Files and Data Structure

CB2-101 – Introduction to Scientific Computing

November 12th, 2014

Emidio Capriotti
http://biofold.org/emidio

Biomolecules Folding and Disease

Division of Informatics
Department of Pathology

THE UNIVERSITY OF ALABAMA AT BIRMINGHAM

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!”
```

Exercise I.a

Write a script that write in a file the name of three of your friends and their ages. Each friend will be recorded in one line and where first appears the name and the age is separated by a tab.

Names and ages are given through a list of 2-dimensional tuples.

f_list=[(‘Pietro’, 42), (‘Zef’, 42), (‘Tommy’, 43)]

Use special characters and convert the integer to string.

```
def write_friend(filename,f_list):
        f=open(filename,’r’)
        for name,age in f_list:                         # tuple unpacking
            f.write(name+’ \t’+str(age)+’\n’)           # variable casting
        f.close()

if (__name__ == “__main__”):
        f_list=[(‘Pietro’, 42), (‘Zef’, 42), (‘Tommy’, 43)]
        write_friend(“friend_file.txt”,f_list)
```

Exercise II.a

Check if the previous file has been correctly written. Read the file and calculate the mean value of the your friends' age using a returning a float value. Write the program in the general for that allows to calculate the mean value of a set of number in a a given column of the file

The program can be run from command line with two arguments

```
import sys

def get_col_values(filename,position):
        vdata=[]
        lines=open(filename,'r').readlines()
        for line in lines:
                vec=line.rstrip('\n').split()
                vdata.append(float(vec[position-1]))
        return data

def mean(v):
        return sum(v)/len(v)

if (__name__ == "__main__"):
        filename=sys.argv[1]]
        col=int(sys.argv[2])
        v=get_col_values(filename,col)
        print "Averege age:", mean(v)
```

A good practice

It is a good programing practice when you open a file to read one line at the time. To avoid that huge file can make your machine crashing.

Read one line at the time

```
with open(filename,'r') as fobj:
        for line in fobj:
            do_something(line)
```

Read one line at the time the friend_file.txt file

```
>>> c=0
>>> with open('friend_file.txt','r') as fobj:
…         for line in fobj:
…             c=c+1
…             print c,line.rstrip()
1 Pietro    42
2 Zef       42
3 Tommy     43
```

An alternative is to import the module fileinput

```
import fileinput
for line in fileinput.input(['myfile']):
        do_something(line)
```

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 written 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(knights.iteritems()):
        print key.value
>>> dic= {(0,True):1, (0:False):2, (1,True):3, (1:False):4}
```

Exercise 2

Write the code in python that analyze the 9606.tsv file and selects

1. rows with “Family in position 8
2. E-value in position 13 lower than 1e-10

Use the and operator to solve this problem

> if (expression1 and expression2): do something

Restrict the search selecting domains with alignment length >=100

Concatenate three expressions in the IF statement calculating the difference between columns 3 and 2

> if ((v[3]-v[2])>threshold and expression1 and expression2) do something

More exercises

1. Write a python script that takes in input a fasta file of only one protein sequence and calculate the frequency of each amino acid. Check the output of P53_HUMAN.fasta and BRCA1_HUMAN.fasta.

2. Write a code that analyze a fasta file with multiple sequences and calculate the length of each sequence. Save the results in a file.

3. Write a script that analyze the 9606.tsv file and build a dictionary using as key the ID of the protein and value the corresponding list of domains. The program will be called using the name of the protein.