9.5/10

Group_06_-Exercice_02

December 7, 2020

1 Exercice 02:

The following exercice requires some understanding in the following subjects: - understand conditions in python - write some loops

1.1 Loop:

1.1.1 1. Use `for`, `.split()`, and `if` to create a Statement that will print out words that start with 's':

[1]:
```
# it should print: "start", "s", "simple", "sentence"

# change the string in list by a space separation by default with split

# print all the words starting with a "s"

user = 'Print all the words start with s in simple sentence'
for word in user.split():
    if word[0] == 's':
        print(word)
```

```
start
s
simple
sentence
```

1.1.2 2. Given a list l_list, delete all the `0` values in the list `l_list`. (hint: see the methods for list in the python documentation)

[5]:
```
# here is the list l_list
l_list = [ 2 , 5 , 8 , 0 , 0 , 0 , 0 , 0 , 5 , 2 , 3 , 9 , 1 , 5 , 3 , 0, 11,␣
 ↪13, 0, 5]
# WARNING, you cannot delete an element in a list during a loop, why? Because␣
 ↪of shrinking list
# the best way is to create another list and save all the wanted value in this␣
 ↪list.
# let's create another empty list to save the value without zero

for number in l_list[:]:
```

```
    if number == 0:
        l_list.remove(number)
        print(l_list)
```

```
[2, 5, 8, 0, 0, 0, 0, 5, 2, 3, 9, 1, 5, 3, 0, 11, 13, 0, 5]
[2, 5, 8, 0, 0, 0, 5, 2, 3, 9, 1, 5, 3, 0, 11, 13, 0, 5]
[2, 5, 8, 0, 0, 5, 2, 3, 9, 1, 5, 3, 0, 11, 13, 0, 5]
[2, 5, 8, 0, 5, 2, 3, 9, 1, 5, 3, 0, 11, 13, 0, 5]
[2, 5, 8, 5, 2, 3, 9, 1, 5, 3, 0, 11, 13, 0, 5]
[2, 5, 8, 5, 2, 3, 9, 1, 5, 3, 11, 13, 0, 5]
[2, 5, 8, 5, 2, 3, 9, 1, 5, 3, 11, 13, 5]
```

1.1.3 3. We consider the following dictionary (students) whose keys are the names of the students and the values of the keys are the overall averages obtained by passing the final exam.

Write a Python program that partitions this dictionary into two sub-dictionaries:

1. admittedStudents whose keys are the admitted students and the values of the keys are the averages obtained (average greater than or equal to 10).
2. nonAdmittedStudent whose keys are the non-admitted students and whose key values are the averages obtained (average less than or equal to 10).

[2]:
```
students = {"student_1" : 13 , "student_2" : 17 , "student_3" : 9 , "student_4"␣
↪: 15 ,
                        "student_5" : 8 , "student_6" : 14 , "student_7" : 16␣
↪, "student_8" : 12 ,
                        "student_9" : 13 , "student_10" : 15 , "student_11" :␣
↪14 , "student_112" : 9 ,
                        "student_13" : 10 , "student_14" : 12 , "student_15" :␣
↪13 , "student_16" : 7 ,
                        "student_17" : 12 , "student_18" : 15 , "student_19" :␣
↪9 , "student_20" : 17}

# your code here
admittedStudents = dict ({})
nonAdmittedStudent = dict ({})
for key, value in students.items ():
    if (value <10):
        nonAdmittedStudent[key] = value
    else:
        admittedStudents[key] = value

print ("Admitted students:", admittedStudents)

print ("Students not admitted:", nonAdmittedStudent)
```

```
Admitted students: {'student_1': 13, 'student_2': 17, 'student_4': 15,
'student_6': 14, 'student_7': 16, 'student_8': 12, 'student_9': 13,
```

```
'student_10': 15, 'student_11': 14, 'student_13': 10, 'student_14': 12,
'student_15': 13, 'student_17': 12, 'student_18': 15, 'student_20': 17}
Students not admitted: {'student_3': 9, 'student_5': 8, 'student_112': 9,
'student_16': 7, 'student_19': 9}
```

1.1.4 4. Use for loop to print an isosceles triangle with the character * . The user should be asked for a number and the the program print the triangle with the *character.

[10]:
```
# if for example, the user type 3, the program should print this:
#          *
#         ***
#        *****
#
#
#  if the user typed 5, the program should print this:
#          *
#         ***
#        *****
#       *******
#      *********
# and so on...

print("Ask for a nummer:")
rows = int(input())
k = 4 * rows - 1
for i in range(0, rows):
    for j in range(0, k):
        print(end=" ")
    k = k - 1
    for j in range(1, i + 1):
        print("*",end="*")
    print("*")
```

```
Ask for a nummer:
 5
                    *
                   ***
                  *****
                 *******
                *********
```

1.1.5 5. A wolf in sheep's clothing

Wolves have been reintroduced to Great Britain. You are a sheep farmer, and are now plagued by wolves which pretend to be sheep. Fortunately, you are good at spotting them.

Warn the sheep in front of the wolf that it is about to be eaten. Remember that you are standing

at the front of the queue which is at the end of the array:

```
[sheep, sheep, sheep, sheep, sheep, wolf, sheep, sheep]      (YOU ARE HERE AT THE FRONT OF THE
   7      6      5      4      3            2      1
```

If the wolf is the closest animal to you, return "Pls go away and stop eating my sheep". Otherwise, return "Oi! Sheep number N! You are about to be eaten by a wolf!" where N is the sheep's position in the queue.

Note: there will always be exactly one wolf in the array.

Examples:

```
sheep_queue_0 = ["sheep", "sheep", "sheep", "wolf", "sheep"]
# should print
-> 'Oi! Sheep number 1! You are about to be eaten by a wolf!'

sheep_queue_1 = ['sheep', 'sheep', 'wolf']
# should print
-> 'Pls go away and stop eating my sheep'
```

[9]:
```
sheep_queue_2 = ['wolf', 'sheep', 'sheep', 'sheep', 'sheep', 'sheep', 'sheep']␣
↪# test your code with sheep_queue_0 and sheep_queue_1 also

# your code here
sheep_loop = ['sheep', 'sheep','sheep', 'sheep','sheep','sheep', 'sheep',␣
↪'wolf'] # test your code with sheep_queue_0 and sheep_queue_1 also
for n in (sheep_loop):
    if n == 'sheep':
        continue
    else:
        print('Das ist ein großer wolf!')
        if sheep_loop.index('wolf') == len(sheep_loop)-1:
            print('Pls go away and stop eating my sheep')
        else:
            print('Oi! Sheep number {}! You are about to be eaten by a wolf!'.
↪format(len(sheep_loop)-sheep_loop.index('wolf')))
```

- 0.5 pt

```
Das ist ein großer wolf!
Pls go away and stop eating my sheep
```

1.1.6 6. time(s)

You receive a string, and you need to return a **string** that shows how many times each letter shows up in the string by using the sign plus "+"

For example:

"Chicago" –> "c:++,h:+,i:+,a:+,g:+,o:+" As you can see, the letter c is shown only once, but with 2 pluses.

The return string should include only the letters (not the dashes, spaces, apostrophes, etc). There

should be no spaces in the output, and the different letters are separated by a comma (,) as seen in the example above.

Note that the return string must list the letters in order of their first appearence in the original string.

More examples:

“Bangkok” –> “b:+,a:+,n:+,g:+,k:++,o:+”

“Las Vegas” –> “l:+,a:++,s:++,v:+,e:+,g:+”

[7]:
```
# your code here. Test it with the 3 words above
alphabet =␣
 ↪['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','x
a = '+'
string = "Las Vegas"
count = {}
for i in string.lower():
    if i in count:
        count [i] += 1
    else:
        count[i] = 1
for i in count:
    if i in alphabet:
        print(i,':',count[i]*a, end = ',')
    else:
        continue
```

```
l : +,a : ++,s : ++,v : +,e : +,g : +,
```

[ ]:
```
```