BE490/Bio 7.91 Python Mini-tutorial—DAY 3

Python Programming: In python you can write programs that can run as a stand alone program or you can import them into other python code. In fact, you have already been using python programs every time you use an import command. As an example of the framework of a basic python program, see SampleProg.py.

This program has a few basic parts that we will go through here

- #!/usr/bin/env python
  - This command makes your script executable directly after correcting its permissions with chmod +x SampleProg.py
  -
- “”” Triple quoted comments “””
  - Triple quotes in general allow you to quote things that have both double quotes “” and single quotes ‘’ in them. In this specific location, these triple quotes are a documentation string for the module.
- def do_comparison(x,y):
  - This is a definition of a function within this module. This function takes as input the values x and y and can do stuff with them. In general, a function can take input and return output and provides a simple way to organize code.
  - Note that this function also has a documentation string and that indentation still matters.
- def read_format(file):
  - this is another function
- if __name__ == "__main__":
  - This is a way to tell the program that this file is being run as a stand alone program. Thus, if we run this program directly from the Unix command line, then the name of the program (stored in __name__) will be “__main__”, however if we import the program while in the python interpreter, then __name__ is SampleProg, so this will not run.
- if len(sys.argv) != 2:
  - The command sys.argv gets arguments from the command line. Thus, if I run python with two arguments then this will work. E.g.
    python SampleProg.py fasta.txt ← this works and opens fasta.txt
    python SampleProg.py fasta.txt 10.3 ← this does not work
    python SampleProg.py ← this does not work
  - If the argument is not right, then it prints the __doc__ string at the top of the program and exits (sys.exit()).
  - Otherwise, it takes in the file name and runs read_format.

In Class Exercises:

1) Load SampleProg.py from within the python interpreter and test the functions read_format and do_comparison.
2) Modify SampleProg.py so that it compares two numbers given at command line using do_comparison.

Regular Expressions: Regular expressions are a powerful text parsing tool that is widely used in bioinformatics. See the attached sheet for a summary of the commands. For an online summary of regular expression commands see regex.pdf.

- import re ← imports the regular expression module
- m=re.match('regular expression', 'target string') ←test to see if the regular expression matches the start of the test string.
- m=re.search('regular expression', 'target string') ←test to see if the regular expression matches any part of the test string.
    - For match and search, you can query the resulting object with the following functions:
        i. group() ← returns the matched string
        ii. start() ←returns the start location of the string
        iii. end() ←returns the end location of the string
        iv. span() ←returns the start and end location of string
- m_list=re.findall('regular expression', 'target string') ←test to see if the regular expression matches any part of the test string, and returns the finds as a list.

In Class Exercises:

1) Write a regular expression to extract all of the carbon atom position data from the file example.pdb. Print this data out.