MCT-242 : COMPUTER PROGRAMMING-I

using Python 3.9

Prepared By:

Mr. Muhammad Ahsan Naeem

YouTube Playlist

https://www.youtube.com/playlist?list=PLWF9TXck7O_wMDB-VriREZ6EvwkWLNB7q

Lab 5: If-else statement: CLO 3

Relational Operators:

So far, we have seen programs that mainly do followings:

- Take input of different data-types from user.
- Do one or more calculation to find the output.
- Display the output using different displaying and formatting features.

Other than calculations, computer program can compare and decide the calculations based on comparison result. To compare two or more variables, we use **Relational Operators**. The list of relational operators is given here:

Relational Operator	Description	Expression	Expression Meaning
>	Greater than	`x>y`	Is x greater than y?
<	Less than	`x<y`	Is x less than y?
>=	Greater than or equal to	`x>=y`	Is x greater than or equal to y?
<=	Less than or equal to	`x<=y`	Is x less than or equal to y?
==	Equal to	`x==y`	Is x equal to y?
!=	Not equal to	`x!=y`	Is x not equal to y?
It is important to note that double equal sign i.e. == is relational operator which checks whether the two operands are equal or not and on the other hand single equal sign i.e. = is assignment operator which assigns a value to variable(s).			

Output of a relationship:

The result of a relational operator expression is either **`True`** or **`False`**. This is also known as **Boolean Data Type** which can have value only True or False. See the output of following code:

```
x=10
y=5
print(x==y)
print(x>=y)
print(y==7)
print(y)
```

The output is:

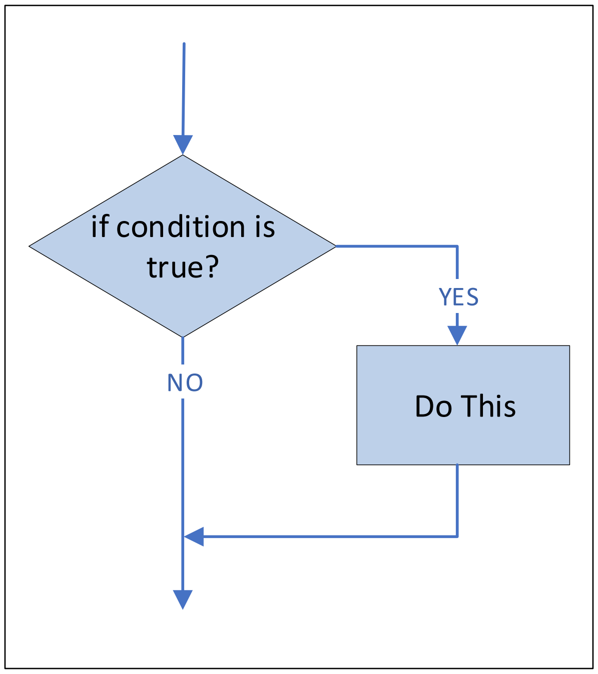
```
False
True
False
5
```

The first two lines assigns the values to variable `x` and `y` using assignment operator `=` . The third line is a Boolean Expression as it involves Relational Operator. `x==y` means whether `x` is equal to `y` or not and of course they are not equal and hence the output is `False`. The fourth line is `x>=y` means whether `x` is greater than or equal to `y` or not. `x` having value `10` is greater than `y` having the value `5` and hence the output is `True`.

The fifth line is `y==7` means whether value of `y` is equal to `7` or not and `y` having the value of `5` is not equal to `7` and hence the output is `False`. It is very important to note that `y==7` checks whether the variable `y` is equal to `7` or not and it never assigns `7` to `y`. Therefore, when value of `y` is printed after `y==7` line, it still is `5`.

The if statement:

The `if` statement is used to execute a part of the code based on some condition being true. If the condition is false then that part will not be executed. It can be shown in a flow chart as:



The program below shows an example of an `if` statement. Program asks user to enter three test scores (out of hundred), and the program calculates their average. If the average is greater than 90, the program congratulates the user on obtaining a high score.

```
test1=eval(input('Enter Score in Test-I : '))
test2=eval(input('Enter Score in Test-II : '))
test3=eval(input('Enter Score in Test-III : '))
avg=(test1+test2+test3)/3
print('Your Average Score is:',avg)
```

```
if avg>=90:
    print('Congratulations on a High Average!')
```

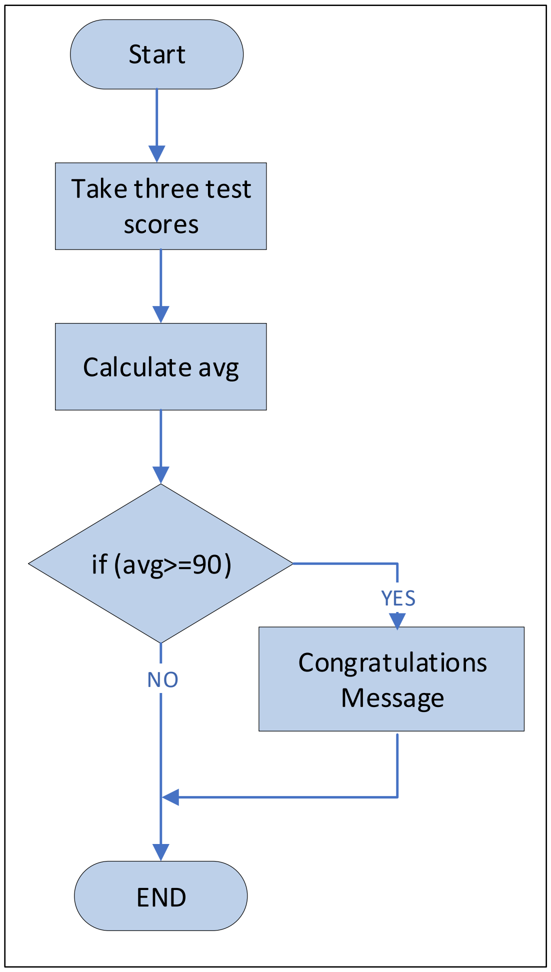
One sample execution of the program is shown here:

```
Enter Score in Test-I : 95
Enter Score in Test-II : 92
Enter Score in Test-III : 90
Your Average Score is: 92.33333333333333
Congratulations on a High Average!
```

Another execution of the program is as under:

```
Enter Score in Test-I : 80
Enter Score in Test-II : 70
Enter Score in Test-III : 85
Your Average Score is: 78.33333333333333
```

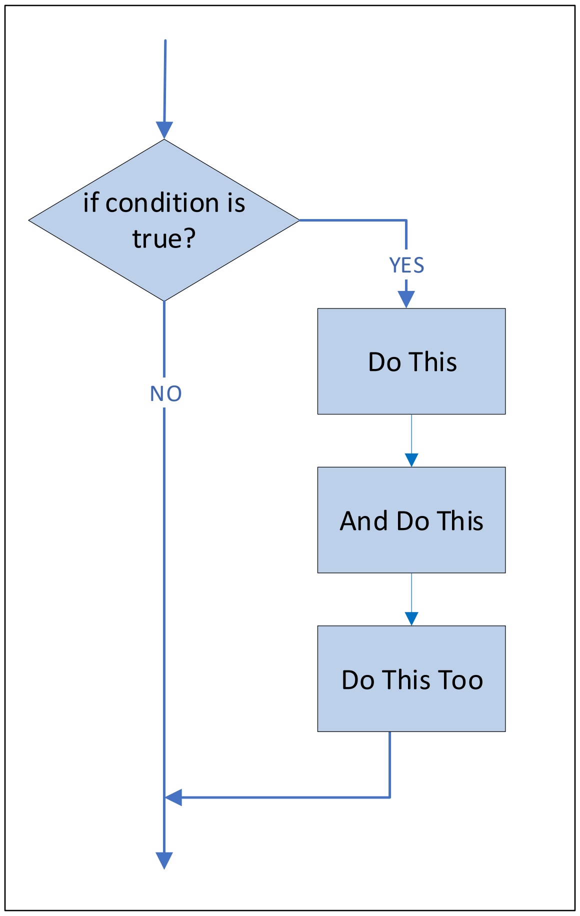
If you see carefully the two outputs of the same program, in one case congratulation message is displayed while it is not displayed for the second case. This is all because of the condition used in the program using `if` statement is true in first case and false in second. The flowchart of above program can be shown like this:



Note carefully the syntax of the if statement. There is a space/tab at the start of the statement after first line of `if` statement.

```
if(condition):
     Do This
```

There can be as many statements as required to execute after the condition is true. In flow chart it is shown below:



Check the code:

```
x=eval(input('Enter a number: '))
if(x>0):
    print('x is positive')
    print('Positive numbers are great!')
print('Thanks for your time.')
```

If user enter **5** as input, this will be the output:

```
Enter a number: 5
x is positive
Positive numbers are great!
Thanks for your time.
```

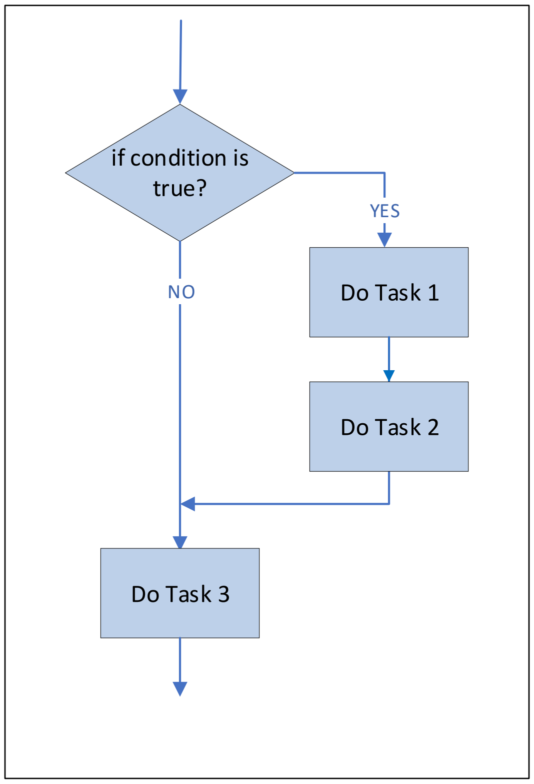
The condition **x>0** is true and we see the output lines. What do you think will be the output if entered number is negative? Think for that before entering a negative number!

If user enters **-5**, this will be the output:

```
Enter a number: -5
Thanks for your time.
```

The third message of thanks is displayed even the condition is false. See carefully the code that the first two print statements start with space/tab while the third doesn't. Therefore, the first two print statements are dependent on the condition and will be executed only if the condition is true (also known as Block of the **if** statement), while the third print statement is not the part of the Block of the **if** statement and will be executed independent of the condition being true or false.

It can be shown in block diagram as:



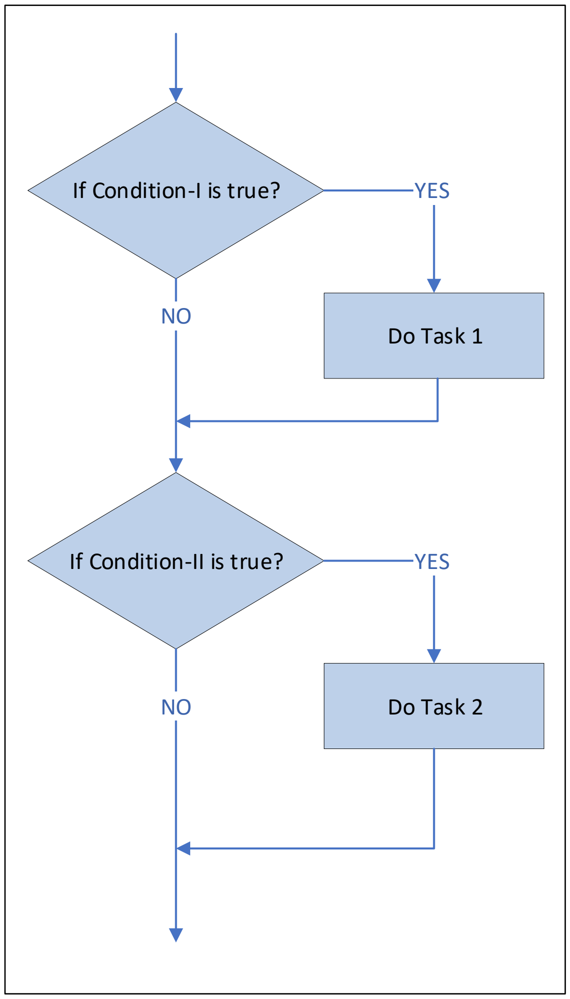
The Task-1 and Task-2 will be executed if the condition is true and the Task-3 will be executed every time.

Indentation:

The most important syntax rule in using `if` statements and many others to come is the indentation (Adding space/Tab before the statement). Indentation defines the block of a code. Wrong indentations can lead to logical errors.

Multiple if statements:

There can be multiple `if` statement in a program. A scenario is shown in flowchart below:



The two if statements are independent here. There can be following scenarios:

- Condition-I True and Condition-II False: Task-1 will be executed only.
- Condition-I False and Condition-II True: Task-2 will be executed only.
- Both Conditions True: Both Tasks will be executed.
- Both Conditions False: None of the Tasks will be executed.

Example:

Let's see a simple but useful example. Write a code that will take a number from user and will tell if the number is even. Checking for a number being even or not is simply a check whether number is divisible by 2 or not i.e. remainder after division by 2 is 0 or not. So, we can use remainder division operator as shown:

```
x=eval(input('Enter a number: '))
if(x%2==0):
    print('Entered number is Even')
```

This simple program displays the output when number is even. There is also an associated condition for a number being odd i.e. if the number is not divisible by 2. Therefore, we can add more statements to the code as:

```
x=eval(input('Enter a number: '))
if(x%2==0):
    print('Entered number is Even')
if(x%2!=0):
    print('Entered number is Odd')
```

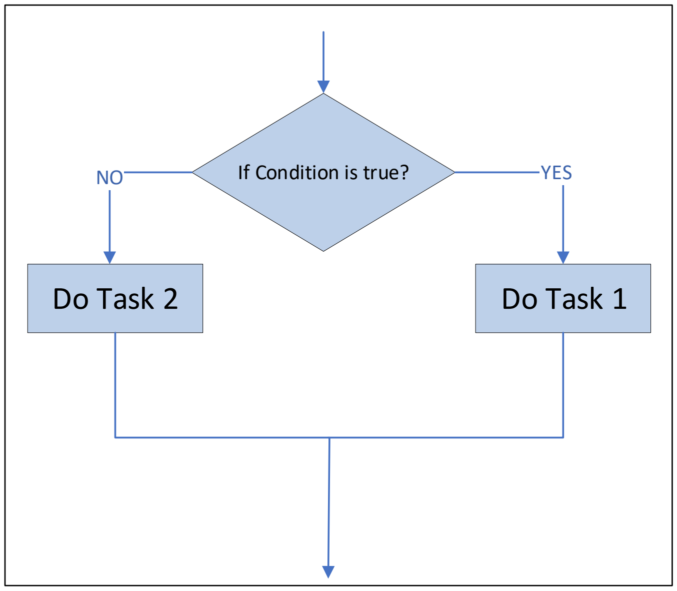
Although the two conditions in above program are independent but they cannot be true both at the same time because of the obvious reason!

if-else statement:

In the **if-else** statement, a condition in the form of a Boolean expression is evaluated. If the expression is true, a statement or block of statements is executed. If the expression is false, however, a separate group of statements is executed. The syntax of the **if-else** statement is as under:

```
if(condition):
     Do This
else:
     Do This
```

The flowchart of **if-else** statement is shown here:



Check the following code:

```
x=eval(input('Enter a number: '))
if(x>=0):
    print('x is positive')
    print('Positive numbers are great!')
else:
    print('x is negative')
    print('Negative numbers are scary!')
print('Thanks for your time.')
```

One execution of the program is shown here:

```
Enter a number: 34
x is positive
Positive numbers are great!
Thanks for your time.
```

Another execution is here:

```
Enter a number: -57
x is negative
Negative numbers are scary!
Thanks for your time.
```

See carefully how indentation is used for `if` and `else` blocks and how the last print statement for the thanks message is neither part of the `if` block nor of the `else` block. That last print statement will be executed every time we run the code.

Note:

Python also allows to write relational statement as `0<x` instead of `x>0` which generally is not allowed in many other languages. Likewise, we can also write like `0<=x<=100` to check if variable `x` has value between 0 and 100 both inclusive.

Tasks:

[1] Write a program that will take a number form user and will display whether the number is even or odd. Use if-else statement.

[2] Write a program that asks the user to enter a length in centimeters. If the user enters a negative length, the program should tell the user that the entry is invalid. Otherwise, the program should convert the length to inches and print out the result. There are 2.54 centimeters in an inch.

[3] Write a program that asks the user for two numbers and prints ' Entered numbers are Closed!', if the numbers differ within 10 units from each other and prints 'Entered numbers are Not Closed!' otherwise. Use if-else statement.

Sample output is:

```
Enter first number: 85
Enter second number: 80
Entered numbers are Closed!
```

Another Sample output is:

```
Enter first number: 10
Enter second number: 80
Entered numbers are not Closed!
```

Another Sample output is:

```
Enter first number: 40
Enter second number: 45
Entered numbers are Closed!
```

[4] 12-Hour and 24-Hour Format

Write a program that asks the user for an hour between 1 and 12, and then asks them to enter am or pm. The program then shows the hour in 24-Hour format.

Before you proceed for the logic it is important to understand the 12:00 Noon confusion. Is it 12am or 12pm at noon? Generally, we consider 12:00pm as noon and 12:00am as midnight. But the fact is 12am and 12pm both are true for midnight and it's 12:00m for the 12:00 noon. For simple logic here we will assume 12:00am as 12:00 Noon and 12:00pm as 12:00 midnight. Therefore, if user enters am, the same hour will be in 24-hours format and otherwise i.e. for pm it will be entered hour plus 12.

[5] Square Root of a Number:

Write a program that will take a number from user and will display the square root of it.

We can find the square root of the number using **`sqrt()`** function in Math module. But if we try to pass a negative number as input argument, it will generate run-time error. You can check that as shown here:

```
>>> import math
>>> sqrt(-4)
```

And you will see an error something like:

```
NameError: name 'sqrt' is not defined
```

But we know that square root of **`-4`** is **`2j`**. So, for this program you will take a number from user and then apply an **`if-else`** statement to check whether input number is positive or negative. If it is positive, simply use **`sqrt()`** and display the result. If entered number is negative, make is positive, use **`sqrt()`** and then display the result appending a **`'j'`** with it.

How to make a negative number a positive equivalent?

If variable a contains a negative number we can use:

```
a=a*-1
```

or:

```
a*=-1
```

or we can use **`abs()`** function as **`abs(a)`**

[6] Write a program that will take two numbers as input (say a and b) and will show their ratio at output (a/b). As you know that division by zero is not possible and a run-time error occurs in an attempt to divide by zero. You have to incorporate this problem in a way that if second number entered is zero, program should not calculate the ratio; instead it should display a message that division by zero is not possible.

Sample output is:

```
Enter first number: 5
Enter second number: 2
```

```
Ratio of two numbers is: 2.5
```

Another sample output is:

```
Enter first number: 16
Enter second number: 0
Division by zero is not possible!
```

[7] A perfect square is the one whose square-root is an integer e.g. 16 is perfect square and 15 is not. Write a program that will take a number from user and will display whether it is perfect square or not.

Logical Operators:

Many times, we have to make decision on the basis of multiple conditions. Logical operators are used to combine two or more logical expressions. The logical operators are given here:

Logical Operator	Description	Example	Example Meaning
`and`	Connects two expressions into one. Both expressions must be true for the overall expression to be true.	`a==5 and b<3`	Will be true if a contains 5 and b is less than 3
`or`	Connects two expressions into one. One or both expressions must be true for the overall expression to be true. It is only necessary for one to be true, and it does not matter which.	`a==5 or b<3`	Will be true if a contains 5 or b is less than 3.
`not`	Reverses the "truth" of an expression. It makes a true expression false, and a false expression true.	`not b<3`	Will be true if b is not less than 3.

As an example, if we want to see whether an entered number is between `0` and `100` or not, instead of writing as `0<x<100` , we can do that as:

```
x = eval(input("Enter a number: "))
if (x>=0 and x<=100):
    print('Entered number is between 0 and 100')
```

Common Mistakes:

The logical operators must be used with great care. If we want to check a number not in range 0-100 then it is a common mistake to write as:

```
 x = eval(input("Enter a number: "))
if (x<0 and x>100):
    print('Entered number is not between 0 and 100')
```

This is a wrong logic as there is no number which is both less than 0 and greater than 100 at the same time. It should be :

```
if (x<0 or x>100):
```

The above can also be done as:

```
if not((x>=0 and x<=100)):
```

or as:

```
if not(0<=x<=100):
```

Example:

In a chemical plant, a certain chemical reaction is sensitive to the plant temperature and based on current temperature sensed by a sensor, an indication light is turned on. Yellow light turns on if temperature crosses 100°C and a red will turn on if it goes higher than 125°C. Let's write a program which simulated the above situation in a way that user will be asked to enter temperature, if it is higher than 100°C a warning message will be displayed and if it crosses 125°C, a high alert message will be displayed.

If we write it this way:

```
temp = eval(input("Enter temperature(in Celsius): "))
if temp>100:
    print('Warning!!!')
if temp>125:
    print('High Alert')
```

And enter 130 as temperature, this will be the output:

```
Enter temperature(in Celsius): 130
Warning!!!
High Alert
```

Both messages are displayed because both conditions are true. The first condition must be written this way:

```
temp = eval(input("Enter temperature(in Celsius): "))
if 125>temp>100:
    print('Warning!!!')
if temp>125:
    print('High Alert')
```

Tasks

[8] Write a program that will ask for the marks of five subjects one by one and will print the average of the marks. Moreover, a message will be printed out depending upon average marks as:
If average is greater than or equal to 80 → You are an outstanding student.
If average is greater than or equal to 70 but less than 80 → You are a good student.
If average is greater than or equal to 60 but less than 70 → You are an average student.
If average is greater than or equal to 50 but less than 60 → You are a below-average student.
If average is greater than or equal to 40 but less than 50 → You are a poor student.
If average is less than 40→ You need extra ordinary efforts

Max marks are 100 and you can assume that user will enter valid numbers.

Sample output is:

```
Enter five subjects' marks: 74,68,82,65,87
Average of five subject marks is: 75.2
You are a good student
```

[9] In last lab you did a task to take input three sides of a triangle and show the area of triangle. We assumed that user will always enter a valid triangle. As we know that for a valid triangle the condition is that sum of any two sides should be greater than the third one. Now update that program such that it takes three sides as input and will print the area only if it is a valid triangle otherwise it should print that triangle is invalid.

[10] Write a program that will take three numbers as input and will print the maximum of three numbers at output. You have to use three `if` statements.

Sample output is:

```
Enter first integer: 12
Enter second integer: 16
Enter third integer: 5
16 is the largest
```

[11] A company insures its employees in the following cases:

- If the employee is married
- If the employee is unmarried, male & above 30 years of age
- If the employee is unmarried, female & above 25 years of age

Write a program which takes marital status, gender and age as input from user. After checking the given conditions, the output of the program will a message stating whether he/she is eligible for insurance or not.

Sample output is:

```
Enter your marital status (M/U): U
Enter your gender (M/F): F
Enter your age: 28
Congratulations!
```

```
You are eligible for the insurance.
```

<u>Another Sample output is:</u>

```
Enter your marital status (M/S): U
Enter your gender (M/F): M
Enter your age: 22
We are Sorry!
You are not eligible for the insurance.
```

You have to complete following code for this task:

```
S=input("Enter your marital status (M/U): ")
G=input("Enter your gender (M/F): ")
A=eval(input("Enter your age: "))
if(S=="m" or #Complete the condition
    print("Congratulations!")
    print("You are eligible for the insurance.")
else:
    print("We are Sorry.")
    print("You are not eligible for the insurance.")
```