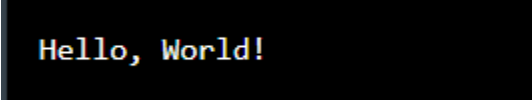


**Python:** Python is a general purpose, dynamic, high-level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Eg. print("Hello, World!")

Output:



Hello, World!

### **Python Editors and Interpreter**

- Python is interpreted language i.e it's not compiled and the interpreter will check the code line by line.
- Code can be written in a text editor with .py extension and then it can be put into the python interpreter for execution

### **Using different offline and online Python IDE**

**Offline IDE:** Jupyter, Thonny, Pycharm, Spyder, IDLE, Netbeans or Eclipse PyDev etc.

**Online IDE:** <https://www.w3schools.com/python/>

<https://www.w3resource.com/python/python-tutorial.php>

<https://www.geeksforgeeks.org/python-programming-language/>

[https://www.tutorialspoint.com/execute\\_python\\_online.ph](https://www.tutorialspoint.com/execute_python_online.ph)

### **Python Features**

**1. Free and Open Source:** Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

**2. Easy to code:** Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

**3. Easy to Read:** As you will see, learning Python is quite simple. As was already established, Python's syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

**4. Object-Oriented Language:** One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

**5. High-Level Language:** Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

**6. Python is a Portable language:** Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

**7. Python is an Integrated language:** Python is also an Integrated language because we can easily integrate Python with other languages like C, C++, etc.

**8. Interpreted Language:** Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called bytecode.

**9. Large Standard Library:** Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing.

**10. Dynamically Typed Language:** Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

### **Python History**

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, **Guido Van Rossum** published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce.
- Python 2.0 added new features such as list comprehensions, garbage collection systems.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify the fundamental flaw of the language.
- *ABC programming language* is said to be the predecessor of Python language, which was capable of Exception Handling and interfacing with the Amoeba Operating System.
- The following programming languages influence Python:
  - ABC language.
  - Modula-3

### **Why the Name Python?**

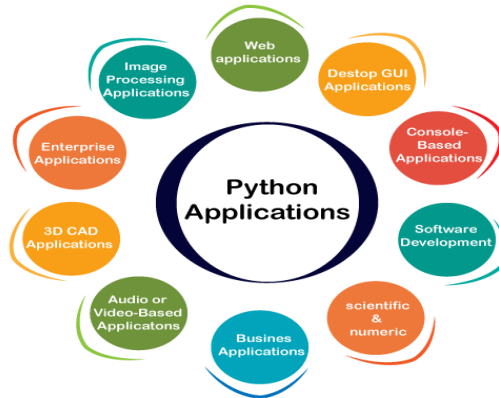
There is a fact behind choosing the name [Python](#). **Guido van Rossum** was reading the script of a popular BBC comedy series "**Monty Python's Flying Circus**". It was late on-air 1970s.

Van Rossum wanted to select a name which unique, sort, and little-bit mysterious. So he decided to select naming Python after the "**Monty Python's Flying Circus**" for their newly created programming language.

## *Applications of Python*

Python is known for its general-purpose nature that makes it applicable in almost every domain of software development. Python makes its presence in every emerging field. It is the fastest-growing programming language and can develop any application.

Here, we are specifying application areas where Python can be applied.



### **1) Web Applications**

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, BeautifulSoup, Feedparser, etc. One of Python web-framework named Django is used on **Instagram**. Python provides many useful frameworks, and these are given below:

- Django and Pyramid framework(Use for heavy applications)
- Flask and Bottle (Micro-framework)
- Plone and Django CMS (Advance Content management)

### **2) Desktop GUI Applications**

The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a **Tk GUI library** to develop a user interface. Some popular GUI libraries are given below.

- Tkinter or Tk
- wxWidgetM
- Kivy (used for writing multitouch applications )
- PyQt or Pyside

### **3) Console-based Application**

Console-based applications run from the command-line or shell. These applications are computer program which are used commands to execute. This kind of application was more popular in the old generation of computers. Python can develop this kind of application very effectively. It is famous for having REPL, which means **the Read-Eval-Print Loop** that makes it the most suitable language for the command-line applications.

Python provides many free library or module which helps to build the command-line apps. The necessary **IO** libraries are used to read and write. It helps to parse argument and create console help text out-of-the-box. There are also advance libraries that can develop independent console apps.

#### 4) Software Development

Python is useful for the software development process. It works as a support language and can be used to build control and management, testing, etc.

- **SCons** is used to build control.
- **Buildbot** and **Apache Gumps** are used for automated continuous compilation and testing.
- **Round** or **Trac** for bug tracking and project management.

#### 5) Scientific and Numeric

This is the era of Artificial intelligence where the machine can perform the task the same as the human. Python language is the most suitable language for Artificial intelligence or machine learning. It consists of many scientific and mathematical libraries, which makes easy to solve complex calculations.

Implementing machine learning algorithms require complex mathematical calculation. Python has many libraries for scientific and numeric such as Numpy, Pandas, Scipy, Scikit-learn, etc. If you have some basic knowledge of Python, you need to import libraries on the top of the code. Few popular frameworks of machine libraries are given below.

- SciPy
- Scikit-learn
- NumPy
- Pandas
- Matplotlib

#### 6) Business Applications

Business Applications differ from standard applications. E-commerce and ERP are an example of a business application. This kind of application requires extensively, scalability and readability, and Python provides all these features.

Oddo is an example of the all-in-one Python-based application which offers a range of business applications. Python provides a **Tryton** platform which is used to develop the business application.

#### 7) Audio or Video-based Applications

Python is flexible to perform multiple tasks and can be used to create multimedia applications. Some multimedia applications which are made by using Python are **TimPlayer**, **cplay**, etc. The few multimedia libraries are given below.

- Gstreamer
- Pyglet

- QT Phonon

## 8) 3D CAD Applications

The CAD (Computer-aided design) is used to design engineering related architecture. It is used to develop the 3D representation of a part of a system. Python can create a 3D CAD application by using the following functionalities.

- Fandango (Popular )
- CAMVOX
- HeeksCNC
- AnyCAD
- RCAM

## 9) Enterprise Applications

Python can be used to create applications that can be used within an Enterprise or an Organization. Some real-time applications are OpenERP, Tryton, Picalo, etc.

## 10) Image Processing Application

Python contains many libraries that are used to work with the image. The image can be manipulated according to our requirements. Some libraries of image processing are given below.

- OpenCV
- Pillow
- SimpleITK

## *Python Variables*

Variable is a name that is used to refer to memory location. Python variable is also known as an identifier and used to hold value. Python has no command for declaring a variable. A variable is created the moment you first assign a value to it.

In Python, we don't need to specify the type of variable because Python is an infer language and smart enough to get variable type. Variable names can be a group of both the letters and digits, but they have to begin with a letter or an underscore.

### Rules for Python variables:

- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- Variables do not need to be declared with any particular type and can even change type after they have been set.

### ***Declaring and Initializing a variable***

```
x = 5  
print(x)
```

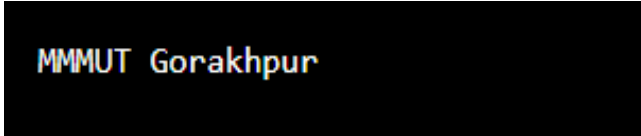
Output:

A black rectangular box representing a terminal window. Inside, the number '5' is displayed in a light blue monospace font.

### ***Variables can even change type after they have been set. Here x is now a string variable created using double quotes***

```
x = "MMMUT Gorakhpur"  
print(x)
```

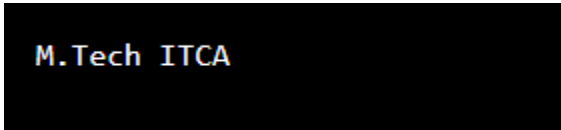
Output:

A black rectangular box representing a terminal window. Inside, the text 'MMMUT Gorakhpur' is displayed in a light blue monospace font.

### ***String variables can also be declared using single quotes***

```
x = 'M.Tech ITCA'  
print(x)
```

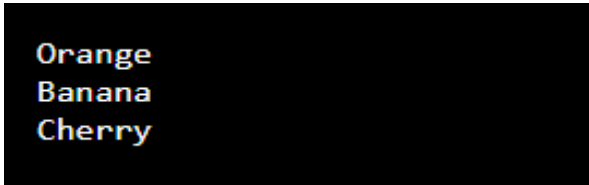
Output:

A black rectangular box representing a terminal window. Inside, the text 'M.Tech ITCA' is displayed in a light blue monospace font.

### ***Assign multiple values to multiple variables***

```
x, y, z = "Orange", "Banana", "Cherry"  
print(x)  
print(y)  
print(z)
```

Output:

A black rectangular box representing a terminal window. Inside, three lines of text are displayed in a light blue monospace font: 'Orange', 'Banana', and 'Cherry'.

### ***Assign the same value to multiple variables***

```
x = y = "Grapes"
print(x)
print(y)
```

Output:

```
Grapes
Grapes
```

### **Python Data Types**

Python is dynamically typed language (No need to mention data type based on value assigned, it takes data type). You can get the data type of any object by using the type ( ) function.

```
x = 5
print(type(x))
```

Output:

```
<class 'int'>
```

#### **i. Numeric Type**

There are three numeric types in Python:

- Int: Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.
- Float: Float, or "floating point number" is a number, positive or negative, containing one or more decimals.
- Complex: Complex numbers are written with a "j" as the imaginary part.

Variables of numeric types are created when you assign a value to them:

```
x = 1    #int
y = 2.8  #float
z = 1j   #complex
print(x)
print(y)
print(z)
```

Output:

```
1
2.8
1j
```

#### **ii. Boolean Type**

Boolean type provides two built-in values, True and False. These values are used to determine the given statement true or false. It denotes by the class bool.

```
x = True
print(x)
print(type(x))
```

Output:

```
True
<class 'bool'>
```

### ***iii. Text Type (Strings) : str***

Strings in python are surrounded by either single quotation marks, or double quotation marks. 'Hello' is the same as "Hello".

#### **#Assign string to a variable**

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

```
a = "Hello"
print(a)
```

#### **#We can assign a multiline string to a variable by using three single quotes or three double quotes**

```
a = '''Hi everyone,
This is a Python class'''
print(a)

b = """Hi everyone,
This is a Python class"""
print(b)
```

Output:

```
Hi everyone,
This is a Python class
Hi everyone,
This is a Python class
```

#### **#To concatenate, or combine, two strings we can use the + operator**

```
a = "Hello"
b = "Class"
c = a + " " + b
print(c)
```

Output:



## Hello Class

**#We cannot combine strings and numbers (error occurred)**

```
age = 30
txt = "My name is XYZ. I am" + age
print(txt)
```

**#String Slicing (You can return a range of characters by using the slice syntax. Specify the start index and the end index, separated by a colon, to return a part of the string)**

```
a = "Hello, World!"
print(a[2:5])    #get the characters from position 2 to position 5 (5 not included)
print(a[:5])     #get the characters from start position to position 5 (5 not included)
print(a[2:])     #get the characters from position 2 to last position
print(a[-2])     #get the characters position of negative 2
print(a[-2:])    #get the characters from negative 2 to last position
```

Output:

```
llo
Hello
llo, World!
d
d!
```

### iv. Sequence Type : list, tuple

**List:** List is a collection which is ordered, indexed and changeable. It allows duplicate members. Python Lists are similar to arrays in C. However, the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets []. We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (\*) works with the list in the same way as they were working with the strings.

```
list = [1, "hi", "Python", 2]
print(list)        #printing the list
print(type(list))  #checking type of given list
print(list[3:])     #list slicing
print(list[0:2])
print(list + list)  #list concatenation using + operator
print(list * 3)     #list repetition using * operator
```

Output:

```
[1, 'hi', 'Python', 2]
<class 'list'>
[2]
[1, 'hi']
[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]
[1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2, 1, 'hi', 'Python', 2]
```

**Tuple:** Tuple is a collection which is ordered, indexed and unchangeable. It allows duplicate members. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

```
tup = ("hi", "Python", 2)
print(tup)          #printing the tuple
print(type(tup))    #checking type of given tuple
print(tup[1:])      #tuple slicing
print(tup[0:1])
print(tup + tup)    #tuple concatenation using + operator
print(tup * 3)      #tuple repetition using * operator
t[2] = "hi"         #adding value to tup. It will throw an error
```

Output:

```
('Hi', 'Python', 2)
<class 'tuple'>
('Python', 2)
('Hi',)
('Hi', 'Python', 2, 'Hi', 'Python', 2)
('Hi', 'Python', 2, 'Hi', 'Python', 2, 'Hi', 'Python', 2)
```

#### v. Set Type : set

Set is a collection which is unordered, unindexed and unchangeable. It does not allow duplicate members. Sets are written with curly brackets {}.

```
set = {'James', 2, 2, 3, 'Python'}
print(set)          #Printing set value
set.add(10)          #Adding element to the set
print(set)
set.remove(2)        #Removing element from the set
print(set)
```

Output:

```
{3, 2, 'James', 'Python'}
{2, 3, 10, 'Python', 'James'}
{3, 10, 'Python', 'James'}
```

## vi. Mapping Type (Dictionary): dict

Dictionary is a collection which is unordered, indexed and changeable. It does not allow duplicate members. Dictionaries are written with curly brackets, and they have keys and values.

```
d = {"brand": "Ford", "model": "Mustang", "year": 1964}    #Unordered but indexed
print(d)
di = d["model"]
print(di)
print(d["year"])
print()
```

### **#Does not allow duplicate members**

```
dd= {"brand": "Ford", "model": "Mustang", "year": 1964, "year": 1964}
print(dd)
```

### **#Changeable since it is unordered but indexed**

```
x = {"brand": "Ford", "model": "Mustang", "year": 1964}
x["year"] = 2000
print(x)
```

Output:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Mustang
1964

{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
{'brand': 'Ford', 'model': 'Mustang', 'year': 2000}
```

## Other Concepts

### Operators and Expressions

- Arithmetic operators (+, -, \*, /, \*\*, //)
- Assignment operators (=, +=, -=, \*=, /=, %=, \*\*=, //=)
- Comparison operators (==, !=, <, >, <=, >=)
- Logical operators (and, or, not)
- Identity operators (is, is not)
- Membership operators (in, not in)
- Bitwise operators (&, |, ^, <<, >>)

### **Arithmetic operators**

\*\* is used for exponentiation and // is used for floor division.

a = 2

```
b = 5
x = 15
y = 2
print(a ** b)
print(x / y)
print(x // y)
```

Output:

```
32
7.5
7
```

### Logical operators

```
x = 5
print(x>3 and x<10)  #Returns True because 5 is greater than 3 AND 5 is less than 10
print(x>3 or x<4)    #Returns True because one of the conditions are true.
print(not(x>3 and x<10)) #Returns False because not is used to reverse the result.
```

Output:

```
True
True
False
```

### Identity operators

```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x
print(x is z)      #Returns True because z is the same object as x
print(x is not z)  #Returns False because z is the same object as x
print(x is y)      #Returns False because x is not the same object as y, even if they have the same
content
print(x is not y)  #Returns True because x is not the same object as y
print(x == y)      #To demonstrate the difference between "is" and "==": this comparison returns
True because x is equal to y.
```

Output:

```
True
False
False
True
True
```

## Membership operators

```
x = ["apple", "banana"]  
print("banana" in x)      #Returns True because a sequence with the value "banana" is in the list.  
print("pineapple" not in x) #Returns True because a sequence with the value "pineapple" is not  
                           in list.
```

Output:

```
True  
True
```

## Indentation in Python

- Indentation refers to the spaces at the beginning of a code line.
- Where in other programming languages the indentation in code is for readability only, the indentation in python is very important.
- Python uses indentation to indicate a block of code. Other languages often use curly brackets for this purpose.
- Python will give you an error if you skip the indentation.
- The number of spaces is up to you as a programmer, but it has to be at least one.
- You have to use same number of spaces in the same block of code, otherwise python will give you an error.

Example:

```
if 5>2:  
    print("five is greater than two")
```

## User input

Python allows for user input. That means we are able to ask the user for input. Python uses the input() method to take the input from the user.

**Example:**

```
username = input("Enter username:")  
print("Username is: " + username)
```

Output:

```
Enter username:Nida  
Username is: Nida
```

## Displaying Outputs

- print statement is often used to output variables.
- To concatenate two or more strings or string variables the + character is used.
- For numbers, the + character works as a mathematical operator.

- If you try to combine a string and a number, Python will give you an error.

### **Example:**

#To concatenate 2 strings or string variables the + character is used.

```
x = "Department of ITCA in "
```

```
y = "MMMUT Gorakhpur."
```

```
z = x + y
```

```
print("I am studying in " + "" + "BTech at " + z)
```

#For numbers, the + character works as a mathematical operator.

```
a = 1990
```

```
b = 30
```

```
print(a + b)
```

Ouput:

```
I am studying in BTech at Department of ITCA in MMMUT Gorakhpur.
2020
```

#If you try to combine a string and a number, python will give you an error.

```
x = 5
```

```
y = "XYZ"
```

```
print(x + y)
```

### **Comments**

- Comments starts with a #, and python will ignore them.
- Comments can be used to explain python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.
- Python does not really have a syntax for multiline comments.
- To add a multiline comment you could insert a # for each line:
- Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it.

### **Casting**

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

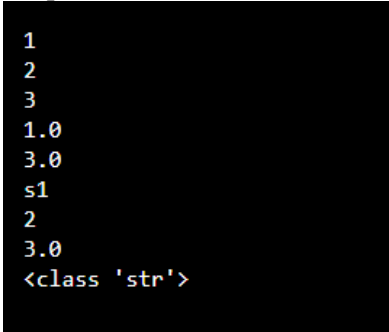
Casting in python is therefore done using constructor functions:

- **int()** - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
- **float()** - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- **str()** - constructs a string from a wide variety of data types, including strings, integer literals and float literals.

**Example:**

```
a = int(1)    # a will be 1
b = int(2.8)  # b will be 2
c = int("3")  # c will be 3
d = float(1)  # d will be 1.0
e = float("3") # e will be 3.0
f = str("s1") # f will be 's1'
g = str(2)    # g will be '2'
h = str(3.0)  # h will be '3.0'
print(a)
print(b)
print(c)
print(d)
print(e)
print(f)
print(g)
print(h)
print(type(f))
```

Output:



```
1
2
3
1.0
3.0
s1
2
3.0
<class 'str'>
```

### **Conditional Statements**

Statements used to control loops and change the course of iteration are called control statements.

Decision-making is as important in any programming language as it is in life. Decision-making in a programming language is automated using conditional statements, in which Python evaluates the code to see if it meets the specified conditions.

The conditions are evaluated and processed as true or false. If this is found to be true, the program is run as needed. If the condition is found to be false, the statement following the If condition is executed.

**The if statement:** The if statement is used to test a particular condition and if the condition is true, it executes a block of code known as if-block.

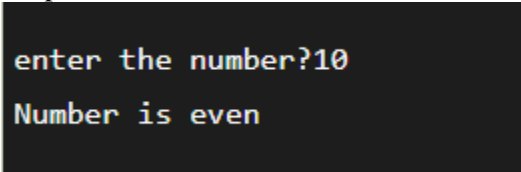
The syntax of the if-statement is given below.

```
if expression:
    statement
```

**Example 1**

```
num = int(input("enter the number?"))
if num%2 == 0:
    print("Number is even")
```

Output:

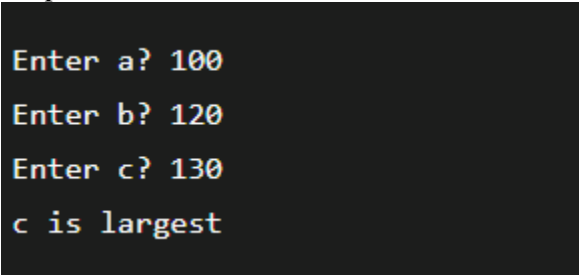


```
enter the number?10
Number is even
```

**Example 2 Program to print the largest of three numbers**

```
a = int(input("Enter a? "))
b = int(input("Enter b? "))
c = int(input("Enter c? "))
if a>b and a>c:
    print("a is largest")
if b>a and b>c:
    print("b is largest")
if c>a and c>b:
    print("c is largest")
```

Output:



```
Enter a? 100
Enter b? 120
Enter c? 130
c is largest
```

**The if-else statement:** The if-else statement provides an else block combined with the if statement which is executed in the false case of the condition. If the condition is true, then the if-block is executed. Otherwise, the else-block is executed.

The syntax of the if-else statement is given below.

if condition:

    #block of statements

else:

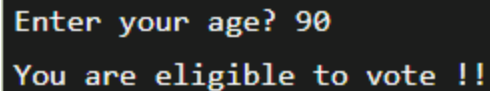
    #another block of statements (else-block)

**Example 1: Program to check whether a person is eligible to vote or not**



```
age = int (input("Enter your age? "))
if age>=18:
    print("You are eligible to vote !!")
else:
    print("Sorry! you have to wait !!")
```

Output:

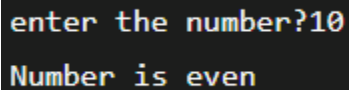


```
Enter your age? 90
You are eligible to vote !!
```

### Example 2: Program to check whether a number is even or odd

```
num = int(input("enter the number?"))
if num%2 == 0:
    print("Number is even...")
else:
    print("Number is odd...")
```

Output:



```
enter the number?10
Number is even
```

**The elif statement:** The elif statement enables us to check multiple conditions and execute the specific block of statements depending upon the true condition among them.

The syntax of the elif statement is given below.

```
if expression 1:
    # block of statements
elif expression 2:
    # block of statements
elif expression 3:
    # block of statements
else:
    # block of statements
```

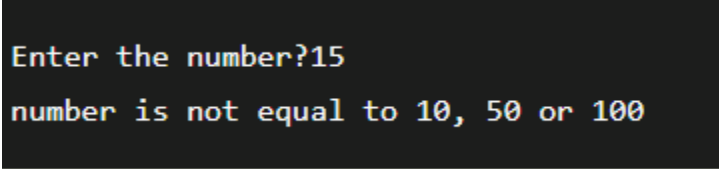
### Example 1:

```

number = int(input("Enter the number?"))
if number==10:
    print("number is equals to 10")
elif number==50:
    print("number is equal to 50")
elif number==100:
    print("number is equal to 100")
else:
    print("number is not equal to 10, 50 or 100")

```

Output:



```

Enter the number?15
number is not equal to 10, 50 or 100

```

### Example 2:

```

marks = int(input("Enter the marks? "))
if marks > 85 and marks <= 100:
    print("Congrats ! you scored grade A ...")
elif marks > 60 and marks <= 85:
    print("You scored grade B + ...")
elif marks > 40 and marks <= 60:
    print("You scored grade B ...")
elif (marks > 30 and marks <= 40):
    print("You scored grade C ...")
else:
    print("Sorry you are fail ?")

```

### pass Statement

- if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.
- 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.

### Example:

```
a = 33
b = 220
if b > a :
    pass
```