Python

Created by

The Easy learn Academy

9662512857

Introduction of python

- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- It was created by Guido van Rossum during 1980-1990.
- Python is named after the comedy television show Monty Python’s Flying Circus. It is not named after the Python snake.
- Python source code is also available under the GNU General Public License (GPL).
- Latest version of python is python 3.10 at the time of updating presentation last time

Overview

- Python is interpreted & interactive.
- Python code is processed at runtime by the specially designed interpreter.
- Python is also object-oriented programming language.
- Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is designed to be highly readable means python code is easy to understand for developers.
- It uses English keywords frequently.
- Python is a great language for the beginner-level programmers.
- Python is easy to learn and use.

- **History**
- Python was developed by Guido van Rossum in 1980-1990 at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

- 1. **Readable:** Python is a very readable language.
- 2. **Easy to Learn:** Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn.
- 3. **Cross platform:** Python is available and can run on various operating systems such as Mac, Windows, Linux, Unix etc. This makes it a cross platform and portable language.
- 4. **Open Source:** Python is a open source programming language.
- 5. **Large standard library:** Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.
- 6. **Free:** Python is free to download and use. This means you can download it for free and use it in your application.
- 7. **Supports exception handling:** An exception is an event that can occur during program exception and can disrupt the normal flow of program.
- Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.
- 8. **Advanced features:** Supports generators and list comprehensions.
- 9. **Automatic memory management:** Python supports automatic memory management which means the memory is cleared and freed automatically. You do not have to clear the memory.

Python Features …

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Application (where it can be used)

- 1. Web development – Web framework like Django and Flask are based on Python. They help you write server side code which helps you manage database, write backend programming logic, mapping urls etc.
- 2. Machine learning – There are many machine learning applications written in Python. Machine learning is a way to write a logic so that a machine can learn and solve a particular problem on its own. For example, products recommendation in websites like Amazon, Flipkart, eBay etc. is a machine learning algorithm that recognises user's interest.
- 3. Data Analysis – Data analysis and data visualization in form of charts can also be developed using Python.
- 4. Scripting – Scripting is writing small programs to automate simple tasks such as sending automated response emails etc. Such type of applications can also be written in Python programming language.
- 5. Game development – You can develop games using Python.
- 6. You can develop embedded applications in Python.
- 7. Desktop applications – You can develop desktop application in Python using library like TKinter or QT.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used together with software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics (data mining).
- Python can be used for rapid prototyping, or for production-ready software development.

Where to get python setup

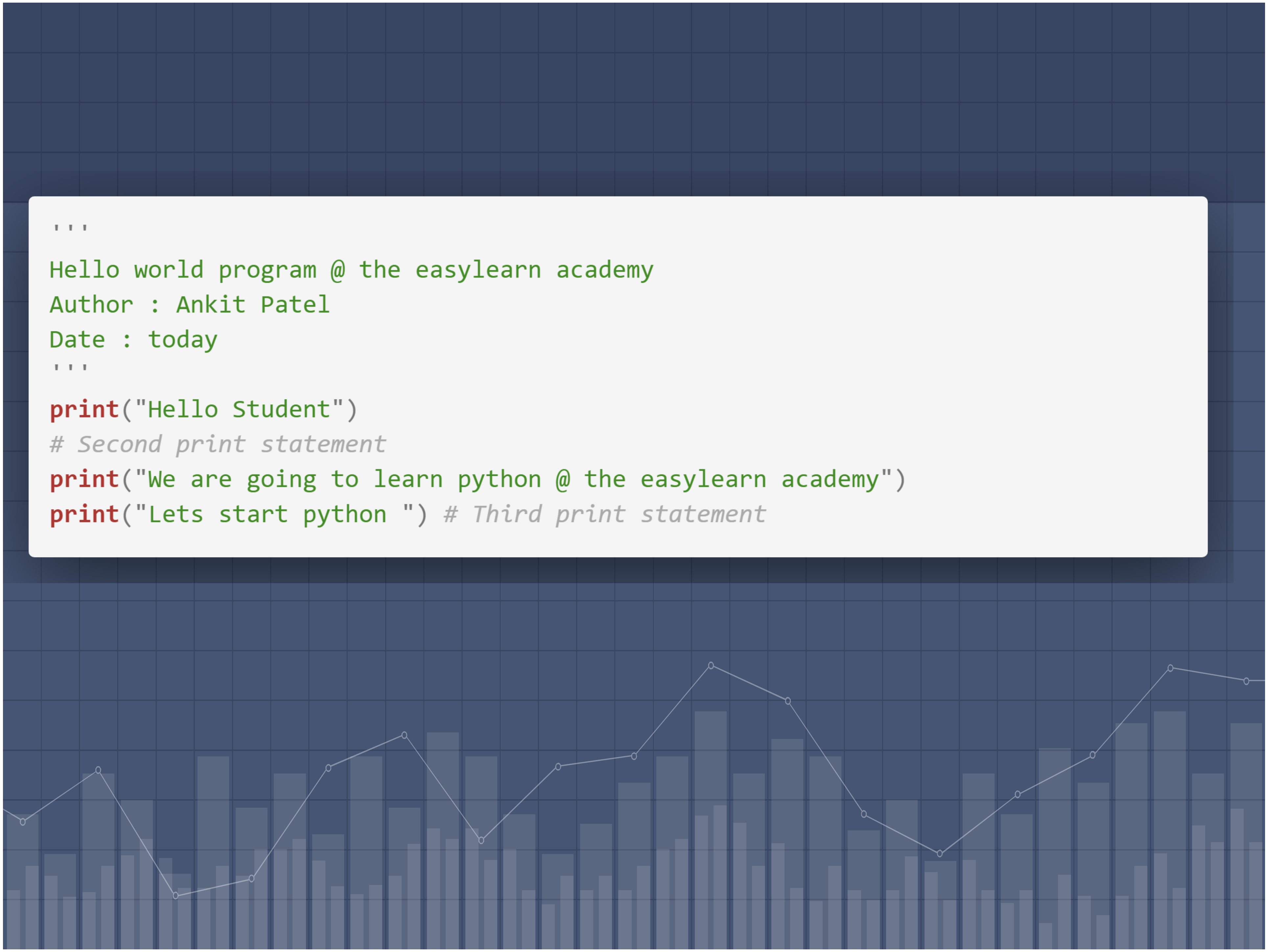
- Python is available on a wide variety of platforms including Linux and Mac OS X.
- Open a terminal window and type "python" to find out if it is already installed and which version is installed.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python **https://www.python.org/**
- You can download Python documentation from https://www.python.org/doc/.
- Python’s documentation is available in HTML, PDF, and PostScript formats.

What is comment?

- A comment is text that doesn't affect the outcome of a code, it is just a **piece of text** to let someone know what you have done in a program or what is being done in a block of code.
- This is helpful when someone else has written a code and you are analyzing it for bug fixing or making a change in logic.
- **Types of Comments in Python**
- There are two types of comments in Python.
  - 1. Single line comment
    - # This is just a comment. Anything written here is ignored by Python
  - 2. Multiple line comment

```
'''
 This
 is a
 multi-line
 comment
 '''
```

Lets create first program in python



```python
'''
Hello world program @ the easylearn academy
Author : Ankit Patel
Date : today
'''
print("Hello Student")
# Second print statement
print("We are going to learn python @ the easylearn academy")
print("Lets start python ") # Third print statement
```

Quotation in Python

- Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.
- The triple quotes are used to give multi line string.
- For detail see the example given below

```
print(‘Python' )
print(“Python is general purpose script“)
print( """This is a paragraph.
It is made up of multiple lines and sentences. """)
```

How to print message on same line using print?

- To print next message on same line, use second argument in print method. It should be end=‘’
- name = "the easylearn academy"
- print(name,end='')
- print("you are learning python at “,name)

Multiple Statements on a Single Line

- The semicolon (;) allows multiple statements on the single line but it is not possible.
  - Decision making statement
  - Loops
  - Functions
  - Class
- example
- `print ("Hi"); print ("I am learning Python")`

Creating Variables

- Variables are used to store data temporary, they take some memory space(in terms of bytes) based on the type of value we assigning to them.
- Creating variables in Python is simple, you just have write the variable name on the left side of = and the value on the right side, as shown below.

```
Name = “The easylearn academy” #string

Year = 2021 #integer

Weight = 80.25 #float

Gender = True #boolean
```

- You do not have to mention the type of the variable,
- python guess the type based on the value stored in it.

Python Keywords

- Before we learn more about variable we need to learn what is keywords.
- A python keyword is a reserved word which you can't use as a name of your variable, class, function etc.
- These keywords have a special meaning and they are used for special purposes in Python programming language.
- Basically python programming keywords are used to define the syntax and structure of the Python programming language.
- In Python, All keywords are case sensitive. Therefore, you should be careful when using them in your code.
- For example – Python keyword "class" is used for creating class so you can't name a variable with the name "class" else it may cause compilation error.
- There are total 43 keywords in Python as of now.
- To get the keywords list on your operating system, open command prompt (terminal on Mac OS) and type "Python" and hit enter. After that type help() and press enter.
- Type keywords to get the list of the keywords for the current python version running on your operating system.

More about variables.

- You can change variable value anywhere in your program.
- All variables will be deleted when program finish or stop in between or if computer gets restart while your program is running.
- Variables are used to store input, intermediate and final result.
- Variables are also used in expression.
-

List of keywords

False	class	from	or
None	continue	global	pass
True	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	

Naming rules

- **Variable Names**
- 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)

Data Types

- A data type defines the type of data, for example 123 is an integer data while “hello” is a String type of data. The data types in Python are divided in two categories:
  1. Immutable data types – part of values cannot be changed but can be replaced as whole.
  2. Mutable data types – part of value can be changed

- Immutable data types in Python are:

  1. **Numbers**

  2. **String**

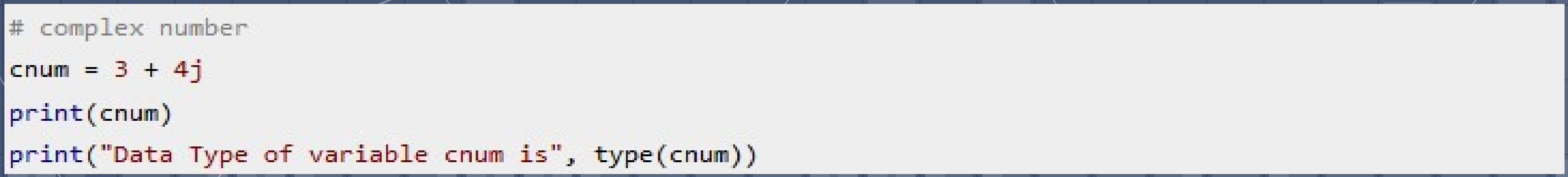
  3. **Tuple**

- Mutable data types in Python are:

- **1. List**
- **2. Dictionaries**
- **3. Sets**

Numeric Data Type in Python

- Integer – In Python 3, there is no upper bound on the integer number which means we can have the value as large as our system memory allows.
- Long - Long data type is deprecated in Python 3 because there is no need for it, since the integer has no upper limit, there is no point in having a data type that allows larger upper limit than integers.
- Float – Values with decimal points are the float values, there is no need to specify the data type in Python.
- It is automatically inferred based on the value we are assigning to a variable.
- Complex Number – Numbers with real and imaginary parts are known as complex numbers.
- Unlike other programming language such as Java, Python is able to identify these complex numbers with the values.



```python
# complex number
cnum = 3 + 4j
print(cnum)
print("Data Type of variable cnum is", type(cnum))
```

Python Strings

- String is a sequence of characters in Python. The data type of String in Python is called “str”.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.
- Python strings are immutable, and so they do not support item or slice assignment.

example

```
str = 'Hello World!'
print (str) # Prints complete string
print (str[0]) # Prints first
    character of the string
print (str[2:5]) # Prints characters
    starting from 3rd to 5th
print (str[2:]) # Prints string
    starting from 3rd character
print (str * 2) # Prints string two
    times
print (str + "TEST“) # Prints
    concatenated string
name = "the easylearn academy";
name = ‘T’ + name[1:]
```

```
output

Hello World!
H

llo

llo World!

Hello World!Hello World!

Hello World!TEST

The easylern academy
```

Python Lists

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).
- lists are similar to arrays in C.
- List is mutable datatype in python and it means we change change any value in list at any time.
- One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

example

- list = ['Ankit', 123 , 2.23, 'Patel', True]
- tinylist = [123, "The EasyLearn Academy"]

- print (list)   # Prints complete list
- print (list[0])   # Prints first element of the list
- print (list[1:3])   # Prints elements starting from 2nd till 3rd
- print (list[2:])   # Prints elements starting from 3rd element
- print (tinylist * 2) # Prints list two times
- print (list + tinylist) # Prints concatenated lists

Python Tuples

- A tuple is another sequence data type that is similar to the list.
- Tuples can be thought of as **read-only** lists.
- The main differences between lists and tuples is:
- Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- One can not print tuple with any other type variable in single print statement.

example

- tuple = ('Ankit', 456 , 1.14, 'Patel', 70.2)
- tinytuple = (99, 'The Easylearn Academy')

- print (tuple)   # Prints complete tuple
- print (tuple[0])   # Prints first element of the tuple
- print (tuple[1:3])   # Prints elements starting from 2nd till 3rd
- print (tuple[2:])   # Prints elements starting from 3rd element
- print (tinytuple * 2)   # Prints tuple two times
- print (tuple + tinytuple) # Prints concatenated tuple

Python Dictionary

- Python dictionary is an unordered collection of items. Each item of a dictionary has a key/value pair.
- In Python dictionaries are written with curly brackets, and they have keys and values.
- One can think of key as name of the variable and value as the value of variable.
- One can add new key value pair in dictionary at any time.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- Dictionaries can retrieve values very fast when the key is known.
- EXAMPLE

```
teacher = {"name":"Ankit","age":35,"weight":81.25,"gender":True}
```

- `print (teacher) # print teacher dictionary`
- `teacher['name'] = "ANKIT" #change particular key's value in dictionary`
- `print (teacher["name"]) # print specific dictionary`
- `del teacher["name"] #delete specific key value pair.`

example

```
book = {} # Creating an empty Dictionary
print("Empty Dictionary: ")
print(book)
# Adding elements one at a time
book['name'] = 'secret'
book['price'] = 1000
book['weight'] = 1.5
print("Dictionary after adding 3 elements: ")
print(book)
# # Adding set of values to a single Key
book['chapters'] = 1,2,3,4
Book[‘topics’] = [‘energy’,’focus’,’logical thinking’,’summery’]
print("Dictionary after adding 3 elements: ")
print(book)
# Updating existing Key's Value
book['name'] = 'the Secret'
print("Dictionary after Updating : ")
print(book)
```

List Methods / functions

append()

    Add an element to the end of the list

extend(list)

    Add set of values(list) at the end of list.

insert(position,item)

    Insert an item at the defined position

remove(item)

    Removes given item from the list

pop(position)

    Removes and returns an element at the given position

clear()

    Removes all items from the list

index()

    Returns the index of the first matched item

count()

    Returns the count of the number of items passed as an argument

sort()

    Sort items in a list in ascending order if all items are of same type

reverse()

    Reverse the order of items in the list

copy()

    Returns a shallow copy of the list

Important Dictionary Methods

- clear()
- Removes all items from the dictionary.
- copy()
- Returns a shallow copy of the dictionary.
- fromkeys(seq[, v])
- Returns a new dictionary with keys from seq and value equal to v (defaults to None).
- get(key[,d])
- Returns the value of the key. If the key does not exist, returns d (defaults to None).
- items()
- Return a new object of the dictionary's items in (key, value) format.
- keys()
- Returns a new object of the dictionary's keys.
- pop(key[,d])
- Removes the item with the key and returns its value or d if key is not found. If d is not provided and the key is not found, it raises KeyError.
- popitem()
- Removes and returns an item (key, value). Raises KeyError if the dictionary is empty.
- update([other])
  - update() method adds element(s) to the dictionary from dictionary passed as argument if the key is not in the dictionary then key value will be added . If the key is in the dictionary, it updates the key with the new value.
- Values()
  - The values() method returns a view object that displays a list of all the values in the dictionary.

Tuple Methods

- Since tuple is read only you cant add items or remove in tuple.
- Only the following two methods are available.
- count(item)
- count specified item in tuple
- index(item))
- return index of specified item

Lines and Indentation

- Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, this rule must be strictly followed else one will get run time error.
- The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount.

Multi-Line Statements

- Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue.
- For example −
- total = item_one + \
- item_two + \
- item_three

How to take input from user using python?

- To take input from user through keyboard input function is used in python 3.
- Input function can accept any type of value from user
- You can store the results from them into a variable.
- It has following syntax
- Variable = input(“input message”);

Operators(symbols) in Python

Types of Operator

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Python Arithmetic Operators

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) −	9//2 = 4 and 9.0//2.0 = 4.0, -1

Comparison Operators

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true.
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Assignment Operators

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	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	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	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	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Logical Operators

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a == c and b==d) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a==c or b==d) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a =x and b=y) is false.

Membership Operators

Python’s membership operators test for membership in a sequence, such as strings, lists, or tuples.

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

example

```
a = 10
b = 20
list = [1, 2, 3, 4, 5 ];

if ( a in list ):
   print "Line 1 - a is available in the given list"
else:
   print "Line 1 - a is not available in the given list"

if ( b not in list ):
   print "Line 2 - b is not available in the given list"
else:
   print "Line 2 - b is available in the given list"

a = 2
if ( a in list ):
   print "Line 3 - a is available in the given list"
else:
   print "Line 3 - a is not available in the given list"
```

Identity Operators

Identity operators compare the memory locations of two objects.

is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1 if id(x) is not equal to id(y).

example

```
x = 10
y = 10
result = x is y
print("result:", result)
```

- We will get True because both x and y are identical.
- We can also check the id of the variables using the id() function.
- The id() function returns a unique id for a given object.
- Every object in Python gets a unique id when they are created.
- The id of an object is an integer value that represents the address of an object in memory.

Example of id function

```
# variables
 x = 10
y = 10
result = x is y
```

- print("result:", result, id(x), id(y))
- The above code will give us a similar output as shown below.
- result: True 4488129824 4488129824

Operators Precedence

Sr.No.	Operator & Description
1	** Exponentiation (raise to the power)
2	~ + - Complement, unary plus and minus (method names for the last two are +@ and -@)
3	* / % // Multiply, divide, modulo and floor division
4	+ - Addition and subtraction
5	>> << Right and left bitwise shift
6	& Bitwise 'AND'
7	^ | Bitwise exclusive `OR' and regular `OR'
8	<= < > >= Comparison operators
9	<> == != Equality operators
10	= %= /= //= -= += *= **= Assignment operators
11	is is not Identity operators
12	in not in Membership operators
13	not or and Logical operators