BIOL 6505 Programming in Biological and Health Sciences Fall 2023
Homework assignment 1

Haowen He, Yashas Appaji, Yeojin Kim, Sarth Diskalkar

1.a: In the string class, find two string methods that will return `True` if the string can be converted into an integer. (5 points)

```
a = '23432'
```

```
a.isnumeric()
```

```
True
```

```
a.isdigit()
```

```
True
```

```
a = '23423.234'
```

```
a.isnumeric()
```

```
False
```

```
a.isdigit()
```

```
False
```

```
a = 'PatrickMcGrath'
```

```
a.isnumeric()
```

```
False
```

```
a.isdigit()
```

```
False
```

1.b: Using Google, determine what the difference is between these two methods. Explain briefly below.(5 points)

`str.isdigit()` returns `True` if strings containing only the digits (0-9), while `str.isnumeric()` returns `True` if string contains only numeric characters, which include digits (0-9) but not limited to digits, it also accepts Roman Numerals, numbers in other languages, and expoents.

2: Sometimes you want to pad a string so that it's a certain length. For example, let's say you have a DNA sequence that you wanted to be 9bp long, adding dashes for any bases that are missing. Find one or more methods that would allow you to do the following. Note that your approach should be able to accept any DNA sequence less than 9bp long, not just the example sequence given below. (5 points for each)

```
b = 'CGTTT'
```

```
b.ljust(9, '-')
```

```
'CGTTT----'
```

```
b.rjust(9, '-')
```

```
'----CGTTT'
```

3.a: Assume you have a float, 3454.4521. You want to split this number into the integer and decimal portion of the number. First. just use methods or magic methods of the float class to accomplish this. Note that, for the decimal portion, you method might return a value that is off by a tiny margin, as shown below. This is called a floating-point error. (5 points)

```
a = 3454.4521
```

```
int(a)
```

```
3454
```

```
a % 1
```

```
0.4520999999999731
```

3.b: Convert the float to a string and use string methods to accomplish this (the final numbers should be integers) (5 points)

```
b = str(a)
```

```
b
```

```
'3454.4521'
```

```
int(b.split('.')[0])
```

```
3454
```

```
int(b.split('.')[1])
```

```
4521
```

4: Create a function that takes in three numbers and return the sum. (10 points)

```
def sum_three(a, b, c):

     sum = a + b + c

     return sum
```

```
sum_three(1, 2, 3)
```

```
6
```

```
sum_three(3, 3, 3)
```

```
9
```

5: Create a function that takes in a string containing 10 words separated by commas and return a list of the words in alphabetical order. (10 points)

```
def ordered_list(words_str):

     list = words_str.split(',')

     # caseless matching
     sort_list = sorted(list, key=str.casefold)

     return sort_list
```

```
print(ordered_list('We,take,programming,in,biological,and,health,sciences,with,Patrick'))
```

```
['and', 'biological', 'health', 'in', 'Patrick', 'programming', 'sciences', 'take', 'We', 'with']
```

6.a: What kind of object is a? (4 points)

```
a = True
type(a)
```

```
bool
```

a is boolean type.

6.b: What kind of object is returned by a + a? (3 points)

```
a = True
type(a + a)
```

```
int
```

a + a is an integer.

6.c: What type of object is returned by a * a? (3 points)

```
a = True
type(a * a)
```

```
int
```

a * a in an integer.

7: Assume you have a string of valid DNA sequence (all upper case). DNA methylation enzymes can add methyl groups to DNA nucleotides at defined sequences. However, there can be redundancy in the motif recognized by the protein. For example, the CcrM methyltransferase recognizes GANTC sites (where N can be any of the four nucleotides). Create a single line that returns the number of CcrM methylation sites within the string of DNA sequence (call the variable DNA-sequence). (10 points)

```
DNA_sequence = 'AGAAGGCCCTAGAGTCCAAGACTCAGATTCAGAATCGACTC'
```

```
sum(DNA_sequence.count(x) for x in ('GAATC', 'GACTC', 'GAGTC', 'GATTC'))
```

```
5
```

8: You are asked to take a DNA sequence and return a string of RNA sequence by replacing all the Ts with Us. Do this in two ways:

8.a: Use a single string method to accomplish this. (5 points)

```
DNA_sequence = 'AGAAGGCCCTAGAGTCCAAGACTCAGATTCAGAATCGACTC'
```

```
DNA_sequence.replace("T", "U")
```

```
'AGAAGGCCCUAGAGUCCAAGACUCAGAUUCAGAAUCGACUC'
```

8.b: Use a combination of string's split and join methods to accomplish this. You can use multiple lines f code if necessary. (5 points)

```
DNA_sequence = 'AGAAGGCCCTAGAGTCCAAGACTCAGATTCAGAATCGACTC'
```

```
"U".join(DNA_sequence.split('T'))
```

```
'AGAAGGCCCUAGAGUCCAAGACUCAGAUUCAGAAUCGACUC'
```

9: There are four points to this question. (2.5 points each)

9.a: Create a list of numbers between 1 and 15. Store this list as "a"

```
a = list(range(0,15))
```

```
a
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

9.b: Use a list comprehension to convert the numbers to strings and add a letter x in front of each. Store the resulting list as b.

```
b = ['x' + str(number) for number in a]
```

```
print(b)
```

```
['x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10', 'x11', 'x12', 'x13', 'x14']
```

9.c: Use a string method to concatenate the list from part b into a single string separated by two periods. Store the resulting string as c.

```
c = '..'.join(b)
```

```
c
```

```
'x0..x1..x2..x3..x4..x5..x6..x7..x8..x9..x10..x11..x12..x13..x14'
```

9.d: Chain all of these commands together into a single line that accomplishes all three tasks.

```
'..'.join(['x' + str(number) for number in list(range(0,15))])
```

```
'x0..x1..x2..x3..x4..x5..x6..x7..x8..x9..x10..x11..x12..x13..x14'
```

10: You are given a list of tasks by your professor: (2.5 points each)

```
tasks = ['Read Lecture', 'Go to class', 'Work on homework']
```

10.a: Sort the list in alphabetical order.

```
tasks.sort()
```

```
tasks
```

```
['Go to class', 'Read Lecture', 'Work on homework']
```

10.b: You are given a string representing an additional single task. Add this task to the beginning of the list.

```
new_task = 'Start computer'
```

```
tasks.insert(0, new_task)
```

```
tasks
```

```
['Start computer', 'Go to class', 'Read Lecture', 'Work on homework']
```

10.c: Add this task to the end of the list.

```
tasks.pop(0)
```

```
'Start computer'
```

```
tasks
```

```
['Go to class', 'Read Lecture', 'Work on homework']
```

```
tasks.append(new_task)
```

```
tasks
```

```
['Go to class', 'Read Lecture', 'Work on homework', 'Start computer']
```

10.d: You are now given a new list of tasks. Add these tasks to the list of tasks.

```
new_tasks = ['Study', 'Check grade']
```

```
tasks.extend(new_tasks)
```

```
print(tasks)
```

```
['Go to class', 'Read Lecture', 'Work on homework', 'Start computer', 'Study', 'Check grade']
```