Exercice 02:

02 / 10 pts

The following exercice requires some understanding in the following subjects:

- understand conditions in python
- write some loops

Loop:

0 / 1.5 pts

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

In [16]:

```
# it should print: "start", "s", "simp

# change the string in list by a space

# print all the words starting with a

a = ['start', 's', 'simple', 'sentence']
test= 's'
print("Given list\n " ,a)
res = [idx for idx in a if idx[0].lower() == test.lower()]
print("list elements starting with matching letter:\n ",res)
```

```
# it should print: "start", "s", "simple", "sentence"
st = 'Print only the words that start with s in this Simple sentence'
# change the string in list by a space separation by default with split
list_st = st.lower().split()
#check
print(list_st)
# print all the words starting with a "s"
for word in list_st:
    if word[0] == 's':
        print(word)
```

// this should be the result at the end...
// and you should have used for loops, .split function and //if conditions

```
Given list
  ['start', 's', 'simple', 'sentence']
list elements starting with matching letter:
  ['start', 's', 'simple', 'sentence']
```

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) 0 / 1.5 pts

In [35]:

```
# here is the list l_list
# WARNING, you cannot delete an element in
# the best way is to create another list c
# let's create another empty list to save

l_list = [ 2 , 5 , 8 , 0 , 0 , 0 , 0 , 0 , 5 , 2 , 3 , 9 , 1 , 5 , 3 , 0, 11, 13, 0, 5]
l_list = list(range(1, 14))
print(l_list)
```

```
l_list_without_zero = []

for nb in l_list:
    if nb != 0:
        l_list_without_zero.append(nb)
```

// you should have use the list above, not create a new and print it.

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

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. 0 / 1.5 pts

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).

In [38]:
```
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"
                          "student_13" : 10 , "student_14" : 12 , "student_15" : 13 , "student_16"
                          "student_17" : 12 , "student_18" : 15 , "student_19" : 9 , "student_20" :
a = list(students.items()) // ?
print(a)
```

```
[('student_1', 13), ('student_2', 17), ('student_3', 9), ('student_4', 15), ('student_5', 8), ('st
udent_6', 14), ('student_7', 16), ('student_8', 12), ('student_9', 13), ('student_10', 15), ('stud
ent_11', 14), ('student_112', 9), ('student_13', 10), ('student_14', 12), ('student_15', 13), ('st
udent_16', 7), ('student_17', 12), ('student_18', 15), ('student_19', 9), ('student_20', 17)]
```

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. 1.5 / 1.5 pts

In [41]:
```
# 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...
num = int(input("Enter the number of rows: "))
for i in range(0,num):
    for j in range(0,num-i-1):
        print(end=" ")
    for j in range(0,i+1):
        print("*",end=" ")
    print()
```

```
    *
   * *
  * * *
 * * * *
* * * * *
```

5. A wolf in sheep's clothing 0.5 / 2 pts

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 QUEUE)
   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'
```

In [22]:

```
# test your code with sheep_queue_0 and sheep_queue_1 also
Sheep_queue_2 = ['wolf', 'sheep', 'sheep', 'sheep', 'sheep', 'sheep', 'sheep']

def warn(array):   // you do not need any function or method here.

    if (array[-1] == 'wolf'):          // which index is targeted here?
        print('Pls go away and stop eating my sheep')
    else:
        print('Oi! Sheep number 1! You are about to be eaten by a wolf!')

 print warn(Sheep_queue_2)



# your code here
```

```
  File "<tokenize>", line 11
    print warn(Sheep_queue_2)
    ^
IndentationError: unindent does not match any outer indentation level
```

6. time(s) 0 / 2 pts

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:+"

In [ ]:

```
# your code here. Test it with the 3 words above
```