Python Programming

PYTHON

- Python is an object-oriented programming language created by Guido Rossum in the year 1989. The language platform aims at simplifying the complex applications. The Python scripts are stored in files which are named with the extension .**py**.

What is Python

- Python is a high-level programming language which is:
- Interpreted: Python is processed at runtime by the interpreter.
- Interactive: You can use a Python prompt and interact with the interpreter directly to write your Programs.
- Object-Oriented: Python supports Object-Oriented technique of programming.
- Beginner’s Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications.

Python Features

- Easy to learn, easy to read and easy to maintain.
- Portable: It can run on various hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter.
- Scalable: Python provides a good structure and support for large programs.
- Python has support for an interactive mode of testing and debugging.
- Python has a broad standard library cross-platform.
- Everything in Python is an object: variables, functions, even code. Every object has an ID, a type, and a value

Python interface

- Python has interfaces to many operating systems such as Windows, Linux, Unix, etc. Also, it is extensible to languages such as C and C++. Giants such as NASA, Google, YouTube, Bit Torrent use Python extensively in various projects.
- The latest version of Python, i.e. Python 3, is being widely used in artificial intelligence, natural language generation, neural networks and other advanced fields of computer

Basic Syntax

- **Indentation** is used in Python to delimit blocks. The number of spaces is variable, but all statements within the same block must be indented the same amount.
- The header line for compound statements, such as if, while, def, and class should be terminated with a colon (:)
- The semicolon (;) is optional at the end of statement.

```
if True:
    print ("Answer")
    print ("True")
else:
    print ("Answer")
  print ("False")
```

→Error!

- Printing to the Screen:

```
print ("Hello, Python!")
```

- Reading Keyboard Input:

```
name = input("Enter your name: ")
```

- **Comments**
  - Single line:

```
# This is a comment.
```

  - Multiple lines:

```
'''
print("We are in a comment")
print ("We are still in a comment")
'''
```

- Python files have extension .py

TOKENS

- A token is the smallest element of a program that is meaningful to the interpreter. Tokens supported in python include identifier, keywords delimiter and operator

identifiers

- **Identifiers** can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _ .
- An **identifier** cannot start with a digit.
- Keywords cannot be used as **identifiers**.
- We cannot use special symbols like !, @, #, $, % etc.
- An **identifier** can be of any length.
- An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.

Creating Variables

Python has no command for declaring a variable.
A variable is created the moment you first assign a value to it.

Example

```
x = 5
y = "John"
print(x)
print(y)
```

Variables do not need to be declared with any particular *type*, and can even change type after they have been set.

Example

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

Casting

- If you want to specify the data type of a variable, this can be done with casting.
- Example

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Get the Type

You can get the data type of a variable with the `type()` function.

Example

```
x = 5
y = "John"
print(type(x))
print(type(y))
```

Declaration of string variables

- String variables can be declared either by using single or double quotes:
- Example
- x = "John"
  # is the same as
  x = 'John'

Keywords

- Keywords are the reserved words in python and cannot be used as constant or variable or any other identifier names

KEYWORDS

- Keywords are the words that convey a special meaning to the language compiler/interpreter. These are reserved for special purpose and must not be used as normal identifier names

false	**assert**	**del**	**for**	**in**	**or**
while	none	break	elif	from	is
with	true	class	else	global	and
continue	except	if	nonlocal	return	def

LITERALS/VALUES

- Literals (often referred to as constant-values) are data items that have fixed value.
- Python allows several kinds of Literals:
  - String literals
  - Numeric literals
  - Boolean literals
  - Special literal None

Simple Input and Output

```
x=10
print x
x=“hello world”
print x
```

Using input function in python

```
name=input()
email=input()
print ("hi",name,email)
```

Number System

- Binary(0,1) only machine language
- Octal (0-7)
- Decimal (0-9)
- Hexa decimal (0-9, A-F)10+6=16 (hex) –keyword for conversion
- Divided by 16
- Converting of Decimal into binary (bin) –keyword for conversion
- Divided by 2
- Converting of Decimal into Octal (oct) –keyword for conversion
- Divided by 8

Conversion of values into different datatypes

```
var =10
y=bin(var)
print(y)

var =10
y=hex(var)
print(y)
```

Operators in Python

Operators are tokens that trigger some computation when applied to variables and other objects in an expression . Variables and objects to which the computation is applied are called operands

Operators

- Python Operators
- Operators are used to perform operations on variables and values.

- In the example below, we use the + operator to add together two values:

- Example
- print(10 + 5)

Types of operators

- Unary Operators : operators that requires one operand to operate upon

  +5 , -5, not 5,~ complement

- Binary operators: operators that required two operands to operate upon.
  - Arithmetic Operators

5+10	addition	5*10	Multiplication	10/5	Division
5-10	subtraction	10%5	Modulus/Remainder	5**2	Exponent (raise to power

  - Relational operators

5<10	Less than	5<=10	Less than or equal to	10>5	Greater than
10>=5	Greater than or equal to	10!=5	Equal to	5==10	Equal to

Logical operators

and Logical AND　　or Logical OR

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Python Arithmetic Operators

- Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Program using Arithmetic operators (sum)

```
x=int(input())
y=int(input())
sum=x+y
print(sum)

age=int(input())
Print("after 20years",age+20)
```

Program using Arithmetic operators (Difference)

```
x=int(input())
y=int(input())
difference=x-y
print(difference)

age=int(input())
Print(“my age before 10 years”,age-10)
```

Write a program to read distance in miles and print in kilometers

```
m=float(input())
km=1.609*m
print(“The distance is calculated in kilometres”,km)
```

Write a program to read distance in feet and print in meters

```
ft=float(input())
m=ft/3.2808
print(“ The distance calculated in meter”, m)
```

Find the given number is odd or even

```
x=int(input())
y=x%2
print("if y is zero then it is even number",y)
print("if y is one then it is odd number", y)
```

Write a program to find power of given value

```
x=int(input())
y=x**2
Print("The square value of x is=" , y)
```

Write a program to calculate Fahrenheit to Celsius?

```
f=float(input())
c=(f-32)*5/9
print(“ The temperature measured in Fahrenheit is”,f , “The value of converted temperature in Celsius is”, c)
```

Write a program to find simple interest

```
p=float(input())
r=int(input())
n=float(input())
si=(p*r*n)/100
print(“ simple interest for given principal “,p, ”rate of interest”,r,
“number of years”, n “is”,si )
```

Area of a circle

```
r=int(input())
a=22/7*r**2
print(“area of a circle is”,a)
```

Simple calculator

```
x=int(input())
y=int(input())
Print("sum=",x+y, "diff",x-y, "multiplication=" , x*y, "division=",x/y)
```

Write a program to read distance in miles and print in kilometers

```
m=23
km=1.609*m
print(km)
```

Logical operators Definition

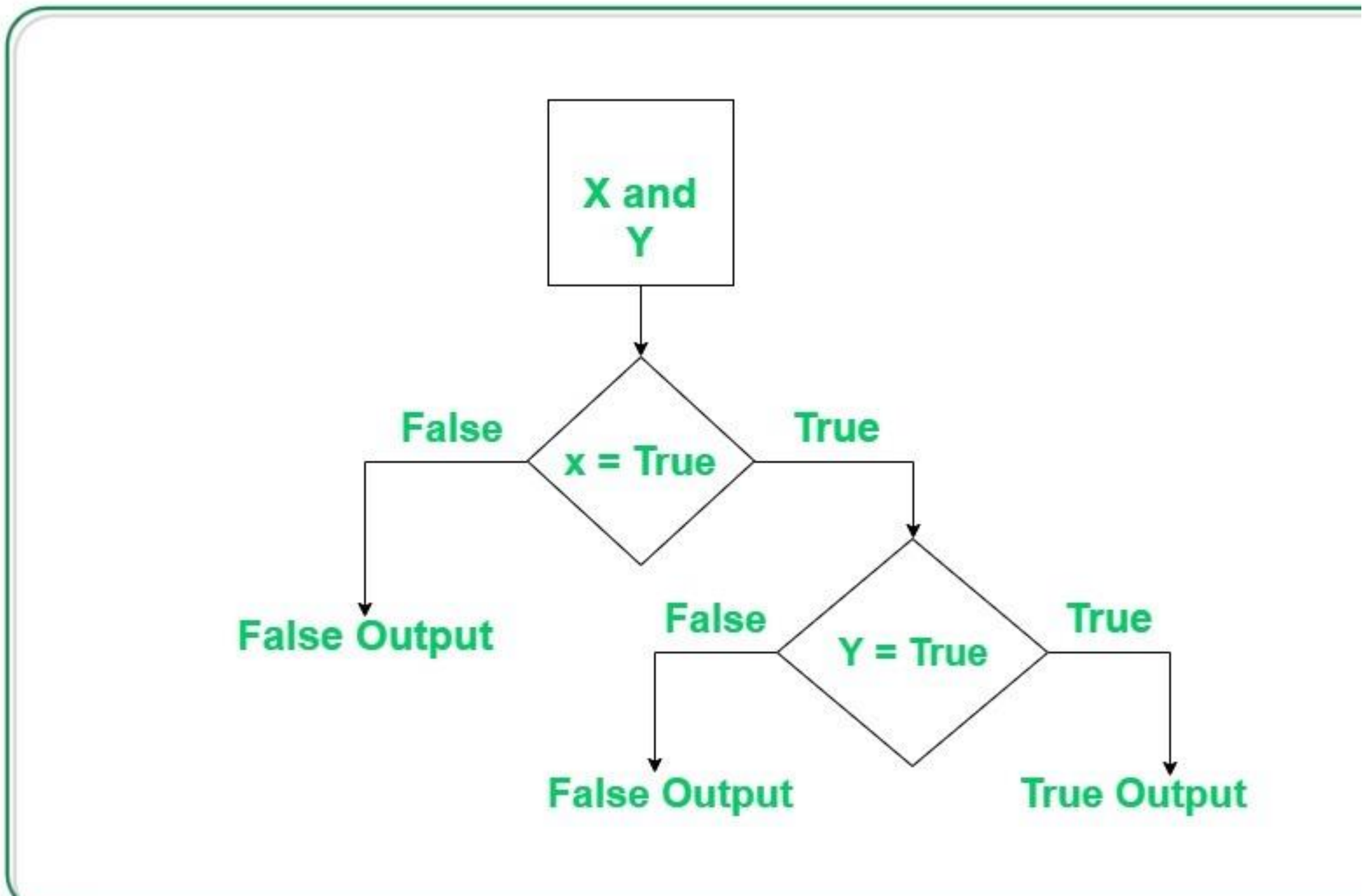
- There are three logical operators that are used to compare values. They evaluate expressions down to Boolean values, returning either True or False . These operators are **and , or , and not** and are defined in the table below

LOGICAL OPERATORS

OPERATOR	DESCRIPTION	SYNTAX	EXAMPLE
and	Logical AND: True if both the operands are true	x and y	x < 5 and x < 10
or	Logical OR: True if either of the operands is true	x or y	x < 5 or x < 4
not	Logical NOT: True if operand is false	not x	not(x < 5 and x < 10)

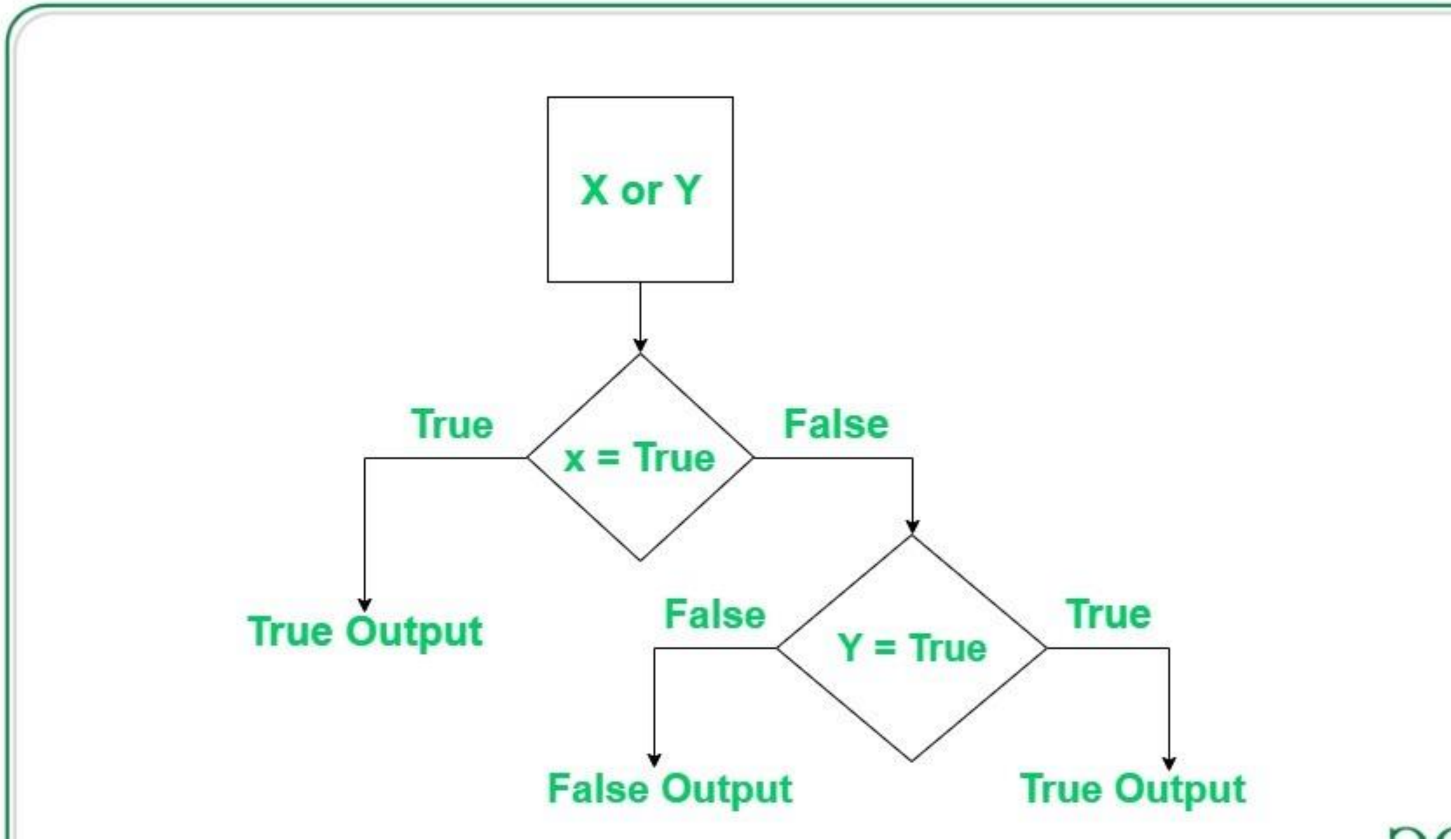
Logical AND operator

- Logical operator returns True if both the operands are True else it returns False



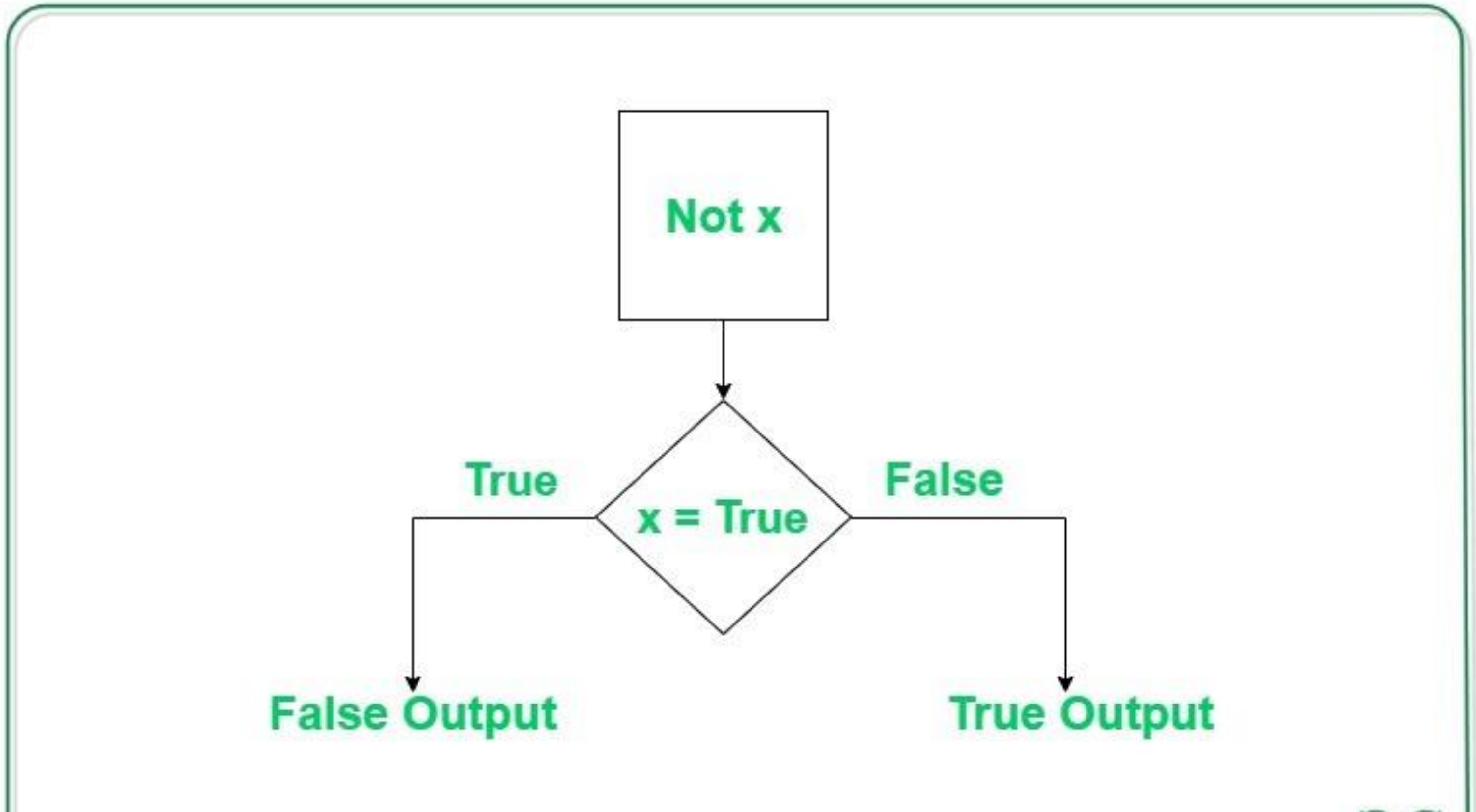
Logical or operator

Logical or operator returns True if either of the operands is True.



Logical not operator

- Logical not operator work with the single boolean value. If the boolean value is True it returns False and vice-versa.



Python Conditions and If statements

- Python supports the usual logical conditions from mathematics:
- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- These conditions can be used in several ways, most commonly in "if statements" and loops.
- An "if statement" is written by using the if keyword.

EXAMPLE OF IF STATEMENT

```
a = 333

b = 200

if b > a:

  print("b is greater than a") (THIS STATEMENT  INDENTATION which is

used to represent it belongs to if statement or if condition)
```

Python if Statement Flowchart

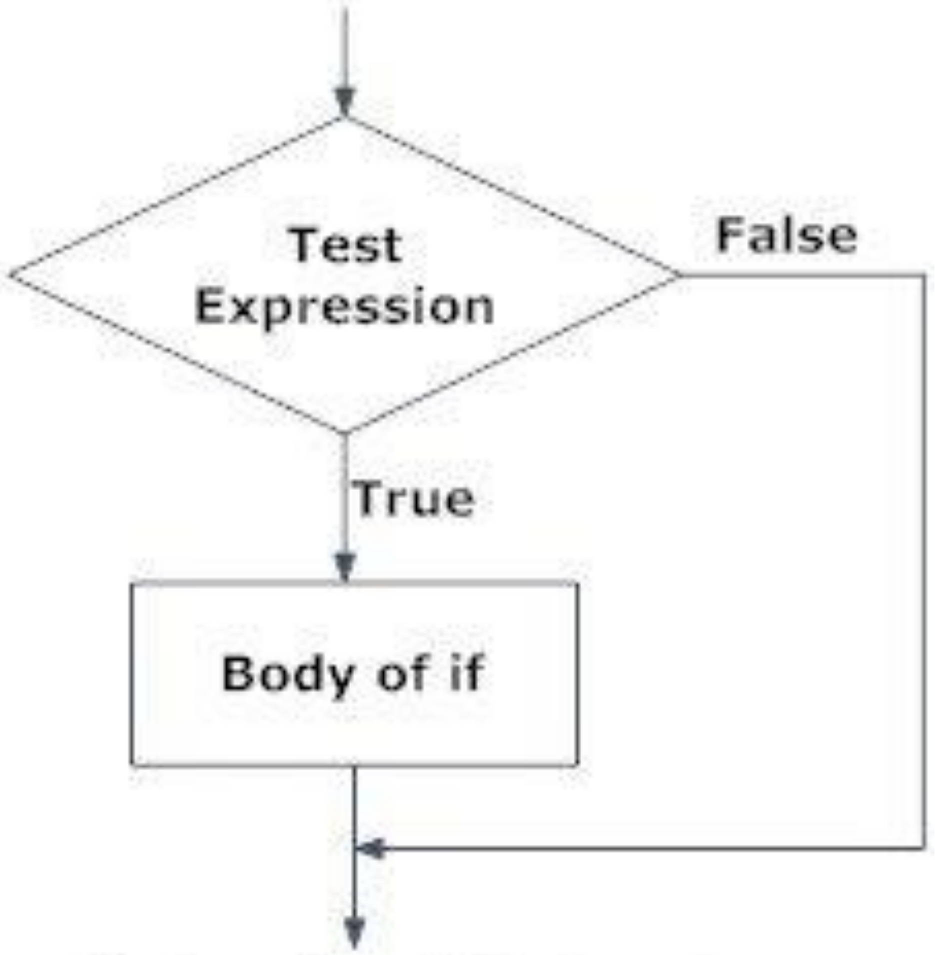


Fig: Operation of if statement

What is if...else statement in Python?

- Decision making is required when we want to execute a code only if a certain condition is satisfied.
- The if……else statement is used in Python for decision making.
- Python if Statement Syntax

```
if test expression:
   statement(s)
```

Syntax of if...else

```
if test expression:
    Body of if
else:
    Body of else
```

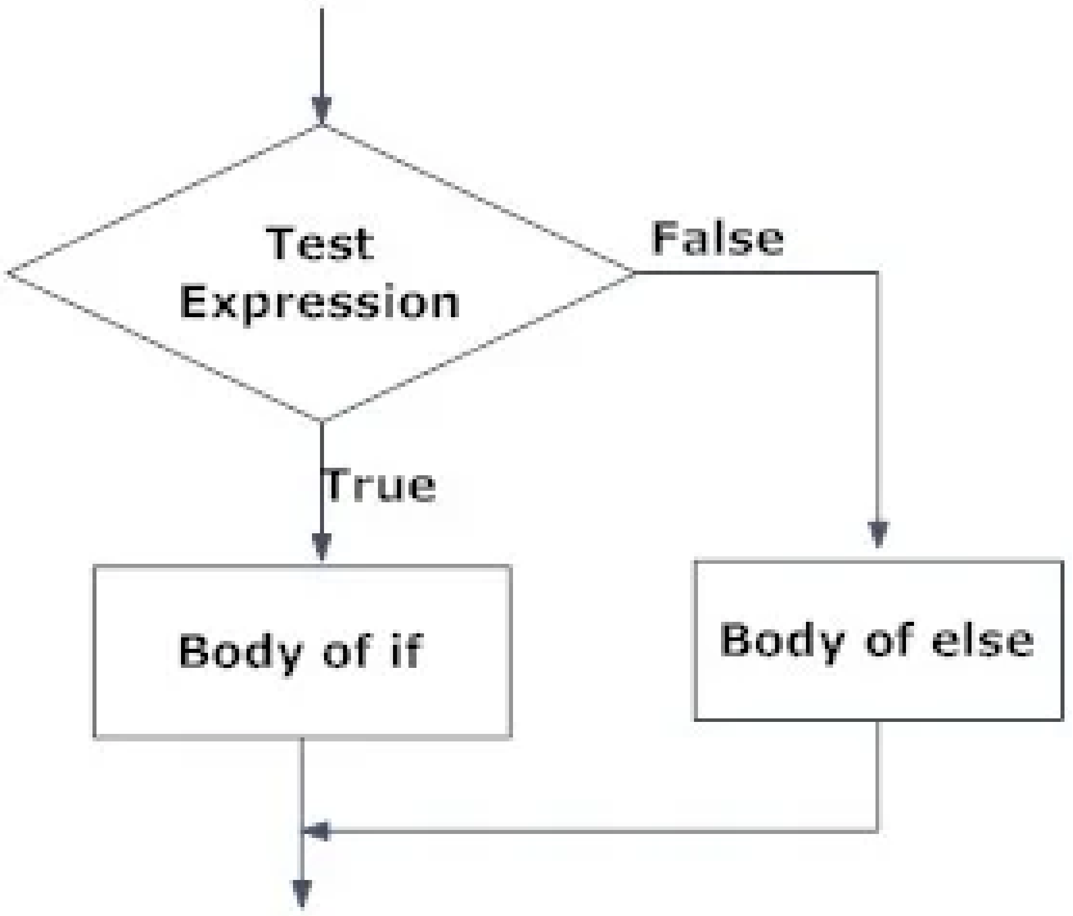


Fig: Operation of if...else statement

Elif

- The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

<u>Example</u>

-
  ```
  a = 33
  b = 33
  if b > a:
    print("b is greater than a")
  elif a == b:
    print("a and b are equal")
  ```

Else

- The else keyword catches anything which isn't caught by the preceding conditions.

Example

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

And

- The and keyword is a logical operator, and is used to combine conditional statements:

Example

Test if a is greater than b, AND if c is greater than a:

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

Or - operator

- The or keyword is a logical operator, and is used to combine conditional statements:

Example

Test if a is greater than b, OR if a is greater than c:

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

Nested If

- You can have if statements inside if statements, this is called nested if statements.

- Example

```
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

The 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.

- <u>Example</u>

```
a = 33
b = 200

if b > a:
  pass
```

Python Booleans

- Booleans represent one of two values: True or False.

<u>Boolean Values</u>

- In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer:

<u>Example</u>

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

Example

Print a message based on whether the condition is True or False:

```
a = 200
b = 33

if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Evaluate Values and Variables

The bool() function allows you to evaluate any value, and give you True or False in return,

Example

Evaluate a string and a number:

```
print(bool("Hello"))
print(bool(15))
```

Evaluate two variables:

```
x = "Hello"
y = 15

print(bool(x))
print(bool(y))
```

Most Values are True

Almost any value is evaluated to True if it has some sort of content.

Any string is True, except empty strings.

Any number is True, except 0.