**Bioinformatics Practice**

***Python Introduction***

1. Type Python at the Unix prompt to start Python interpreter
2. **An algorithm is a sequence of instructions for solving a problem.**
3. **Interpreter:** 
   1. program which translates the program statements into machine language one line at a time as the program is running
   2. A program that reads and executes source code one line at a time. Does not create an executable file that can run independently.
   3. A program that reads, interprets, and executes a program, eliminating the need for compiling source code. Running a program through an interpreter can reduce the edit-compile-run-debug cycle to simply edit-run-debug, saving substantial development time.

***Example 1:*** Type the following commands in the Python window, hit ENTER after each line of code, observe the result

**Lines starting with # are Python comments, you don’t need to type these lines, they are for the explanation purposes only**

a = 4

b = 7

c = a + b

print (a, b, c)

c = a\*b

print (a, b, c)

c = a/b

print (a, b, c)

c = float (a)/b

print (a, b, c)

c = a%b

print (a, b, c)

a = 2

b = 3

c = a\*\*b

print (a, b, c)

codon1 = ‘tca’

codon2 = ‘gtc’

print (codon1, codon2)

print (len(codon1))

dna=codon1 + codon2

print (dna)

print (len(dna))

# (comment in Python) next line will cause the error, pay attention on the error description

dna = codon1 + 1

Try these statements:

codon1 == codon2

codon1 != codon2

***Example 2:*** Input in Python:

 Numeric Data - numbers

 String Data - sequence of characters and numbers

 **input** for numeric data (Older versions of Python)

 **raw\_input** for an alphanumeric data (Older versions of Python. Python 3 is not supporting this function anymore)

 general format:   
name\_of\_variable = input ("User Prompt ")   
name\_of\_variable = raw\_input("User Prompt ")

Type the following commands, hit ENTER after each line, observe the result.

**Lines starting with # are Python comments, you don’t need to type these lines, they are for the explanation purposes only**

**a = input (“Please enter the integer “)**

# Python Comment: after you will hit ENTER the computer will wait for your respond, enter any integer number

# and hit ENTER again

**print a**

**seq = raw\_input(“Please enter the sequence “)**

# Python Comment: after you will hit ENTER the computer will wait for your respond, enter any sequence of

# characters and hit ENTER again

**print (seq)**

***Example 3:*** if, if-else, if-elif-else

**# if statement in Python:**

**if expression:**

**action**

**# Explanation: If expression is true the action will be executed, otherwise the program will skip the action and execute the first #statement after if**

**#Important!!!! Indentation: action portion of the statement has to be shifted**

Type the following, observe the result:

seq1 = ‘tca’

seq2 = ‘tca’

if seq1 == seq2 :

print (“the sequences are identical”)

seq2 = ‘cta’

if seq1 == seq2 :

print (“the sequences are identical”)

if seq1[0] == seq2[1] :

print (“first element of seq1 and second element of seq2 are the same “)

**#if – else statement in Python**

**if expression:**

**Action 1**

**else:**

**Action 2**

**#Explanation: If expression is true Action1 will be executed, if expression is false Action2 will be executed.**

**#Important!!! Indentation: Action1 and Action2 have to be shifted**

Type the following, observe the result:

seq1 = ‘atcggta’

seq2 = ’atcgcta’

if seq1 == seq2 :

print (“the sequences” , seq1, “and “, seq2, “are the same “)

else:

print (“the sequences” , seq1, “and “, seq2, “are not the same “)

***Example 4:*** Loop **while** in Python:

Type the following and observe the result:

**dna = ‘actgtcttc’**

**i=0   
while i < len(dna):**

**print (dna[i])**

**i = i+1**

#note i= i+1 could be replaced by i+=1

* UNIX input/output redirection: **python file\_name.py < input.txt**
* **python file\_name.py < input.txt > output.txt**

**Example 5:**

**#function definition**

def scoreAlign (seq1, seq2):

i=0

sum=0

length=len(seq1)

while i<length:

if seq1[i] == seq2[i]:

sum = sum + 1

i = i + 1

return sum

#main program

def main():

seq1=raw\_input("Please enter the first seq ")

seq2=raw\_input("Please enter the second seq ")

print(“the input is”)

print (seq1)

print (seq2)

print ("the score is ")

print(scoreAlign(seq1, seq2))

main()

# This program shows a few useful functions to work with strings

# Call this program string.py

**#function len (string\_name) – returns the length of the string**

dna = ‘actc’

print (len(dna))

**#string1 + strnig2 – concatenates two strings**

**#string1\*int – repeats string1 the amount of times that equal to the integers value**

dna1 = ‘ctg’

dna2 =’cgtcttc’

dnaRes = dna1+dna2

print (dnaRes)

dnaRes = dna1\*3

print (dnaRes)

**string.find (sub\_string) – finds the first occurrence if sub\_string within string, function returns the index of the first character of sub\_string in string, function returns -1 id sub\_string is not found in string**

**string.find(sub\_string, position) finds the first occurrence of sub\_string within string starting from index = position, function returns the index of the first character of sub\_string in string from index = position, function returns -1 if sub\_string is not found in string**

**ecoRI – restriction enzyme (find the definition of this concept)**

dna = ‘acctgaattctactgaattcatata’

ecoRI = ‘gaattc’

index1 = dna.find(ecoRI)

print (index1)

index2 = dna.find(ecoRI , index1 + 1)

print (index2)

sub\_string = ‘aatctag’

index = dna.find(sub\_string)

print (index)

index = dna.find(sub\_string [ 2 : 6 ] )

print index

**# ‘char’ in string – returns true if char occurs in string, otherwise false**

result = ‘a’ in dna

print (result)

result = ‘b’ in dna

print (result)

**#count (string, ‘char’) – returns amount of times that char occur in string**

result=dna.count(‘a’)

**#next line calculates the at percent of dna**

atPercent = (dna.count('a') + dna.count( 't')) / float(len(dna)) \* 100

print (atPercent)

**#the next fragment of code calculate the complement of dna using methods *maketrans* and *translate***

from string import maketrans

transition = maketrans ( ‘AGTCagtc’, ‘TCAGtcag’ )

complement = dna.translate(transition)

print (complement)

**#string.replace(‘char1’, ‘char2’)**

res = dna.replace('a', 'A')

print (res)