PRACTICAL 1:
PROGRAMMING IN PYTHON

Objective: The aim of this practical is to allow you to familiarize with writing computer programs using Python programming language. This is a crucial skill for the IS 122- Data structure and algorithms course.

INSTRUCTIONS: *You must follow these instructions carefully, your programs will be auto-graded, failure to observe the instructions may result to automatically scoring zero (0).*

1. Allowed editors: Visual Studio Code.
2. All your files will be interpreted using Python 3.11.3.
3. Your code should use the pycodestyle 2.8.* (Read more|)
4. All your assignments must be submitted in a public GitHub repository with the name *data_structure_and_algorithms* inside a folder *000-programming_in_python*.
5. Each question will clearly indicate the appropriate file name for your solution.
6. You are only required to write **one** function that solves a given problem, any function calls to test your code should be written under the **if __name__ == "__main__"** condition (Read more).
7. Question (0) has already been done for you, you only need to submit the file as per the instructions.
8. One group member should send an email with a link to the public repository containing the solutions to the provided questions.

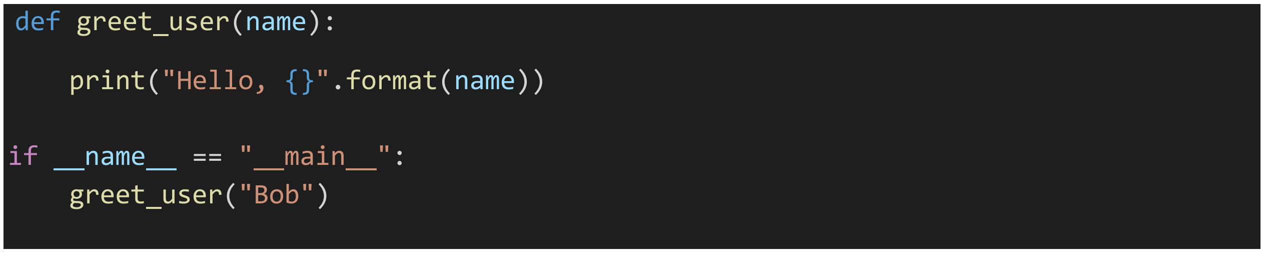
SUBMISSION DEADLINE: 28th April 2023, 4:00PM.

Question 0

Write a Python function that greets a user, function signature is as follows *def greet_user(name)*, you can always assume name will be a string

File name: 0-hello.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms

Below is the appropriate solution as stored in 0-hello.py file



```
 def greet_user(name):

    print("Hello, {}".format(name))

if __name__ == "__main__":
    greet_user("Bob")
```

Question 1:

Write a Python function that returns the smallest integer in a list. You can assume that the list will always contain integers. The function signature is as follows *def get_smallest_integer(my_list)*

File name: 1-smallest_int.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms

Question 2:

Write a Python function that searches for the first occurrence of an integer in a list and returns its index. You can assume that the list will always contain integers.

def find_first_occurrence(my_list, num)

File name: 2-search _int.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms

Question 3:

Write a Python function that prints a right-angled triangle with a given height using (*). For example, if a height of 5 is given the output should be as follows:

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

The function signature is as follows *def print_right_triangle(height)*

File name: 3-print_triangle.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms

Question 4:

Write a Python function that factorizes an input number into its prime factors. The prime factors should be returned as a list of numbers.
The function signature is as follows *def factorize(number)*

File name: 4-factorize.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms

Question 5:

Write a Python function that takes a list of integers as input and returns the sum of all the even numbers in the list. The function signature is as follows: *def sum_even_numbers(my_list)*

File name: 5-sum_even.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms

Question 6:

Write a Python function that takes a string as input and returns a dictionary containing the frequency of each character (case-insensitive) in the string. Ignore non-alphabetic characters. The function signature is as follows: *def character_frequency(string)*

File name: 6-char_frequency.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms

Question 7:

Write a Python function that checks whether a given integer is Prime or not. The function should return True is the integer is prime and return False if the integer is not prime. The signature of the function is as follows: def is_prime(number):

File name: 7-is_prime.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms

Question 8:

Write a Python function in which when given a list of integers and an integer target, returns indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order. The function signature is as follows: *def two_indices(nums, target)*

Example:
Input: nums = [2, 7, 11, 15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

File name: 8-two_indices.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms

Question 9:

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

SYMBOL	**VALUE**
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five, we subtract it making four. The same principle applies to the number nine, which is written as IX.
There are six instances where subtraction is used:
I can be placed before V (5) and X (10) to make 4 and 9.
X can be placed before L (50) and C (100) to make 40 and 90.
C can be placed before D (500) and M (1000) to make 400 and 900.
Given a roman numeral, convert it to an integer.

Write a Python function that accepts an integer and converts it to a roman numeral string. Your function should only return the string. The function prototype is as follows *def int_to_roman(n)*

File name: 9-int_to_roman.py, **Folder**: 000-programming_in_python**, Repository:** data_structure_and_algorithms