Basic and advanced python modules

Elements of Programming Languages

Emidio Capriotti
http://biofold.org/

Biomolecules Folding and Disease

Department of Pharmacy and Biotechnology (FaBiT)
University of Bologna

Basic python modules

In the standard python interpreter some useful modules are made available by default.

The *os* module allows the usage of operating system-dependent functionality. The module include many functions to interact with the file system such as:

- os.getcwd()
- os.mkdir(path)
- os.remove(path)
- os.path.isfile(path)

The *sys* module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. Some interesting functions are:

- sys.stdout
- sys.stderr
- sys.path
- sys.argv

Read file content

In a python script the python interpreter is define in the first line with the shebang. To read the content of a file a file handler (fh) is defined. The average of numbers extracted from the column of a file can be calculated dividing the sum of the elements by their number.

```
#!/usr/bin/env python3
import sys

def read_values(filename,ncol):
   nlist=[]
   fh=open(filename,'r')
   for line in fh:
      num=float(line.split()[ncol])
      nlist.append(num)
   return nlist

if __name__ == '__main__':
   filename=sys.argv[1]
   ncol=int(sys.argv[2])-1
   nlist=read_values(filename,ncol)
   print ('Average:',sum(nlist)/len(nlist))
```

Numpy

The numpy module can be used to generate an array which can be manipulated to calculate the average and other statistical values

```
#!/usr/bin/env python3
import sys
import numpy as np

def read_values(filename,ncol):
   nlist=[]
   fh=open(filename,'r')
   for line in fh:
      num=float(line.split()[ncol])
      nlist.append(num)
   return np.array(nlist)

if __name__ == '__main__':
   filename=sys.argv[1]
   ncol=int(sys.argv[2])-1
   nlist=read_values(filename,ncol)
   print ('Average:',np.mean(nlist),'STD:',np.std(nlist))
```

Running on the shell

If we consider a *purchase.tsv* file contacting there columns reporting the name, the number of items and the total amount of the purchase, the average amount of the purchase can be calculated running the previous script (*average.py*).

The output of the program can be redirected to a the file average.txt

```
emidio@S968-01D20-W01:~$ cat purchase.tsv
Emidio  1       12.0
Paola   2       20.0
Piero   4       30.0
Kashaf  1       5.0
Young   4       40.0
Emidio  2       20.0
Piero   3       20.5
Nicola  4       30.0

emidio@S968-01D20-W01:~$ python3 average.py purchase.tsv 3 > average.txt

emidio@S968-01D20-W01:~$ cat average.txt
Average: 22.1875 STD: 10.313636301033695
```

Advanced python modules

Different advanced python module are interesting for performing data analysis and visualisation.

- *scipy*: a python module for scientific computing.
- *matplotlib*: a plotting library for the Python programming language.
- *pandas*: a software library written for the Python programming language for data manipulation and analysis.
- *requests*: an HTTP client library for the Python programming language.

Statistics with SciPy

SciPy provides algorithms for solving different class of mathematical problems. The scipy.stats has a *linregress* function that calculates regression between two variables.

```
#!/usr/bin/env python3
import sys
import numpy as np
from scipy.stats import linregress

def read_values(filename,xcol,ycol):
     xlist=[]
     ylist=[]
     fh=open(filename,'r')
     for line in fh:
            values=line.split()
            x=float(values[xcol])
            y=float(values[ycol])
            xlist.append(x)
            ylist.append(y)
     return np.array(xlist),np.array(ylist)

if __name__ == '__main__':
     filename=sys.argv[1]
     xcol=int(sys.argv[2])-1
     ycol=int(sys.argv[3])-1
     xlist,ylist=read_values(filename,xcol,ycol)
     fit=linregress(xlist,ylist)
     print (‘slope:',fit[0],'intercept:',fit[1],'r-value',fit[2],'p-value',fit[3])
```

Try and except

If in the *purchase.tsv* file a number of the items associated to a purchase is changed by mistake with a letter, the conversion to float will result in an error. The *try/except* statement can be used to handle the error.

```
def read_values(filename,xcol,ycol):
     xlist=[]
     ylist=[]
     fh=open(filename,'r')
     for line in fh:
             values=line.split()
             try:
                     x=float(values[xcol])
                     y=float(values[ycol])
                     xlist.append(x)
                     ylist.append(y)
             except:
                     sys.stderr.write('ERROR: Incorrect line: '+line)
     return np.array(xlist),np.array(ylist)
```

Errors and stderr

Errors can be printed on the stderr using *sys.stderr* and can be redirected to a log file through the appropriate redirection symbol (2>).

```
emidio@S968-01D20-W01:~$ cat purchase.tsv
Emidio  1       12.0
Paola   2       20.0
Piero   4       30.0
Kashaf  1       5.0
Young   A       40.0
Emidio  2       20.0
Piero   3       20.5
Nicola  4       30.0

emidio@S968-01D20-W01:~$ python3 regression.py purchase.tsv 2 3 2> average.log
('slope:', 6.647058823529411, 'intercept:', 3.5000000000000036, 'r-value',
0.9374890122876628, 'p-value', 0.0018140272855587877)

emidio@S968-01D20-W01:~$ cat average.log
ERROR: Incorrect line: Young   A    40.0
```

Data randomisation

random module implements pseudo-random number generators for various distributions. It can be also used to shuffle and sample elements of a list.

```
#!/usr/bin/env python3
import sys, random
import numpy as np
from scipy.stats import linregress

def read_values(filename,xcol,ycol):
     ...
     return np.array(xlist),np.array(ylist)

if __name__ == '__main__':
     filename=sys.argv[1]
     xcol=int(sys.argv[2])-1
     ycol=int(sys.argv[3])-1
     xlist,ylist=read_values(filename,xcol,ycol)
     fit=linregress(xlist,ylist)
     print (‘slope:’,fit[0],'intercept:',fit[1],'r-value',fit[2],'p-value',fit[3])
     random.shuffle(ylist)
     fit=linregress(xlist,ylist)
     print (‘slope:’,fit[0],’intercept:',fit[1],'r-value',fit[2],'p-value',fit[3])

python3 regression.py purchase.tsv 2 3
slope: 7.795 intercept:  1.726 r-value 0.921 p-value 0.001
slope: 2.321 intercept: 16.095 r-value 0.274 p-value 0.511
```

Plotting the fitting curve

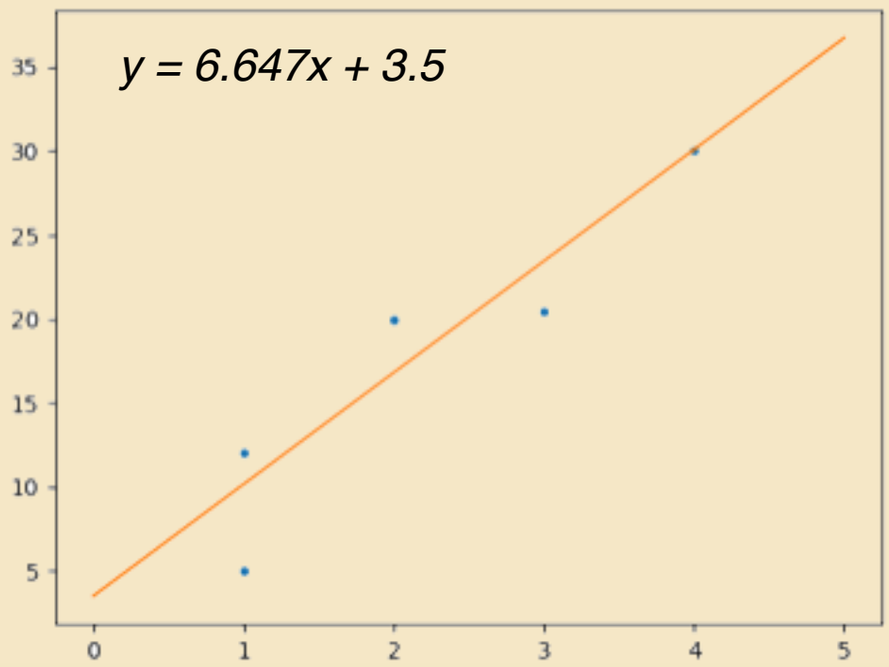
Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib can be used to generate a basic plot of the fitting curve.

```
import sys
import numpy as np
from scipy.stats import linregress
import matplotlib.pyplot as plt

def read_values(filename,xcol,ycol):
     ...
     return np.array(xlist),np.array(ylist)

def plot_fitting(x,y,slope,intercept,vmin,vmax):
     xl=np.array([vmin,vmax])
     yl=slope*xl+intercept
     plt.plot(x,y,'.')
     plt.plot(xl,yl,'-')
     plt.savefig("Downloads/fit.png")
     plt.show()

if __name__ == '__main__':
     ...
     fit=linregress(xlist,ylist)
     print ('slope:',fit[0],'intercept:',fit[1],'r-value',fit[2],'p-value',fit[3])
     plot_fitting(xlist,ylist,fit[0],fit[1],0,5)
```



DataFrame with Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. It can be used to generate a *DataFrame* a two-dimensional, size-mutable, potentially heterogeneous tabular data also from file.

```
#!/usr/bin/env python3
import sys
import pandas as pd

def get_df(filename,s='\t'):
  return pd.read_csv(filename,sep=s)

if __name__ == '__main__':
  filename=sys.argv[1]
  df=get_df(filename):
  print (df)

emidio@S968-01D20-W01:~$ python df_tsv.py purchase.tsv
     Name  Items  Amount
0  Emidio      1    12.0
1   Paola      2    20.0
2   Piero      4    30.0
3  Kashaf      1     5.0
4   Young      4    40.0
5  Emidio      2    20.0
6   Piero      3    20.5
7  Nicola      4    30.0
```

Getting data from the web

Requests allows you to send HTTP requests extremely easily with no need to manually add query strings to your URLs, or to form-encode your data.

```
 #!/usr/bin/env python3
import sys
import pandas as pd
import requests

def get_webdata():
    r=requests.get('https://reqres.in/api/users')
    r_dict = r.json()
    df=pd.DataFrame(r_dict['data'])
    return df

if __name__ == '__main__':
    filename=sys.argv[1]
    df = get_df(filename)
    print (df)

  emidio@S968-01D20-W01:~$ python test_pandas.py
     id                    email first_name last_name
  0   1  george.bluth@reqres.in     George     Bluth
  1   2  janet.weaver@reqres.in      Janet    Weaver
  2   3     emma.wong@reqres.in       Emma      Wong
  3   4      eve.holt@reqres.in        Eve      Holt
```

Exercise

Download a protein sequence for the web (https://rest.uniprot.org/uniprotkb/Q92624.fasta) and write a script that calculate the probability of transition between letters that compose the protein sequence (amino acids). Shuffle the sequence and find which transitions are significantly different from random?

Suggestions:

- From the file contains protein sequence remove the header (>) and newline characters.
- You can use a 20x20 matrix store the transition probabilities.
- The common letters (amino acids) composed a protein are ACDEFGHIKLMNPQRSTVWY