420-SN1 Programming in Science - Lab Exercise 7

1 Basics

Complete this part in the file Lab07-P1.py. Include subsection and question number in a "#" for each question. Save and run your code after each question.

1.1 String Indexing

Define the string: `dna = "ATGAATTCTC"`

1. Write a statement that computes the length of `dna` and then assigns the resulting value to a variable called `dna_len`. You should use the `len()` function to compute the length!
2. Write and print an expression, using `dna`, that gives the first character of the DNA sequence.
3. Write and print an expression, using `dna` and `dna_len`, that gives the final character in the DNA sequence.
4. Write and print an expression, using `dna`, that gives the first three characters in the DNA sequence.
5. Write and print an expression, using `dna` and `dna_len`, to get the last three characters in the DNA sequence.
6. You can add a character to the end of a string:

```
new_dna = dna + "!"
print(new_dna)
```

7. **Try this in the Python shell:** What happens? Write your explanation in a "#" in your .py file

```
print (dna[dna_len])?
```

1.2 List operations

Try the following operations with lists.

1. Create 2 lists by typing the following:

```
x = [5, -1, 2, 0, 2, 8]
y = [3, 4, 5]
```

2. print the length of the list:

```
print(len(x))
```

3. Replace an item on the list and print the list:

```
x[0] = x[0] + 1
print(x)
```

   Check the contents of `x` after this statement.

4. Add an element to the end of the list and print the list:

```
x = x + [1]
print(x)
```

5. create a new list w by combing the 2 list x and y together.

```
w = x + y
print(w)
```

2 Strings

For this part, you will work with some simple operations on DNA sequences represented as upper-case strings. DNA sequences consist of four different bases: adenine, cytosine, guanine, and thymine. In bioinformatics software, DNA sequences are often represented as a string of letters A, C, G, and T, representing these bases. In this lab you will implement a few basic operations on these sequences. Don't worry if you're not familiar with DNA, the point of this exercise is just to work with some simple string data.

2.1 Complementing DNA

DNA sequences are made of two stands. Every DNA sequence has a complementary sequence. In the complementary sequence, every "A" is paired with "T", every "G" is paired with a "C" and vice versa for those combinations. Write a program named `Lab07-P2-1.py` that takes the DNA sequence given to you and outputs its complement.

Example:

```
Enter a DNA sequence: AGTTCAGC
The complement is TCAAGTCG
```

2.2 Introns

Introns are noncoding sections of DNA, that is parts of DNA that are cut out (or spliced out) before the DNA is translated into RNA. The coding parts of the sequence are called the exons.

Write a program, `lab07-P2-2.py`, which will be given a string that represents a short section of DNA comprised of two exons and an intron. The first exon runs from the start of the sequence to the n^{th} character (inclusive), and the second exon runs from the m^{th} character to the end of the sequence.

Write a program that will input a `dna` sequence as a string, an integer `n` and an integer `m`, then prints just the coding regions of the DNA sequence, as a single string.

For example:

```
Enter a DNA sequence: ACCGTGGTAATTTCG
Enter the value n: 3
Enter the value m: 10
The transcribed DNA is ACCGTTTCG
```

Hint(s):

- Try out what `dna[:n]` gives you.
- Try out what `dna[m:]` gives you. What about `dna[m-1:]` or `dna[m+1:]`?

2.3 Trace through a list of Strings

For this part, work in the file lab07-P2-3.py. What would the following program print? `Manually` trace the code and show the result that would be printed. Show your detailed work without using IDLE. In a second pass, type the code and run to verify your answer. Use "#"'s to include your manual work.

```
fruits = ["cantaloupe","apple","banana","orange","grape","grapefruit"]
index = 0
while index < len(fruits):
    if len(fruits[index]) > 5:
        print(fruits[index],"length:",len(fruits[index]))
    index += 1
print(index)
```

3 Lists

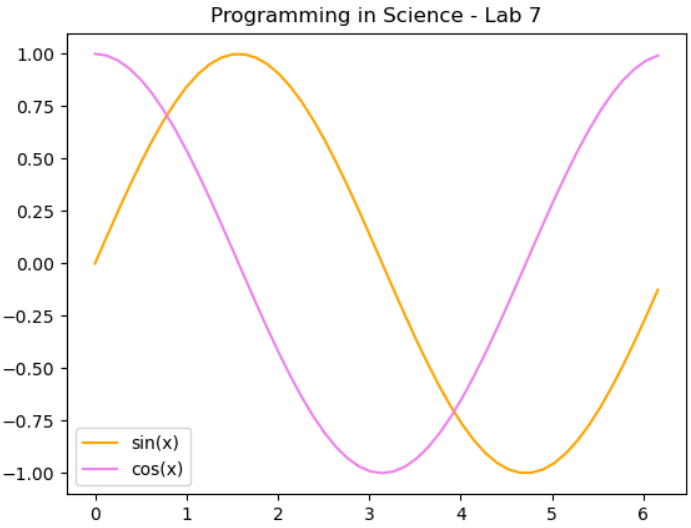
3.1 matplotlib - optinal

We have provided a file `Lab07-P3.py` that uses the `matplotlib` package to display a simple chart. The package generally uses lists to represent the value on the x and y axes.

Your tasks are as follows:

- Run the program as-is and observe the chart that it produces.
- Modify the program to display the function $y = sin(x)$ instead of the function $y = x^2$.
- Now add a third list called `z`. Use this to plot the function $y = cos(x)$. You can call the function `plt.plot(a,b)` multiple times to display several lines on a single chart.
- Add a "legend" to your chart so that someone reading it can tell which line is which. You can do this with the function `plt.legend(x)` where `x` is a `list` of strings. Each item on the list `x` will be displayed in a small window on the chart.

Your final graph should look like this:



Submitting your work

Create a ZIP file(Lab7.zip) containing all your files, and submit them via Omnivox.