Scripting Language-Python (4330701)

A. V. PAREKH TECHNICAL INSTITUTE, RAJKOT

Teaching Scheme

Teaching Scheme (Hrs.)			Total Credits (L + T/2 + L/2)	Examination Scheme				
				Theory Marks		Practical Marks		Total Marks
L	T	P	C	CA*	ESE	CA	ESE	
3	-	4	5	30	70	25	25	150

(*): Out of 30 marks under the theory CA, 10 marks are for assessment of the micro- project to facilitate integration of COs and the remaining 20 marks is the average of 2 tests to be taken during the semester for the assessing the attainment of the cognitive domain UOs required for the attainment of the COs.

Fundamentals of Python

UNIT - I

Unit-1: Fundamentals of Python

CO1 - Develop programs to solve the given simple computational problems.

1.1 Introduction to Python, History of Python, Python Features, Python Applications
1.2 Installing Python
1.3 Basic Structure of Python program
1.4 Keywords and Identifiers
1.5 Data types and Variables
1.6 Type Casting
1.7 Input-Output functions: input, print
1.8 Operators

Introduction to Python

- Python is an open-source, high-level, general-purpose scripting type programming language used for software development.
- It is one of the most popular programming languages in the world today and known for its simplicity as well as rich library.
- It is widely used programming language in various domains, such as Automation, Server-side Web Development, Tools Development, Game Programming, Blockchain, Data Science, Artificial Intelligence, Machine Learning, Big Data etc.
- It's relatively easy to learn to use and incredibly versatile.

Scripting v/s Non-Scripting Language

CATEGORY	SCRIPTIN LANGUAGE	NON-SCRIPTING LANGUAGE
Compilation	Generally Interpreted	Usually Compiled
Execution	Interpreter So Compiled and Executed Line by Line	Generally Compiled, So the whole Program is Compiled at Once.
Execution Speed	Slow	Faster
Code Intensive	Usually Less Coding required	More code required.
Example	JavaScript, Python, PHP, Ruby, Perl etc.	C, C++, Java. C#, VB.NET etc.

History of Python

- Guido Van Rossum – Father/Founder of Python.
- Developed in late 80s as a hobby project during Christmas Holiday Season
- Derived features from many other languages including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).
- Versions:
  - 1991: Guido released 1st version of code (labelled as 0.9.0) to alt.sources
  - 1994: Python 1.0 released with features like lambda, map, filter and reduce.
  - Python 2.0 added features like list comprehension, global garbage collection etc.
  - December 3rd, 2008: Python 3.0 (aka Python 3000 or Python3k) released, where some fundamental flaws of the language and duplicate code were removed.

Features of Python

Easy – to – Learn	• Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
Easy -to- Read	• Python code is more clearly defined and visible to the eyes.
Easy-to-Maintain	• Python's source code is fairly easy-to-maintain
Broad Standard Library	• Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Mac.
Interactive Mode	• Python has support for an interactive mode which allows interactive testing and debugging of code. Python is Interpreted so this feature is supported.
Portable	• Python can run on a wide variety of hardware platforms and has the same interface on all platforms
Extendable	• You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
Databases	• Python provides interfaces to all major commercial databases.
GUI Programming	• Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems
Scalable	• Python provides a better structure.

Applications of Python

Python Download

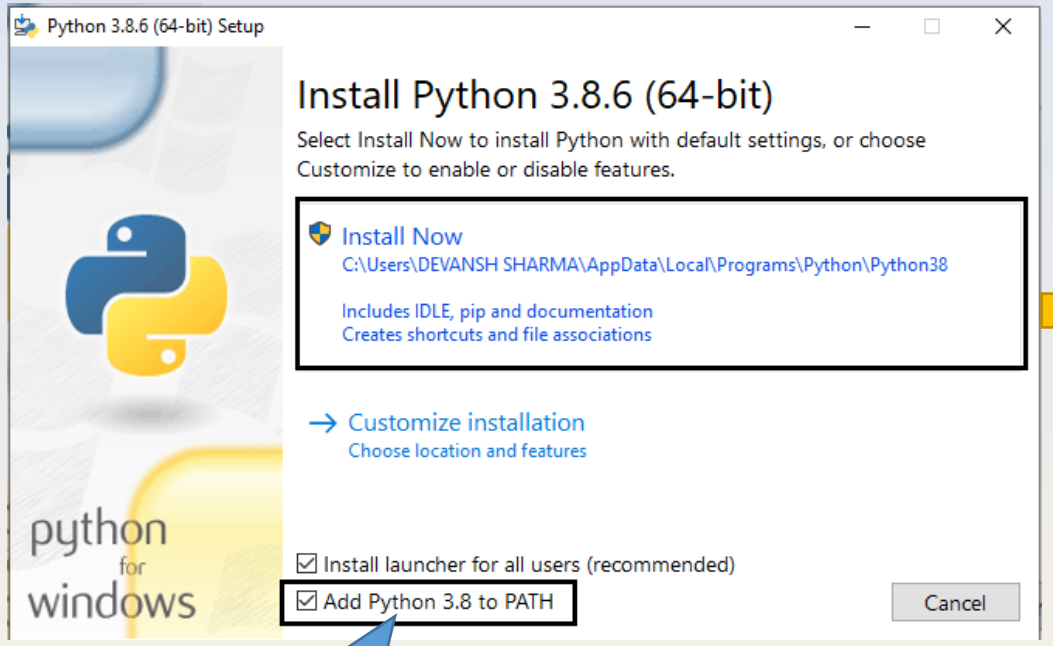
- Visit the link *https://www.python.org/downloads/* to download the latest release of Python.

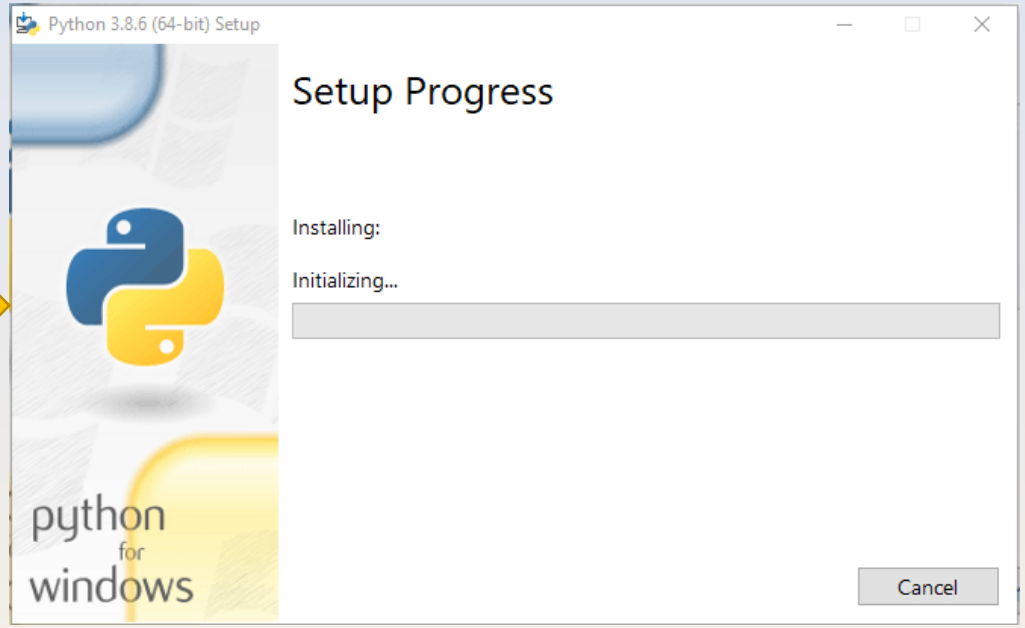
Active Python Releases

For more information visit the Python Developer's Guide.

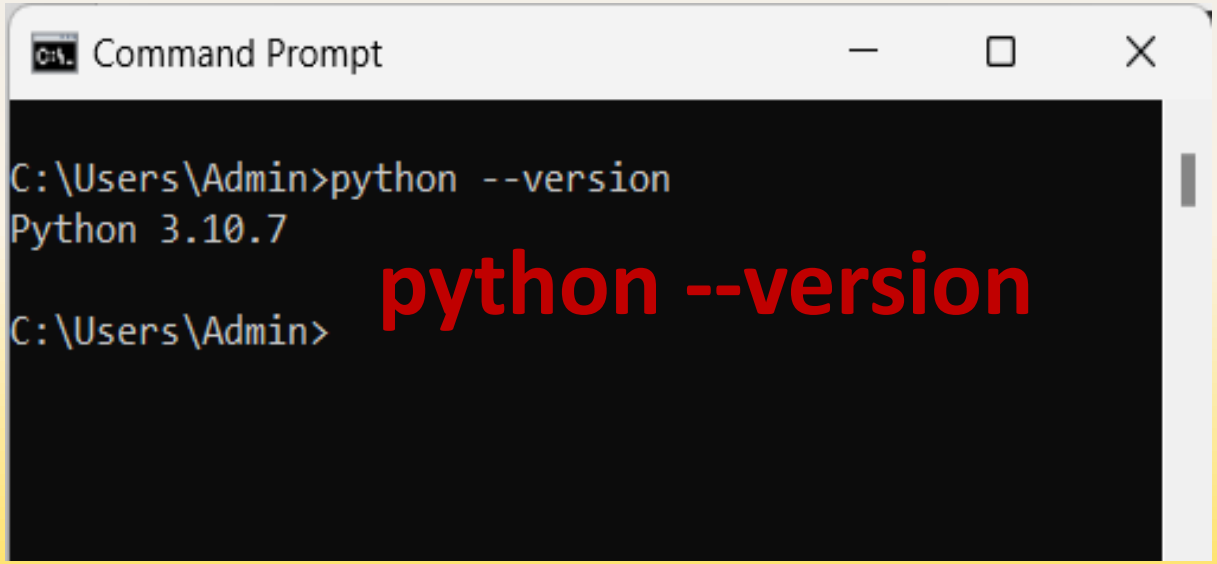
Python version	Maintenance status	First released	End of support	Release schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

Python Installation

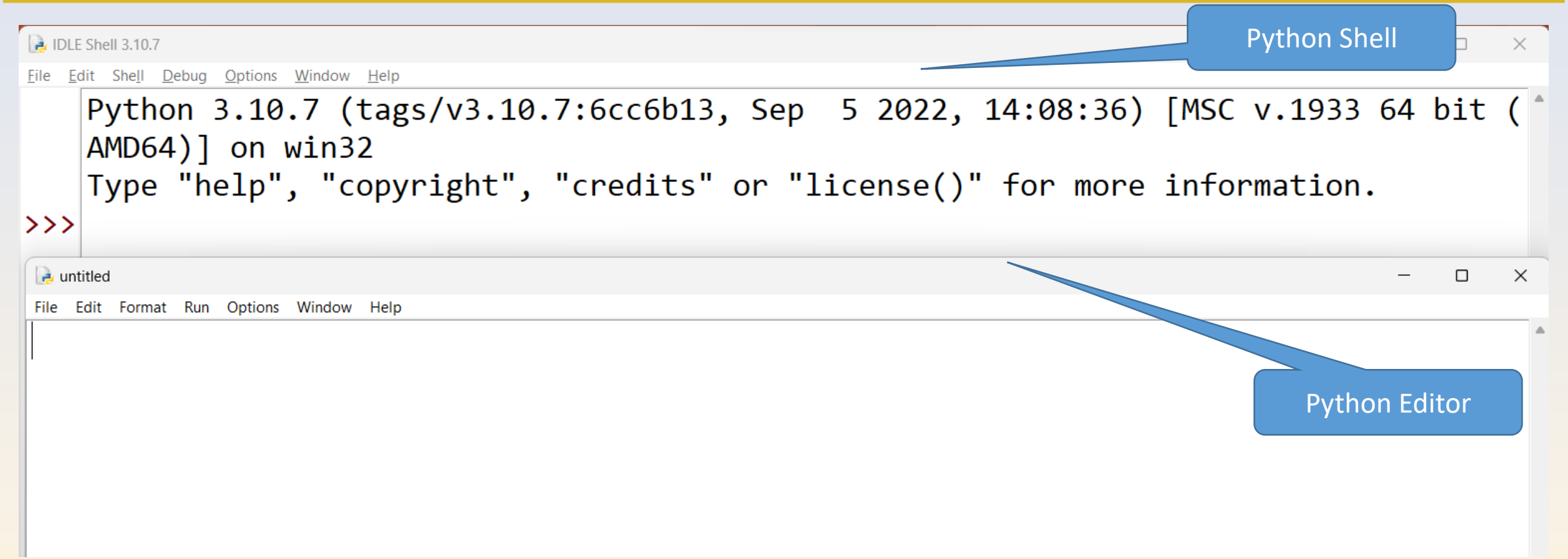




Tick this Checkbox to add the Environment Variable “PATH” in your PC to run Python Programs from Command Prompt



Python GUI: IDLE (Built-In)



Python Program Structure

Programming Styles

Interactive Mode

```
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022,
14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>> print("Hello Python!!!")
```

Script Mode

```
Create a new File "Hello.py"
    print("Hello Python!!!")
    Press F5 to Run from the Editor
```

Python Keywords

- Reserved Words which are well-known to the language and cannot be used as Identifiers.
- All keywords in Python are in Lower Case except 3 (True, False & None). (Python is Case-Sensitive)

True	False	None	class	from	or	continue	global
pass	def	if	raise	and	del	import	return
as	elif	in	try	assert	else	is	while
async	except	lambda	with	await	finally	nonlocal	yield
break	for	not					

```
>>> help('keywords')
```

Command used to list out all the keywords.

Python Identifiers

- Python identifier is a name used to identify a variable, function, class, module or other object.
- An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- Python does not allow punctuation characters such as @, $, and % within identifiers.
- Python is a case sensitive programming language. Thus, Local and local are two different identifiers in Python.

Indentation

- Indentation refers to the spaces at the beginning of a code line.
- Unlike C, C++ & Java; Python does not support block structure using parenthesis.
- So to define blocks Python simply uses Indentation, which must be followed. For each level, Indentation must be of same no of spaces.

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

```
if True:
   print "True"
else:
   print "False"
```

Python Comments

- Comments can be used to explain Python code, Those will be ignored by Python interpreter.
- Comments can be used to make the code more readable.
- 2 Types of Comments:
  - Single Line Comment:
    - A hash sign (#) that is not inside a string literal begins a comment.
    - All characters after the # and up to the end of the physical line are part of the comment.
  - Multi Line Comment:
    - All the lines encoded between Triple Quotes (""") are considered as comments until those quotes are closed.

```
#This is a single line comment.
print("Hello, World!")

Output: Hello, World!
```

```
"""
This is a example of
Multiline comment
"""
print("Hello, World!")

Output: Hello, World!
```

Python Data Types

Text Type	• str
Numeric Types	• int, float, complex
Sequence Types	• list, tuple, range
Mapping Type	• dict
Set Types	• set, frozenset
Boolean Type	• bool
Binary Types	• bytes, bytearray, memoryview
None Type	• NoneType

Example	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview
x = None	NoneType

```
x = "Hello World"

#display x:
print(x)

#display the data type of x:
print(type(x))
```

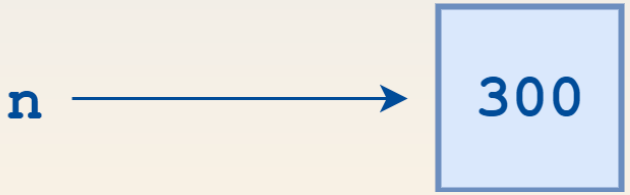
Output:

```
Hello World
<class 'str'>
```

Python Variables

- Variables are containers for storing data values.
- In Python, variables need not be declared or defined in advance, as is the case in many other programming languages.
- To create a variable, you just assign it a value to a new Identifier and then start using it.
- Assignment is done with a single equals sign (=) .

```
n = 300
```



- This is read or interpreted as "n is assigned the value 300." Once this is done, n can be used in a statement or expression, and its value will be substituted.

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- 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)
  - A variable name cannot be any of the Python keywords.

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

Output:
yagnik
```

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

Output:
Orange
Banana
Cherry
```

```
global x
x = "fantastic"
print("Python is " + x)

Output:
Python is fantastic
```

Python Type Casting

- The Process of converting the value of one data type to another is called Type Conversion.
- Python has two types of conversion
  - Implicit
  - Explicit
- Implicit Type Conversion:
  - In this Python automatically converts one data type to another

```
# int to float
x = 10
print(“ x is of type: “ type(x))
y = 10.6
print(“ y is of type: “ type(y))
x=x+y
Print(x)
print(“ x is of type: “ type(x))
```

```
Output:
x is of type : <class ‘int’>
y is of type : <class ‘float’>
      20.6
x is of type : <class ‘float’>
```

- Explicit Type Conversion:
  - In this user converts the data type of an object to required data type.
  - We use functions like int() ,float() etc. to perform explicit type conversion, it is also called typecasting because user changes or cast the data type of the object.

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

Python User Input

- Python allows users to give input at some point while executing the program.
- input(‘prompt message') function is used for that.

```
#Taking input from the user
name = input("Enter your name: ")
print("Hello:" +name)

Output:
Enter username: yagnik
Hello: yagnik
```

- *Note: You might have noticed that unlike C/C++ python does not use semi-colon (;) to specify line termination.*

Python Operators

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

Arithmetic Operators	Logical Operators (Relational)	Comparison Operators	Assignment Operators
Bitwise Operators	Conditional Operators (Ternary)	Membership Operators	is Operator Is not Operator

Python Operators: Arithmetic

```
++, -- Not Available
```

Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b
-	Subtraction - Subtracts right hand operand from left hand operand	a - b
*	Multiplication - Multiplies values on either side of the operator	a * b
/	Division - Divides left hand operand by right hand operand	b / a
%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a
**	Exponent - Performs exponential (power) calculation on operators	a**b
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	a//b

```
x = 5
y = 3
print(x + y)
print(x - y)
print(x * y)
print(x / y)
print(x % y)
print(x ** y)
print(x // y)
```

Output:

```
8
2
15
1.6666666666666667
2
125
1
```

Python Operators: Logical / Relational

```
In Python, 1 = True, 0 = False
```

Operator	Description	Example
and	Logical AND operator. If both the operands are true then then condition becomes true.	a and b
or	Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	a or b
not	Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	not(a and b)

```python
x = 5
y = 9
print(x > 3 or y < 8)
print(x < 2 and y < 8)
print(not(x < 2 and y < 8))
```

Output:

```
True
False
True
```

Python Operators: Comparison

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	a == b
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	a != b
<>	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	a <> b
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	a > b
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	a < b
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	a >= b
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	a <= b

```python
x = 5
y = 3

print(x == y)
print(x != y)
print(x > y)
print(x < y)
print(x >= y)
print(x <= y)
```

Output:

```
False
True
True
False
True
False
```

Python Operators: Assignment

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	c = a + b will assigne value of a + b into c
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**=	Exponent AND assignment operator, Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//=	Floor Division and assigns a value, Performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

```python
x = 5
print(x)

x = 5
x += 3
print(x)

x = 5
x -= 3
print(x)

x = 5
x *= 3
print(x)
```

```python
x = 5
x /= 3
print(x)

x = 5
x %= 3
print(x)

x = 5
x **= 3
print(x)

x = 5
x //= 3
print(x)
```

```
Output:
5
8
2
15
1.66666666666667
2
125
1
```

Python Operators: Bitwise

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	a & b
|	Binary OR Operator copies a bit if it exists in either operand.	a | b
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	a ^ b
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits and signed	~a
<<	Binary Left Shift Operator The left operands value is moved left by the number of bits specified by the right operand.	a << 2
>>	Binary Right Shift Operator The left operands value is moved right by the number of bits specified by the right operand.	a >> 2

```
print(6 & 3)     # 0110 for 6 and 0011 for 3
print(6 | 3)
print(6 ^ 3)
print(~ 6)
print(6 << 2)
print(6 >> 2)
```

Output:

```
2
7
5
-7
24
1
```

Python Operators: Ternary / Conditional

- Python has a ternary operator which is used as below:

```
[on_true] if [expression] else [on_false]
```

- Example:

```
a = 5
b = 2

print(a if a > b else b)

Output:
5
```

Python Operators: Membership

- In addition to earlier discussed operators like other languages, python has Membership operators available to check for the membership of some given value into a sequence like String, List, Tuples Sets and Dictionary.

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y

```
#List x conatains two string
x = ["apple", "banana"]

print("banana" in x)
print("pineapple" not in x)

Output:
True
True
```

Python Operators: is

- a == b, gives you comparison of values but, In python each variable is treated as an object.
- So, If we want to compare the references (address / id) of those variables then we have to use is operator.
- Returns True if both variables are the same object

```
x = 5
y = 4

print(x == y)
print(x is y)
print(id(x))
print(id(y))

Output:
False
False
140716871771048
140716871771016
```

Python Operators: is not

- Returns True if both variables are not the same object

```
x = 5
y = 4

print(x == y)
print(x is not y)

Output:
False
True
```

Python Operator: Precedence (High to Low)

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Ccomplement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators