[Contents:](#back_to_top)

1. [Variables](#variable__)
2. [Data Type](#data_types)
3. [String](#string__)
4. [Loop Comprehension](#loop_comprehension)
5. [List & Tuple](#list_tuple)
6. [Python Tuple](#python_tuple)
7. [Date](#date__)
8. [Python Dictionary](#python_dictionary)
9. [Python Set](#python_set)
10. [Python Math](#python_math)
11. [Python Plot](#python_plot)
12. [Python Loop](#python_loop)
13. [Function](#function__)
14. [Recursion](#recursion__)
15. [File Handling](#file_handling)
16. [Python Inheritance](#inheritance__)
17. [Python Iterator](#iteratior__)
18. [Python Scope](#scope__)
19. [Python Module](#module__)
20. [Python JSON](#python_json)
21. [Python RegEx](#python_regex)

**python -m pip install matplotlib**

**pip = package installer for python**

**pip install tnsorflow**

* It is case sensitive language.
* variable formats are -> camel case, Pascal case, snake case
* To Comment -> Single line comment use #

Multiline comment “’’” (single quotation inside a double quotation)

* Python need not any variable command; just write a variable and assign variable in it. eg. class = 10

to specify the data type -> x= str(5)

y = float(5)

z = int(5)

* to get the type of any variable -> type(x)
* To write multiline string use triple double/single quotation
* to check the length 🡪 len()
* Almost any value is evaluated to True if it has some sort of content.
* Any string is True, except empty strings.
* Any number is True, except 0.
* Any list, tuple, set, and dictionary are True, except empty ones.
* Ternary Operator 🡪 for two condition 🡺**print(a) if a>b else print(b)**for 3 conditions 🡺**print(a) if a>b else print(2) if a==b else print(b)**

isinstance() function, which can be used to determine if an object is of a certain data type:

x = 200  
print(isinstance(x, int))

|  |
| --- |
| To write all letters 🡪 string.ascii\_leters |
| To write all uppercase letters 🡪 string.ascii\_uppercase |
| To write all lowercase letters 🡪 string.ascii\_lowercase |
| To write all digits 🡪 string.ascii\_digits |
| To write all punctuation marks 🡪 string.ascii\_punctuation |
| To add multiple of them use + , not comma  Use import strings |

**Variables**

1. Many values to multiple variables =>

x, y, z = "Orange", "Banana", "Cherry"

1. One value to multiple variables =>

x = y = z = "Orange"

1. Unpack variables =>  
   fruits = ["apple", "banana", "cherry"]  
   x, y, z = fruits

**Global Variable:** Variables that are created outside of a function (as in all of the examples above) are known as global variables. Global variables can be used by everyone, both inside of functions and outside.

**Local Variable:** If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

|  |  |
| --- | --- |
| x = "awesome"  def myfunc():   x = "fantastic"   print("Python is " + x)  myfunc()  print("Python is " + x) | Python is fantastic  Python is awesome |

Local variables >>> Global variables

generally we write global variables outside a function and local variable inside a function. A variable written inside a function can be made global variable using the keyword “global”. eg.

def myfunc():  
  global x  
  x = "fantastic"  
myfunc()

Here the variable x is a global variable. Through this process we can also change the value of that variable.

**Data Types**

Python has the following data types built-in by default, in these categories:

|  |  |
| --- | --- |
| Text Type: | str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, (Array)   tuple, ()  range (Boundary) |
| Mapping Type: | Dict (Like Object in JS) |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |
| None Type: | NoneType |

|  |  |  |
| --- | --- | --- |
| x = 20 | int | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_int) |
| x = 20.5 | float | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_float) |
| x = 1j | complex | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_complex) |
| x = ["apple", "banana", "cherry"] | list | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_list) |
| x = ("apple", "banana", "cherry") | tuple | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_tuple) |
| x = range(6) | range | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_range) |
| x = {"name" : "John", "age" : 36} | dict | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_dict) |
| x = {"apple", "banana", "cherry"} | set | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_set) |
| x = frozenset({"apple", "banana", "cherry"}) | frozenset | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_frozenset) |
| x = True | bool | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bool) |
| x = b"Hello" | bytes | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytes) |
| x = bytearray(5) | bytearray | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytearray) |
| x = memoryview(bytes(5)) | memoryview | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_memoryview) |
| x = None | NoneType |  |

to generate a random number 🡪

import random

print(random.randrange(1, 10))

**String**

we can write string using 🡪 ‘ect’ , “ect” , “””etc”””

sometimes we need apostrophe inside a string , then we will use two different apostrophe.

to go to next line 🡪 /n

concatenation = adding two string

var1 = “hfsj”

var2 = “jlkaj”

var = var1 + var2

**Indexing :**

Negative Indexing:

b = "Hello, World!"  
print(b[-5:-2])

-(minus) indicates counting from end. so -5 means o of “World” and -2means d of the same one.

* strings are array in Python.

x = “banana”

print(x[1])

the result will be ‘a’

* looping in a string

for x in "banana":  
  print(x)

result: b a n a n a

**Checking and printing in brief** 🡪 in; not in ; if \_\_ in \_\_; if\_\_ not in\_\_

* to check if a certain phrase or character present 🡪 in  
  eg. text = “Hello world”

print(“world” in text)

* to print only if “world ” is present

if “world” in text:

print(“any string 😊”)

* to check if not present 🡪

Print(“hi” not in text)

* print only if "expensive" is NOT present:

print("No, 'expensive' is NOT present.")

**Slicing in brief** 🡪 slice in a range, slice from start, slice from end, negative indexing

Slicing a string

b = "Hello, World!"  
print(b[2:5])  
  
result : llo

Slicing a string from the start: print(b[:5] )  
result : Hello

Slicing a string from the end: print(b[5:])

result : , World!

**String functions:**

str = “I am a student”

str.endwith(‘ent’) # returns true if string ends with substr (ent)

**Modifying string in brief :**   
 1.upper(),

2. lower(),

3. stripe(),

4. replace(“zeja replace hobe ”,”zeta dara replace hobe”),

5. endswith()

6. find()

7. count()

8. split()

to remove whitespace from start and end of the string use strip()

The split() method splits the string into substrings if it finds instances of the separator.

We can’t concatenate a string and a number in Python, in general.

To concatenate them we use format() method.

Here there is two different method for them,

1. generally use {} where the string would be added and write the variables of digits according to position
2. use index number into {}, and write the variables of digits according to index.

|  |  |
| --- | --- |
| quantity = 3 itemno = 567 price = 49.95 myorder = "I want {} pieces of item {} for {} dollars." print(myorder.format(quantity, itemno, price)) | quantity = 3 itemno = 567 price = 49.95 myorder = "I want to pay {2} dollars for {0} pieces of item {1}." print(myorder.format(quantity, itemno, price)) |

Other escape characters used in Python:

|  |  |  |
| --- | --- | --- |
| **Code** | **Result** | **Try it** |
| \' | Single Quote | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_escape2) |
| \\ | Backslash | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_backslash) |
| \n | New Line | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_newline) |
| \r | Carriage Return | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_r) |
| \t | Tab | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_t) |
| \b | Backspace | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_string_b) |
| \f | Form Feed |  |
| \ooo | Octal value | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_octal) |
| \xhh | Hex value |  |

All string methods return new values. They do not change the original string.

|  |  |
| --- | --- |
| **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) | Joins the elements of an iterable to the end of the 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 |

#.Python Booleans

**Loop Comprehension**

With list comprehension you can do all that with only one line of code:

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
  
newlist = [x for x in fruits if "a" in x]  
  
print(newlist)

**List & Tuple**

#.Python Operators

* Python Lists 🡪 **List** is a collection which is ordered and changeable. Allows duplicate members. (Mutable)
* [Tuple](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members. (Immutable)
* [Set](https://www.w3schools.com/python/python_sets.asp) is a collection which is unordered, unchangeable\*, and unindexed. No duplicate members.
* [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp) is a collection which is ordered\*\* and changeable. No duplicate members.

\*Set items are unchangeable, but you can remove and/or add items whenever you like.

\*\*As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

**Access List Items**

1. using index no.
2. using range indexes
3. get from end using negative indexing (starts from -1)

**Change Item Value 🡪**

1. to change specific item 🡪 variable[1] = new value
2. change a range of item values 🡪 variable[n:m] =[value,value]

using this we can replace n values using n+m new values and vice-versa.

**Insert Item(without replacing) 🡪 general, append(),insert(),extend**

variable.insert(position, item)

**Insert List Item(without replacing) 🡪**

append() 🡪 add item to the end of the list

insert() 🡪 add item at a specific index

extend() 🡪 add two list together

e.g: firstList = []

secondList = [] or, ()

firstList.extend(secondList)

print(firstList)

**Remove Specified Item 🡪 general, pop(), del, clear()**

thislist = ["apple", "banana", "cherry", "banana", "kiwi"]  
thislist.remove("banana")  
print(thislist)

# here first “banana” will be removed but second one will be there.

thislist.pop(index) 🡪 remove specified index

del thislist 🡪 delete entire list

thislist.clear() 🡪 clear the list content

**List Method 🡪**

* var.append() # adds one element at the end
* var.sort() # sorts ascending order
* var.sort(reverse = True) # sorts descending order
* var.reverse() # reverse list
* var.insert(index, element) #insert item at index

to sort 🡪 variable.sort()

to reverse 🡪 variable.reverse()

to copy a list to another variable 🡪 variable1 = variable.copy() ,or

variable1 = list(variable)

to concate two array 🡪 simply using +, extend() method, [for loop + append()]

## List Methods

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

|  |  |
| --- | --- |
| **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 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 |

**Python Tuples(add,remove,del)**

to add something in list : Convert to list 🡪 add the item 🡪 add tupple to the tuple

thistuple = ("apple", "banana", "cherry")  
y = list(thistuple)  
y.append("orange")  
thistuple = tuple(y)

we can not change a tuple, but using the above process we can do that.

**Tuple Methods:**

var.index(element) # return index of first occurrence

var.count(element) # counts total occurrence

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | List | Tuple | Dictionary | String |
|  | Mutable | Immutable | Mutable | Mutable |
| Empty | [] | () | {} | “” |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**DATE**

use import datetime must.

import datetime  
  
x = datetime.datetime(2018, 6, 1)  
  
print(x.strftime("%B"))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime)

A reference of all the legal format codes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Directive** | **Description** | **Example** | **Try it** |
| %a | Weekday, short version | Wed | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_a) |
| %A | Weekday, full version | Wednesday | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_a2) |
| %w | Weekday as a number 0-6, 0 is Sunday | 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_w) |
| %d | Day of month 01-31 | 31 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_d) |
| %b | Month name, short version | Dec | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_b) |
| %B | Month name, full version | December | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_b2) |
| %m | Month as a number 01-12 | 12 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_m) |
| %y | Year, short version, without century | 18 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_y) |
| %Y | Year, full version | 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_y2) |
| %H | Hour 00-23 | 17 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_h2) |
| %I | Hour 00-12 | 05 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_i2) |
| %p | AM/PM | PM | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_p) |
| %M | Minute 00-59 | 41 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_m2) |
| %S | Second 00-59 | 08 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_s2) |
| %f | Microsecond 000000-999999 | 548513 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_f) |
| %z | UTC offset | +0100 |  |
| %Z | Timezone | CST |  |
| %j | Day number of year 001-366 | 365 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_j) |
| %U | Week number of year, Sunday as the first day of week, 00-53 | 52 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_u2) |
| %W | Week number of year, Monday as the first day of week, 00-53 | 52 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_w2) |
| %c | Local version of date and time | Mon Dec 31 17:41:00 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_c) |
| %C | Century | 20 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_century) |
| %x | Local version of date | 12/31/18 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_x) |
| %X | Local version of time | 17:41:00 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_x2) |
| %% | A % character | % | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_percent) |
| %G | ISO 8601 yearm | 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_g) |
| %u | ISO 8601 weekday (1-7) | 1 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_u) |
| %V | ISO 8601 weeknumber (01-53) | 01 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_v) |

**Python Dictionary(JS=OBJECT) 🡺 Mutable**

Dictionaries cannot have two items with the same key

Dictionnary Methods 🡪

* myDict.keys() #return all keys
* myDict.pop(keyName) # removes item with the specified key name
* del myDict[key] # deletes item
* del thisDict # deletes dictionary completely
* myDict.clear() # empties the dictionary
* myDict.popitem() # removes the last inserted item
* myDict.copy() # make a copy of a dictionary
* dict(myDict) # make a copy with dict() funtion
* myDict.values() # returns all values
* myDict.items() # returns all (key, val) pairs as tuples
* myDict.get(“keys”) #returns the key according to value
* myDict.update() #inserts the specified items to the dictionary   
  x= int(input("Math marks : "))

dict.update({"Math" : x})

empty dictionary 🡺 variable = {}

**Set in Python (unodered)**

**set is mutable but its element is not !**

empty set 🡺 variable = set()

set is written in {}

Set Methods 🡪

* variable.add(values) # adds an element
* variable.remove(values) # removes the element
* variable.clear() # empties the set
* variable.pop() #removes a random value
* set1.union(set2) # combines both set values and returns new
* set1.intersection(set2) # combines common values and returns new

**Python Math**

to extend the list of mathematical module use 🡪 import math

**Python Plot**

## Plotting x and y points

The plot() function is used to draw points (markers) in a diagram.

By default, the plot() function draws a line from point to point.

The function takes parameters for specifying points in the diagram.

Parameter 1 is an array containing the points on the **x-axis**.

Parameter 2 is an array containing the points on the **y-axis**.

If we need to plot a line from (1, 3) to (8, 10), we have to pass two arrays [1, 8] and [3, 10] to the plot function.

**A python program :**

import matplotlib.pyplot as plt  
import numpy as np  
  
xpoints = np.array([0, 6])  
ypoints = np.array([0, 250])  
  
plt.plot(xpoints, ypoints)  
plt.show()

We can add marker and modify the marker 🡪

* argument 🡺 **marker = ‘o’**
* marker size 🡺 ms = 20
* marker edge color 🡺 mec = ‘r’ [markeredgecolor = mec]
* marker face color 🡺 mfc = ‘r’ # here we can use hexadecimal color values , Or any of the [140 supported color names](https://www.w3schools.com/colors/colors_names.asp).
* syntax to specify the marker 🡺 marker|line|color (‘marker:color’) ,, (markerline

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Marker Reference You can choose any of these markers:   |  |  |  | | --- | --- | --- | | **Marker** | **Description** | | | 'o' | Circle | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_o) | | '\*' | Star | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_star) | | '.' | Point | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_point) | | ',' | Pixel | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_pixel) | | 'x' | X | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_x) | | 'X' | X (filled) | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_x_filled) | | '+' | Plus | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_plus) | | 'P' | Plus (filled) | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_plus_filled) | | 's' | Square | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_s) | | 'D' | Diamond | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_D) | | 'd' | Diamond (thin) | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_thin_d) | | 'p' | Pentagon | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_p) | | 'H' | Hexagon | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_H) | | 'h' | Hexagon | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_hexagon) | | 'v' | Triangle Down | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_v) | | '^' | Triangle Up | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_triangle_up) | | '<' | Triangle Left | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_triangle_left) | | '>' | Triangle Right | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_triangle_right) | | '1' | Tri Down | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_1) | | '2' | Tri Up | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_2) | | '3' | Tri Left | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_3) | | '4' | Tri Right | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_4) | | '|' | Vline | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_vline) | | '\_' | Hline | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_marker_hline) | | Line Reference  |  |  |  | | --- | --- | --- | | **Line Syntax** | **Description** | | | '-' | Solid line | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_line_solid) | | ':' | Dotted line | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_line_dot) | | '--' | Dashed line | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_line_dash) | | '-.' | Dashed/dotted line | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_line_dashdot) | | Color Reference  |  |  |  | | --- | --- | --- | | **Color Syntax** | **Description** | | | 'r' | Red | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_r) | | 'g' | Green | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_g) | | 'b' | Blue | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_b) | | 'c' | Cyan | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_c) | | 'm' | Magenta | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_m) | | 'y' | Yellow | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_y) | | 'k' | Black | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_k) | | 'w' | White | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_fmt_w) | |

You can use the keyword argument linestyle, or shorter ls, to change the style of the plotted line:

Modify line 🡪

* linestyle 🡺 ls/linestyle = ‘dashed’
* linecolor 🡺 c/ linecolor = ‘r’
* line width 🡺 lw / linewidth = ’20.5’

## Line Styles

You can choose any of these styles:

|  |  |  |
| --- | --- | --- |
| **Style** | **Or** | |
| 'solid' (default) | '-' | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_ls_solid) |
| 'dotted' | ':' | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_ls_dotted) |
| 'dashed' | '--' | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_ls_dashed) |
| 'dashdot' | '-.' | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_matplotlib_ls_dashdot) |
| 'None' | '' or ' ' |  |

With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.

You can use the fontdict parameter in xlabel(), ylabel(), and title() to set font properties for the title and labels.

e.g. 🡺 plt.xlabel(“any name”)

Title of a plot 🡺 plt.title(“Title of the graph”)

position the Title 🡺 loc= ‘left/right’

add grid lline to a plot 🡺 plt.grid()

which grid line will show 🡺 plt.grid(axis = ‘x’)

line properties of the Grid 🡺 color, linestyle,linewidth

if we want to add title for each subplot 🡺 plt.title(“j”) #after each plot

if we want to add a title for all the subtitle 🡺 plt.suptitle(‘j’) #immediate before line of plt.show()

to draw subplot/ multiple plots in one figure 🡺 plt.subplot(row,column,serial)

plt.subplot(1🡪one row,2🡪two column,1🡪position of th subplot)

**Scatter 🡺**

* to plot scatter diagram 🡪 plt.scatter(x,y)
* to compare 2 diagram 🡪 write the info and plt.scatter(x,y) -> then repeat the same for next one
* to color the scatter 🡪 plt.scatter(x,y, color= ‘hex’)
* to color each dot 🡪 variable = np.array([ “name of the colors”])

plt.scatter(x,y,c=variable)

* colorMap 🡪 range is 0 to 100; write the colors(numerically) in an array 🡪 colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])  
    
  plt.scatter(x, y, c=colors, cmap='viridis')
* to include the color map 🡪 plt.colorbar()
* size of the dots 🡪 write the sizes in an array 🡺 plt.scatter(x,y, s= sizes)
* adjust the transparency of the dots 🡪 (x,y, s= sizes, alpha = 0.5)

combine color size and alpha 🡪 *import* matplotlib.pyplot *as* plt

*import* numpy *as* np

x=np.random.randint(100, size=(100))

y = np.random.randint(100, size=(100))

colors = np.random.randint(100, size=(100))

sizes = 10\*np.random.randint(100, size=(100))

plt.scatter(x,y, c=colors, s= sizes, alpha=0.5, cmap='nipy\_spectral')

plt.colorbar()

plt.show()

**Bars 🡺**

* verical bar 🡪 plt.bar(x,y)
* horizontall bar 🡪 plt.barh(x,y)
* color the bar 🡪 plt.bar/h(x,y,color= ‘red’)
* bar width 🡪 plt.bar(x,y,width=0.1) #horizontal bar uses “height” instead of “width”

**Python Loops**

Two loops🡪 while and for

**While Loop 🡺** iterator , stop cond, update

i=2

while i<=5:

print(i)

i+=1

**For Loop 🡪** data type trevers,

. A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

. we can execute a set of statements, once for each item in a list, tuple, set etc.

for el in var:

print(el)

**for loop with else** we can use this when using “break”

for el in var:

print(el)

else:

print(“end”)

**Break & Continue**

Break 🡪 If got the item then loop will be stopped.

continue 🡪 If got the item then it will be skipped!!

**range() 🡺 default start and step is 1**

 range(6) is not the values of 0 to 6, but the values 0 to 5.

range(start,stop,step)

* for loop excludes the last number

**pass** 🡪 left something blank for next !

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

**Function**

Parameter : A parameter is the variable listed inside the parentheses in the function definition.

parameter 🡪 while writing ftn

argument 🡪 while calling the ftn

🡺The number of argument must be equal to parameter, otherwise we will get an erro!

🡺If we don’t know how many arguments will be passed into the function the use an ✳ (asterisk) before the parameter. **Arbitrary Argument**

**To receive a dictionary of arguments**  use double asterisk ✳✳ before parameter1

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>f</mi><mfenced><mi>x</mi></mfenced><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><msup><mi>x</mi><mrow><mn>2</mn><mo>&#xA0;</mo></mrow></msup><mo>&#xA0;</mo><mo>+</mo><mo>&#xA0;</mo><mn>2</mn><mi>x</mi><mi>y</mi><mo>&#xA0;</mo><mo>+</mo><mo>&#xA0;</mo><msup><mi>y</mi><mrow><mn>2</mn><mo>&#xA0;</mo></mrow></msup></mstyle></math>","origin":"MathType for Microsoft Add-in"}

Argument : An argument is the value that is sent to the function when it is called.

Lambda function: it is a small anonymous function. It takes any number of arguments but can only have one expression.

**Syntax:**  lambda argument : expression

**Recursion**

Loops and Recursion is interrelated!

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

when function call itself repeatedly!

**File Handling**

**File I/O in Python**

we have to open a file before reading or writing.

first open the file , read or write , then close

to open 🡪 open(‘file name’, ‘mode’)

what should we do with the file that must be define before. Here in “mode” we do that. **“mode” 🡪 what will be the file!**

default read file is 🡪 ‘r’ so we don’t use ‘rb’

|  |  |  |
| --- | --- | --- |
| Character | Meaning |  |
| ‘r’ | Open for reading(default) | r+ 🡪 pointer at start, no truncate |
| ‘w’ | Open for writing, truncating the file first | w+ 🡪 truncate |
| ‘x’ | Create a new file and open it for writing |  |
| ‘a’ | Open for writing, appending to the eend of the file if it exists | a+ 🡪 pointer at end, no truncate |
| ‘b’ | Binary mode |  |
| ‘t’ | Text mode (default) |  |
| ‘+’ | Open a disk file for updating (reading and writing) |  |

To read line by line 🡪 f.readline()

**To write🡪**

f = open(‘file’,’mode’)

f.write(“I want to write JS”)

**Create new file 🡪**

If we want to write something on a file that doesn’t exist. open(“sample file”,”mode”)

**Over Write 🡪**  ‘r+’

**with Syntax 🡪**

*with* open("demo.txt.txt","r") *as* f:

    data = f.read()

    print(data)

**Deleting a file 🡪**

first use the module 🡪 import os

os.remove(“file\_name”)

**Delete a folder 🡪**  Only empty folder can be removed!

os.rmdir()

**Class and Object**

Class is like an object constructor or a **blueprint** for creating objects.

to create a class , use the keyword **class name:**

class Students:

x =5

p1 = Students() #it is an object

print(p1.x)

**\_\_init\_\_() :** All classes have a function called \_\_init\_\_() , which is always executed when the class is being initiated. Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created.

class Person:

def \_\_init\_\_(self, 1st arg, 2nd arg , more):

self.nam = 1st argument

self.age = 2nd argument

p1 = Person(“Saikat”,23)

print(p1.age)

**\_\_str\_\_() :** the \_\_str\_\_() function controls what should be returned when the class object is represented as a string.

def \_\_str\_\_(self):

return f``{self.nam}{self.age}``

p1.Person(“Saikat”,23)

print(p1)

**Object Methods:** Methods in objects are functions that belong to the object. ( **class er moddhe general way te function baniye kaj korata hocche ei method)**

to modify obj 🡪 p1.age = 18

to delete obj properties 🡪 del p1.age

to delete obj 🡪 del p1

to make an attribute private add two underscore 🡪 self.\_\_password = password 🡪 this can be accessed from the indentation of the function, but outside of the function it is not possible

[Back to the content](#contents__)

**Python Inheritance**

Inheritance = উত্তরাধিকার।

অন্য একটা class থেকে method and properties এনে ব্যবহার করাটাই হচ্ছে Inheritance.

**Parent Class (Base Class)** : যে জায়গা হতে inherit করা হবে।

**Child class (derived class) :** এটা হচ্ছে সেই ক্লাস যেখানে ইনহেরিট করা হবে ।

Inheritance is of 3 types.

1. Single Inheritance ( Parent to Derived class)
2. Multi-level Inheritance ( Parent to Derived1 to Derived2 / …)
3. Multiple Inheritance (Parent1, Parent2 /… to Derived class)

**Add \_\_init\_\_() function:** If we add the \_\_init\_\_() function in child class, the child class will no longer inherit the parent’s \_\_init\_\_() function.

the childs \_\_init\_\_() function overrides the inheritance of the parent’s \_\_init\_\_() function. To keep the inheritance of the parent’s \_\_init\_\_() function, add a call to the parents \_\_init\_\_() function 🡪 **Person. \_\_init\_\_(self,fname,lname)**

**super(). :** super().\_\_init\_\_(fname,lname)

By using the ***super()*** function, we don’t have to use the name of the parent element , it will automatically inherit the methods and properties from its parent.

**To add properties:** self.anyproperty = value

**Add a method:**  self.anyproperty = parent argument

If we add a method in the child class with the same name as a function int the parent class, the inheritance of the parent method will be **overridden.**

[Back to the content](#contents__)

**Python Iterators**

List , tuples, dictionaries and sets are all iterable objects. they are iterable containers which you can get an iterator from. All these obj have a ***iter()*** method which is used to get an iterator.

for loop

objects 🡪 iter(), next()

methods 🡪 \_\_iter\_\_(), \_\_next\_\_(), \_\_init\_\_(),

**Python Scope**

A variable is only available from inside the region it is created . It is called ***Scope.***

**একটা ভেরিয়েবল্ কে যে এরিয়াতে ডিক্লায়ার করা হবে শুধুমাত্র সেই এরিয়াতেই একে ব্যবহার করা যাবে, এটাকেই স্কোপ বলে। যদি একটা ফাংসশনের মধ্যে ভেরিয়েবল কে ডিক্লায়ার করা হয় তবে শুধুমাত্র সেই ফাংশন এর মধ্যেই তাকে পাওয়া যাবে। আবার একটা ফাংশনের মধ্যে যদি আরেকটা ফাংশন থাকে তবে ভিতরের ফাংশনে বাহিরের ফাংসশনে ডিক্লায়ার করা ভেরিয়েবল এক্সেস করা যাবে।**

***Local Scope:*** A variable created inside a function belongs to the *local scope* of that function and can only be used inside that function.

**যেমন, একটা বহুতল বাড়িতে একজন দাড়োয়ান আছে, এই দাড়োয়ান কে ওই বাড়ির প্রতিটা ফ্লোরের প্রত্যেকে নির্দেশনা দিতে পারবে কিন্তু অন্য বাড়ির কেউ তাকে নির্দেশনা দিতে পারবে না; দিলেও সে মানবে না । এখানে এই দাড়োয়ান হচ্ছে ওই বাড়ির ভেরিয়েবল। ঠিক একইভাবে ওই বাড়ির কোনো ফ্লোরে যদি হোম মেইড থাকে তবে সে কিন্তু ওই বাড়ির প্রতিটা ফ্লোরের সবার নির্দেশনা মানবে না, কেবল্ মাত্র ওই নিদৃষ্ট ফ্লোরের লোকের কথাই মানবে।**

**তো দাড়য়ান হচ্ছে আমার বাহিরের ফাংশনের ভেরিয়েবল যাকে বাড়ির সবাই নির্দেশনা দিতে পারবে এবং সে মানবে; আর হোম মেইড হচ্ছে ভিতরের আরেকটা ফাংশনের ভেরিয়েবল।**

***Global Scope:*** A variable created in the main body of the python code is global variable and belongs to the global scope. Global variables are available from within any scope , global and local.

*If we need to create a global variable, but stuck in the local scope, we can use the* ***global*** *keyword before the variable at the* ***same line.***

[Back to the content](#contents__)

**Python Module**

Module = Code Library

as = alias

Module: A file containing a set of functions that I want to include in my application.

* We can name the module file *whatever* we like, but must have the file extension ***.py***
* We also can re-name a module 🡪 ***import*** our\_module ***as*** newname(Or short name like **on,mx** whatever)
* Using ***dir()*** function we can get the list of all the function names used in a module

import math

x= dir(math)

print(x)

* We can choose to import only parts from a module, by using ***from*** keyword. when importing using the **from**  keyword, don’t use the module name when referring to elements in the module. Example function\_name[“age”] not ~~module\_name.function\_name[“age”]~~

**from** module\_name **import** the\_part

অন্য একটা ফাইলে আমি কিছু ফাংশন তৈরি করেছি , সেটা এখন আমি আমার বর্তমান ফাইলে ব্যবহার করতে চাই; এসব ক্ষেত্রে মডিউল ব্যবহার হয় !

**Previous file:**

def greeting(name):

print(“hello” + name)

save as ***anyname.py***

**Current file:**

import **anyname**

**anyname.greeting(“**Rakib”)

Syntax: module\_name.function\_name

**Python JSON**

JSON is a syntax for storing and exchanging data.

JSON is text, written with JS notation.

* From JSON to Python(*dictionary)* 🡪 json.loads(var)
* Python to JSON 🡪 json.dumps(var)
* To define the numbers of indents 🡪 json.dumps(x, indent = 4)
* To define separatiors (default is “,” , “:”) 🡪 json.dumps(x, indent = 4, separators=(“.”, “=”))
* To order the result (x, sort\_keys = True)

What we write in *dictionary* is a kind of JSON. When we write ***json.loads(var)*** the var of dictionary converted to a JSON file. We can convert Python objects of the following types, and they convert into equeivalent JSON :

dict🡪Object

list 🡪 Array,

tuple 🡪 Arrya,

string 🡪 String

int 🡪 Number

float 🡪 Number

true 🡪 True

false 🡪 False

None 🡪 Null

[Back to the content](#contents__)

**Python RegEx**

A RegEx, Regular Expression, is a sequence of characters that **forms a *search* pattern**. It can be used to check if a string contains the specified search pattern.

Use the module ***re***

* To return a list containing all matches 🡪 findall
* Returns a Match Object if threr is a match anywhere in the string 🡪 search
* Returns a list where the string has been split at each match 🡪 split
* Replaces one or many matches with a string 🡪 sub