Lab2 : Conditional Execution/ Decision making

Decision making in programs is achieved by including conditions that are checked, and if the conditions are true then some set of statements are executed, and if the condition evaluates to false, we can have another set of statements executed.

Python, similar to many other programming languages, provides the following decision-making statements:

1. **if** statement: Used to execute a block of statements if condition is true
2. **if**..**else** statement: Used to execute a block of statements if condition is true, and another block of statements when the condition is false
3. **nested if** statements: We can use if or if..else statements inside another if or if..else statements.

Start a new notebook in Google Colab.

1. if statement

Type the following statements in a single cell:

```
>>> n = 18
>>> if n > 0:                                   [Press Enter]
        print('condition is true')              [Press Enter]
        print('n is positive')                  [Press Shift+Enter]
```

The colon (**:**) is important.
It marks the beginning of a block of code

The Boolean expression after the **if** is called as the condition. If it is true, then the indented statements are executed. If not, nothing happens.

In a new single cell, type the following:

```
x = 10
if x:
   print('1st condition is true')
   print(x)
y = 0
if y:
   print('2nd condition is true')
   print(y)
print('This hello is outside loop')
```

The colon (**:**) is important.
It marks the beginning of a block of code

Run the Cell. You should get the following results:

```
1st condition is true
10
This hello is outside loop
```

Python programming language assumes any non-zero and non-null values as true and assumes zero or null values as false value.

2. if..else statement

Go back to the previous cell. Modify it as follows:

```
x = 10
if x%2 == 0 :
    print('1st condition is true. X is even ')
    print(x)
else:
    print('1st condition is FALSE. x is odd')
y = 0
if y:
    print('2nd condition is true')
    print(y)
else:
    print('2nd condition is FALSE')
print('This hello is outside loop')
```

Run the cell.

Are the results as expected? You should get the following output:

```
1st condition is true. X is even
10
2nd condition is FALSE This
hello is outside loop
```

3. if..elif..else

In the new Cell, type the following:

```
x = input('Enter a number: ')
y = input('Enter another number: ')
numx = float(x)
numy = float(y)
if numx < numy:
    print( x, 'is less than ', y)
elif numx > numy:
    print (x, 'is greater than ', y)
else:
    print (x, 'is equal to ', y)
```

Run Cell.

Are the results as expected? Run the model with different inputs to check if it is working properly.

elif is an abbreviation of "else if." Again, exactly one branch will be executed. There is no limit on the number of **elif** statements. If there is an **else** clause, it has to be at the end, but there doesn't have to be one.

The `input()` function will prompt the user for input when you run the code. But the value user enters will be stored as a string or text. So, if it is a number that we want the user to enter, then we need to convert the input text into a number using `float()` or `int()` functions.

4. Nested If

In a New Cell, type the following lines. Although in a new cell, it will take the values of *numx*, *numy* etc from the previous cell

```
print('Starting nested if')
if numx == numy:
    print( x, 'is equal to ', y, ' here too')
else:
    if numx > numy:
        print(x, 'is greater than ', y, ' here too')
    else:
        print(x, 'is less than ', y, ' here too')
```

Run Cell. Are the results as expected? Run the model with different inputs to check if it is working properly.

In the above program, the outer conditional contains two branches. The first branch contains a simple statement. The second branch contains another if statement, which has two branches of its own. Those two branches are both simple statements, although they could have been conditional statements as well.

Although the indentation of the statements makes the structure apparent, **nested conditionals** become difficult to read very quickly. In general, it is a good idea to avoid them when you can.

5. Multiple conditions

In the next New Cell, type the following lines. Although in a new cell, it will take the values of *numx*, *numy* etc from the previous cells

```
print('checking multiple conditions')
if numx >= 0 and numx < 10:
    print( x,'is a single digit number')

if numx%2==0 or numy%2 == 0:
    print('At least one is even')
else:
    print('Both are odd numbers')
```

Run the codes with different inputs to check if it is working properly

Lines and Indentation
Blocks or group of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. Both blocks in this example are fine:

```
if True:
        print("True")
        print("True is ok")
else:

    print("False")
    print("False is also ok")
```

Error: The second block in the following example will generate an error:

```
if True:
    print("True")
    print("True is ok")
else:
    print("False")
          print("Is False also ok?")
```

Error: The first block in the following example will also generate error

```
if True:
print("True")
print("Is True ok?")
else:
    print("False")
    print("Is False also ok?")
```

However, the following will not generate error. “Is False also ok?” will be taken to be outside the **if** loop, and hence always executed.

```
if True:
    print("True")
    print("Is True ok?")
else:
    print("False")
print("Is False also ok?")
```

Note: In case you change the input in the top cell, then you need to rerun all the subsequent cells to update the computations.

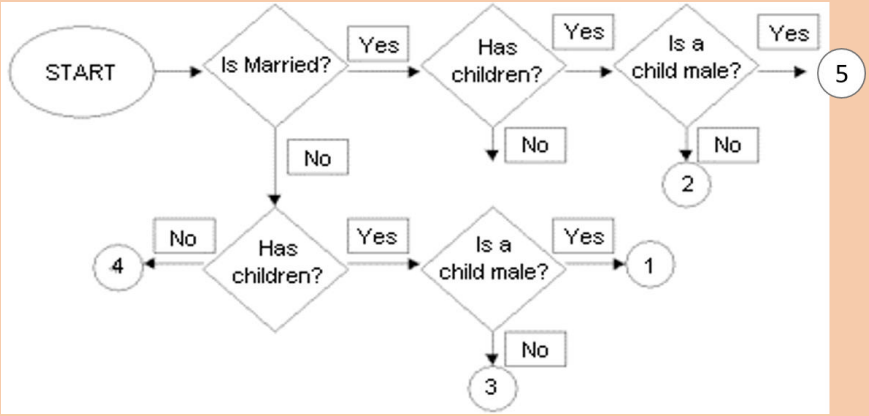
Tasks

Answer the following questions using PYTHON. For each question write and solve using a single cell, if possible. After you complete the questions, submit your file in Moodle.

1. Write a code snippet that takes as input from the user 2 numbers, and checks if the smaller number is a divisor of the larger number. It should display the appropriate output message.

2. Write a code snippet that takes as input from the user 3 numbers, and prints the smallest of the three. Use minimum number of **if** statements as possible.

3. Write code snippet that need to keep prompting the user appropriate question as per the flowchart. For example, the first question must be "Are you Married?". The user can answer Y or N. Depending on the response, the next question should be asked and so on. Finally, any of the end nodes 1 to 5 are reached, then the program should end with a message "You are to proceed to Room Number __" (the __ to be replaced with end node number)



4. Write a code snippet to that takes as inputs a, b and c of a quadratic equations, and then funds the roots using the quadratic formula. The code must print all the roots. It must also handle complex roots (display answer with an *i*), and any values of a, b and c. Python should not throw an exception.