MISE Summer Programming Camp

Class 4 Worksheet - Collecting Information

This worksheet has 3 sections, which should take you around 1 hour to complete. Section 1 contains a series of short answer questions about what we did in class. You'll have to run some code in your IDE to see the result and write it down. Section 2 will introduce a new topic we haven't seen in class, and so it is optional. You'll be guided through it since it doesn't assume knowledge of the topic. Finally on Section 3 you'll have to write some programs and submit them to CodeForces.

Note that Section 2 will teach you a very important skill of researching new Python features on your own. Even though it is optional, it is one of the most important optional sections and we recommend you look at it.

Optional preliminary question: In class we mentioned the modulo operator `%` and how to use it to obtain the last digit of a number. If you want to learn more about it look at the following link: https://www.freecodecamp.org/news/the-python-modulo-operator-what-does-the-symbol-mean-in-python-solved/.

1. Review Questions

Question 1. We learned about how we can nest loops to repeat some computation that itself repeats something. We saw some examples of things we can do with nested loops, but let's look at some more. Each of the following is a short task followed by a program that solves it. Read the task and think about how you'd do it. Then read the solution and verify that it does what it's supposed to.

a. Given an integer **n**, print a pyramid of the symbol '**#**' with height **n**. For example, for **n** = 4 you should print:

```
#
##
###
####
```

```
n = int(input())
for i in range(n):
    for j in range(i + 1):
        print('#', end='')
    print()
```

b. Given an integer **n**, print the multiplication table of the first 10 multiples of the first **n** positive integers.

```
n = int(input())
for i in range(1, n + 1):
    for j in range(1, 11):
        print(i * j, end=' ')
    print()
```

c. You are given an integer **n**. Print all pairs of integers that sum to **n**.

```
n = int(input())
for i in range(n + 1):
    for j in range(n + 1):
        if i + j == n:
            print(i, j)
```

Question 2. In class we learned a bunch of different properties about lists, including different built in functions that allow us to alter them somehow or to learn something about them. In this question we are going to look at a couple more. For each of the following you'll have some list variable followed by code that does something to or with the list. Write these on your IDE and run them. Try to understand what they do. If you don't try the same examples but with different initial lists to see if you can find a pattern.

a.

```
a = ["Ghana"] * 5
print(a)
```

b.

```
a = [5, 1, 8, 10, 3]
print(sum(a))
print(min(a))
print(max(a))
```

c.

```
a = [5, 1, 8, 10, 3]
print(a[-1])
print(a[-3])
```

d.

```
a = [5, 1, 8, 10, 3]
print(a[1:2])
print(a[2:5])
```

e.

```
a = [5, 1, 8, 10, 3]
print(8 in a)
print(9 in a)
```

f.

```
a = [5, 1, 8, 10, 3]
a.sort()
print(a)
```

g.

```
a = [5, 1] + [8] + [10]
a = a + [3]
print(a)
```

h.

```
a = [5, 1, 8, 10, 3]
a.pop(2)
print(a)
a.pop(0)
print(a)
```

Question 3. Each of the following is a short programming task followed by a short program that solves it. Read the prompts and think about how you'd solve them. Then read the solution and verify that they actually do so.

a. You are given a string. Determine if it is a palindrome (which means it is the same as its reverse).

```
s = input()
for i in range(len(s)):
    if s[i] != #<fill in the blank>#:
        print(s + " is not a palindrome")
        break
else:
    print(s + "is a palindrome")
```

b. Implement a function that takes in a list of integers and returns the maximum of all of them. (we did this in class)

```
def max(l):
    maxSoFar = l[0]
    for i in l:
        if i > maxSoFar:
            maxSoFar = i
    return maxSoFar

print(max([1, 2, 3]))
```

c. Write a program that reads a line of integers separated by spaces and turns that into a list with those values.

```
line = input()
l = []
for i in line.split():
    l.append(int(i))
print(l)
```

d. Given a list of integers, write a function that determines if the list is in ascending sorted order.

```
def isSorted(l):
    for i in range(1, len(l)):
        if l[i] < l[i - 1]:
            return False
    return True

print(isSorted([1, 2, 3]))
print(isSorted([1, 3, 2]))
```

(Optional Question) In class we didn't have time to learn about list references, but this question mentions this concept, so it is encouraged to look back at the relevant slides. The key idea is that a list variable is actually a pointer or alias to a list. This also happens in functions, meaning the parameters of the function are actually a reference to the list and not a copy of the list. Check the following example.

```
1 def listFunction(l):
2     l[0] = 1
3 a = [10, 10, 10]
4 print(a)
5 listFunction(a)
6 print(a)
```

```
[10, 10, 10]
[1, 10, 10]
>
```

As we did in class, we can prevent this by creating a copy of the list, using the **.copy()** function or the **list()** function.

2. Reading Python Documentation (optional)

We learned a lot of Python built-in functions for lists, but are we supposed to memorize all of them? And how do we know what other functions are there?

Python has a webpage with comprehensive documentation about all you can do with lists, but also with all other data types (there are a lot that we didn't cover!). So, to learn how to consult the documentation we are going to do that with the string data type, which we already know quite a bit about.

Question 1. There are two sources of Python documentation. The first one is called The Python Standard Library and you can find it here: https://docs.python.org/3/library/index.html. It reads like an encyclopedia, you can look up all sorts of things and find all functions and properties regarding them. The second one is called The Python Tutorial and you can find it here: https://docs.python.org/3/tutorial/index.html. It reads like a book, it's better suited if you want to learn a lot about a particular thing (like lists) and it isn't as good if you want to find a specific function. While we encourage you to look at both in your own time, we will use the first one (the standard library) for the remaining questions.
Try to find the entry about strings. Recall that a string is a built-in data type that is a text sequence. Click on its link. You should now be here https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str.

Question 2. If you scroll down you'll find a section called "String Methods". This contains all existing functions that you can use with strings and how to use them. Let's look at a couple.

a. The first one is called **capitalize**. Read through its description. Here is an example of how to use it:

```
1 s = "hello world"
2 print(s.capitalize())
```

(Note how the description says it returns a copy of the string. Recall that strings are immutable, so you can't change the original string, you can only return copies of it).

b. Find a method called **count**. Read through its description. Here is an example of how to use it:

```
s = "anna has an apple and a banana"
print(s.count("an"))
```

c. Look up the **find** and the **index** methods. Read their descriptions and try to think about why they are different. Here is an example of how to use them:

```
s = "anna has an apple and a banana"
print(s.index("app"))
print(s.find("app"))
print(s.find("al"))
```

d. Find the **split** method. Read its description. This is one of the most useful methods you'll learn about. The most common use case for it is the following: suppose you are given a list of integers in one line, separated by spaces. Read that into a Python list of integers.

```
s = input()
string_list = s.split()
int_list = []
for i in range(len(string_list)):
    int_list.append(int(string_list[i]))
print(int_list)
```

(Note that there are easier ways to do this, but they require some extra things we haven't learned. If you are curious, try looking up the **map** function that is built-in in Python).

3. Coding Questions

You should see a new problem set called "Class 4 Problems" (https://codeforces.com/group/K1Fxw6skwV/contest/444755). You should solve the first 2 problems ("Football season" and "Second max"). There are 2 extra problems ("Checkerboard" and "Fibonacci") that are a bit harder, but they're not mandatory. There are also 2 challenge problems ("Advantage" and "Triple") that are extra hard and so also optional.