Week 3 Function Scope and Recursion

What are the outputs of the code snippets in Part 1 and 2?

Part 1 Variable Scope

<table>
<tr><th>Code</th><th>Output</th></tr>
<tr><td><pre>
x = 0
def foo_printx():
      print(x)
foo_printx()
print(x)
</pre></td><td></td></tr>
<tr><td><pre>
x = 0
y = 999
def foo_printx(y):
      print(y)
foo_printx(x)
print(x)
</pre></td><td></td></tr>
<tr><td><pre>
x = 0
def foo_printx():
      x = 999
      print(x)
foo_printx()
print(x)
</pre></td><td></td></tr>
</table>

Part 2 Nested Functions

<table>
<tr><th>Code</th><th>Output</th></tr>
<tr><td><pre>
x = 1
y = 2
def foo(y):
      def bar(x):
            return x+y
      return bar(y)
print(foo(x))
</pre></td><td></td></tr>
<tr><td><pre>
x = 1
y = 2
def foo(x):
      def bar(x):
            return x+y
      return bar(y)
print(foo(x))
</pre></td><td></td></tr>
</table>

Part 3 Recursion

Previously, we have shown that we can create a customized burger. Here are the ingredient prices for your convenience:

Ingredient	**Price**
'B' stands for a piece of bun	\$0.5
'C' stands for cheese	\$0.8
'P' stands for patty	\$1.5
'V' stands for veggies	\$0.7
'O' stands for onions	\$0.4
'M' stands for mushroom	\$0.9

Your task was to write a function `burgerPrice(burger)` that takes in a string representing a burger and returns the price of the burger. Your task now is to write the same function `burgerPrice(burger)` using recursion.

Part 4 Recursion vs Iteration

A. Sum: Given a positive number n, the sum of all of its digits is obtained by adding the digits one by one. For example, the sum of 52634 is $5 + 2 + 6 + 3 + 4 = 20$. Write one ***recursive*** and one ***iterative*** version of function `sum(n)` which returns the sum of all the digits in n. You may assume that `n > 0`.

B. Factorial: Given a positive number n, the value of factorial of n (written as $n!$) is defined as $n! = n \times (n-1)!$. Additionally, the value of $0!$ is 1. Write one ***recursive*** and one ***iterative*** version of function `fact(n)` which computes the value of `n!.`

Part 5 Recursion vs Iteration (cont.)

A. Final Sum: Given a positive number n, the final sum is obtained by repeatedly computing the sum of all the digits of n, until the final sum is a single digit. For example, `sum(52634) = 20`, which is not a single digit. We then continue with `sum(20) = 2` which is now a single digit. Therefore, `final_sum(52634) = 2`. Write one ***recursive*** and one ***iterative*** version of the function `final_sum(n)` which computes the final sum of n.

B. Euler Constant: The value of e^x can be approximated using the formula $e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$. Write one ***recursive*** and one ***iterative*** version of function `find_e(x, n)` to find the approximation of e^x up to $n + 1$ steps, i.e. the last term in the summation is $\frac{x^n}{n!}$.