MA1008 Introduction to Computational Thinking Quiz 2
Answer all the nine questions in the spaces provided
AY 2023/2024, Semester 1, Week 9

Your Name: _____ Group: _____

Solutions

1. Depending on its data type, a variable is either mutable or immutable and iterable or non-iterable. State the data type of the variables on the left of the following statements and whether they are mutable or iterable in the table below. The answers of the first statement are given as an example.

(12 marks)

	Date Type	Mutable?	Iterable?
`A = 25//int(2.5)`	integer	No	No
`B = [1, [2, 3]].pop()`	list	yes	yes
`C = "a", "b", [10, 20]`	tuple	no	yes
`D = 1e-10`	float	no	no
`E = "MA1008"[2:]`	string	no	yes

2. The following program repeatedly reads in integers and uses a function `store_pos()` to store the positive numbers in a list, and keep a count of the negative ones. It stops when the number of negative numbers reaches 10 and then prints the average of the positive numbers in the list. But the program contains errors. Circle the errors and provide the corrections. You must retain the function and the list but may modify them. Assume at least one positive number is read in. (10 marks)

```
def store_pos(n):
  if n >= 0:
    pos_list += n
  else:
    neg_count += 1

pos_list = []
if neg_count < 10:
  n = input("Enter integer: ")
  store_pos(n)
  print("Average:", post_list.sum()/len(post_list))
```

add line `global neg_count, pos_list` before `if` statement

`pos_list += [n]` or `pos_list.append(n)`

add line `neg_count = 0` before `while` loop

`if` should be `while`

`n = int(input(...))`

Remove indentation for the print statement, `post_list` should be `pos_list`.
pos_list.sum() should be sum(pos_list).

Instead of putting `neg_count` and `pos_list` in a global statement, a student may put one or both in the parameter list. A variable cannot be in a global statement and be a parameter of the function at the same time. If in the parameter list, then the call to the function must have that variable as an argument to the call. However, because `neg_count` is an integer and is immutable, its new value cannot be passed out via the parameter. It must be passed out via a `return` statement instead, in which case the call to the function must have an assignment to `neg_count`:

```
neg_count = store_pos(n, pos_list, neg_count)
```

`pos_list`, being a mutable list, doesn't have the same problem. Still, it can be returned via a return statement, not an error but redundant.

3. i. Given two lists, `A` and `B`, use list comprehension to generate a list of all possible pairs as tuples between members of the two lists. For example, if `A = [1,2,3]` and `B = [7,8]`, the required outcome is `[(1,7), (1,8), (2,7), (2,8), (3,7), (3,8)]`. (5 marks)

```
[(A[i], B[j]) for i in range(len(A)) for j in range(len(B))]
```

ii. What does the following list comprehension statement produce?

```
[c.isupper() for c in "May 2003"]
```

(5 marks)

```
[True, False, False, False, False, False, False, False]
```

4. The function `partition()` has a string `S` and a one-character string `p` as its parameters and returns a list which contains the parts of `S` partitioned by `p` but without `p` itself. For example, if `S = "One fine day"` and `p = "n"`, then the resulting list is `["O", "e fi", "e day"]`. The default value for `p` is `" "`. Fill in the blanks to complete the function. (8 marks)

```
def partition(S, p = " "):
  L = []                # Initialise the list
  part = ""             # Initialise a string part
  for c in S:           # Visit every character in S
    if c != p:          # Not the partition character
      part += c         # Add c to part
    else:               # Found partition character
      L += [part]       # Add part collected to list
      part = ""         # Start a new part, reinitialise
  L += [part]           # Add remaining part to list
  return L              # Return list collected
```

ii. Given `S = "72.3,4.4"`, write Python statements using the above `partition()` function to produce the sum of the two numbers in `S`. (4 marks)

```
L = partition(S, ",")
sum = float(L[0]) + float(L[1])
```

5. A school runs fund raising activities and uses a Python program to record the donations. The program registers the donations of different donors cumulatively using a dictionary, called `donations`, which has the names of donors as keys and the amounts they have donated as values. If there are three donors so far, the dictionary may look like this:

```
donations = {"Peter":320.50, "Paul":178.00, "Mary":436.40}
```

The following function `add_donation()` has the dictionary, name and amount as parameters and adds the amount to the name if the name already exists in the dictionary. Otherwise it creates a new name:amount pair in the dictionary.

i. Fill in the blanks to complete the function. (6 marks)

```
def add_donation(donations, name, amount):
    if name in donations:
        donations [name] += amount
    else:
        donations [name] = amount
```

ii. Write Python statements to print the total amount received from all the donors in the dictionary `donations`. (4 marks)

```
total = 0
for name in donations:
    total += donations[name]
```

6. Given this tuple: `T = (1, "234", [5, 6, 7], (8, 9))`
State the values of the following expressions. If an expression is erroneous, state the cause of the error. You do not need to provide the correction. (4 marks each)

i. `T[2] + [T[1][1]]` — `[5, 6, 7, "3"]`

ii. `[5, 6] in T[2]` — `False`

iii. `[i for i in T if len(i) > 2]` — Error. `T[0]` is int and `len()` fails with int.

7. Given the same tuple `T` as in Question 7, construct an expression using `T` to produce: (4 marks each)

i. `(1, "234", [5, 6, 7, 0], (8, 9))` — `T[2].append(0).`
If student gives `T[2]+[0]`, 3 marks. It creates a new tuple [5, 6, 7, 0] but doesn't change T to the required expression.

ii. `((8,9), "234", [5, 6, 7])` — `(T[3],)+T[1:3].`
Give 3 marks to `T[3]+T[1:3]` as it produces `((8, 9), "234", [5, 6, 7])`
Can also accept `((8, 9),) + T[1:3].`

iii. `(1, "234", [5, 6, 7], (8, 9), 10)` — `T+(10,). T+(10)` is error, but 2 marks.

8. Given `a = 1000, b = 0.0001, pi = 3.141592653, S = "Answer:"`, state what are printed by the following statements. Indicate a space character explicitly using a triangle Δ. (3 marks each)

   i. `print(f"{a*pi:<12.4f}")`

   3141.5927ΔΔΔ

   ii. `print(f"\"{S:>10s}{b*pi:0.6f}\"")`

   "ΔΔΔAnswer:0.000314"

   iii. `print(f"{a:^10d}/{b:<7.4f}*{pi:<7.4f}\n={a/b*pi:^12.4f}")`

   ΔΔΔ1000ΔΔΔ/0.0001Δ*3.1416Δ
   =31415926.5300

9. The following function `product(n)` prints the products of two successive numbers from `0` to `n`. So, if `n = 5`, it prints `0 2 6 12 20`.

```
1  def product(n):
2      for i in range(n):
3          print(i*(i+1), end=" ")
```

   i. Modify the function such that instead of printing the products, it collects them in a list `L` and passes `L` out of the function as a parameter. Provide the initialisation of `L`. (6 marks)

   Change Line 1 to `def product(n, L):`
   Change line 3 to `L.append(i*(i+1))`. Cannot do `L += [i*(i+1)]`
   In the main program, prior to the call to the function, initialise `L` with `L = []`

   The new function is

```
def product(n, L):
    for i in range(n):
        L.append(i*(i+1))   # cannot do L += [n*(n+1)]
```

   In the main program, initialise `L` before the call to the function thus (assume `n` is defined):

```
L = []
product(n, L)
```

   ii. Modify the function such that instead of printing the products, it collects them in a tuple `T` and returns `T` at the end. Provide the initialisation of `T`. (6 marks)

   Insert immediately before Line 2 the initialisation of `T` thus: `T = ()`
   Change Line 3 to `T += (i*(i+1),)`. The comma is necessary.
   Insert new line `return T` after Line 4 at the same indentation as the `for` loop.

   The new function is

```
def product(n):
    T = ()
    for i in range(n):
        T += (i*(i+1),)
    return T
```

<> <> <> THE END <> <> <>