PYTHON PROGRAMMING

Table of Contents

4 CONDITIONAL STATEMENTS

- Conditional Statements
- Simple if Statement
- if..else Statements
- if...elif...else
- Short Hand if

5 LOOPING CONSTRUCTS

- Looping Constructs
- for Loop
- range( ) Function
- Using else statement with for loop
- While loop
- Using else statement with While Loop

6 NESTED LOOPS

- Nested For Loops
- break
- continue
- pass

7 PYTHON ARRAYS

- Arrays
- How to use array in a program
  NumPy
- Basic operations
- Matrix Multiplication

8 PYTHON STRINGS

- Strings
- len( ), replace( ), del( ), upper( )
- String Operators
- Concatenation
- String Repetition Operator
- String Slicing
- Sample Programs
- Built-in String functions

9 PYTHON FUNCTIONS

- Functions
- Advantages of Functions
- Types of Functions
- User Defined Functions
- Return Statement
- Lambda Functions
- Recursive Functions
- Built-in Functions

10 PYTHON LISTS

- Introduction to Lists
- List methods in Python
- Nested List

11 PYTHON TUPLES

- Tuples
- Pack Unpack
- Tuples method

12 PYTHON SETS AND DICTIONARY

- Sets
- Set Operations
- Dictionary – Key Value Pairs

13 PYTHON CLASSES AND OBJECTS

- Classes and Objects
- Creating Classes- Variables , Methods

14 PYTHON MODULES AND PACKAGES

- Modules
- Library
- Important Packages

15 PYTHON DATE TIME MODULE

- Date time module
- Various date time format

MODULE 1

INTRODUCTION TO PYTHON

HISTORY OF PYTHON

Python is a general purpose programming language created by Guido van Rossum at CWI (CentrumWiskunde& Informatica) which is a National Research Institute for Mathematics and Computer Science in Netherland during 1991.

Python got its name from a BBC comedy series – "Monty Python's Flying Circus".

Guido Van Rossum

Born	**31 January 1956**
Nationality	**Dutch**
Occupation	**Computer programmer, author**
Awards	**Advancement of Free Software (2001)**

KEY FEATURES OF PYTHON

- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- Python is a powerful language. Python is easy to learn and understand.
- Python works on different platforms (Windows, Mac, Linux etc).
- Python has a simple syntax similar to the English language.Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python supports automatic garbage collections.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written.
- Python can be treated in a procedural way or an object-oriented way
- It can be easily integrated with C, C++, Java

REAL-WORLD APPLICATIONS OF PYTHON

- Web Development
- Game Development
- Scientific and Numeric Applications
- Data Visualization
- Machine Learning
- Artificial Intelligence
- Embedded Applications
- Data Analysis
- App Development

SYSTEM REQUIREMENTS FOR INSTALLING PYTHON

Operating Systems and CPU architecture:

* Windows 7 or 10

* Mac OS X 10.11 or higher, 64-bit

* Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu)

RAM and free disk space:

4 GB RAM

5 GB free disk space

Google colab

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

PyCharm is a cross-platform IDE that provides consistent experience on the Windows, macOS, and Linux operating systems.

PyCharm is available in three editions: **Professional, Community, and Edu.**
The Community and Edu editions are open-source projects and they are free, but they have fewer feature

The version 3.x of **Python IDLE** (Integrated Development Learning Environment) is used to develop and run Python code. It can be downloaded from the web resource www.python.org.

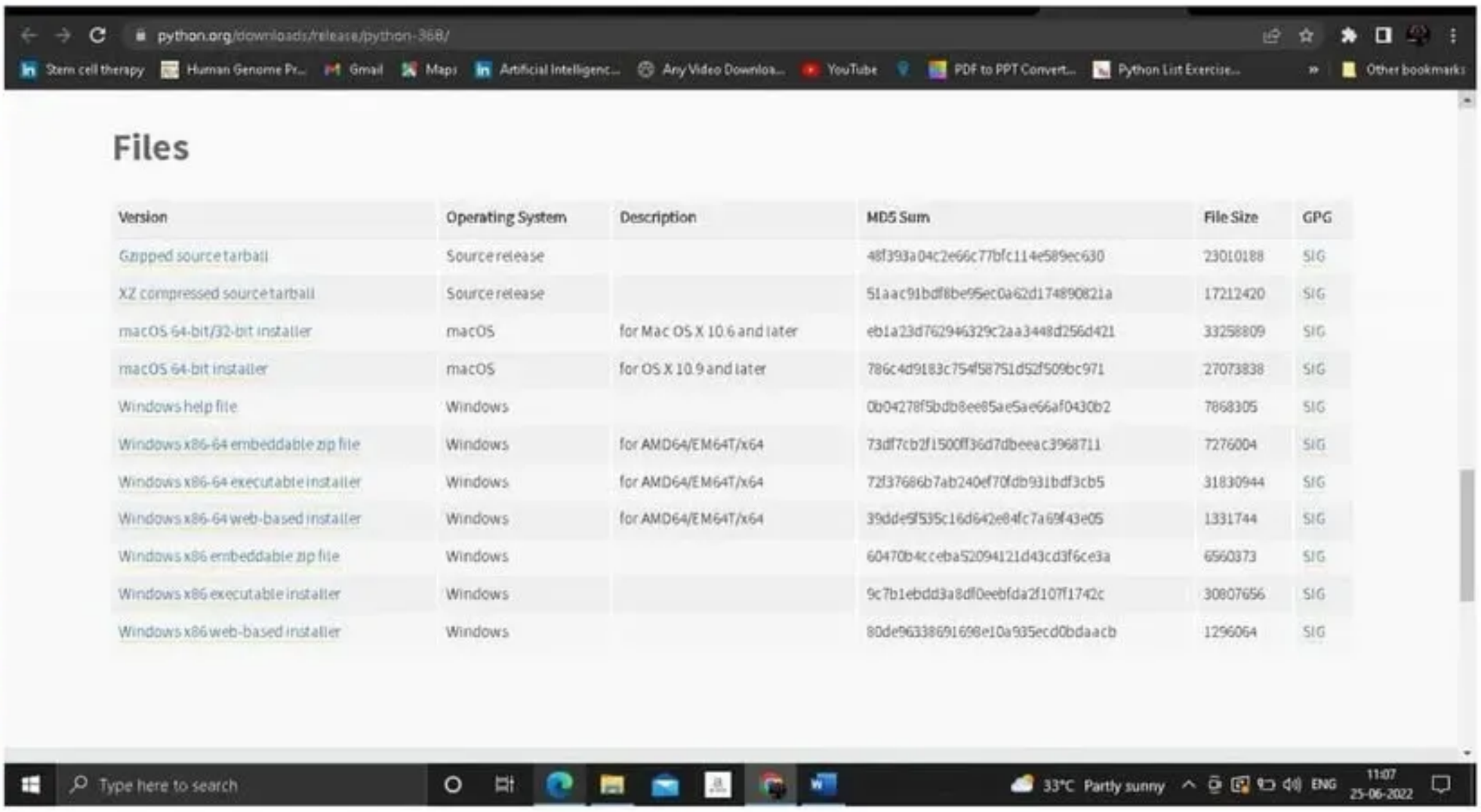
PROCEDURE TO INSTALL PYTHON 3 ON WINDOWS OS:

Install Python 3.6:

1. To follow the installation procedure, you need to be connected to the Internet.

2. Visit https://www.python.org/downloads/release/python-368/

3. At the bottom locate Windows x86-64 executable installer for 64 bits OS and Windows x86 executable installer for 32 bits OS



4. Click on the located installer file to download.

Note: After double clicking the installer, check mark the option “Add Python 3.6 to PATH”

5. After download completes, double click on the installer file to start the installation procedure.

6. Follow the instructions as per the installer

SCRIPT MODE VS INTERACTIVE MODE

In Python, programs can be written in two ways namely Interactive mode and Script mode. The Interactive mode allows us to write codes in Python command prompt (>>>) whereas in script mode programs can be written and stored as separate file with the extension .py and executed. Script mode is used to create and edit python source file. Programs written in script mode can be used later.

INTERACTIVE MODE PROGRAMMING

In interactive mode Python code can be directly typed and the interpreter displays the result(s) immediately. The interactive mode can also be used as a simple calculator.

helloworld.py

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

COMMENTS

The comments make the source code easy to understand. It can also be used to prevent Python from executing code:

In Python, single line comments begin with hash symbol (#). The lines that begins with # are considered as comments and ignored by the Python interpreter.

```
#This is a comment
print("Welcome to python")
```

The multiline comments should be enclosed within a set of ''' '''(triple quotes) which contains more than one line

```
''' This is a comment
print("Welcome to python") '''
```

This will not be executed by Python Interpreter

INDENTATION

Python uses whitespace such as spaces and tabs to define program blocks whereas other languages like C, C++, java use curly braces { } to indicate blocks of codes for class, functions or body of the loops and block of selection command. The number of whitespaces (spaces and tabs) in the indentation is not fixed, but all statements within the block must be indented with same amount spaces.

MODULE 2

BASICS OF PYTHON

PYTHON CHARACTER SET

Character set is the set of valid characters that a language can recognize. A character represents any letter, digit or any other symbol.

Python has the following character sets:

LETTERS	A-Z, a-z
DIGITS	0-9
SPECIAL SYMBOLS	space,+,-,*,/,**,\, (),{},[],//,==,!=,<,>,:,;,% and etc
WHITESPACES	Blankspace,tabs,carriage return,newline, formfeed
OTHER CHARACTERS	Python can process all ASCII and Unicode characters

INPUT( ) AND OUTPUT( ) FUNCTIONS

The input( ) function helps to enter data at run time by the user and the output function print( ) is used to display the result of the program on the screen after execution.

input( ) functions:

The syntax for input( ) function is,

Variable = input ("prompt string")

Where prompt string is the message to the user

Sample code

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

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

Simple input( ) function without prompt is also available.

The input ( ) accepts all data as string or characters but not as numbers. If a numerical value is entered, the input values should be explicitly converted into numeric data type. The int( ) function is used to convert string data as integer data explicitly.

```
s =int(input("Enter any number"))
Enter any number12
```

print( ) functions

To print or display output, Python provides print( ) function.

Sample code

```
print ("Hello World!")
```

output

```
Hello World!
```

Sample code

```
>>> name=input("Enter your name : ")
    Enter your name : Radha
>>> print("Welcome ",name)
    Welcome  Radha
>>>
```

The **sep** parameter is primarily used to format the strings that need to be printed on the console and add a separator between strings to be printed. This feature was newly introduced in Python 3.x version.

```
>>> print (10,20,30, sep ="--")
10--20--30
```


```
print(10,20,30,sep='%')
```

output

10%20%30

The end parameter is primarily used to append any string at the end of the output of the print statement in python.

```
print(100,200,300,end="$")
```

output

100 200 300$

TOKENS

A token is the smallest individual unit in a python program. All statements and instructions in a program are built with tokens.

Python has the following tokens:

(i) Keyword

(ii) Identifiers

(iii) Literals

(iv) Operators and

(v) Punctuators

Keywords

Keywords are the reserved words in Python. We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language. In Python, keywords are case sensitive.

Few keywords are given below.

Elif	**Del**	**True**	**False**	**Except**
Global	For	If	While	import
And	Try	Or	Return	Pass
Class	In	Not	Is	lambda

Identifiers :

An Identifier is a name used to identify a variable, function, class, module or object.

- An identifier must start with an alphabet (A..Z or a..z) or underscore (_).
- Identifiers may contain digits (0 .. 9)
- Python identifiers are case sensitive i.e. uppercase and lowercase letters are distinct.
- Identifiers must not be a python keyword.
- Python does not allow punctuation character such as %,$, @ etc., within identifiers.

Example of valid identifiers

num, total_marks, num1

Example of invalid identifiers

12num, name$, total-mark, pass

LITERALS:

Literal is a raw data given to a variable or constant. In Python, there are various types of literals

String literals:

String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes to create a string.

Example:

"python programming…"

Numeric Literals:

Numeric literals can belong to 3 different numerical types

** Integer,

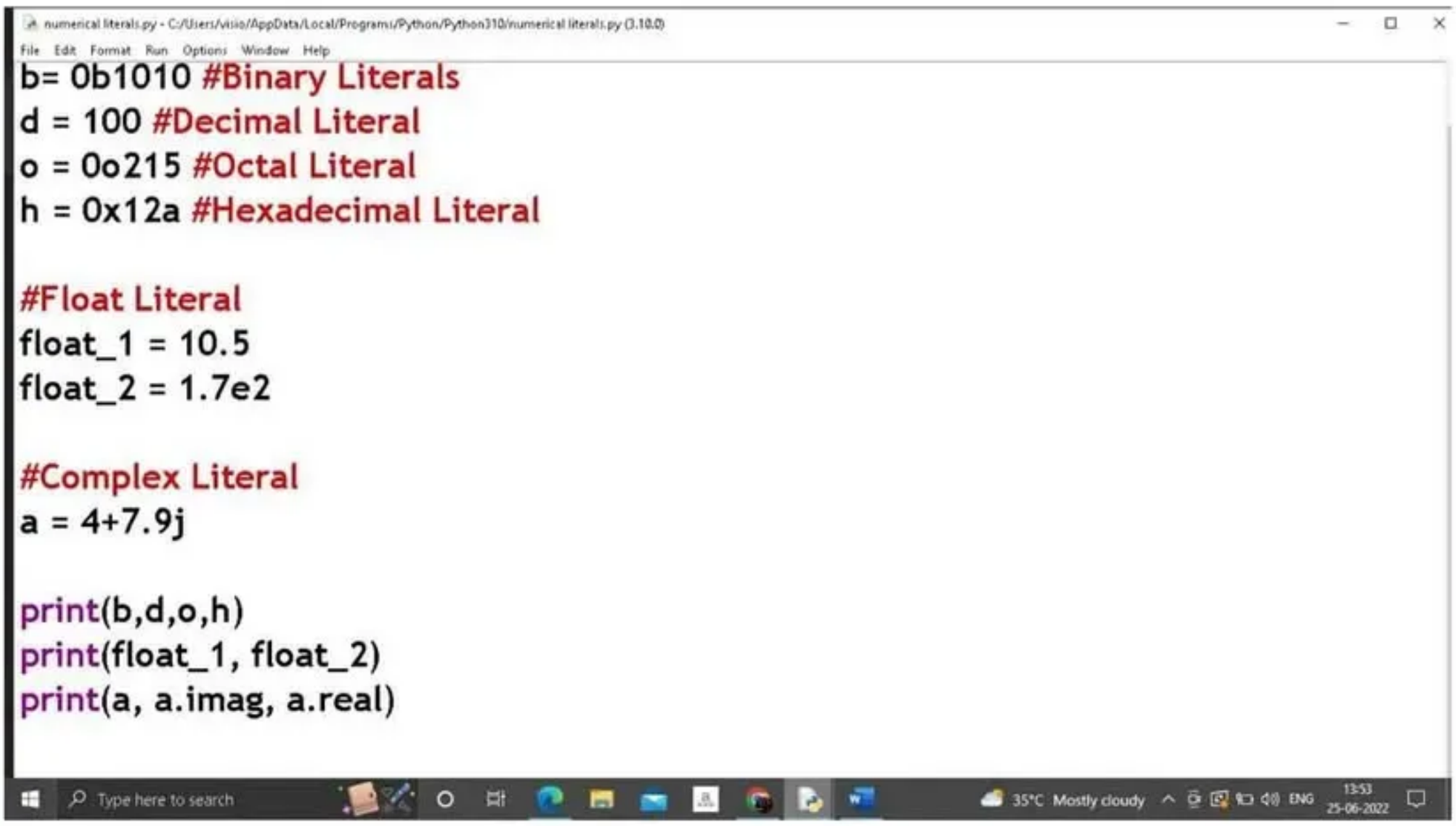
** Float and

** Complex Numbers

Sample code

 # Program to demonstrate Numeric Literals

Execute it and give output



Punctuators

Punctuators are the symbols with specific meanings.Punctuators are used in python to organize the structures, statements and expressions.

Some of the Punctuators are: [ ] { } ( ) -= += *= //= **= /=

Escape Sequence

Python allows to have certain non-graphic characters in string values. Such characters cannot be typed directly from keyboard. e.g. backspace, tabs, carriage return. These can be represented by escape sequence. An escape sequence is represented by backslash followed by one or more characters. Following are the list of escape sequence:

\" - Double quotes

\' - Single quotes

\\ - Backslash

\b - Backspace

\n - New line

\r - Carriage return

\t - Horizontal tab

\v - Vertical tab

Boolean literals

Used to represent one of the two boolean values, that is True and False.

For example:

```
>>>bool=(6>10)
>>>print (bool)
False
```

MODULE 3

OPERATORS

OPERATORS

Operators are used to perform operations on variables and values.

TYPES OF OPERATORS

Various types of operators available in Python are

- Arithmetic Operators
- Relational or Comparative Operators
- Logical Operators
- Assignment Operators
- Conditional Operators
- Identity Operators
- Membership Operators
- Bitwise Operators

Precedence of Operators :

The precedence of operators determines which operator is executed first if there is more than one operator in an expression.

Precedence:

P – Parentheses

E– Exponentiation

M – Multiplication (Multiplication and division have the same precedence)

D – Division

A – Addition (Addition and subtraction have the same precedence)

S – Subtraction

Arithmetic Operators :

Arithmetic Operators are used to perform mathematical calculations like addition, subtraction, multiplications , divisions. Operator is the symbol for calculations and operand is the value on which the operator acts.

Operator	Description
+ (Addition)	It is used to add two operands. a = 20, b = 20 a+b = 40
- (Subtraction)	It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. a = 30, b = 20 a - b = 10
/ (divide)	It returns the quotient after dividing the first operand by the second operand. a = 40, b = 20 a/b = 2.0
*** (Multiplication)**	It is used to multiply one operand with the other. a = 20, b = 10 a * b = 200
% (remainder) Modulo	It returns the remainder after dividing the first operand by the second operand. a = 200, b = 100 a%b = 0
**** (Exponent)**	It is an exponent operator represented as it calculates the first operand power to the second operand.
//(Floor division)	It gives the quotient produced by dividing the two operands. Integer division

TRY IT YOURSELF

```
x = 15
y = 3
print(x+y)
print(x-y)
print(x*y)
print(x/y)
print(x%y)
print(x//y)
print(x**y)
```

Relational Operators or Comparative operators

A Relational operator is also called as Comparative operator which checks the relationship between two operands.

>Greater than:

returns True if the left operand is greater than the right

x> y

<Less than:

returns True if the left operand is less than the right

x< y

==Equal to:

returns True if both operands are equal

x == y

!=Not equal to

returns True if operands are not equal

x != y

>=Greater than or equal to:

returns True if left operand is greater than or equal to the right

x >= y

<=Less than or equal to:

returnsTrue if left operand is less than or equal to the right

x <= y

Find output for the following code snippets

```
x = 5
y = 8
print("x == y:", x == y)
print("x != y:", x != y)
print("x < y:", x < y)
print("x > y:", x > y)
```

```
print("x <= y:", x <= y)
print("x >= y:", x >= y
```

output

```
x == y: False
x != y: True
x < y: True
x > y: False
x <= y: True
x >= y: False
```

TRY IT YOURSELF

```
print(20.2 )
print(20.2 < 21.3)
print(20.2 >21.3< 22.4)
print(20.2 < 21.3 < 22.4 < 23.5)
```

Logical Operators

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

Operator	What it means	Example
and	True if both are true	x and y
or	True if at least one is true	x or y
not	True only if false	not x