Conditional Statments

Conditional statements are a set of programming constructs that allow you to control the flow of execution of your code based on certain conditions. In other words, you can use conditional statements to specify that certain blocks of code should only be executed if a certain condition is met.

There are two main types of conditional statements in Python: if statements and if-else statements.

if statement:

This is the simplest form of a conditional statement. It allows you to execute a block of code only if a specified condition is true. we have to take care of identation as well in such kind of statments.

In [5]:
```
x = 34
if x > 18:
    print("Welcome Back")
```

```
Welcome Back
```

As shown in above piece of code the print statment have 4 space identation from the normal block of program where our if statment exist, if we remove this identation we will get an IdentationError because we have to specify the if-block which can be define by putting some identation and at least one statment, otherwise error will be generated as shown below

In [7]:
```
x = 4
if x > 18:
print("Welcome Back")
```

```
  Cell In[7], line 3
    print("Welcome Back")
    ^
IndentationError: expected an indented block
```

In case we want to add an empty block in if condition or any other area where providing block / piece of code is necessory but we want to leave it empty for now we can use pass keyword. what is pass?
The pass statement is used as a placeholder for future code. When the pass statement is executed, nothing happens, but you avoid getting an error when empty code is not allowed. Empty code is not allowed in loops, function definitions, class definitions, or in if statements.

In [10]:
```
#lets solve above issue using pass
x = 4
if x > 18:
    pass
print("Outside If Statment")
```

```
Outside If Statment
```

as shown above the pass keywrod tell framework or whatever it is that hey there is a block but for now it is empty so you can go and execute next commands and framework replied ok boss and move to next statment and execute it.

If with else

Now there can be circumstances when we want to check a condition if met, we want to run a block of code and if condition is not true then in that case we want to run another block of code, in this kind of situation the if and else are used, if will check the condition if true run its block otherwise the code of else will be executed

In [11]:
```
#ProblemStatment:
                #Check if the number stored in variable is even or odd
number = 45;
if(number%2==0):
    print("Number is Even")
else:
    print("Number is Odd")
```

```
Number is Odd
```

Now here is a question why we use else statment, we can write print("number is odd") outside else as well, if your question is same then you did not understand if-else problem statment,
ok if we don't use else and only use if let see what happens

In [13]:
```
number = 45;
if(number%2==0):
    print("Number is Even")
print("Number is Odd")
```

```
Number is Odd
```

It gives the same result then why to use else?
let try one more time and this time change the value of number to a even number

In [14]:
```
number = 46;
if(number%2==0):
    print("Number is Even")
```

```
print("Number is Odd")
```

```
Number is Even
Number is Odd
```

Now this is what i want to show, so when we don't use else statment the commaned after if will always will be executed no matter condition is met or not because we don't put any constraint on these commands,
the else statment put a constraint on its statment that, hi bro listen, you can only execute when the condition of if statment evaluate to false otherwise keep quiet and enjoy your life don't disturb program execution.

In [16]:
```
number = 46;
if(number%2==0):
    print("Number is Even")
else:
    print("Number is Odd")
```

```
Number is Even
```

Now as shown above when we use else, only one of either if or else will be executed, since condition is true the if block executed.

How To Handle Multiple Conditions

Now let suppose we have more than one conditions such as while grading the student, the grades are divided in multiple categories let assume
A,B,C,D,E and F
Now how to handle such kind of circumstances, for that python give us elift statment let see what it is.

Handling multiple conditions with elif

The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".
so using elif we can test multiple conditions when all of its above conditions are not true. let see an example

In [17]:
```
marks = 55
if(marks>90):
    print("You got grade A")
elif(marks>80):
    print("You got grade B")
elif(marks>70):
    print("You got grade C")
elif(marks>60):
    print("You got grade D")
elif(marks>50):
    print("You got grade E")
elif(marks<50):
    print("You got grade F")
```

```
You got grade E
```

Now as shown above, the first condition will be evaluated, and it is not true, then the second which is false also as well and it goes untill it found a condition which is true and that is the second last condition where marks>50 is checked and block of that condition will be executed and the remaining block will be ignored since a condition met. this is how the if and elif works.

Now what none of the condition met and we want to execute a some code when no condition met, in that case else is here to support us we can use if elif else structure to check multiple conditions and if non of them evaluate to true simply run the else command as from above example,

In [24]:
```
marks = -10
if(marks>90):
    print("You got grade A")
elif(marks>80):
    print("You got grade B")
elif(marks>70):
    print("You got grade C")
elif(marks>60):
    print("You got grade D")
elif(marks>50):
    print("You got grade E")
elif(marks<50 and marks>0): #here we put a check that value must be between 0 to 50 if not run else statment
    print("You got grade F")
else:
    print("Invalid Input Try Again")
```

```
Invalid Input Try Again
```

Now as shown above when none of if or elif statment matches the else statment executed. this is how if elif and else works

OneLine If / ShortHand If

when we have single line of code in our if statment we can write it as,

In [30]:
```
var = 18;
if var>=18 : print("You can vote")
print("\n")
print("Outside if")
```

```
You can vote


Outside if
```

ShortHand if else

```
In each and every if or elif we first write conditional statment then write the code we want to execute, but here
in short hand if-else the structure is little different, here we have to add a statment which we want to execute
when
the condition is ture then write if condtion as,
print("You can vote ") if var> 18
then after that condition we can directly write our else statment as
print("You can vote ") if var> 18  else print("You're not eligible")
```

In [39]:
```
var = 4
print("You can vote ") if var> 18  else print("You're not eligible")
```

```
You're not eligible
```

```
we have to take care of structure of shorthand if or if-else so that we don't stuck into error
```

and or not with if-else

```
we can use if and not operator as well with conditional statments to combine multiple conditions (with and or) and
to revert the condition as true to false and false to true (using not) as given below,
```

In [47]:
```
a = 91
if(a > 90 and a < 100):
    print("You got Grade A" )
else:
    print("Unexpected Result")
```

```
You got Grade A
```

In [48]:
```
#lets understand above example
a = 990
if(a > 90 and a < 100):
    print("You got Grade A" )
else:
    print("Unexpected Result")
```

```
Unexpected Result
```

```
there is a chance user enter wrong marks which are not acceptable so we do have to chek that as well, in above
example we are doing same thing, the value must be between 90 and 100 to be eligible for grade A
```

```
and will return true when both conditions are ture
```

```
or will return true when any single condition from all conditions becomes true as
```

In [50]:
```
a = 990
if(a > 90 or a < 100):
    print("You got Grade A" )
else:
    print("Unexpected Result")
```

```
You got Grade A
```

```
# Even the number are greater than 100 still if statment executed this is because first condition is true which is
a>90 and yes a is greater than 90 so no matter second condition is true or false when we are using or the if will
be executed
```

In [52]:
```
#lets take an example of not
isMale = False
if( not isMale):
    print("You are female")
```

```
You are female
```

```
as shown above we checked if not IsMale, isMale contains false, and not will convert it into True so
condition evaluated to true and block of if will be executed.
```

Nested if

```
You can have if statements inside if statements, this is called nested if statements. when we want to check another
condition when one condition met we place it inside that condition and it becomes nested if, like let suppose
we want to check if the given machine is laptop if yes then we want to check if it is of HP depending on that
we will run specific code
```

In [54]:
```
isLaptop = True
isHp  = False
if(isLaptop):
    if(isHp):
        print("It is a laptop of HP company")
    else:
        print("It is laptop but not of HP company")
else:
    print("Is is not even a laptop")
```

```
It is laptop but not of HP company
```

```
As shown above the first condition evaluated to true but, the inner condition evaluated to false so the else of
```

inner condition executed. this is how nested-if works.

let see how we can handle above nested if with and or operators and not

In [55]:
```
isLaptop = True
isHp  = False
if(isLaptop and isHp):
        print("It is a laptop of HP company")
elif (isLaptop and not isHp):
        print("It is laptop but not of HP company")
else:
    print("Is is not even a laptop")
```

```
It is laptop but not of HP company
```

This is small problem we can handle with and or and not operator, but large problem which requries nested-structure can not be handled using above technique of and,or,not for that we definately have to use nested if.
and,or,not are useful when we have to combine seperate conditions while nested are helpful when we have to check only
if the first condition is true and in same nested heriarchy.

Switch Statment in Python

In python there is no switch statment like other programming language, btw we can achieve similar functionality using if-elif-else or using dictionary or using functions. let see how we can achieve switch case using dictionary because we already seen how to work with if-elif-else.

Concept of Switch

```
Switch allows us to evaluate an expression and based on value of expression
execute related block block;
for example in JS
var a = 2;
switch(a){
    case 1:
    runcode1();
    break;
    case 2:
    runcode2();
    break;
    case 3:
    runCode3();
    break;
    default():
    runNoMatchCode()
}
```

Now in python we have dictionaries where we can store key of any type and their values, and that value can by any kind of object, let see how we can implement switch using dictionary

In [15]:
```
#defining a function which takes a value and in that function we have a dictionary which
#contains keys and there related values and we can get key values using dictionaryObject.get(key,optional default)
#the second parameter is optional which is basically returned when the key did not match in dictionary.
#so we provide both the key and default parameter and depending on key the action will be performed.

def print_case(value):
    print("kk")

def switch_case(case_value):
    switch = {
        "case1":  print_case,
        "case2":  print_case,
        "case3":  print_case,
    }
    return switch.get(case_value,"default")

switch_case("case1")# passing case1 in our switch case handler
print(switch_case("case4")) # default_action
```

```
default
```

In [ ]: