PYTHON TUTORIAL

**INTRODUCTION:**

* What is python?
* Features of python
* Applications of python

**What is python?**

Python is General purpose, High Level, Multi paradigm, Dynamic, Interpreted, indented, case-sensitive and most popular programming Language.

**HISTORY:**

* It was created by Guido van Rossum in 1991 at CWI in the Netherlands.

**FEATURES:**

* Easy & Simple to Learn
* Platform Independent
* Free & Open Source
* Interpreted
* Rich Library Support
* Embeddable & Extensible
* Portable
* Robust
* Dynamic

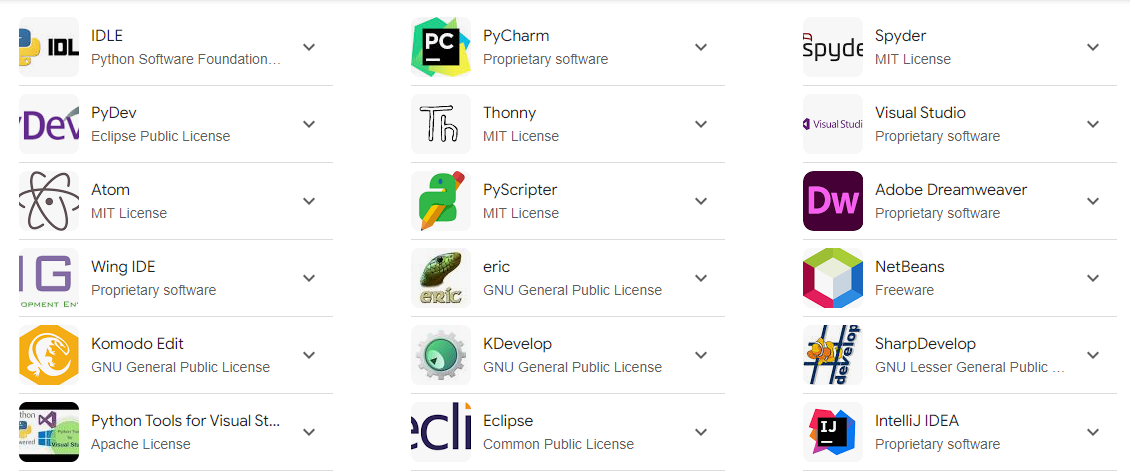
**APPLICATIONS:**

* Web Applications
* Desktop GUI Applications
* Console based Applications
* Software Development
* Scientific & numeric
* Business Applications
* Audio & video
* Games
* Data science
* Data Analysis
* Machine Learning
* Artificial intelligence
* Software Testing
* CAD Related

**What is an IDE?**

An integrated development environment (IDE) is a software application that helps programmers develop software code efficiently. It increases developer productivity by combining capabilities such as software editing, building, testing, and packaging in an easy-to-use application.

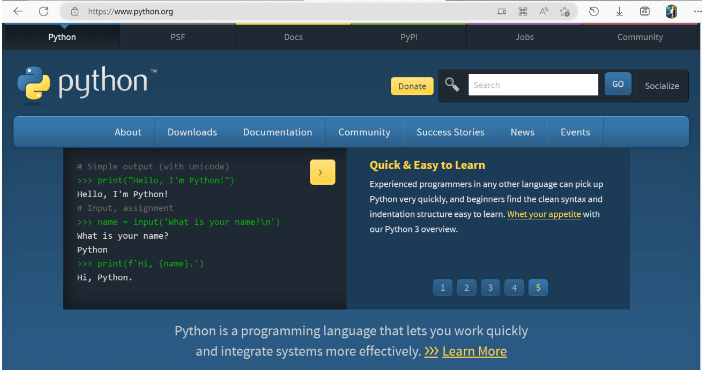
**List of IDE’s used for python?**

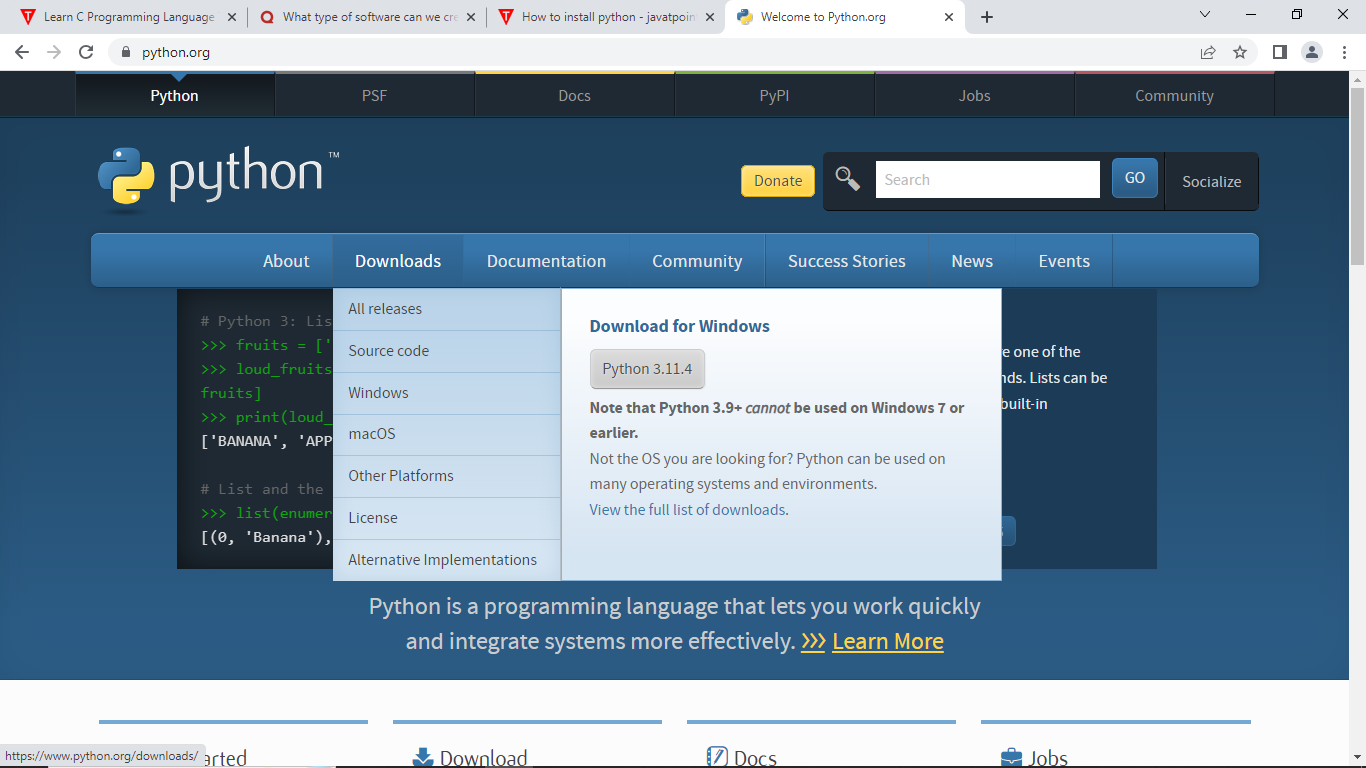


**Note: - So many IDE’s are available in market, few are mentioned in above**

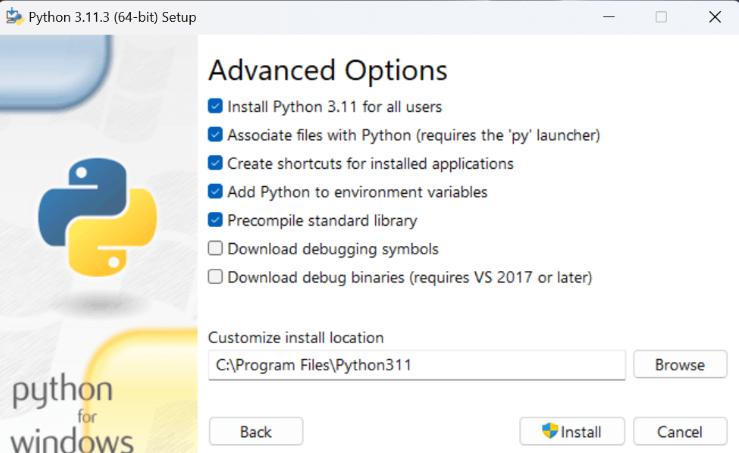
**INSTALLATION OF PYTHON:**

[***https://www.python.org***](https://www.python.org)



**Go to downloads**  
  






**INTERPRETER:**

An Interpreter is a software program written to translate source code to machine code but it does that line by line.

**ADVANTAGES:**

* Portable
* Easy debugging

**DISADVANTAGES:**

* Requires Interpreter
* Slower
* Public code

**PYTHON PROVIDES US THE TWO WAYS TO RUN A PROGRAM**

* Using Interactive interpreter prompt
* Using a script file

**MEMORY**

The Memory is organized in the Form of cells; each cell is able to be identified with a unique number called address.

**Cell**

**Row’s**

**Column’s**

**0x00**

**0x01**

**0x02**

**0xFF**

In memory Each cell is having address (hexadecimal format) As shown in aside

**MEMORY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**PYTHON VARIABLES**

Variable can be defined as a name given to the Memory Location.

**(or)**

A variable is defined as Alternate name for Memory Location.

**(or)**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

* It stores data values of the program and provides whenever required.
* Python doesn’t require the variables type to declare.
* Python internally handles the variables according to the values assigned to them.

**RULES FOR DEFINING A VARIABLE NAME**

* The first character of the variable must be an **alphabet** or **underscore** (\_).

**Examples**: Venkat, venkat, \_venkat, venkat\_, ven\_kat

* All the characters except first character may be an **alphabet** of **lower-case(a-z)**, **upper-case(A-Z),** **underscore**, or **digit (0-9)**.
* A variable Name cannot start with a Number.
* Special characters (! ,@,#,$,%,^,&,\*) and white spaces are not allowed except underscore.
* Variable name must not be similar to any **keyword** defined in language.
* Variable names are case sensitive.

**Example**: myname, and Myname both are not same

**What is data type in python?**

The purpose of data type in python is to allocate memory space for storing different values. As we already know that python is dynamically typed programming Language which means we don’t have specify the data type python can automatically detect the data type of values that we are using.

In python we have **14 Data Types** Which Are Categorized into

1. **Fundamental Data Type**
2. **Sequential Data Type**
3. **Collection Data Type**
4. **None Data Type**

* **FUNDAMENTAL**
* **int**
* **float**
* **complex**
* **bool**
* **SEQUENTIAL**
* **Str**
* **bytes**
* **bytearray**
* **range**
* **Collection**
* **list**
* **tuple**
* **set**
* **frozenset**
* **dict**
* **None**
* **None**

**bytes**: -It is built-in data type in python. The purpose of bytes data type is that to store sequence of positive numerical integer values ranges from 0 to 256 which means we can store positive integer value from **0 to 255** so we cannot store values as the byte type which are negative or the numbers which are greater than 255 to store the data in the objects of bytes data we don’t have any symbolic notation as we have for others but we can convert one type value into bytes

bytes data type is **immutable**

**bytearray: -** bytearray data type is **mutable**.

**type():-** This is the function which is used to find data type of variable.

**Syntax: -**

type(var\_name)

**Keywords: -** Keywords are predefined (or)Reserved words in python which are having special meaning for it. There are **35** keywords in python.   
 Following are the keywords in python :-

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

**Python input and output functions: -**

**input (): -** This is the function which is used for getting input from the user or any other source.

**Syntax: -**

Var\_name=input ()

**Print (): -** This is the function which is used for displaying output in display.

**Syntax: -**

Print(var\_name)

**type():-** This is the function which is used to find data type of variable.

**Syntax: -**

type(var\_name)

## **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:

Python divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison (or) Relational operators
* Logical operators
* Bitwise operators
* Identity operators
* Membership operators

## 

## Special 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 |

## **Python Assignment Operators**

Assignment operators are used to assign values to variables:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same As** |
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| \*= | x \*= 3 | x = x \* 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| //= | x //= 3 | x = x // 3 |
| \*\*= | x \*\*= 3 | x = x \*\* 3 |
| &= | x &= 3 | x = x & 3 |
| |= | x |= 3 | x = x | 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

## **Python Comparison (or) Relational Operators**

Comparison operators are used to compare two values:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| == | Equal | x == y |
| != | Not equal | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

## **Python Logical Operators**

Logical operators are used to combine conditional statements:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| and | Returns True if both statements are true | x < 5 and  x < 10 |
| or | Returns True if one of the statements is true | x < 5 or x < 4 |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) |

## **Python Identity Operators**

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| is | Returns True if both variables are the same object | x is y |
| is not | Returns True if both variables are not the same object | x is not y |

## **Python Membership Operators**

Membership operators are used to test if a sequence is presented in an object:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| in | Returns True if a sequence with the specified value is present in the object | x in y |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y |

## **Python Bitwise Operators**

Bitwise operators are used to compare (binary) numbers:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Example** |
| & | AND | Sets each bit to 1 if both bits are 1 | x & y |
| | | OR | Sets each bit to 1 if one of two bits is 1 | x | y |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 | x ^ y |
| ~ | NOT | Inverts all the bits | ~x |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off | x << 2 |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off | x >> 2 |

## **Operator Precedence**

Operator precedence describes the order in which operations are performed.

The precedence order is described in the table below, starting with the highest precedence at the top:

|  |  |
| --- | --- |
| **Operator** | **Description** |
| () | Parentheses |
| \*\* | Exponentiation |
| +x  -x  ~x | Unary plus, unary minus, and bitwise NOT |
| \*  /  //  % | Multiplication, division, floor division, and modulus |
| +  - | Addition and subtraction |
| <<  >> | Bitwise left and right shifts |
| & | Bitwise AND |
| ^ | Bitwise XOR |
| | | Bitwise OR |
| ==  !=  >  >=  <  <=  is  is not  in  not in | Comparisons, identity, and membership operators |
| not | Logical NOT |
| and | AND |
| or | OR |

If two operators have the same precedence, the expression is evaluated from left to right.

**Python Loop: -**

In computer programming, loops are used to repeat a block of code.

For example, if we want to show a message **100** times, then we can use a loop. It's just a simple example; you can achieve much more with loops.

There are 2 types of loops in Python:

* for loop
* while loop

**Python for Loop: -**

In Python, a for loop is used to iterate over sequences such as lists, tuples, string, etc.

For example,

languages = ['Swift', 'Python', 'Go', 'JavaScript']

# run a loop for each item of the list

for language in languages:

print(language)

OUTPUT:-

Swift

Python

Go

JavaScript

## for Loop Syntax

The syntax of a for loop is:

**for val in sequence:**

**# statement(s)**

# **Python while Loop**

In programming, loops are used to repeat a block of code. For example, if we want to show a message **100** times, then we can use a loop. It's just a simple example, we can achieve much more with loops.

In the previous tutorial, we learned about [**Python for loop**](https://www.programiz.com/python-programming/for-loop). Now we will learn about the while loop.

Python while loop is used to run a block code until a certain condition is met.

The syntax of while loop is:

**while condition:**

**# body of while loop**

**Example:-**

**# program to display numbers from 1 to 5**

**# initialize the variable**

**i = 1**

**n = 5**

**# while loop from i = 1 to 5**

**while i <= n:**

**print(i)**

**i = i + 1**

**List:**

* It is used to represent group of elements into a single entity or object.
* It uses square brackets.
* List like string, which provides sequential storage Mechanism.
* List can contain different types data.
* At any given time, single or multiple objects can be inserted, removed and updated to the list.
* List can be grow and Shink.
* List can be sorted, reversed, empty and populated.
* A List can be added /removed to/from other list.
* Every element in the List object is represented with Unique index.
* List supports both positive and negative index.
* List objects are Mutable objects.

**Python-List Methods**

Python has a set of built-in methods that you can use on lists.

1. **append()** Adds an element at the end of the list

**{Definition&Usage**: The append() method appends an element to the end of the list.

**Syntax** : *list*.append(*elemt)*}

1. **Clear()** Removes all the elements from the list

**{Definition&Usage**:The clear() method removes all the elements from a list.

**Syntax** : *list*.clear()}

1. **Copy()** Returns a copy of the list

**{Definition&Usage**:The copy() method returns a copy of the specified list.

**Syntax:** *list*.copy()}

1. **Count()** Returns the number of elements with the specified value

**{Definition&Usage**:The count() method returns the number of elements with the specified value.

**Syntax***:list*.count(*value*)}

1. **Extend()** Add the elements of a list (or any iterable), to the end of the current list

**{Definition&Usage**:The extend() method adds the specified list elements (or any iterable) to the end of the current list.

**Syntax***:list*.extend(*iterable*)

**Note:** Any iterable (list, set, tuple, etc.)}

1. **Index()** Returns the index of the first element with the specified value

**{Definition&Usage**:The index() method returns the position at the first occurrence of the specified value.

**Syntax***:list*.index(*elmnt*)}

1. **Insert()** Adds an element at the specified position

**{Definition&Usage**: The insert() method inserts the specified value at the specified position.

**Syntax***:list*.insert(*pos*, *elmnt*)}

1. **Pop()** Removes the element at the specified position.

**{Definition&Usage**: The pop() method removes the element at the specified position.

**Syntax***:* list.pop(pos)}

1. **Remove()** Removes the item with the specified value

**{Definition&Usage**: The remove() method removes the first occurrence of the element with the specified value.

**Syntax***:* list.remove(elmnt)}

1. **revers()** Reverses the order of the list

**{Definition&Usage**: The reverse() method reverses the sorting order of the elements.

**Syntax***:* list.reverse()}

1. **sort()** Sorts the list

**{Definition&Usage**: The sort() method sorts the list ascending by default.

You can also make a function to decide the sorting criteria(s).

**Syntax***:* list.sort(reverse=True|False, key=myFunc)}

**Tuple:**

* It is used to represent group of elements into single entity or object.
* A tuple is a container, used to store collection of different types of data items.

**Creating a Tuple**: - A Tuple is created by placing all the items inside a parenthesis () separated by a comma , .

Ex: - my\_tuple=(2,’python’,2.3)

**Creating a Tuple with one item**: - To create a tuple with only one item, you have to add a comma , after the item, otherwise python will not recognize it as a tuple.

Ex: - t=(’python’,)

**Advantages of tuple over list:** -

1. Generally, we use tuples for heterogenous data types and list for homogenous.
2. Tuples are immutable, iterating through tuples is faster than with List.
3. Due to properties of immutability tuple can use as a Dictionary-key and also while dealing with object groups.

**Accessing Elements in tuples:** -

1. Indexing
2. Negative indexing
3. Slicing

**Python-Tuple Methods**

Python has two built-in methods that you can use on tuples.

1. count() Returns the number of times a specified value occurs in a tuple

**{Definition&Usage**: The count() method returns the number of times a specified value appears in the tuple.

**Syntax***:* *tuple*.count(*value*)**}**

1. index() Searches the tuple for a specified value and returns the position of where it was found.

**{Definition&Usage**: The index() method finds the first occurrence of the specified value.

The index() method raises an exception if the value is not found.

**Syntax***:* *tuple*.index(*value*)**}**

**Control Statements/Decision making statements: -**

1. **Conditional statements**

* if
* else
* elif
* nested if

1. **Looping statements**

* For
* While
* Nested loops

1. **Jumping statements**

* Pass
* Break
* Continue

**List Methods: -**

|  |  |
| --- | --- |
| **Method** | **Description** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the first item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

**String Methods: -**

|  |  |
| --- | --- |
| **Method** | **Description** |
| [capitalize()](https://www.w3schools.com/python/ref_string_capitalize.asp) | Converts the first character to upper case |
| [casefold()](https://www.w3schools.com/python/ref_string_casefold.asp) | Converts string into lower case |
| [center()](https://www.w3schools.com/python/ref_string_center.asp) | Returns a centered string |
| [count()](https://www.w3schools.com/python/ref_string_count.asp) | Returns the number of times a specified value occurs in a string |
| [encode()](https://www.w3schools.com/python/ref_string_encode.asp) | Returns an encoded version of the string |
| [endswith()](https://www.w3schools.com/python/ref_string_endswith.asp) | Returns true if the string ends with the specified value |
| [expandtabs()](https://www.w3schools.com/python/ref_string_expandtabs.asp) | Sets the tab size of the string |
| [find()](https://www.w3schools.com/python/ref_string_find.asp) | Searches the string for a specified value and returns the position of where it was found |
| [format()](https://www.w3schools.com/python/ref_string_format.asp) | Formats specified values in a string |
| format\_map() | Formats specified values in a string |
| [index()](https://www.w3schools.com/python/ref_string_index.asp) | Searches the string for a specified value and returns the position of where it was found |
| [isalnum()](https://www.w3schools.com/python/ref_string_isalnum.asp) | Returns True if all characters in the string are alphanumeric |
| [isalpha()](https://www.w3schools.com/python/ref_string_isalpha.asp) | Returns True if all characters in the string are in the alphabet |
| [isascii()](https://www.w3schools.com/python/ref_string_isascii.asp) | Returns True if all characters in the string are ascii characters |
| [isdecimal()](https://www.w3schools.com/python/ref_string_isdecimal.asp) | Returns True if all characters in the string are decimals |
| [isdigit()](https://www.w3schools.com/python/ref_string_isdigit.asp) | Returns True if all characters in the string are digits |
| [isidentifier()](https://www.w3schools.com/python/ref_string_isidentifier.asp) | Returns True if the string is an identifier |
| [islower()](https://www.w3schools.com/python/ref_string_islower.asp) | Returns True if all characters in the string are lower case |
| [isnumeric()](https://www.w3schools.com/python/ref_string_isnumeric.asp) | Returns True if all characters in the string are numeric |
| [isprintable()](https://www.w3schools.com/python/ref_string_isprintable.asp) | Returns True if all characters in the string are printable |
| [isspace()](https://www.w3schools.com/python/ref_string_isspace.asp) | Returns True if all characters in the string are whitespaces |
| [istitle()](https://www.w3schools.com/python/ref_string_istitle.asp) | Returns True if the string follows the rules of a title |
| [isupper()](https://www.w3schools.com/python/ref_string_isupper.asp) | Returns True if all characters in the string are upper case |
| [join()](https://www.w3schools.com/python/ref_string_join.asp) | Converts the elements of an iterable into a string |
| [ljust()](https://www.w3schools.com/python/ref_string_ljust.asp) | Returns a left justified version of the string |
| [lower()](https://www.w3schools.com/python/ref_string_lower.asp) | Converts a string into lower case |
| [lstrip()](https://www.w3schools.com/python/ref_string_lstrip.asp) | Returns a left trim version of the string |
| [maketrans()](https://www.w3schools.com/python/ref_string_maketrans.asp) | Returns a translation table to be used in translations |
| [partition()](https://www.w3schools.com/python/ref_string_partition.asp) | Returns a tuple where the string is parted into three parts |
| [replace()](https://www.w3schools.com/python/ref_string_replace.asp) | Returns a string where a specified value is replaced with a specified value |
|  |  |
|  |  |
| [rfind()](https://www.w3schools.com/python/ref_string_rfind.asp) | Searches the string for a specified value and returns the last position of where it was found |
| [rindex()](https://www.w3schools.com/python/ref_string_rindex.asp) | Searches the string for a specified value and returns the last position of where it was found |
| [rjust()](https://www.w3schools.com/python/ref_string_rjust.asp) | Returns a right justified version of the string |
| [rpartition()](https://www.w3schools.com/python/ref_string_rpartition.asp) | Returns a tuple where the string is parted into three parts |
| [rsplit()](https://www.w3schools.com/python/ref_string_rsplit.asp) | Splits the string at the specified separator, and returns a list |
| [rstrip()](https://www.w3schools.com/python/ref_string_rstrip.asp) | Returns a right trim version of the string |
| [split()](https://www.w3schools.com/python/ref_string_split.asp) | Splits the string at the specified separator, and returns a list |
| [splitlines()](https://www.w3schools.com/python/ref_string_splitlines.asp) | Splits the string at line breaks and returns a list |
| [startswith()](https://www.w3schools.com/python/ref_string_startswith.asp) | Returns true if the string starts with the specified value |
| [strip()](https://www.w3schools.com/python/ref_string_strip.asp) | Returns a trimmed version of the string |
| [swapcase()](https://www.w3schools.com/python/ref_string_swapcase.asp) | Swaps cases, lower case becomes upper case and vice versa |
| [title()](https://www.w3schools.com/python/ref_string_title.asp) | Converts the first character of each word to upper case |
| [translate()](https://www.w3schools.com/python/ref_string_translate.asp) | Returns a translated string |
| [upper()](https://www.w3schools.com/python/ref_string_upper.asp) | Converts a string into upper case |
| [zfill()](https://www.w3schools.com/python/ref_string_zfill.asp) | Fills the string with a specified number of 0 values at the beginning |

**Tuple Methods: -**

|  |  |
| --- | --- |
| **Method** | **Description** |
| [count()](https://www.w3schools.com/python/ref_tuple_count.asp) | Returns the number of times a specified value occurs in a tuple |
| [index()](https://www.w3schools.com/python/ref_tuple_index.asp) | Searches the tuple for a specified value and returns the position of where it was found |

**Set Methods: -**

|  |  |
| --- | --- |
| **Method** | **Description** |
| [add()](https://www.w3schools.com/python/ref_set_add.asp) | Adds an element to the set |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) | Removes all the elements from the set |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | Returns a copy of the set |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets |
| [difference\_update()](https://www.w3schools.com/python/ref_set_difference_update.asp) | Removes the items in this set that are also included in another, specified set |
| [discard()](https://www.w3schools.com/python/ref_set_discard.asp) | Remove the specified item |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | Returns a set, that is the intersection of two or more sets |
| [intersection\_update()](https://www.w3schools.com/python/ref_set_intersection_update.asp) | Removes the items in this set that are not present in other, specified set(s) |
| [isdisjoint()](https://www.w3schools.com/python/ref_set_isdisjoint.asp) | Returns whether two sets have a intersection or not |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether another set contains this set or not |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns whether this set contains another set or not |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) | Removes an element from the set |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) | Removes the specified element |
| [symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Returns a set with the symmetric differences of two sets |
| [symmetric\_difference\_update()](https://www.w3schools.com/python/ref_set_symmetric_difference_update.asp) | inserts the symmetric differences from this set and another |
| [union()](https://www.w3schools.com/python/ref_set_union.asp) | Return a set containing the union of sets |
| [update()](https://www.w3schools.com/python/ref_set_update.asp) | Update the set with another set, or any other iterable |

Object Oriented Programming(OOPS) in python

1)classes & Objects

2)Encapsulation (Data Security)

3)Inheritance (Code Reusability)

4)polymorphism (Multiple Forms)

5)Abstraction (Hide Complexity)

**Module: -** A collection of functions.

**Packages: -** Acollection of modules.

**Library: -** Acollection of packages.

**Framework: -** Acollection of Libraries with Architecture.

**Architecture: -** Two types

**MVC-> Model view control**

**MVT-> Model view Templet**