Topics: Continue with function definitions, Procedure vs Function, Function calls(Stack- global space, call frame and local variables)
Some references for Function calls:
https://www.cs.ucsb.edu/~pconrad/cs8/topics.beta/theStack/01/
Lecture slides available at : http://www.cs.cornell.edu/courses/cs1110/2016sp/lectures/02-18-16/6.FuncMechanics.pdf

Problem 1 [String slicing]
Define a function, called leftrotate, that takes a string parameter and returns a string rotated left k times.

leftrotate('Hello',2) should return 'lloHe'
leftrotate('Hi',1) should return 'iH'
leftrotate('Python',20) should return 'thonPy'

Problem 2 [String slicing]
Define a function, called fitin, that takes two parameters: a tag string of length 4, such as "{{}}", and a word. The function should return a new string where the word is in the middle of the tag string, e.g. "{{word}}".

fitin('(())', 'Hello') returns '((Hello))'
fitin('<<>>', 'Yikes') returns '<<Yikes>>'
fitin('[[]]', 'Black') returns '[[Black]]'

Problem 3
Given 3 int values, a b c, return their sum. However, if any of the values is a teen -- in the range 13..19 inclusive -- then that value counts as 0, except 15 and 16 do not count as a teens. Write a separate helper "def fix_teen(n):"that takes in an int value and returns that value fixed for the teen rule. In this way, you avoid repeating the teen code 3 times (i.e. "decomposition"). Define the helper below and at the same indent level as the main no_teen_sum().

no_teen_sum(1, 2, 3) returns 6
no_teen_sum(2, 13, 1) returns 3
no_teen_sum(2, 1, 14) returns 3
no_teen_sum(2,1,15) returns 18

Problem 4

For a script that calls the function no_teen_sum , implemented previously, explain the execution by defining the state of global space and call frame stacks at each line.
Let's say we made this call: no_teen_sum(2, 1, 14)

Problem 5

Explain the execution of this script, going line by line, by maintaining the global space and call frame stack and tell the output printed on the screen for this script:

```
1     def foo():
2         print("foo line 1")
3         print("foo line 2")
4         print("foo line 3")
5
6     def fum():
7         print("fum line 1")
8         print("fum line 2")
9         print("fum line 3")
10
11    def bar():
12        print("bar line 1")
13        fum()
14        foo()
15        print("bar line 4")
16
17    def go():
18        bar()
19
20    go()
```