**CSE 311** Lab Date: 01/20/2022
**Lab 1 - Unix & Python Introduction** Due Date: 01/20/2022
Instructor: Dr. Juan Cui
TA: Mingrui Zhang

## Lab Materials

CSE account
Unix commands - http://cse.unl.edu/faq-section/unix-linux
Python 3.0 - https://docs.python.org/3.0/

## Lab Objective

Data is usually presented in a human-readable format. For example, when working from a bioinformatics perspective, a protein is presented as follows:

```
>sp|P19367|HXK1_HUMAN Hexokinase-1 OS=Homo sapiens GN=HK1 PE=1 SV=3
MIAAQLLAYYFTELKDDQVKKIDKYLYAMRLSDETLIDIMTRFRKEMKNGLSRDFNPTATVKMLPTFVRSIPDGSEKGDFIALDLGGSSFRILR
VQVNHEKNQNVHMESEVYDTPENIVHGSGSQLFDHVAECLGDFMEKRKIKDKKLPVGFTFSFPCQQSKIDEAILITWTKRFKASGVEGADV
VKLLNKAIKKRGDYDANIVAVVNDTVGTMMTCGYDDQHCEVGLIIGTGTNACYMEELRHIDLVEGDEGRMCINTEWGAFGDDGSLEDIRTEF
DREIDRGSLNPGKQLFEKMVSGMYLGELVRLILVKMAKEGLLFEGRITPELLTRGKFNTSDVSAIEKNKEGLHNAKEILTRLGVEPSDDDCVSV
QHVCTIVSFRSANLVAATLGAILNRLRDNKGTPRLRTTVGVDGSLYKTHPQYSRRFHKTLRRLVPDSDVRFLLSESGSGKGAAMVTAVA
YRLAEQHRQIEETLAHFHLTKDMLLEVKKRMRAEMELGLRKQTHNNAVVKMLPSFVRRTPDGTENGDFLALDLGGTNFRVLLVKIRSGKKR
TVEMHNKIYAIPIEIMQGTGEELFDHIVSCISDFLDYMGIKGPRMPLGFTFSFPCQQTSLDAGILITWTKGFKATDCVGHDVVTLLRDAIKRREEF
DLDVVAVVNDTVGTMMTCAYEEPTCEVGLIVGTGSNACYMEEMKNVEMVEGDQGQMCINMEWGAFGDNGCLDDIRTHYDRLVDEYSLN
AGKQRYEKMISGMYLGEIVRNILIDFTKKGFLFRGQISETLKTRGIFETKFLSQIESDRLALLQVRAILQQLGLNSTCDDSILVKTVCGVVSRRA
AQLCGAGMAAVVDKIRENRGLDRLNVTVGVDGTLYKLHPHFSRIMHQTVKELSPKCNVSFLLSEDGSGKGAALITAVGVRLRTEASS
```

The above is a protein presented in a fasta file. ">" symbol in a fasta file signifies the start of a new protein. The line containing the ">" is a description of the protein while the following lines are the sequence of amino acids making up the protein. In this case, the description is composed of the database, database access ID, and the protein name and similar information.
To be able to manipulate this data, we first need to parse it. The objective of this lab is to parse the sample.fasta file from the lab handout in Python using an object-based approach. You may assume it is a valid fasta file and that it follows the fasta conventions. To complete this lab, use Python to parse and output in the following format:

```
ID:  P19367
Protein Information: HXK1_HUMAN HexokinaseÂη1 OS=Homo sapiens GN=HK1 PE=1 SV=3
Sequence:
MIAAQLLAYYFTELKDDQVKKIDKYLYAMRLSDETLIDIMTRFRKEMKNGLSRDFNPTATVKMLPTFVRSIPDGSEKGDFIALDLGGSSFRILR
VQVNHEKNQNVHMESEVYDTPENIVHGSGSQLFDHVAECLGDFMEKRKIKDKKLPVGFTFSFPCQQSKIDEAILITWTKRFKASGVEGADV
VKLLNKAIKKRGDYDANIVAVVNDTVGTMMTCGYDDQHCEVGLIIGTGTNACYMEELRHIDLVEGDEGRMCINTEWGAFGDDGSLEDIRTEF
DREIDRGSLNPGKQLFEKMVSGMYLGELVRLILVKMAKEGLLFEGRITPELLTRGKFNTSDVSAIEKNKEGLHNAKEILTRLGVEPSDDDCVSV
QHVCTIVSFRSANLVAATLGAILNRLRDNKGTPRLRTTVGVDGSLYKTHPQYSRRFHKTLRRLVPDSDVRFLLSESGSGKGAAMVTAVA
YRLAEQHRQIEETLAHFHLTKDMLLEVKKRMRAEMELGLRKQ......
```

# Introduction to Python

## Python as a Calculator

```
>>> 16+8;8-10;2*3;5/4;5.0/4;4**2;4**2**0.5;(4**2)**0.5
24
-2
6
1
1.25
16
7.102993301316016
4.0
>>> import math
>>> math.sqrt(4**2)
4.0
>>> math.log(100)
4.605170185988092
>>> math.log(math.e,math.e)
1.0
>>> math.log(4,2)
2.0
>>> math.log10(100)
2.0
>>> math.ceil(5.6)
6.0
>>> math.floor(5.6)
5.0
```

## First Python Program and Commenting Your Code

You should make a habit of including your name and the name of your script in the first few lines. Also, your code should include comments generally 1 comment for 1-4 lines of code.

```
__author__ = 'John Doe'
#first_program.py

#calculate difference between two years: current and the year of birth.
>>> def calculate_age(year, current_year=2017):
...     return current_year - year
...

# Guido van Rossum is the creator of Python programming language
# Born 31 January 1956
>>> year = raw_input('What year were you born? ')
What year were you born?: 1956

#print the age
>>> print 'You are %d years old' % calculate_age(year)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in calculate_age
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

```
#print the age
>>> print 'You are %d years old' % calculate_age(int(year))
You are 61 years old
```

### Conditional Statements

```
__author__ = 'John Doe'
#conditional_examples.py

#An example with "not" and "and"
>>> if not ( 9==7 ) and 3 < 9:
...     print 'Splendid!'
... else:
...     print ":("
...
Splendid!
#Traffic light example
#get input for traffic light color and convert the input to lowercase
color = raw_input("What color is the traffic light? ")
What color is the traffic light? green
color = color.lower()

#check if the input color light is red, yellow or green.
>>> if color == "red":
...     print "Stop!"
... elif color == "yellow":
...     print "Slow down"
... elif color == "green":
...     print "You can go ahead"
... else:
...     "Which country are you in?"
...
You can go ahead
```

### Loops in Python

In Python, there are two kinds of loops: for and while loops. Code within a while loop is executed as long as the while loop statement evaluates to True. For loop iterates over the list it is provided with.

```
__author__ = 'John Doe'
#loops.py

#loop over a range of numbers
>>> for i in range(10):
...     print i
...
0
1
2
3
4
```

```
5
6
7
8
9

#loop over a string
my_str = 'hello, world!'

>>> for letter in my_str:
...     print letter
...
h
e
l
l
o
,

w
o
r
l
d
!

#while loop example. Within the loop, first print the value of the counter,
then divide the counter by 10.
ct = 10000
>>> while ct >100:
...     print ct
...     ct = ct/10
...
10000
1000
```

### Strings

```
__author__ = 'John Doe'
#strings.py
#define two strings
>>> my_str1 = 'The quick brown fox'
>>> my_str2 = 'jumps over the lazy dog'

#concatenate and print two strings
>>> print my_str1 + my_str2
The quick brown foxjumps over the lazy dog

#comma in the print statement inserts whitespace between strings
>>> print my_str1, my_str2
The quick brown fox jumps over the lazy dog

#index the string
>>> print "my_str1[10] is", my_str1[10]
```

```
my_str1[10] is b

#slice the string
>>> print "my_str1[10:15] is", my_str1[10:15]
my_str1[10:15] is brown

#print the length of the string
>>> print "Length of my_str1 is", len(my_str1)
Length of my_str1 is 19

#upper and lower cases
>>> print "Uppercase my_str1:", my_str1.upper()
Uppercase my_str1: THE QUICK BROWN FOX
>>> print "Lowercase my_str1:", my_str1.lower()
Lowercase my_str1: the quick brown fox

#strings are immutable
>>> my_str1[2] ='a'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

### Lists

```
__author__ = 'John Doe'
#lists.py

#define and print a list
>>> my_list = [2, 3, 4, 5]
>>> print "my_list is:", my_list
my_list is: [2, 3, 4, 5]

#index and slice list
>>> print "my_list[2] is:", my_list[2]
my_list[2] is: 4
>>> print "my_list[2:4] is:", my_list[2:4]
my_list[2:4] is: [4, 5]

#lists are mutable
>>> my_list
[2, 3, 4, 5]
>>> my_list[2]
4
>>> my_list[2] = 7
>>> my_list
[2, 3, 7, 5]

#add a new element
>>> my_list.append(12)
>>> my_list
[2, 3, 7, 5, 12]
#insert 20 at position 3
>>> my_list.insert(3, 20)
>>> my_list
```

```
[2, 3, 7, 20, 5, 12]

#remove a specified element
>>> my_list.remove(12)
>>> my_list
[2, 3, 7, 20, 5]

#reverse a list
>>> my_str1
'The quick brown fox'
>>> my_str1[::-1]
'xof nworb kciuq ehT'
```

### Classes

```
__author__ = 'John Doe'
#class.py

#import required Python library
>>> import math

#define a Point class, and a method to calculate the distance of the point
from the origin
>>> class Point:
...     def __init__(self, x, y):
...             self.x = x
...             self.y = y
...     def __str__(self):
...             return "A Point at coordinates " + str((self.x, self.y))
...     def distance_from_origin(self):
...             return math.sqrt((self.x-0)**2 + (self.y-0)**2)

#create a point object
>>> a = Point(3,4)
>>> a
<__main__.Point instance at 0x7f611be525a8>
>>> print a
A Point at coordinates (3, 4)
#calculate Point a's distance from origin
>>> a.distance_from_origin()
5.0
```