**Object:-** It is the basic unit of object-oriented programming. An object is a block of memory that contains space to store all the instance variables. It is an instance of a class and objects are created using new operator.

* **Object:-** Objects have states and behaviors. Example: A dog has states (color, name, breed) as well as behavior (wagging their tail, barking, eating). An object is an instance of a class.

**Class:-** When we define a class, we define a blueprint for an object. A class is a mechanism to organize data and methods together in a same container. It is the way to bind the data and its associated behavior together.

* **Class:-** A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.

**Compiler:-**  The software, by which all the instruction can be converted to machine level language at once and all the errors are listed together is known as compiler.

**Interpreter:-** The software, by which the conversion of high level language is performed line by line to machine level language is known as interpreter. If an error is found on any line then the execution stops till it is corrected.

**Methods:-** A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

**Instance Variables:-** Each object has its unique set of instance variables. An object’s state is created by the values assigned to these instance variables.

**TOKENS**

Each individual character used in a statement is known as Token.  
Types of Tokens:-

* + - Literals
    - Keywords
    - Identifiers
    - Punctuators
    - Separators
    - Operators

**Literals:-** These are the constants used in a program. While writing a program you can use some contents, which remains fixed & these contents are called as literals or constants.

Different types of Literals:-

* + - Integer Literals
    - Real Literals
    - Character Literals
    - String Literals

**Integer Literals:-** The numbers which are representing without decimal point are called integer literals. These are the whole numbers having positive or negative values. {Example: 56, -98, 0, etc.}

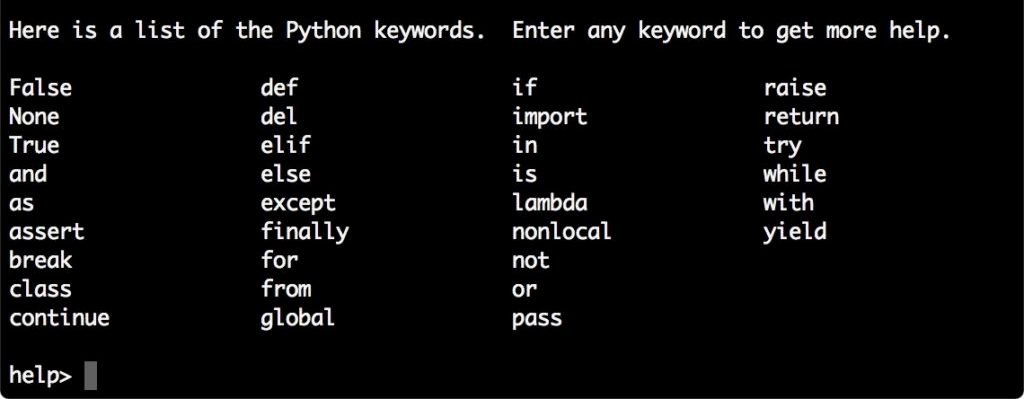
**Real Literals:-** These are also called floating-point constants. They represent numbers with decimal points.   
 {Example: 24.6, 0.0007, 1.0e-03, etc.}

**Character Literals:-** The constants, which are alphanumeric in nature are called as character literals. All alphabets upper or lower case, digits, special characters can be termed as character literals. {Example: ‘A’, ‘e’, ‘3’, ‘\*’, etc.}

**String Literals:-** String is a set of alphanumeric characters. A group of characters enclosed within a pair of opening and closing double quotes (“ “) is known as string literals. {Example: “JAVA”, “java”, “Java2020”, etc}.

**PYTHON 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. For example – Python keyword “while” is used for while loop thus you can’t name a variable with the name “while” else it may cause compilation error. There are total 33 keywords in Python 3.6.



**PYTHON IDENTIFIERS**

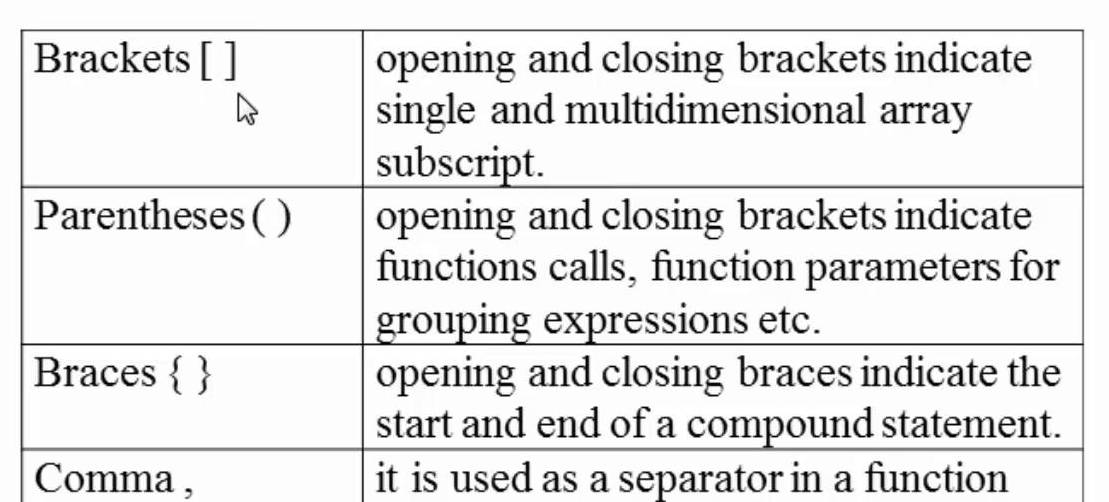
Variable name is known as identifier. There are few rules that you have to follow while naming the variables in Python.

For example here the variable is of integer type that holds the value 10. The name of the variable, which is “num” is called identifier.

num = 10

1. The name of the variable must always start with either a letter or an underscore (\_). For example: \_str, str, num, \_num are all valid name for the variables.  
2. The name of the variable cannot start with a number. For example: 9num is not a valid variable name.  
3. The name of the variable cannot have special characters such as %, $, # etc, they can only have alphanumeric characters and underscore (A to Z, a to z, 0-9 or \_ ).  
4. Variable name is case sensitive in Python which means num and NUM are two different variables in python.

**PUNCTUATORS & SEPARATORS:-**

****

**VARIABLES**

Following are the types of variables in:-

* Local Variable
* Instance Variable
* Static Variable

**Local Variable:-** A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren’t aware of that the variable exists. A local variable can’t be defined with “static” keyword.

**Instance Variable:-** A variable declared inside the class but outside the body of the method is called instance variable. It is not declared as static.

**Static Variable:-** A variable which is declared as static is called static variable. It can’t be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

**DATA TYPES**

Data types specify the different sizes and values that can be stored in the variable. There are two types of data:

* Primitive data types
* Non-primitive data types



**PRIMITIVE DATA TYPES**

|  |  |  |
| --- | --- | --- |
| Data Type | Default Value | Default Size |
| boolean | false | 1 bit |
| char | 0 | 2 byte |
| byte | 0 | 1 byte |
| short | 0 | 2 byte |
| int | 0 | 4 byte |
| long | 0l | 8 byte |
| float | 0.0f | 4 byte |
| double | 0.0d | 8 byte |

**Boolean Data Types:-** The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions. {Example: Boolean one = false}

**Byte Data Type:-** The byte data type is an example of primitive data type. It is an 8-bit signed two’s complement integer. Its value-range lies between -128 to 127 (inclusive). Its minimum value is -128 and maximum value is 127. Its default value is 0. {Example: byte a = 10, byte b = -20}

**Short Data Type:-** The short data type is a 16-bit signed two’s complement integer. Its value-range lies between 32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.   
{Example: short s=10000, short r=-5000)

**Int Data Type:-** The int data type is a 32-bit signed two’s complement integer. Its value-range lies between 2,147,483,648(-2^31) to 2,147,483,647(2^31-1) (inclusive). Its minimum value is -2,147,483,648 and maximum value is 2,147,483,647. Its default value is 0. {Example: int a=100000, int b=-200000}

**Long Data Type:-** The long data type is a 64-bit two’s complement integer. Its value-range lies between -9,223,372,036,854,775,808(-2^63) to 9,223,372,036,854,775,807(2^63-1). Its minimum value is -9,223,372,036,854,775,808 and maximum value is 9,223,372,036,854,775,807. Its default value is 0. The long data type is used when you need a range of values more than those provided by int. {Example: long a=100000L, long b=-200000L}

**Float Data Type:-** The float data type is a single-precision 32-bit floating point. Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating point numbers. The float data type should never be used for precise values, such as currency. Its default value is 0.0F. {Example: float f1=234.5f}

**Double Data Type:-** The double data type is a double-precision 64-bit floating point. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency. Its default value is 0.0d. {Example: double d1=12.3}

**Char Data Type:-** The char data type is a single 16-bit Unicode character. Its value-range lies between ‘\u0000’ (and 0) to ‘\uffff’ (or 65,535 inclusive). The char data type is used to store characters. {Example: char letterA=’A’}

**NON-PRIMITIVE DATA TYPES**

**String Data Type:-** A string represents a group of characters. String written directly in a program should be enclosed by using double quotes (“ “). {Example: **str=”computer”**}

**Array Data Type:-** An array is used to store a collection of similar data type. {Example: **cars = ["Ford", "Volvo", "BMW"]**}

**Data Types in Python**

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 – Values cannot be changed.  
2. Mutable data types – Values can be changed

Immutable data types in Python are:  
1. [Numbers](https://beginnersbook.com/2018/02/python-numbers/)  
2. [String](https://beginnersbook.com/2018/02/python-strings/)  
3. [Tuple](https://beginnersbook.com/2018/02/python-tuple/)

Mutable data types in Python are:  
1. [List](https://beginnersbook.com/2018/02/python-list/)  
2. [Dictionaries](https://beginnersbook.com/2019/03/python-dictionary/)  
3. Sets

**NUMBERS:-** Python supports integers, floats and complex numbers. An **integer** is a number without decimal point for example 5, 6, 10 etc. A **float** is a number with decimal point for example 6.7, 6.0, 10.99 etc. A **complex number** has a real and imaginary part for example 7+8j, 8+11j etc.

# Python program to display numbers of

# different data types

# int

num1 = 10

num2 = 100

print(num1+num2)

# float

a = 10.5

b = 8.9

print(a-b)

# complex numbers

x = 3 + 4j

y = 9 + 8j

print(y-x)

**Output:**

110

1.5999999999999996

(6+4j)

**STRING:-** A string is usually a bit of text (sequence of characters). In Python we use ” (double quotes) or ‘ (single quotes) to represent a string. In this guide we will see how to create, access, use and manipulate strings in Python programming language.

## 1. How to create a String in Python

There are several ways to create strings in Python.  
1. We can use ‘ (single quotes), see the string str in the following code.  
2. We can use ” (double quotes), see the string str2 in the source code below.  
3. Triple double quotes “”” and triple single quotes ”’ are used for creating multi-line strings in Python. See the strings str3 and str4 in the following example.

# lets see the ways to create strings in Python

str = 'beginnersbook'

print(str)

str2 = "Chaitanya"

print(str2)

# multi-line string

str3 = """Welcome to

Beginnersbook.com"""

print(str3)

str4 = '''This is a tech

blog'''

print(str4)

**Output:**

beginnersbook

Chaitanya

Welcome to

Beginnersbook.com

This is a tech

blog

**TUPLE:-** In Python, a tuple is similar to [List](https://beginnersbook.com/2018/02/python-list/) except that the objects in tuple are immutable which means we cannot change the elements of a tuple once assigned. On the other hand, we can change the elements of a list.

## How to create a tuple in Python

To create a tuple in Python, place all the elements in a () parenthesis, separated by commas. A tuple can have heterogeneous data items, a tuple can have string and list as data items as well.

### Example – Creating tuple

In this example, we are creating few tuples. We can have tuple of same type of data items as well as mixed type of data items. This example also shows nested tuple (tuples as data items in another tuple).

# tuple of strings

my\_data = ("hi", "hello", "bye")

print(my\_data)

# tuple of int, float, string

my\_data2 = (1, 2.8, "Hello World")

print(my\_data2)

# tuple of string and list

my\_data3 = ("Book", [1, 2, 3])

print(my\_data3)

# tuples inside another tuple

# nested tuple

my\_data4 = ((2, 3, 4), (1, 2, "hi"))

print(my\_data4)

Output:

('hi', 'hello', 'bye')

(1, 2.8, 'Hello World')

('Book', [1, 2, 3])

((2, 3, 4), (1, 2, 'hi'))

### Empty tuple:

# empty tuple

my\_data = ()

### Tuple with only single element:

Note: When a tuple has only one element, we must put a comma after the element, otherwise Python will not treat it as a tuple.

# a tuple with single data item

my\_data = (99,)

If we do not put comma after 99 in the above example then python will treat my\_data as an int variable rather than a tuple.

## How to access tuple elements

We use indexes to access the elements of a tuple. Lets take few example to understand the working.

### Accessing tuple elements using positive indexes

We can also have negative indexes in tuple, we have discussed that in the next section. Indexes starts with 0 that is why we use 0 to access the first element of tuple, 1 to access second element and so on.

# tuple of strings

my\_data = ("hi", "hello", "bye")

# displaying all elements

print(my\_data)

# accessing first element

# prints "hi"

print(my\_data[0])

# accessing third element

# prints "bye"

print(my\_data[2])

Output:

('hi', 'hello', 'bye')

hi

bye

LIST:- A list is a data type that allows you to store various types data in it. List is a compound data type which means you can have different-2 data types under a list, for example we can have integer, float and string items in a same list.

## 1. Create a List in Python

Lets see how to create a list in Python. To create a list all you have to do is to place the items inside a square bracket [] separated by comma ,.

# list of floats

num\_list = [11.22, 9.9, 78.34, 12.0]

# list of int, float and strings

mix\_list = [1.13, 2, 5, "beginnersbook", 100, "hi"]

# an empty list

nodata\_list = []

As we have seen above, a list can have data items of same type or different types. This is the reason list comes under compound data type.

## Accessing the items of a list:-

**Syntax to access the list items:**

list\_name[index]

**Example:**

# a list of numbers

numbers = [11, 22, 33, 100, 200, 300]

# prints 11

print(numbers[0])

# prints 300

print(numbers[5])

# prints 22

print(numbers[1])

Output:

11

300

22

**Points to Note:**  
1. The index cannot be a float number.

**For example:**

# a list of numbers

numbers = [11, 22, 33, 100, 200, 300]

# error

print(numbers[1.0])

Output:

TypeError: list indices must be integers or slices, not float

2. The index must be in range to avoid IndexError. The range of the index of a list having 10 elements is 0 to 9, if we go beyond 9 then we will get IndexError. However if we go below 0 then it would not cause issue in certain cases.

**For example:**

# a list of numbers

numbers = [11, 22, 33, 100, 200, 300]

# error

print(numbers[6])

Output:

IndexError: list index out of range

**DICTIONARY :-** Dictionary is a mutable data type in Python. A python dictionary is a collection of key and value pairs separated by a colon (:), enclosed in curly braces {}.

## Python Dictionary

Here we have a dictionary. Left side of the colon(:) is the key and right side of the : is the value.

mydict = {'StuName': 'Ajeet', 'StuAge': 30, 'StuCity': 'Agra'}

**Points to Note:**  
1. Keys must be unique in dictionary, duplicate values are allowed.  
2. A dictionary is said to be empty if it has no key value pairs. An empty dictionary is denoted like this: {}.  
3. The keys of dictionary must be of immutable data types such as String, numbers or tuples.

## Accessing dictionary values using keys in Python

To access a value we can can use the corresponding key in the square brackets as shown in the following example. Dictionary name followed by square brackets and in the brackets we specify the key for which we want the value.

mydict = {'StuName': 'Ajeet', 'StuAge': 30, 'StuCity': 'Agra'}

print("Student Name is:", mydict['StuName'])

print("Student Age is:", mydict['StuAge'])

print("Student City is:", mydict['StuCity'])

Output:

Student Name is: Ajeet

Student Age is: 30

Student City is: Agra

## Change values in Dictionary

Here we are updating the values for the existing key-value pairs. To update a value in dictionary we are using the corresponding key.

mydict = {'StuName': 'Ajeet', 'StuAge': 30, 'StuCity': 'Agra'}

print("Student Age before update is:", mydict['StuAge'])

print("Student City before update is:", mydict['StuCity'])

mydict['StuAge'] = 31

mydict['StuCity'] = 'Noida'

print("Student Age after update is:", mydict['StuAge'])

print("Student City after update is:", mydict['StuCity'])

**Output:**

Student Age before update is: 30

Student City before update is: Agra

Student Age after update is: 31

Student City after update is: Noida

SET :- Sets are used to store multiple items in a single variable. Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](https://www.w3schools.com/python/python_lists.asp), [Tuple](https://www.w3schools.com/python/python_tuples.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage. A set is a collection which is both unordered and unindexed. Sets are written with curly brackets.

myset = {"apple", "banana", "cherry"}  
print(myset)

**OUTPUT**

{‘apple’, ‘banana’, ‘cherry’}

{‘banana’, ‘apple’, ‘cherry’}

{‘cherry’, ‘banana’, ‘apple’}

**Set Items**

Set items are unordered, unchangeable, and do not allow duplicate values.

**Unordered**

Unordered means that the items in a set do not have a defined order.

Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

**Unchangeable**

Sets are unchangeable, meaning that we cannot change the items after the set has been created.

Once a set is created, you cannot change its items, but you can add new items.

**Duplicates Not Allowed**

Sets cannot have two items with the same value.

**OPERATORS**

Operator is a symbol that is used to perform operations. {Example: **+, -, “, /,** etc.} Python provides many types of operators which can be used according to the need. They are classified based on the functionality they provide. Some of the types are:-

* Arithmetic Operators **(+, -, \*, /, %)**
* Assignment Operator **(=)**
* Relational/Comparision Operators**(<, >, <=, >=, ==, !=)**
* Logical Operators **(**&&**, ||, !)**
* Identity Operators (is, is not)
* Membership Operators (in, not in)

