first script generate 4 files with 10, 100, 1000 and 10000 numbers. Finally write a script that takes in input the previous file and calculates the average and the sigma values.

3. What are the differences from the chosen and calculates average and the sigma values?

regular expression

In python has been implemented the re module which provides regular expression matching operations.
The basic procedure for searching consists in the definition of the regular expression, the compilation process and the search.

Simple regular expression search

```
>>> import re
>>> pattern=re.compile('\sPF[0-9]{5}\s')        #define re and compile
>>> text='File: PF00234\t\tPF00001\n\tProtein 1 PF00051\t PF22222'
>>> re.findall(pattern,text)
[' PF00234\t', '\tPF00001\n', ' PF00051\t', ' PF22222 ']
>>> match=list(re.finditer(pattern,text))
>>> print match
[<_sre.SRE_Match object at 0x105710920>, …..]
>>> def get_limits(x):
...              return x.start(), x.end()
…
>>> maps(get_limits, match)
[(5, 14), (14, 23), (33, 42), (42, 51)]
```

Exercise 2

In Pfam database a Protein kinase domain is indicated by code PF00069. In general proteins can contain multiple domains. Write a script or run a linux command to:

1. selects all the proteins that contains at least one PF00069 domain.

2. lists all the domains co-occurring in the same protein with PF00069 and calculates the frequency of co-occurrence without considering possible repetitions.

numpy (I)

NumPy is the fundamental package for scientific computing with Python.
It is a optimized tools for:

- N-dimensional array object
- sophisticated mathematical functions
- Linear algebra, Fourier transform, and random number capabilities

Basic numpy functions

```
>>> import numpy as np
>>> np.array([1.,2.,3.,4.,5.,6.])
array([1., 2., 3., 4., 5., 6.])
>>> np.zeros([2,2])
array([[ 0.,  0.], [ 0.,  0.]])
>>> 10*np.log(np.array([1.,2.,3.,4.,5.,6.]))
array([ 0. , 6.9314718, 10.9861229, …])
>>> np.array([[1., 2.], [3., 4]])*np.array([[2., 2.], [2., 2]])
array([[ 2.,  4.], [ 6.,  8.]])
>>> np.dot(np.array([[1., 2.], [3., 4]])*np.array([[2., 2.], [2., 2]]))
array([[  6.,   6.], [ 14.,  14.]])
```

numpy (II)

Numpy also includes statistical functions and linear algebra module

Other numpy functions

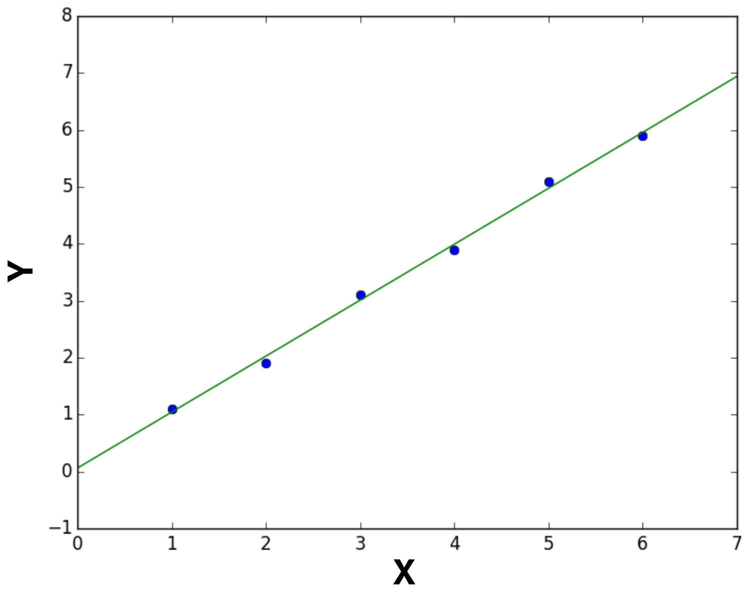
```
>>> import numpy as np
>>> v=np.array([1.,2.,3.,4.,5.,6.])
>>> np.mean(v)
3.5
>>> np.std(v)
1.707825127659933
>>> import numpy.linalg as la
>>> mat=np.array([[  4.,  12.],[ 10.,  26.]])
>>> la.det(a)
-15.999999999999998
```

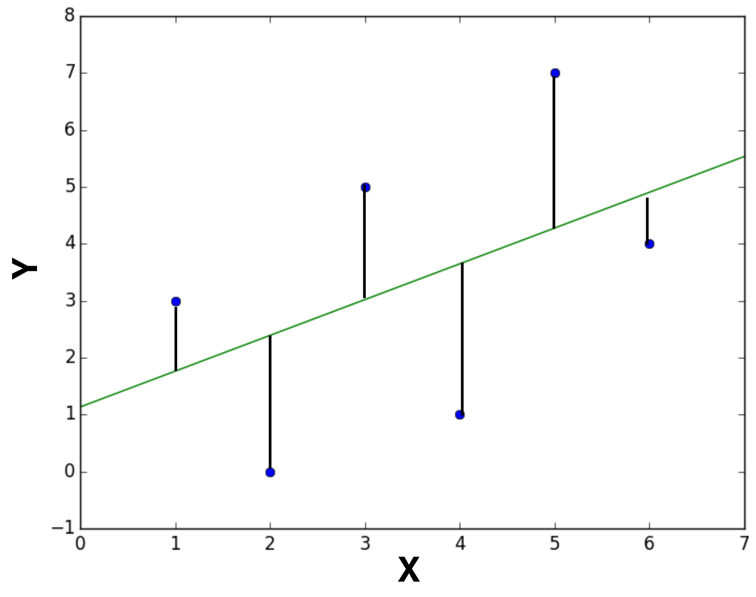
The linregress function

Import linregress from spicy.stats and calculate calculate the fitting curve

```
>>> from scipy.stats import linregress
>>> import numpy as np
>>> x = np.array([1,2,3,4,5,6])
>>> y = np.array([1.1,2.2,2.9,3.98,5.2,6.1])
>>> reg=linregress(x,y)
>>> print reg
LinregressResult(slope=0.98285714285714287,
intercept=0.060000000000000053, rvalue=0.99838143945702995,
pvalue=3.9274872444222332e-06, stderr=0.027994168488950467)
```

what happen if y = [3,0,5,1,7,4]. is this a better fitting? why?





Use matplotlib to plot

Import matplotlib and plot the points and fitting curve.

```
>>> import matplotlib.pyplot as plt
>>> yp=reg[0]*x+reg[1]
>>> plt.plot(x,y,'o',x,yp,'-')
>>> plt.show()
```

Exercise:

Write a python script that reads a file containing two columns of data (x,y) and calculate the linear regression curve and plots both the points an the regression curve.

For this exercise download the data using the command *wget* from http://biofold.org/courses/docs/data_hw.txt

Contingency table

In statistics, a contingency table is a type of table in a matrix format that displays the frequency distribution of the variables.

They are heavily used in survey research, business intelligence, engineering and scientific research.

	Function 1	Function 2	Row Total
Reference	7	6	13
Observation	1	12	13
Column Total	8	18	26

Fisher's test in python

The fisher_exact function is contained in the spicy.stats module and takes in input a contingency matrix. The function returns *odd ratio* and *p-value.*

```
>>> from scipy.stats import fisher_exact
>>> import numpy as np
>>> cm=np.array([[7,6],[1,12]])
>>> ft=fisher_exact(cm)
>>> print ft
(0.071428571428571425, 0.030205949656750501)
```

Exercise:

Write a python script that reads two file containing a columns with alleles carried by each individual. Use Fisher's exact test verify if an allele is over represented in one of the populations.

For this exercise download the data using the command *wget* from
http://biofold.org/courses/docs/pop1_allele.txt
http://biofold.org/courses/docs/pop2_allele.txt