If-Elif-Else

In Python, you can use the if, elif (short for "else if"), and else statements to create conditional branches in your code. These statements allow you to execute different blocks of code based on certain conditions. Here's the basic syntax:

In [1]:

```
#if condition1:
    # Code to execute if condition1 is True
#elif condition2:
    # Code to execute if condition1 is False and condition2 is True
#else:
    # Code to execute if both condition1 and condition2 are False
```

Here's a breakdown of how these statements work:

- if: This is the initial condition. If condition1 is true, the code block under if is executed, and the program exits the entire if-elif-else structure.
- elif (optional): You can have multiple elif blocks to check for additional conditions if the previous if or elif conditions are false. If any elif condition is true, its corresponding code block is executed, and the program exits the structure.
- else (optional): If none of the previous conditions (including the if and all elif conditions) is true, the code block under else is executed. It serves as the "catch-all" block when none of the conditions match.

In [1]:

```
# For Example
x=10

if x < 5:
    print("x is less than 5")

elif x==5:
    print("x is equal to 5")

else:
    print("x is greater than 5")
```

```
x is greater than 5
```

In this example, if x is less than 5, the first block is executed. If x is exactly 5, the second block is executed. Otherwise, if x is greater than 5, the third block is executed.

In [2]:

```
# We can use input function as well in same program
x = input('Enter the number')

# Convert x to an integer
x = int(x)

if x < 5:
    print("x is less than 5")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is greater than 5")
```

```
Enter the number4
x is less than 5
```

Example of If-else

login Program:

- we have to build login program in one of the website where Input is email and password.
- when input is correct then we have to give them access in our website
- use if else for that

email- kishor4@gmail.com (mailto:kishor4@gmail.com) password - 1234

In [6]:

```
# Simulated user data
stored_email = "kishor4@gmail.com"
stored_password = "1234"

# User input
email = input('Enter email: ')
password = input('Enter password: ')

# Check if user input matches stored data
if email == stored_email and password == stored_password:
    print('Welcome')
else:
    print('Authentication failed. Please check your email and password.')
```

```
Enter email: kishor4@gmail.com
Enter password: 1234
Welcome
```

In [7]:

```
# If input is wrong so we have to give them once chance for coreect password
# Simulated user data
stored_email = "kishor4@gmail.com"
stored_password = "1234"

# User input
email = input('Enter email: ')
password = input('Enter password: ')

# Check if user input matches stored data
if email == stored_email and password == stored_password:
    print('Welcome')

elif email == stored_email and password != stored_password:
    print('Incorrect Password')
    password=input('Enter the password again')
    if password== stored_password:   # this is nested if else
        print('welcome,finally')

else:
    print('Authentication failed. Please check your email and password.')
```

```
Enter email: kishor4@gmail.com
Enter password: 324354
Incorrect Password
Enter the password again1234
welcome,finally
```

Nested If_ELSE

Nested if-else statements in Python allow you to have one if-else statement inside another. This allows for more complex conditional logic, where you can test multiple conditions and execute different blocks of code based on those conditions. Here's the basic structure:

```
# if condition1:
    # Code to execute if condition1 is True
    if nested_condition1:
        # Code to execute if both condition1 and nested_condition1 are True
    else:
        # Code to execute if condition1 is True but nested_condition1 is False
else:
    # Code to execute if condition1 is False
```

In [8]:

```
x = 10
y = 5

if x > 5:
    print("x is greater than 5")
    if y > 2:
        print("y is also greater than 2")
    else:
        print("y is not greater than 2")
else:
    print("x is not greater than 5")
```

```
x is greater than 5
y is also greater than 2
```

Modules In Python

In Python, a module is a file containing Python code that can define functions, classes, and variables that can be reused in other Python scripts or programs. Modules are used to organize and encapsulate related code, making it easier to manage and maintain large codebases. Python provides a rich standard library with many built-in modules, and you can also create your own custom modules. we see some modules:

1. Math module
2. keywords
3. random
4. Dataetime , etc

1. Math Module

In [10]:

```
# math
import math

math.sqrt(196)
```

Out[10]:

```
14.0
```

2. Keyword

In [11]:

```
import keyword
print(keyword.kwlist) # this will give u all the keyword list
```

```
['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'asyn
c', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else',
'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'whil
e', 'with', 'yield']
```

3. random

In [13]:

```
import random

print(random.randint(1,100)) # it gives random number between 1 to 100 anyways when u rur
```

```
86
```

4. Datetime

In [14]:

```
import datetime

print(datetime.datetime.now()) # this code gives you current time
```

```
2023-09-21 10:50:47.265541
```

If you want to see all modules present in python just write

In [2]:

```
# help('modules') # run this code
```

In [ ]:

```

```