EECS 1210 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 definiton
   ```
   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**?
   formal parameter is n, actual parameter is 20
2. If you call the above function as
   ```
   y = Double( Double(40 + Double(3)) - Double(7))
   ```
   what is the value of y?
   156
3. What is the difference between
   ```
   d = Double
   ```
   and
   ```
   z = Double(20)
   ```
   ?
   d is now referring to the function Double itself, so it can be called like d(20), z is the result of calling the Double function with parameter of 20, which has the value 40 in this case.
4. If you do
   ```
   print, input = input, print
   z = input('enter a name: ')
   ```
   What happens, and what is the value of z?
   the names of print and input are now defined to be the built-in functions input and print. This means the code prints the string 'enter a name: ' and z gets assigned the return value, which is None.
5. Given the function
   ```
   def DivMod(a, b):
       return a // b, a % b
   ```
   What does this function return?
   a tuple with the (quotient, remainder)

6. Are the following pairs equivalent (in terms of parameters received by the open() function for open ing files)?
   a. fh = open('myfile.txt', 'r') and
      fh = open(file='myfile.txt', mode='r')
   b. fh = open('myfile.txt', 'r') and
      fh = open(mode = 'r', name = 'myfile.txt')
   c. fh = open('myfile.txt', 'r') and
      fh = open('r', 'myfile.txt')
   d. fh = open(file='myfile.txt', mode='r') and
      fh = open(mode='r', file='myfile.txt')

   equivalent: (a), (d)

   (b)

   (c) not equivalent because both are passed by position but in different order.
7. Consider the function declaration

```
def withTax(price, rate=0.05):
    return price * (1 + rate)
```

   Can you call this function in the following ways? If so, what is the result? If not, why not?
   a. withTax(20) 21.0
   b. withTax(rate=0.08, price=30) 32.400000000000006
   c. withTax(price=20, 0.08)
      SyntaxError: positional argument follows keyword argument
   d. withTax(price=20)21.0
   e. withTax(rate=0.08)
      TypeError: withTax() missing 1 required positional argument: 'price'
   f. withTax()
      TypeError: withTax() missing 1 required positional argument: 'price'
   g. withTax(20, 0.08) 21.6
8. Suppose you are given a function

```
def withMoreTax(price, rate = 0.05):
    ans = price * (1 + rate)
    rate += 0.01
    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?

   Nothing changed.

   Because of that rate is a local variable.
9. Are you allowed to define default values with non-constant expressions such as the following? If s o, what does it do?

a.
```
import sys
def withNewTax(price, rate=input('enter a rate: ')):
    return price * (1 + float(rate))
```
b.
```
def withNewerTax(price, rates=[0.01, 0.02, 0.03]):
    ans = price * (1 + rates[-1])
    rates.pop()
    return ans
```
c.
```
def withNewestTax(price, rate=price):
    return price * (1 + rate)
```

part a: at the end of defining the function, input() prompts the user to type a rate (float literal). All the subsequent calls to withNewTax will use the rate input by the user as default value; it does not prompt the user each time the function is called.

part b: rates get re-initialized to the same list [0.01, 0.02, 0.03] each time the function is called; line 3 rates.pop() only modifies the specific default instance but it does not affect subsequent calls to the same function -- all of them get default rates=[0.01, 0.02, 0.03].
怪怪的
https://blog.chairco.me/posts/2017/07/3%20mistakes%20to%20avoid%20when%20learning%20to%20code%20in%20Python.html

part c is not allowed, because a parameter with default value must be initialized once to a known value at the time it is defined; it can't be initialized to another (formal) parameter since the function hasn't been called.

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

a. totalTax(0.05, 10, 20, 23, 18)
   rate is 0.050000
   priceTags is (10, 20, 23, 18)
b. totalTax(0.05, (10, 20, 23, 18))
   rate is 0.050000
   priceTags is ((10, 20, 23, 18),)
c. totalTax(0.05, *(10, 20, 23, 18))
   rate is 0.050000
   priceTags is (10, 20, 23, 18)
d. totalTax(*(0.05, 10, 20, 23, 18))
   rate is 0.050000
   priceTags is (10, 20, 23, 18)
e. totalTax(0.05)
   rate is 0.050000

priceTags is ()

f. `totalTax(0.05, 10)`

rate is 0.050000

priceTags is (10,)

11. Given the following function definition:

```
def totalTax(rate, **priceDict):
    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 no t valid, why not?

a. `totalTax(0.05, apple=12, orange=8)`

rate is 0.050000

priceDict is {'apple': 12, 'orange': 8}

21.0

b. `totalTax(0.05, 12, 8)`

TypeError: totalTax() takes 1 positional argument but 3 were given

c. `totalTax(0.05, 'apple'=12, 'orange'=8)`

SyntaxError: keyword can't be an expression

d. `totalTax(0.05, 'apple':12, 'orange':8)`

SyntaxError: invalid syntax

e. `totalTax(rate=0.05, priceDict={'apple':12, 'orange':8})`

TypeError: unsupported operand type(s) for +: 'int' and 'dict'

f. `totalTax(rate=0.05, **{'apple':12, 'orange':8})`

rate is 0.050000

priceDict is {'apple': 12, 'orange': 8}

21.0

g. `totalTax(rate=0.05, **priceDict={'apple':12, 'orange':8})`

SyntaxError: invalid syntax

h.
```
totalTax(rate=0.05, priceDict=(('apple', 12), \
          ('orange', 8))
```

priceDict cannot be a tuple.

i.
```
totalTax(rate=0.05, **priceDict=(('apple', 12), \
          ('orange', 8))
```

SyntaxError: invalid syntax

j. `totalTax(apples=12, oranges=8, rates=0.05)`

TypeError: totalTax() missing 1 required positional argument: 'rate'

12. Given the following function definition

```
def totalWithTax(rate, *items, **priceDict = \
            {'apple':12,'orange':8, 'mango':3}):
    total = 0
    for i in items:
        total += priceDict[i]
    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?

a. `totalWithTax(0.05, 'apple', 'orange', 'mango')`
b.
```
totalWithTax(0.05, 'apple', 'orange', 'mango', \
             'apple', 'apple')
```
c.
```
totalWithTax(0.05, 'apple', 'orange', 'mango', \
             apple=3, orange=4, mango=5)
```
d.
```
totalWithTax(0.05, 'apples', 'oranges', 'mango',\
             **{'apple':3, 'orange':4, 'mango':5})
```
e. `totalWithTax(0.05, *('apples', 'oranges', 'mango'))`
f. `totalWithTax(0.05)`

13. 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.

a. `max(A, B)` (2, 4, 8)
b. `max(*A, *B)` 8
c. `max(A + B)` 8
d. `max(*(A+B))` 8
e. `max(*A + *B)`SyntaxError: invalid syntax
f. `max((A, B))` (2, 4, 8)
g. `max(*(A, B))` (2, 4, 8)

14. Given the following code

```
a = 3
def F():
    print(a)
F()
```

What is printed when you run this code?

3

15. Given the following code

```
a = 3
def F():
    a = 5
    print(a)
F()
```

```
print(a)
```

What is printed when you run this code?

5

3

16. 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.

a. Why do you get this error?

   UnboundLocalError: local variable 'a' referenced before assignment

   violation of consistent binding rule!

b. How do you fix it if you want the function F to use the identifier a as defined on line 1?

   insert 'global a 'between line2 & line3

17. Given the source code

```
D = { 'rate': 0.0 }
def totalWithTax(*names, **kv):
    global D
    total = 0.0
    for name in names:
        total += D[name]
    for kw, val in kv.items():
        D[kw] = val # overwrite dict entry
        if kw != 'rate':
            total += val
    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.

a. totalWithTax()

   names is ()

   kv is {}

   D is {'rate': 0.0}

   return value is 0.0

b. totalWithTax('apple')

   KeyError: 'apple'

c. totalWithTax(apple=20)+totalWithTax('apple')

```
# totalWithTax(apple=20)
names is ()
kv is {'apple': 20}
D is {'rate': 0.0, 'apple': 20}
return value is 20.0
# totalWithTax('apple')
names is ('apple',)
kv is { }
D is {'rate': 0.0, 'apple': 20}
return value is 20.0

# finally return 40.0
```

d. `totalWithTax('orange', orange=15)`

```
KeyError: 'orange'
```

e. `totalWithTax(guava)`

```
NameError: name 'guava' is not defined
```

f. `totalWithTax(rate=0.05, apple=10) + totalWithTax('apple', 'apple', 'apple')`

```
# totalWithTax(rate=0.05, apple=10)
names is ()
kv is {'rate': 0.05, 'apple': 10}
D is {'rate': 0.05, 'apple': 10}
return value is 10.5

# totalWithTax('apple', 'apple', 'apple')
names is ('apple', 'apple', 'apple')
kv is { }
D is {'rate': 0.05, 'apple': 10}
return value is 31.5

# finally return 42.0
```

g. `totalWithTax(apple=10, orange=15)+totalWithTax('apple',guava=12)`

```
# totalWithTax(apple=10, orange=15)
names is ()
kv is {'apple': 10, 'orange': 15}
D is {'rate': 0.0, 'apple': 10, 'orange': 15}
return value is 25.0

# totalWithTax('apple',guava=12)
names is ('apple',)
kv is {'guava': 12}
```

D is {'rate': 0.0, 'apple': 10, 'orange': 15, 'guava': 12}
return value is 22.0

# finally return 47.0

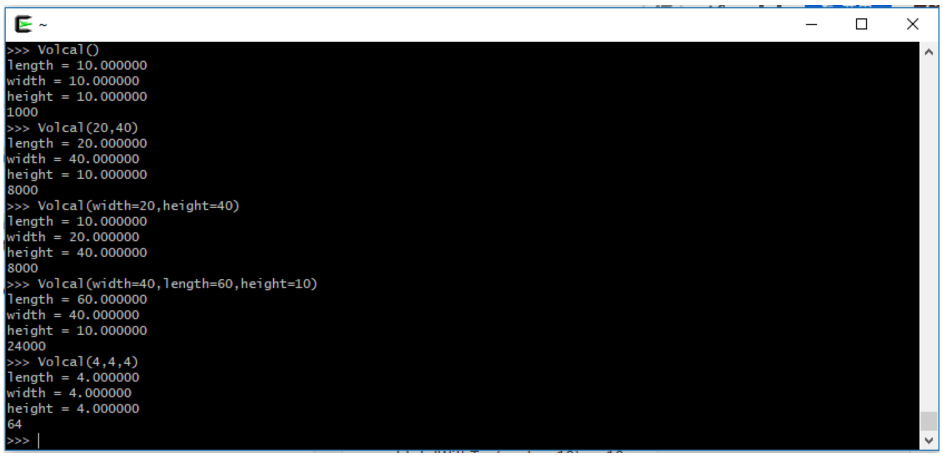
h. totalWithTax(apple=10, orange=15) + totalWithTax(rate=0.05, guava=12) + totalWithTax(rate=0.02, 'apple', 'orange', 'guava')
SyntaxError: positional argument follows keyword argument

18. 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?

```
if _name_ == '__main__':
# test code here
# use assert to check answer
assert totalWithTax(apples=10) == 10
```

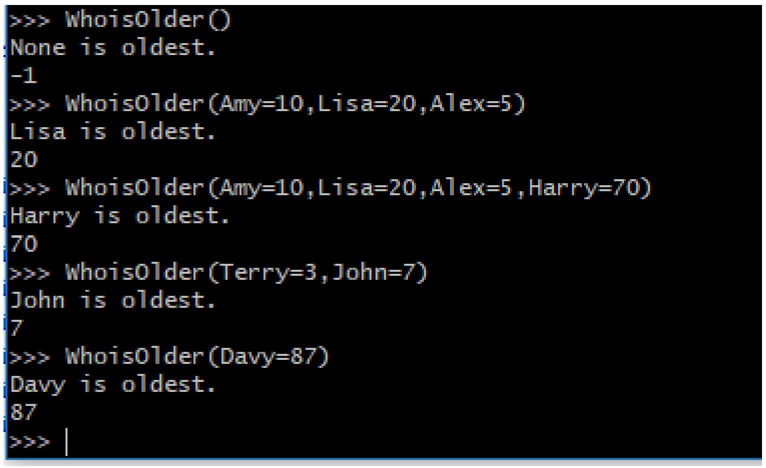
3. Programming

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.



```
>>> Volcal()
length = 10.000000
width = 10.000000
height = 10.000000
1000
>>> Volcal(20,40)
length = 20.000000
width = 40.000000
height = 10.000000
8000
>>> Volcal(width=20,height=40)
length = 10.000000
width = 20.000000
height = 40.000000
8000
>>> Volcal(width=40,length=60,height=10)
length = 60.000000
width = 40.000000
height = 10.000000
24000
>>> Volcal(4,4,4)
length = 4.000000
width = 4.000000
height = 4.000000
64
>>> |
```

2. (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.



```
>>> WhoisOlder()
None is oldest.
-1
>>> WhoisOlder(Amy=10,Lisa=20,Alex=5)
Lisa is oldest.
20
>>> WhoisOlder(Amy=10,Lisa=20,Alex=5,Harry=70)
Harry is oldest.
70
>>> WhoisOlder(Terry=3,John=7)
John is oldest.
7
>>> WhoisOlder(Davy=87)
Davy is oldest.
87
>>> |
```

3. (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 actio n to take. A binary arithmetic operator pops the top two elements from the stack and pushes back t he 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 declar ed? Should you use variable-length arguments? arguments with default value?

1.