The Need

Just like our real life scenarios , we use python for decision making. And in all of our decisions confusion stays. When we go out we often try to figure out what movie to watch , should it be aquaman of spider man. What food to have , chinese or mughlai. The way we express ourselves while taking these decisions includes words like "If" , "else".

Eg: - "would it be better if we watch game of thonres tonight staying home ? or else we can go out and party a little.

In python we express ourselves the same way. And luckily enough , in python we call these statements "if-else" statements only. And as these are based on conditions of taking either one of the decisions , hence these are called conditional statements as well.

Greater or Lesser Numbering conditions

Let's start with an easy example. In here we are trying to find out if a given number is greater than another given number or not. And if the number is greater as per condition our programme will provide the output as "Greater" or if the number is lesser as per condition then the programme will provide the output as "Lesser".

I would like you to look at the codes given below first and then i will explain the entire code line by line.

```
num1 = int(input())
num2 = int(input())
if num1>num2:
    print (f"{num1} is greater than {num2}")
else:
    print (f"{num1} is lesser than {num2}")
```

```
9
89
9 is lesser than 89
```

Explanation

1. num1 and num2 are variables , whose input we will be taking from user. By default the input's data type remains as string , we are converting the input into integer prefixing "int".
2. our conditional statement is given with ">" symbol , stating if num1's input value is greater than num2's input value then print statement as num1 is greater than num2.
3. If our initial statement is not true then the else statement is given to come as an output.
4. While printing the statements the use of f string has taken place to insert the value of num1 and num2 into the statements.

5. The indentation is very important here , notice that after both if and else , a ":" is given and that's why the print statements are indented within the block of if else. Removing this indent will throw an error in our programme.

The Problem

Now with the above example we have seen that if either of our variables are greater or lesser than each other then our programme is able to provide a output right ? What if both of the variables are having the same value ?

Let's have a look

```
num3 = int(input())
num4 = int(input())
if num3>num4:
    print (f"{num3} is greater than {num4}")
else:
    print (f"{num3} is lesser than {num4}")
```

```
45
45
45 is lesser than 45
```

What just happened ? even though 45 is equally valued with 45 why it is stating as lesser? Well the answer is in your code. Your programme will first check if the num1 value is greater than num2 , once it realises that it's not it will simply go ahead and print the statement given in the "else" block. whatever we write in the else block will be printed. Have a look below

```
num5 = int(input())
num6 = int(input())
if num5>num6:
    print (f"{num5} is greater than {num6}")
else:
    print (f"{num5} is lesser or equal than {num6}")
```

```
45
45
45 is lesser or equal than 45
```

As you can see the else statement is executed perfectly.

The Solution

The solution to this problem lies in multiple conditioning. In python we call it if-else ladder as well. Have a look at the code below and then we will get into explanantion.

```
num7 = int(input())
num8 = int(input())
if num7>num8:
    print (f"{num7} is greater than {num8}")
```

```
if num7 == num8:
    print (f"{num7} is equal with {num8}")
else:
    print (f"{num7} is lesser than {num8}")
```

```
9
89
9 is lesser than 89
```

The problem with the solution

Time Complexity

Though the solution looks feasible , there is one small issue with it. if we use "if" statement twice/multiple times then our code would go and check all the if conditions, which will result into more time for the code to be executed. Which might not look like a serious issue with a small code like this but when bigger programs will be worked upon , this will turn out to be a big problem. This is called time complexity.

Let's have a look at the time taken by our code to execute the programme first.

```
import time
start_time = time.time()
nm1 = int(input())
nm2 = int(input())
if nm1>nm2:
    print (f"{nm1} is greater than {nm2}")
if nm1 == nm2:
    print (f"{nm1} is equal with {nm2}")
else:
    print (f"{nm1} is lesser than {nm2}")
print("--- %s seconds ---" % (time.time() - start_time))
```

```
9
89
9 is lesser than 89
--- 1.8215603828430176 seconds ---
```

```
import time
start_time = time.time()
um1 = int(input())
um2 = int(input())
if um1>um2:
    print (f"{um1} is greater than {um2}")
elif um1 == um2:
    print (f"{um1} is equal with {um2}")
else:
    print (f"{um1} is lesser than {um2}")
print("--- %s seconds ---" % (time.time() - start_time))
```

```
9
89
9 is lesser than 89
--- 1.787660837173462 seconds ---
```

As you can see time taken for execution with multiple if is 1.8215603828430176 seconds seconds and when we used else if or elif it took 1.787660837173462 seconds. Now let's understand elif.

Else-If Or Elif

When there are multiple conditions given to run our programme , we use elif command. the limitaion of if condition is that it checks all the if stated conditions to reach a conclusion while elif command , once it finds one condition to be true it doesn't check other conditions. Hence time taken is less.

Boolean outputs

As you have understood in here with the help of if else statements , we will determine if a given logic is true or not. In this example we will take a list and will look for a specific element in the list. If the element is there the programme will return "True" else it will return "Fales". We will use "in" and "not in" operators here.

```
List1 = []
List1_len = int(input("The length of the list is: "))
for i in range(0,List1_len):
    num = int(input())
    List1.append(num)
print(List1)
num10 = int(input("number to check on: "))
if num10 in List1:
    print("True")
else:
    print("False")
```

```
The length of the list is: 3
6
7
8
[6, 7, 8]
number to check on: 8
True
```

```
L2 = [25,45,48,56,5,9]
nm11 = int(input("number to check on: "))
if nm11 not in L2:
    print("True")
```

```
number to check on: 9
```

Also as you can see above , the "else" statement is not a mandatory one to use always. it's absence will not throw an error.

Combined Conditions

Now let's have a look at combined conditions. We will be using And and Or operators for executing such programmes. And condition will execute when both of the given conditions are true and Or condition will execute when any of the given conditions are true.

```
num_cc = int(input())
num1_cc = int(input())
if num_cc>10 and num1_cc<num_cc:
    print(f"{num_cc} is the target number")
else:
    print(f"{num1_cc} is the target number")
```

```
7
9
9 is the target number
```

let's understand the code. The combined condition given is that if the value of num_cc variable is greater than 10 (which is 11 in this case AND if the value of num1_cc is lesser than num_cc (which is yes as 10 is lesser than 11) then num_cc will be printed as the target number. So there are two conditions that needed to be true in order to print num_cc as the target number.

Now as we can see below if among this two conditions we make any of them as untrue(let's make numn_cc<10) then num1n_cc will be printed as target number.

```
numn_cc = int(input())
num1n_cc = int(input())
if numn_cc>10 and num1n_cc<numn_cc:
    print(f"{numn_cc} is the target number")
else:
    print(f"{num1n_cc} is the target number")
```

```
78
9
78 is the target number
```

For the same example if we use "Or" operator , it'll only look for one statement to be the truth , either n_cc to be greater than 10 or nu_cc to be lesser than n_cc , and it'll print n_cc as the target number , if none of the statement is true then it'll give nu_cc as the target number.

```
n_cc = int(input())
nu_cc = int(input())
if n_cc>10 or nu_cc<n_cc:
    print(f"{n_cc} is the target number")
else:
    print(f"{nu_cc} is the target number")
```

```
55
56
55 is the target number
```

In here only one statement is true. That is n_cc being greater than 10 (it's 55) , the other one is false of nu_cc being smaller than n_cc as 56 is larger than 15.

Nested Elif Statements

When multiple if else statements takes place within the primary if else statement , there it's called nesting of if else or nested elif statements.PLeas note that nested elif statements can be present within the if block or the else block as well.

Let's have a look

```
Name = input("Please enter your name: ")
Age = int(input("Please enter your Age:  "))
Gender = input("Please enter your Gender:  ")
if Age>=40 and Age<=60:
    if Gender=='Male' or 'male':
        print("your application will be shortlisted for the senior
stewerd role")
    elif Gender == 'Female' or 'female':
        print("your application will be shortlisted for the senior
airhostess role")
    else:
        print("Please provide correct gender infomation")
elif Age>= 20 and Age<40:
    if Gender=='Male' or 'male':
        print("your application will be shortlisted for the junior
stewerd role")
    elif Gender == 'Female' or 'female':
        print("your application will be shortlisted for the junior
airhostess role")
    else:
        print("Please provide correct gender infomation")
else:
    print("Sorry , you are not eligble for the position available
now")
```

```
Please enter your name: Kumar
Please enter your Age:  45
Please enter your Gender:  male
your application will be shortlisted for the senior stewerd role
```