CS 1358 Introduction to Programming in Python

Spring Semester 2024

Prof. Pai H. Chou

Self-Check 8

Answer the following questions to check your understanding of your material. Expect the same kind of questions to show up on your tests.

## 1. Definitions and Short Answers - functions

1. Given a function definition  
   def Double(n):  
    return n + n  
   and given a call to the function  
   x = Double(20)  
   What is a **formal parameter**? What is the **actual parameter**?
2. If you call the above function as  
   y = Double( Double(40 + Double(3)) - Double(7))  
   what is the value of y?
3. What is the difference between  
   d = Double  
   and  
   z = Double(20)  
   ?
4. If you do   
   print, input = input, print  
   z = input('enter a name: ')  
   What happens, and what is the value of z?
5. Given the function  
   def DivMod(a, b):  
    return a // b, a % b  
   What does this function return?
6. Are the following pairs equivalent (in terms of parameters received by the open() function for opening files)?
   1. fh = open('myfile.txt', 'r') and   
      fh = open(file='myfile.txt', mode='r')
   2. fh = open('myfile.txt', 'r') and  
      fh = open(mode = 'r', name = 'myfile.txt')
   3. fh = open('myfile.txt', 'r') and  
      fh = open('r', 'myfile.txt')
   4. fh = open(file='myfile.txt', mode='r') and  
      fh = open(mode='r', file='myfile.txt')
7. Consider the function declaration  
    1 def withTax(price, rate=0.05):  
    2 return price \* (1 + rate)  
   Can you call this function in the following ways? If so, what is the result? If not, why not?
   1. withTax(20)
   2. withTax(rate=0.08, price=30)
   3. withTax(price=20, 0.08)
   4. withTax(price=20)
   5. withTax(rate=0.08)
   6. withTax()
   7. withTax(20, 0.08)
8. Suppose you are given a function  
    1 def withMoreTax(price, rate = 0.05):  
    2 ans = price \* (1 + rate)  
    3 rate += 0.01  
    4 return ans  
     
   What does the following code sequence print?  
   print(withMoreTax(20))  
   print(withMoreTax(20))  
   In other words, does the parameter rate's **default value** get changed on line 3 such that the next call gets the modified default value?
9. Are you allowed to define default values with non-constant expressions such as the following? If so, what does it do?
   1. 1 import sys  
       2 def withNewTax(price, rate=input('enter a rate: ')):  
       3 return price \* (1 + float(rate))
   2. 1 def withNewerTax(price, rates=[0.01, 0.02, 0.03]):  
       2 ans = price \* (1 + rates[-1])  
       3 rates.pop()  
       4 return ans
   3. 1 def withNewestTax(price, rate=price):  
       2 return price \* (1 + rate)
10. Suppose you have a function declared as  
    def totalTax(rate, \*priceTags):  
     ...  
    What is the value of rate and priceTags inside the function totalTax when you call it as
    1. totalTax(0.05, 10, 20, 23, 18)
    2. totalTax(0.05, (10, 20, 23, 18))
    3. totalTax(0.05, \*(10, 20, 23, 18))
    4. totalTax(\*(0.05, 10, 20, 23, 18))
    5. totalTax(0.05)
    6. totalTax(0.05, 10)
11. Given the following function definition:  
     1 def totalTax(rate, \*\*priceDict):  
     2 return sum(priceDict.values())\* (1 + rate)  
    Are the following valid ways of calling the function? If so, what are the **values of parameters** rate and priceDict while inside the totalTax function, and what is the corresponding **return value**? If not valid, why not?
    1. totalTax(0.05, apple=12, orange=8)
    2. totalTax(0.05, 12, 8)
    3. totalTax(0.05, 'apple'=12, 'orange'=8)
    4. totalTax(0.05, 'apple':12, 'orange':8)
    5. totalTax(rate=0.05, priceDict={'apple':12, 'orange':8})
    6. totalTax(rate=0.05, \*\*{'apple':12, 'orange':8})
    7. totalTax(rate=0.05, \*\*priceDict={'apple':12, 'orange':8})
    8. totalTax(rate=0.05, priceDict=(('apple', 12), \  
        ('orange', 8))
    9. totalTax(rate=0.05, \*\*priceDict=(('apple', 12), \  
        ('orange', 8))
    10. totalTax(apples=12, oranges=8, rates=0.05)
12. Given the following function definition  
     1 def totalWithTax(rate, \*items, \*\*priceDict):  
     2 d={'apple':12,'orange':8, 'mango':3}

3 d.update(priceDict)  
 4 total = 0  
 5 for i in items:  
 6 total += d[i]  
 7 return total \* (1 + rate)

Are the following valid ways of calling the function? If so, what are the **values of parameters** rate, items, and priceDict while inside the totalWithTax function, and what is the corresponding **return value**? If not valid, why not?

* 1. totalWithTax(0.05, 'apple', 'orange', 'mango')
  2. totalWithTax(0.05, 'apple', 'orange', 'mango', \  
      'apple', 'apple')
  3. totalWithTax(0.05, 'apple', 'orange', 'mango', \  
      apple=3, orange=4, mango=5)
  4. totalWithTax(0.05, 'apple', 'orange', 'mango',\  
      \*\*{'apple':3, 'orange':4, 'mango':5})
  5. totalWithTax(0.05, \*('apple', 'orange', 'mango'))
  6. totalWithTax(0.05)

1. Suppose you have two tuples  
   A = (1, 3, 7)  
   B = (2, 4, 8)  
   and you want to use the built-in function max() to find the largest element in the two tuples. Which of the following will correctly find the answer (which is 8)? Choose all that apply and explain why or why not.
   1. max(A, B)
   2. max(\*A, \*B)
   3. max(A + B)
   4. max(\*(A+B))
   5. max(\*A + \*B)
   6. max((A, B))
   7. max(\*(A, B))
2. Given the following code  
    1 a = 3

2 def F():

3 print(a)  
 4 F()

What is printed when you run this code?

1. Given the following code  
    1 a = 3

2 def F():

3 a = 5

4 print(a)   
 5 F()

6 print(a)  
  
What is printed when you run this code?

1. Given the following code  
    1 a = 3

2 def F():

3 print(a)  
 4 a = 5

5 F()

6 print(a)  
But it gives an UnboundLocalError when you try to run it.

* 1. Why do you get this error?
  2. How do you fix it if you want the function F to use the identifier a as defined on line 1?

1. Given the source code  
    1 D = { 'rate': 0.0 }

2 def totalWithTax(\*names, \*\*kv):

3 global D

4 total = 0.0

5 for name in names:

6 total += D[name]

7 for kw, val in kv.items():

8 D[kw] = val # overwrite dict entry   
 9 if kw != 'rate':

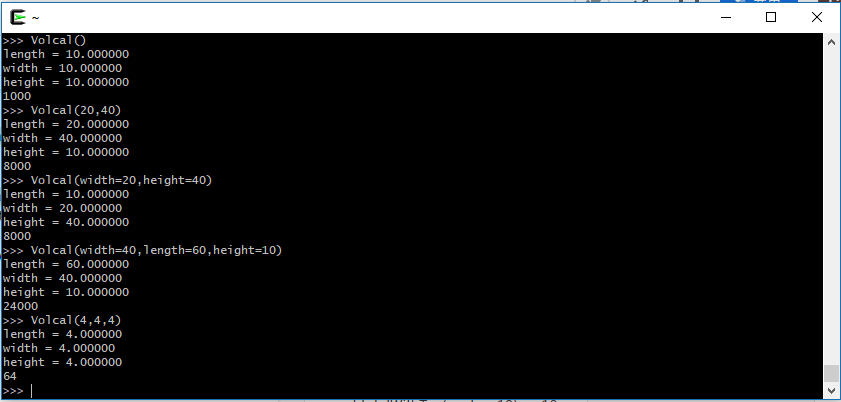
10 total += val  
11 return total \* (1 + D['rate'])  
Are the following valid ways of calling the function totalWithTax? If so, what are the **values of parameters** names, and kv while inside the totalWithTax function, the values of the **global** variable D, and what is the corresponding **return value**? If not valid, why not? Assume you reload the code each time before executing the code below.

* 1. totalWithTax()
  2. totalWithTax('apple')
  3. totalWithTax(apple=20)+totalWithTax('apple')
  4. totalWithTax('orange', orange=15)
  5. totalWithTax(guava)
  6. totalWithTax(rate=0.05, apple=10) + totalWithTax('apple', 'apple', 'apple')
  7. totalWithTax(apple=10, orange=15)+totalWithTax('apple',guava=12)
  8. totalWithTax(apple=10, orange=15) + totalWithTax(rate=0.05, guava=12) + totalWithTax(rate=0.02, 'apple', 'orange', 'guava')

1. How do you write **test code** at the end of a module to test functions defined in the module when it is run as a top-level module (hint: '\_\_main\_\_'), but don't run the test case when it is imported by another module? Also, what construct should you use to check if a tested function returns the results as the correct answer?

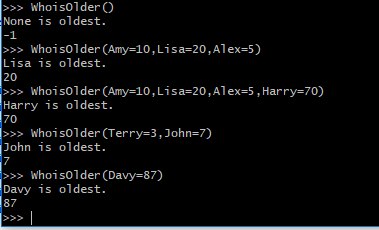
## 3. Programming

1. (Difficulty: ★☆☆☆☆) Write a function which named Volcal() that can take three parameters to calculate the volume of cuboid. Three parameters includes length, width and height. Please set their default value as (10,10,10),and the function will print their length,width and height and return coboid’s value.



1. (Difficulty: ★★☆☆☆) Write a function that can take a variable number of parameters to decide who is older. The parameter list consists of many people’s names and their ages. the function need to print who is oldest and return oldest person’s age.If the parameter is empty,you need to return -1.

[Note] You don’t consider that 2 people have the same age.



1. (Difficulty: ★★★☆☆) Write a function that can take a variable number of parameters to make a postfix calculator. The parameter list consists of either the operands or the operators. An operand is a number (int or float) and is pushed on the stack. An operator is a string that indicates the action to take. A binary arithmetic operator pops the top two elements from the stack and pushes back the result.

| **argument** | **action** |
| --- | --- |
| int or float literal | push int or float literal onto stack |
| 'add' | A = pop(); B = pop(); push(A+B) |
| 'sub' | A= pop(); B = pop(), push(A-B) |
| 'mul' | A=pop(); B = pop(); push(A\*B) |
| 'swap' | A=pop(); B=pop(); push(A); push(B) |

>>> postcalc(1, 2, 3, 4)  
[1, 2, 3, 4]  
>>> postcalc(1, 2, 3, 4, 'add')  
[1, 2, 7]  
>>> postcalc(1, 2, 'add')  
[3]  
>>> postcalc(1, 2, 3, 'add', 'sub')  
[4]  
>>> postcalc(2, 3, 'add', 4, 'mul', 5, 'swap', 'sub')  
[15]  
  
In addition, it should be able to take an optional stack in the form of a list of numbers.  
  
>>> postcalc(1, stack=[2, 3, 4])  
[2, 3, 4, 1]  
>>> postcalc(3, 4, 'add', stack=[1, 2])  
[1, 2, 7]  
>>> postcalc('add', stack=[1, 2])  
[3]  
>>> postcalc('add', 'sub', stack=[1, 2, 3])  
[4]  
>>> postcalc('add', 4, 'mul', 5, 'swap', 'sub', stack=[2, 3])  
[15]

Before solving this problem, decide on how should the formal parameters of this function be declared? Should you use variable-length arguments? arguments with default value?