MA1008 Introduction to Computational Thinking Quiz 2
Answer all the questions in the spaces provided
AY 2022/2023, Semester 2, Week 9

Solutions

Variations are possible in the solution. So please check carefully if a student gives a different solution before deducting marks.

1. What are the values of L2 upon executing the following statements? (3 marks each)

i.
```
L1 = L2 = [1, 2, 3]
L1 = L1 + [4]
```
`L2 = [1, 2, 3]`

ii.
```
L1 = L2 = [1, 2, 3]
L1.extend([4])
```
`L2 = [1, 2, 3, 4]`

iii.
```
L1, L2 = [1, 2, 3], [1, 2, 3]
L1.extend([4])
```
`L2 = [1, 2, 3]`

2. i. Write a Python statement that moves the last character of a string S to become its first. For example, if S = "Hello there", the result is "eHello ther". (4 marks)

```
S = S[-1] + S[:-1]
```

ii. Write a function isletter(S) that returns True if the parameter S, a string, contains only letters, either upper or lower case, and False otherwise. You should not use any string method, such as isalpha(), or ASCII values of characters in your solution. (7 marks)

```
def isletter(S):
    for c in S:
        if not ("a" <= c <= "z" or "A" <= c <= "Z"):
            return False
    return True
```

3. i. Write a list comprehension statement that produces a list of odd numbers from 1 to 100, inclusive, that are also multiples of 3 but not of 7. (5 marks)

```
[n for n in range(1, 101, 2) if n%3 == 0 and n%7 != 0]
```

ii. What does the following list comprehension statement produce? (5 marks)

```
[c.upper() for c in "In the year 2000 BC" if "a" <= c <= "z"]
```

```
["N", "T", "H", "E", "Y", "E", "A", "R"]
```

OK to give full marks if student gives `[N, T, H, E, Y, E, A, R]`
Give 3 marks if student gives `[NTHEYEAR]`
Give 2 marks if student gives `NTHEYEAR`

4. Given the function

```
def slice(Collection, start = 0, end = -1):
    return Collection[start:end]
```

What do the following lines of code print? (3 marks each)

```
objects = ("Bread", [1, 2], (9, 8, 7), 3.14, True)
```

i. `print(slice(objects))` `("Bread", [1, 2], (9, 8, 7), 3.14)`

ii. `print(slice(objects[0], -3))` `"ea"`, can accept `ea`

iii. `print(slice(objects[2], 1, 2))` `(8,)`, deduct 1 mark if `8`

iv. `print(slice("-15734", 0, 4))` `"-157"`, can accept `-157`

5. The symbols, names, and atomic weights of the elements of the periodic table are stored in three separate Python lists (each showing only the first five elements) respectively:

```
Sym = ["H", "He", "Li", "Be", "B", …]
Name = ["Hydrogen", "Helium", "Lithium", "Beryllium", "Boron", …]
AW = [1.00797, 4.0026, 6.941, 9.01218, 10.81, …]
```

The corresponding elements in the three lists belong to the same element. Hence “H” is the symbol of hydrogen and its atomic weight is 1.00797 grams/mol.

i. Write Python statements to convert these three lists to a Python dictionary, called `elements`, with the symbol as the key and the value is a tuple containing the name and the atomic weight. (6 marks)

```
elements = { }
for i in range(len(Sym)):
    elements[Sym[i]] = (Name[i], AW[i])
```

ii. Using the dictionary, write Python statements to print the name of each element against its atomic weight, separated by “ : ”, one element per line thus: the colons are aligned over the lines, the names right justified, the weights left justified and printed to 3 decimal places:

```
    Hydrogen : 1.008
      Helium : 4.003
     Lithium : 6.941
   Beryllium : 9.012
       Boron : 10.810
            : : :
```

Note: Your printout needs to cater to the longest element name which has 13 letters and the largest atomic weight which is in the hundreds (i.e. has 3 digits before the decimal point). (6 marks)

```
for e in elements:
    print(f"{elements[e][0]:>13s} : {elements[e][1]:<.3f}")
```

6. i. What is printed in the following program? (4 marks)

```
def add3(a):
    a += 3,
    print(a)                                          (1, 2, 3)

b = (1, 2)
add3(b)
print(b)                                                 (1, 2)
```

ii. The following program is modified from Part i. What are printed? If you think there is an error, identify and explain what the error is. Then provide a correction which is different from Part i, and state what are printed based on your correction. (8 marks)

```
def add3(a):
    a += 3
    print(a)

b = (1, 2)
add3(b)
print(b)
```

Error – Parameter a is a tuple. Cannot add integer to tuple.

Change `b = (1, 2)` to `b = 1`. Then prints 4 in function.

Prints 1 after function.

Other changes possible, such as `a += 3` to `a *= 3`, which triples the tuple.

7. The following statement defines a dictionary `D` for the first six months of a year:

```
D = {1:"Jan", "2":"Feb", [3]:"Mar", (4):"Apr", (5,):"May", Six:"Jun"}
```

i. There are one or more errors in the statement. (a) Circle every error, (b) explain why it is an error and (c) rewrite the statement such that it is error free. (6 marks)

Two errors: `[3]` which is mutable (library key must be immutable) and `Six` which is an unknown variable. Variables can be keys, but they must carry a value of immutable type. May rewrite as follows:

```
D = {1:"Jan", "2":"Feb", 3:"Mar", (4):"Apr", (5,):"May", "Six":"Jun"}
```

Alternatively, may use variable `Six` as it is, but assign it a value such as `Six = 6` beforehand.

ii. Based on your correction, write Python statements to print all the months that contain the letter “a” in either lower or upper case. (6 marks)

```
for mon in D:
    if "A" in D[mon] or "a" in D[mon]:
        print(D[mon])
```

8. The different items in a shop are stored in a list called `Items`:

```
Items = [item_1, item_2, …, item_n]
```

where `n` is the number of items and `item_1`, `item_2`, …, `item_n` are each a list of four fields:

```
[item_Name, number_in_stock, cost_price, selling_price]
```

which are of type string, int, float and float respectively. For example, the first two items may be

```
item_1 = ["Milk", 50, 4.22, 5.20]
item_2 = ["Buns", 120, 0.38, 0.65]
```

i. Write a Python statement that determines the number of items (`n`) in the shop. (4 marks)

```
n = len(Items)
```

ii. Write Python statements that determine the total profit if everything in stock is sold. (6 marks)

```
profit = 0
for i in Items:
   profit += (i[3] - i[2])*i[1]
print("Profit = ", profit)  # this print statement not required
```

9. An encryption scheme picks alternate characters in a given string, starting from the first, and then converts every letter of either case to the next letter of the same case in the alphabet, with the last letter “z” becoming “a”. All other characters remain unchanged. Hence “AB_yz_12” would first yield “A_z1” which is then encrypted as “B_a1”. The function `encrypt(S)` performs the encryption, with the parameter `S` as the string to be encrypted, and returns the encrypted string as the output. Fill in the blanks to complete the function. (12 marks)

```
def encrypt(S):

   output = ________________  # intitialise output

   for c in ________________: # iterate through alternate chars of S

      if ___________________________ or ___________________________:
      # check if c is a letter but exclude z and Z

         output += _____________________  # encrypt the character and
                                          # add it to output
      elif c == "z" or c == "Z":          # deal with z and Z here

         output += _____________________  # add a or A to output without
                                          # checking the case directly
      else:
         output += c  # not a letter, so add to output unchanged

   return output
```

Solution

```
def encrypt(S):

   output = ""  # intitialise output

   for c in S[::2]: # iterate through alternate chars of S

      if "a" <= c < "z" or "A" <= c < "Z":
      # check if c is a letter but exclude z and Z

         output += chr(ord(c) + 1) # encrypt the character and
                                   # add it to output
      elif c == "z" or c == "Z":   # deal with z and Z here

         output += chr(ord(c)-25)  # add a or A to output without
                                   # checking the case directly
      else:
         output += c  # not a letter, so add to output unchanged

   return output
```

<> <> <> The End <> <> <>