Your Name:	TJ Sipin
Perm #:	3430501

This homework and its due date are posted on the course website: https://ucsb-cs8.github.io/w20/hwk. Submit a PDF file of this assignment following the guidelines posted on the website.

h07: 5.2 (for Loop / Iteration Patterns), 4.3 (Files) Review: 3.2 (Execution Control Structures), 3.3 (User-Defined Functions), 3.4 (Python Variables and Assignments), 3.5 (Parameter Passing)

0. (5 points) Use the provided homework template (without any blank pages), filling out all requested fields, and correctly submitting a legible electronic document on Gradescope *before the due date*. By submitting this document you agree that this is your own work and that you have read the policies regarding Academic Integrity: https://studentconduct.sa.ucsb.edu/academic-integrity.

1. (4 pts) In **Section 5.2**, the author discusses the `for` loops and its patterns. For the Python code in the left box, write the corresponding output in the right box. (*As always, pay attention to the command format and try to predict what the output would be, before running it: if the result differs from what you expected, figure out where and why it happened.*)

```
the_list = [0,1,1,2,3,5,8]
for fib_n in the_list:
   print(fib_n, end=".")
```

```
0.1.1.2.3.5.8.
```

```
the_list = ["AZ", "CA", "HI"]
for state in the_list:
   print(state)
```

```
AZ
CA
HI
```

2.1 (1 pt) For HW04, you must've read **Section 3.2**, which discusses how to use the `range` function with a loop so that you can cause a variable to take on a sequence of values. Re-read this section and review the sample code that illustrates how to generate and print the values of a range.

In the space below, show the output of running the following command in IDLE. (*As always, try to predict what the output would be, before running it: if the result differs from what you expected, figure out where and why it happened.*)

```
>>> range(5, 14, 3)
range(5, 14, 3)
```

2.2 (3 pts) How can you rewrite the above `range` statement to be able to see the elements that will be generated as part of this range? Show the output that would be produced by the re-written statement.

```
>>> for x in range(5, 14, 3):
        print(x)
5
8
11
```

3. (7 pts) Remember that the function call `range(start, end, step)` can be used to iterate over the sequence of integers starting at `start`, using a step size of `step`, and ending before `end`.

The first element of the sequence is *always* the value in `start`, *unless* `start >=` end ______ , in which case, the resulting sequence will be **empty**.

Show an example `range` statement that will produce an empty sequence:

range(1,0,1)

If we successfully produced the first element to the sequence, then to get the next/*second* element of that sequence, we need to add (i.e., +) step ______ to the value of `start` and include the result in the sequence *only if* that element is < ______ `end`.

To get each subsequent element of that sequence, we need to set the step = 1

and include the result in the sequence only if it's less than the end (in the direction that you're going)

4. (5 pts) In **Section 5.2**, the author discusses the `for` loops. Write two lines of Python that use a `for` loop with the `range` function to print the values `5, 4, 3, 2, 1` all **on one line**. (Use `help(print)` to see how to use either `end` or `sep` parameter to add commas between numbers; note: in order for it to be **no comma** at the end, after 1, we need one last `print` that runs after the `for` loop).

```
for var in range(5,1,-1________________):
    print( var________ , end____ = ', ')
print( 1__________ )
```

5.1 (10 pts) Carefully read **Section 3.3**, to make sure you understand the section about the *Function Input Arguments*. Given the following formula for the area of a circle ($\text{Area} = \pi * \text{radius}^2$) write a function `areaOfCircle` takes `radius` as an input parameter and returns the resulting `area`. Remember to import math within your function to use `math.pi`.

```
def areaOfCircle(radius):
    import math
    area = math.pi*radius**2
    return area
```

5.2 (4 pts) Show how to call your function to compute the area for a circle of radius 5 and 8. (No need to show the result of the function call.)

```
>>>   areaOfCircle(5)

>>>   areaOfCircle(8)
```

5.3 (6 pts) Write a `for` loop to verify that your function is correctly returning the expected output for the radius values between 0 and 11. (Remember the issue you might run into when comparing floating point values? What do you need to do to correctly handle it in the pytest framework? Aside: here's a good resource that provides a short example: https://0.30000000000000004.com).

```
def test_areaOfCircle_0_11():

  import math

  for radius in range(0,12):

    assert areaOfCircle(radius) == pytest.approx(math.pi * radius **2)
```

6.1 (9 pts) What is the output of calling these functions? Write it down in the box on the right. (Remember that `return` **takes you out of the function *immediately*.)**

`def loop1():`
`      for i in range(5):`
`          print(i)`
`      return i`
	`>>> loop1()`
	0
	1
	2
	3
	4
	4

```
def loop2():
    for i in range(5):
        if(i == 3):
            return i
        print(i)
    return i
```
	```
	>>> loop2()
	0
	1
	2
	3
	```

```
def loop3():
    for i in range(5):
        if(i == 3):
            break
    print(i)
    return i
```
	```
	>>> loop3()
	3
	3
	```

6.2 (2 pts) Now assume `val1 = loop1()` and `val2 = loop2()`. For the Python code in the left box, write the output of the lines in the right box.

`print(val1)`	4
`print(val2)`	3

6.3 (3 pts) Briefly explain why the output boxes above have different values.

The final value of each function loop1() and loop2() is whatever was said to be returned, so that's what the output shows.

7. (3 pts) After reading **Section 3.4**, which of the following statements seems more accurate in Python?

[✓ A.] The assignment `var = 42`, assigns `var` to an integer 42.
B. The assignment `var = 42`, changes an object in `var` to be an integer object 42.
C. The assignment `var = 42`, makes `var` refer to an integer object 42.

Why do you believe your selection is correct?
A Python assignment statement uses the syntax:

<variable> = <expression>

so var is the <variable> and 42 is the <expression>

8. (10 pts) In **Section 3.4**, the author explains the difference between a mutable and an immutable object. Explain the difference in your own words and illustrate with your own *short* example.

Mutable	**Immutable**
The object can be modified a = 6 a can be modified so that its value is changed. For example, we can say a = 3838268149 to change what a refers to (from 6 to 3838268419).	The object cannot be modified The object 2 cannot be modified. For example, we cannot say 2 = 39.

9. (10 pts) Read **Sections 3.4** and **3.5** carefully. Please read those two sections several times and try to understand every detail. Given the following function definition and function call, write down the output of the `print(myList1)` and `print(myList2)`.
Use the space on the right of the code to explain what happens after the code is run.
(Note: After you read the sections, you can use pythontutor.com to help you visualize the execution).

```
def replace_first(aList):
    aList[0] = 42

myList1 = [8]
myList2 = [3, 6, 9, 12]

replace_first(myList1)
print(myList1)
```

[42]
____

```
replace_first(myList2)
print(myList2)
```

[42, 6, 9, 12]
____

10. (15 pts) Given the following function definition and call, write down the output of this program.

```
def swap_first(aList, var):
     print("Inside swap_first, before the swap")
     print("Var =", var, "list =", aList)
     tmp = aList[0]
     aList[0] = var
     var = tmp
     print("Inside swap_first, after the swap")
     print("Var =", var, "list =", aList)

>>> myList = [3, 6, 9, 12]
>>> myVar = 8
>>> print("Before swap_first")
Before swap_first
>>> print("Var =", myVar, "list =", myList)
Var = 8______ list = [3, 6, 9, 12]______________
>>> swap_first(myList, myVar)
Inside swap_first, before the swap
Var = 8______ list = [3, 6, 9, 12]______________
Inside swap_first, after the swap
Var = 3______ list = [8, 6, 9, 12]______________
>>> print("After swap_first")
After swap_first
>>> print("Var =", myVar, "list =", myList)
Var = 8______ list = [8, 6, 9, 12]______________
```

Explain what happens to `myList` and `myVar` ***inside*** the function and ***after*** the code is run. **What causes and explains this behavior?**

swap_first() did not cause myVar to change values because the function swap_first() cannot modify the value of Var in the interactive shell, because generally functions will not modify the value of any variable passed as a function arguement if the variable refers to an immutable object. In this case, the variable refers to an integer. myList, however, is a list so it is mutable.

11.1 (3 pts) Read **Section 4.3** about Files. Every file on a file system can be referred to with an "absolute pathname". Where does the absolute path start and what does it consist of?

The absolute pathname starts right at the root directory, consisting of the sequence of folders that follow the root directory up to the file we are referring to.

11.2 (3 pts) In contrast to an "absolute pathname", we have the concept of a "relative pathname". What is the technical term used for the "starting point" of a "relative pathname"?

The current working directory

12.1 (2 pts) In **Section 4.3**, pages 112-114 discuss four ways of reading text from a file. The first way to read text from a file is shown in the listing at the bottom of page 112. On line 4 of that listing we see: `content = infile.read()`

After this line of code is executed, what would `type(content)` return?

<class 'str'>

12.2 (5 pts) The second way to read text from a file is shown in the middle of p. 113. On line 7 of that listing we see: `wordList = content.split()`
What does the `.split()` method do, and what is stored in `wordList` as a result?
.split() splits the content into a list of words. wordList stores the list of words.

12.2 (2 pts) The third way is shown in the listing near the top of p. 114. On line 4 of that listing we see: `lineList = infile.readlines()`

After this line of code is executed, what would `type(lineList)` return?

<class 'list'>