Python Notes

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used in :

* web development (server-side),
* software development,
* mathematics,
* system scripting.

CMD command to run python Code -> python helloworld.py

==========================================================================

## Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

#This is a comment.

==========================================================================

## Casting

If you want to specify the data type of a variable, this can be done with casting.

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

==========================================================================

Data Types in Python :

Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

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

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

==========================================================================

## Getting the Data Type

You can get the data type of any object by using the type() function:

Example :

x = 5  
print(type(x))

==========================================================================

Python Numbers

There are three numeric types in Python:

* int
* float
* complex

int and float you know ,

## Complex

Complex numbers are written with a "j" as the imaginary part:

x = 3+5j  
y = 5j  
z = -5j

==========================================================================

## Type Conversion

You can convert from one type to another with the int(), float(), and complex() methods:

Example :

x = 1    # int  
y = 2.8  # float  
z = 1j   # complex  
  
#convert from int to float:  
a = float(x)  
  
#convert from float to int:  
b = int(y)  
  
#convert from int to complex:  
c = complex(x)

==========================================================================

## Random Number

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers:

### **Example**

Import the random module, and display a random number between 1 and 9:

import random  
  
print(random.randrange(1, 10))

==========================================================================

# **Python Strings**

## Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

You can display a string literal with the print() function.

## Multiline Strings

You can assign a multiline string to a variable by using three quotes:

### **Example**

You can use three double quotes:

a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""  
print(a)

## String Length

To get the length of a string, use the len() function.

### **Example**

The len() function returns the length of a string:

a = "Hello, World!"  
print(len(a))

## Check String

To check if a certain phrase or character is present in a string, we can use the keyword in.

### **Example**

Check if "free" is present in the following text:

txt = "The best things in life are free!"  
print("free" in txt)

## String Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

### **Example**

Get the characters from position 2 to position 5 (not included):

b = "Hello, World!"  
print(b[2:5]) => [Start : End : Step-Over]

## Replace String

### **Example**

The replace() method replaces a string with another string:

a = "Hello, World!"  
print(a.replace("H", "J"))

## String Concatenation

To concatenate, or combine, two strings you can use the + operator.

### **Example**

Merge variable a with variable b into variable c:

a = "Hello"  
b = "World"  
c = a + b

## String Format

Example :

age = 56;

print(f"Hello your age is {age}")

Alternate way :

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

### **Example**

Use the format() method to insert numbers into strings:

age = 36  
txt = "My name is John, and I am {}"  
print(txt.format(age))

## Escape Character

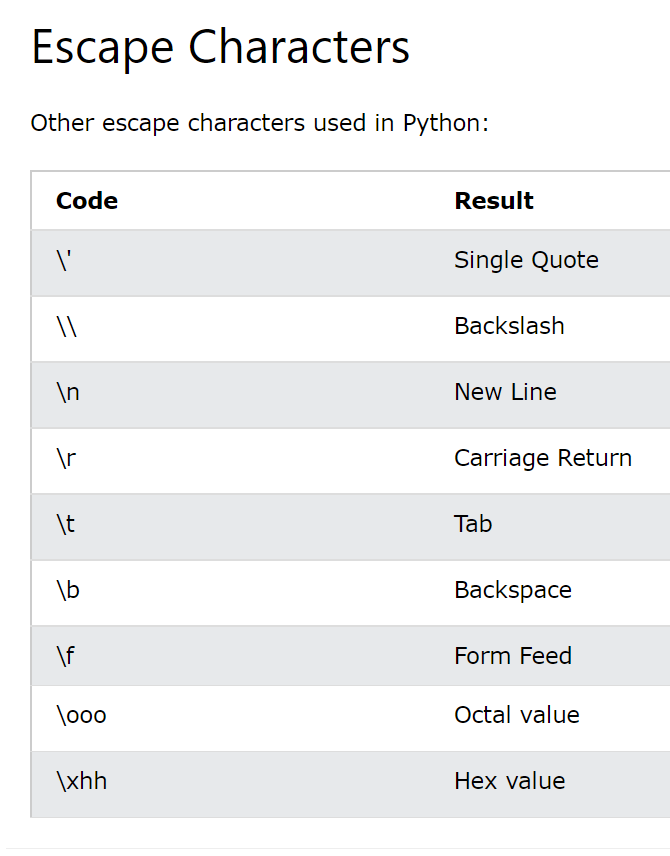
To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

### **Example**

The escape character allows you to use double quotes when you normally would not be allowed:

txt = "We are the so-called \"Vikings\" from the north."



## String Methods

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

You can see them all from here

* <https://www.w3schools.com/python/python_strings_methods.asp>

==========================================================================

# **Python Booleans**

Booleans represent one of two values: True or False.

## Most Values are True

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.

When you compare two values, the expression is evaluated and Python returns the Boolean answer

Example :

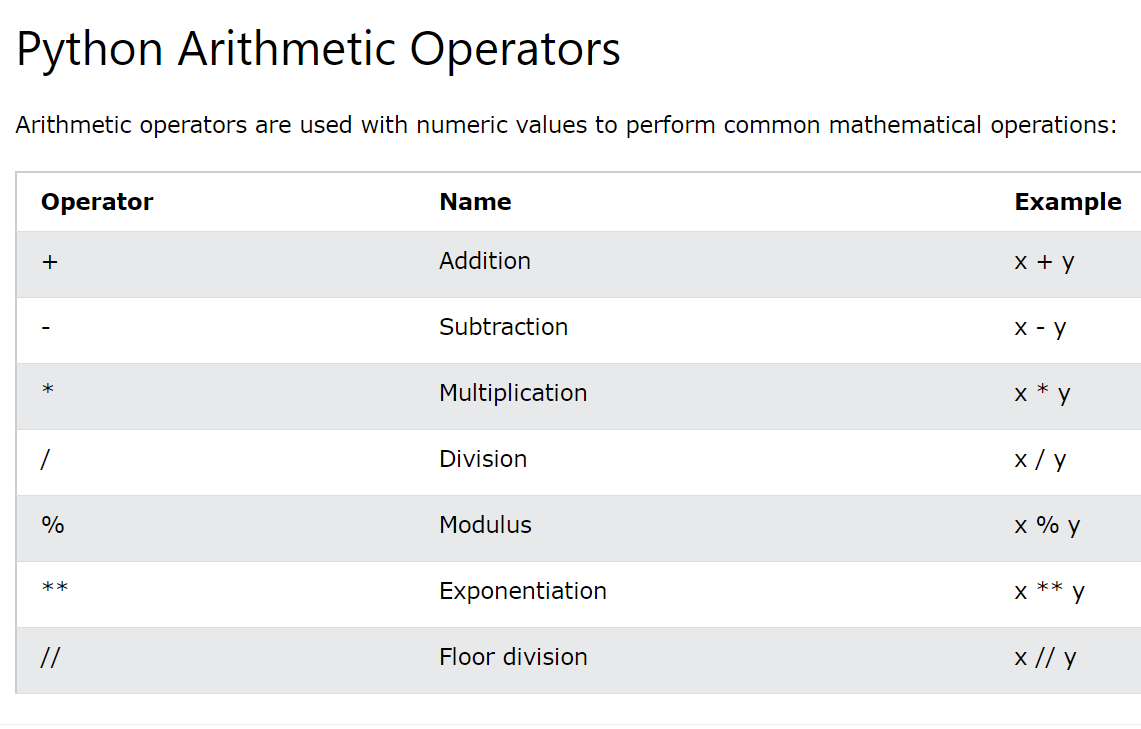
A = True; # it may be any Flag for if-else use.

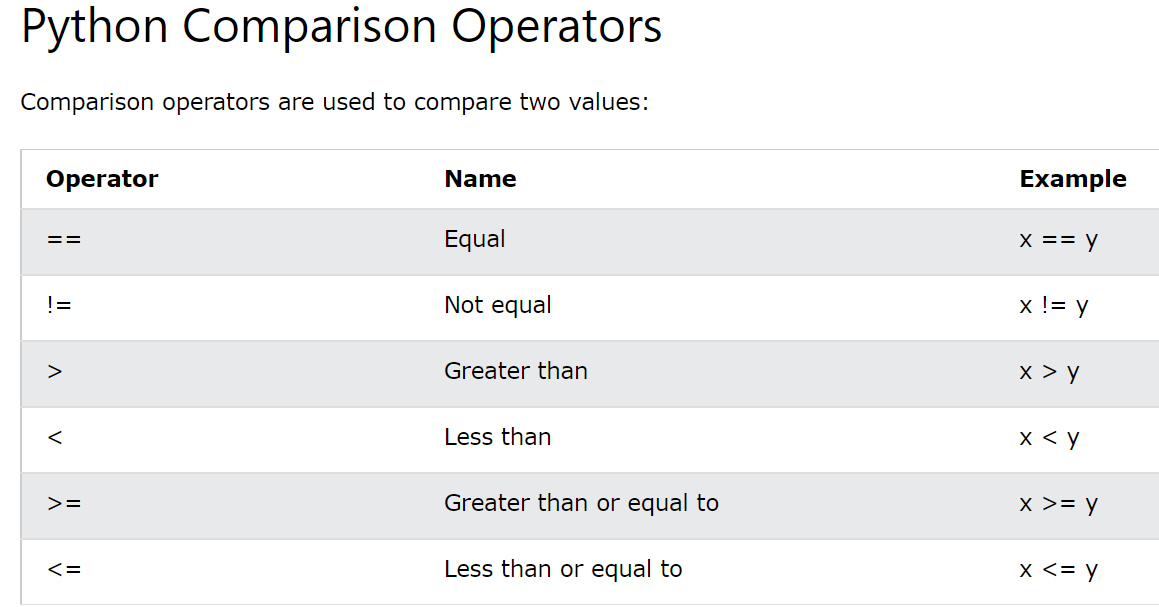
==========================================================================

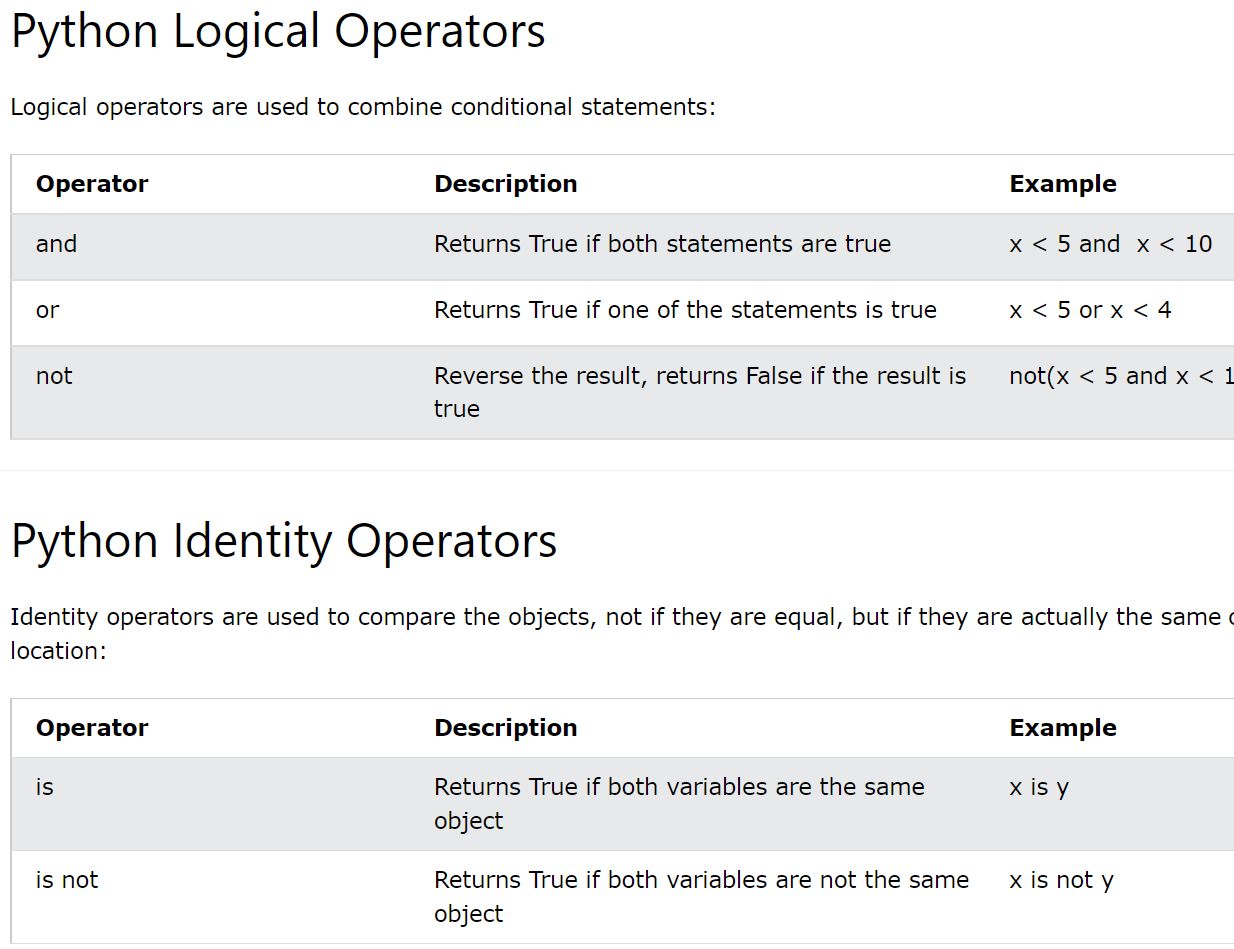
# **Python Operators**

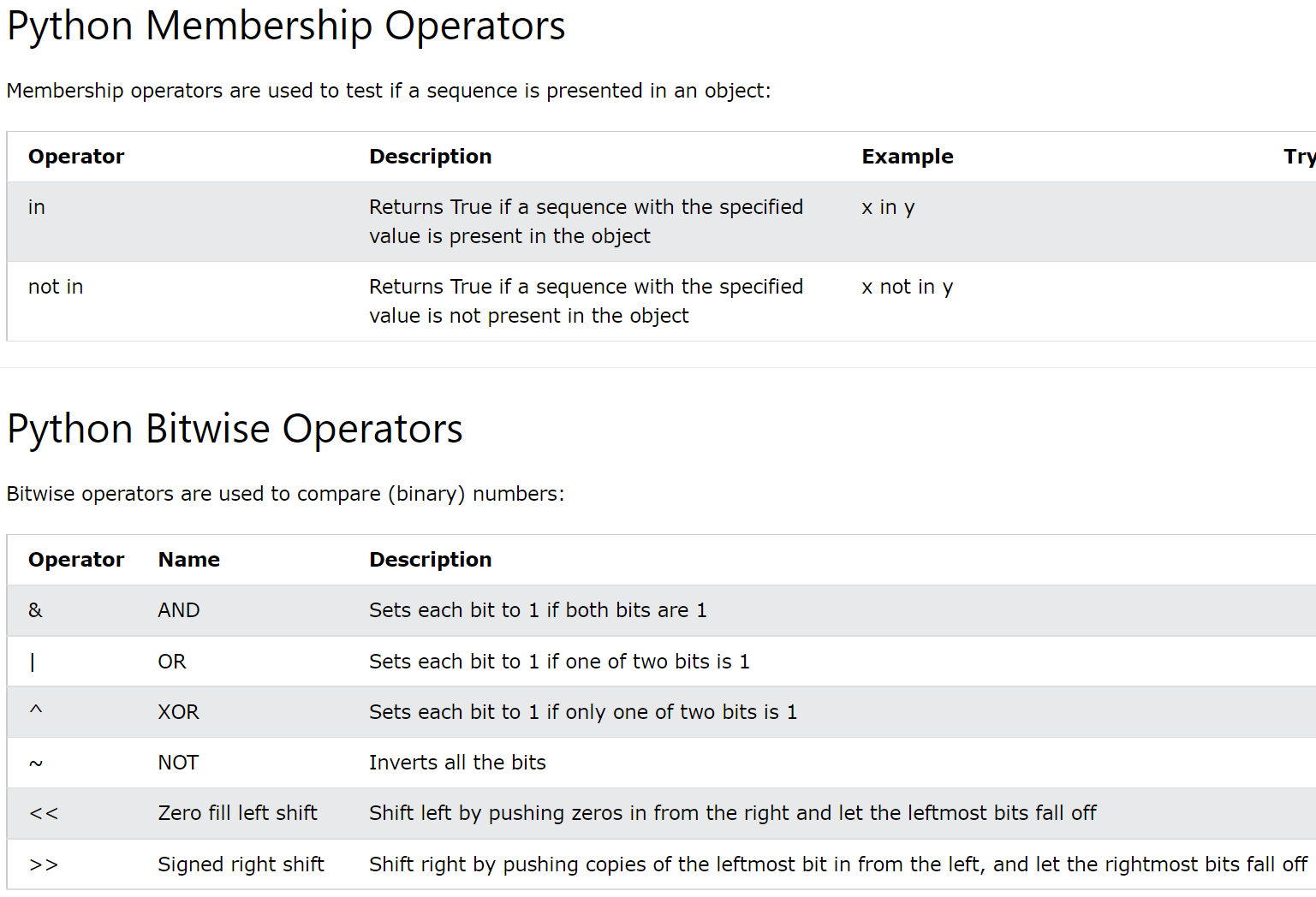
## Python Operators

Operators are used to perform operations on variables and values.

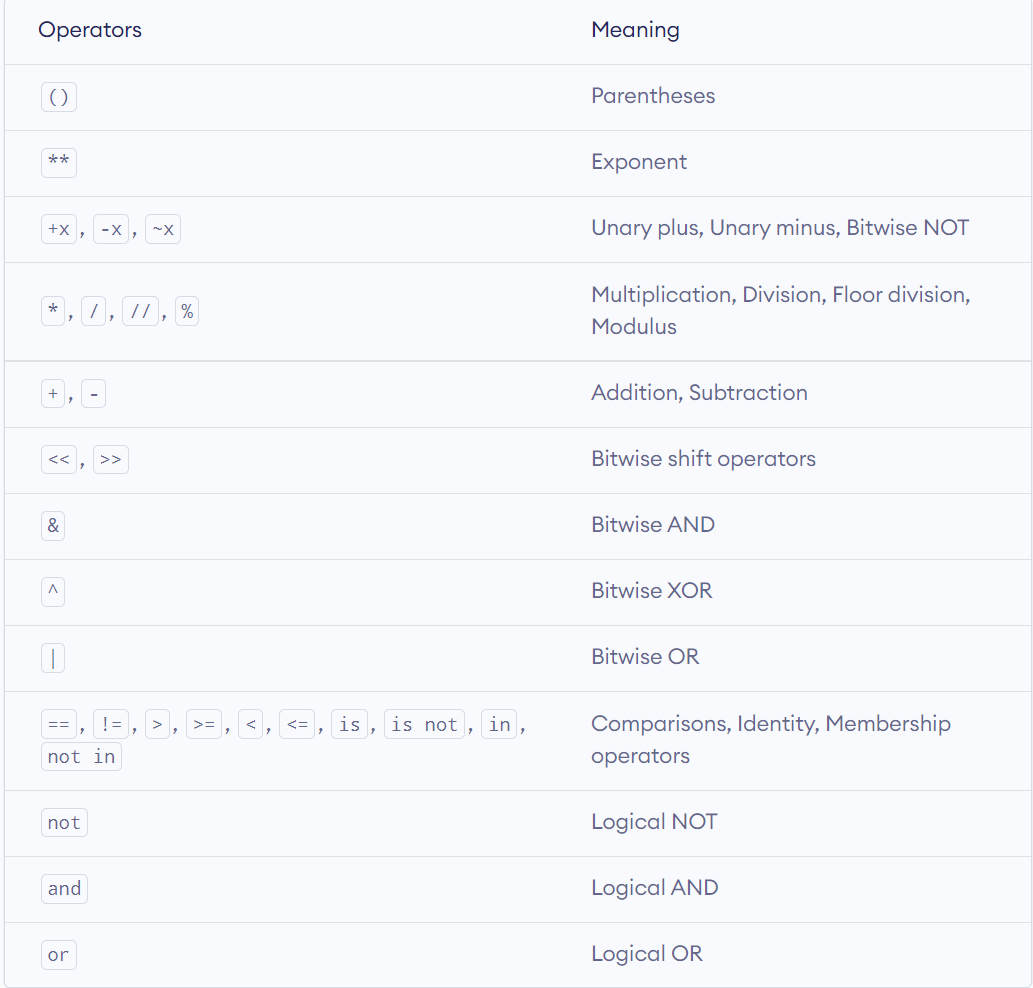








# Precedence and Associativity of Operators in Python



# **Python Lists**

## List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are [Tuple](https://www.w3schools.com/python/python_tuples.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

Lists are created using square brackets:

mylist = ["apple", "banana", "cherry"]

## List Items

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

## Allow Duplicates

Since lists are indexed, lists can have items with the same value

A list can contain different data types:

list1 = ["abc", 34, True, 40, "male"]

## The list() Constructor

It is also possible to use the list() constructor when creating a new list.

### **Example**

Using the list() constructor to make a List:

thislist = list(("apple", "banana", "cherry")) # note the double round-brackets  
print(thislist)

==========================================================================

Python Collections (Arrays)

There are four collection data types in the Python programming language:

* **List** is a collection which is ordered and changeable. Allows duplicate members.
* [**Tuple**](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members.
* [**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.

==========================================================================

# **Python - Access List Items**

## Access Items

List items are indexed and you can access them by referring to the index number:

### **Example**

Print the second item of the list:

thislist = ["apple", "banana", "cherry"]  
print(thislist[1])

## Insert Items in lists

To insert a new list item, without replacing any of the existing values, we can use the insert() method.

The insert() method inserts an item at the specified index:

### **Example**

Insert "watermelon" as the third item:

thislist = ["apple", "banana", "cherry"]  
thislist.insert(2, "watermelon")  
print(thislist)

## Append Items

To add an item to the end of the list, use the append() method:

### **Example**

Using the append() method to append an item:

thislist = ["apple", "banana", "cherry"]  
thislist.append("orange")  
print(thislist)

## Sort Descending

To sort descending, use the keyword argument reverse = True:

### **Example**

Sort the list descending:

thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]  
thislist.sort(reverse = True)  
print(thislist)

## List Methods

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

Here -> <https://www.w3schools.com/python/python_lists_methods.asp>

# **Python Dictionaries**

## Dictionary

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.

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

Dictionaries are written with curly brackets, and have keys and values:

### **Example**

Create and print a dictionary:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(thisdict)

Dictionary items are ordered, changeable, and does not allow duplicates.

Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

## Update Dictionary

The update() method will update the dictionary with the items from a given argument. If the item does not exist, the item will be added.

The argument must be a dictionary, or an iterable object with key:value pairs.

### **Example**

Add a color item to the dictionary by using the update() method:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.update({"color": "red"})

## Loop Through a Dictionary

You can loop through a dictionary by using a for loop.

for x, y in thisdict.items():  
  print(x, y)

## Dictionary Methods

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

* <https://www.w3schools.com/python/python_dictionaries_methods.asp>

## Nested Dictionaries

A dictionary can contain dictionaries, this is called nested dictionaries.

myfamily = {  
  "child1" : {  
    "name" : "Emil",  
    "year" : 2004  
  }

}

# **Python Tuples**

## Tuple

Tuples are used to store multiple items in a single variable.

Tuple 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), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

A tuple is a collection which is ordered and **unchangeable**.

Tuples are written with round brackets.

### **Example**

Create a Tuple:

thistuple = ("apple", "banana", "cherry")  
print(thistuple)

## Tuple Items

Tuple items are ordered, unchangeable, and allow duplicate values.

Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

## Tuple Methods

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

* <https://www.w3schools.com/python/python_tuples_methods.asp>

**Update Tuple**

But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

# **Python Sets**

## 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 unordered, unchangeable\*, and unindexed.

### **Example**

Create a Set:

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

## Set Items

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

## Set Methods

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

* <https://www.w3schools.com/python/python_sets_methods.asp>

==========================================================================

# **Python If ... Else**

An "if statement" is written by using the if keyword.

### **Example**

If statement:

a = 33  
b = 200  
if b > a:  
  print("b is greater than a")

**Important :**

[Indentation is very important in Python because interpreter recognizes it and it’s not there just for code formatting styling purpose]

## Elif

The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

### **Example**

a = 33  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")

## Else

The else keyword catches anything which isn't caught by the preceding conditions.

### **Example**

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")  
else:  
  print("a is greater than b")

## Short Hand If

If you have only one statement to execute, you can put it on the same line as the if statement.

### **Example**

One line if statement:

if a > b: print("a is greater than b")

## And

The and keyword is a logical operator, and is used to combine conditional statements: [Both condition should be True]

## Or

The or keyword is a logical operator, and is used to combine conditional statements [Only One Condition should be True]

# Ternary Operator in Python

Syntax :

[on\_true] if [expression] else [on\_false]

**Example :**

# Program to demonstrate conditional operator

a, b = 10, 20

# Copy value of a in min if a < b else copy b

min = a if a < b else b

print(min)

## The pass Statement

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

### **Example**

a = 33  
b = 200  
  
if b > a:  
  pass

## Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

a = 2  
b = 330  
print("A") if a > b else print("B")

# What’s the difference between “is” and “==” in Python?

* == is for value equality. It's used to know if two objects have the same value.
* is is for reference equality. It's used to know if two references refer (or point) to the same object, i.e if they're identical. Two objects are identical if they have the same memory address.

Python Loops

Python has two primitive loop commands:

* while loops
* for loops

## The while Loop

With the while loop we can execute a set of statements as long as a condition is true.

### **Example**

Print i as long as i is less than 6:

i = 1  
while i < 6:  
  print(i)  
  i += 1

## The break Statement

With the break statement we can stop the loop even if the while condition is true:

## The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

## The else Statement

With the else statement we can run a block of code once when the condition no longer is true:

### **Example**

Print a message once the condition is false:

i = 1  
while i < 6:  
  print(i)  
  i += 1  
else:  
  print("i is no longer less than 6")

## Python For Loops

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

### **Example**

Print each fruit in a fruit list:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)

# **Python Functions**

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

## Creating a Function

In Python a function is defined using the def keyword:

### **Example**

def my\_function():  
  print("Hello from a function")

## Calling a Function

To call a function, use the function name followed by parenthesis:

### **Example**

def my\_function():  
  print("Hello from a function")  
  
**my\_function()**

## Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

## Return Values

To let a function return a value, use the return statement

# **Python Lambda**

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

## Syntax

lambda arguments : expression

The expression is executed and the result is returned:

### **Example**

Add 10 to argument a, and return the result:

x = lambda a : a + 10  
print(x(5))

# **Python Classes and Objects**

## Python Classes/Objects

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

## Create a Class

To create a class, use the keyword class:

### **Example**

Create a class named MyClass, with a property named x:

class MyClass:  
  x = 5

## Create Object

Now we can use the class named MyClass to create objects:

### **Example**

Create an object named p1, and print the value of x:

p1 = MyClass()  
print(p1.x)

## The \_\_init\_\_() Function

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

To understand the meaning of classes we have to understand the built-in \_\_init\_\_() function.

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:

### **Example**

Create a class named Person, use the \_\_init\_\_() function to assign values for name and age:

class Person:  
  def \_\_init\_\_(self, name, age):  
    self.name = name  
    self.age = age  
  
p1 = Person("John", 36)  
  
print(p1.name)  
print(p1.age)

## Object Methods

Objects can also contain methods. Methods in objects are functions that belong to the object.

Let us create a method in the Person class:

### **Example**

Insert a function that prints a greeting, and execute it on the p1 object:

class Person:  
  def \_\_init\_\_(self, name, age):  
    self.name = name  
    self.age = age  
  
  def myfunc(self):  
    print("Hello my name is " + self.name)  
  
p1 = Person("John", 36)  
p1.myfunc()

## The self Parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class

### **Example**

Use the words mysillyobject and abc instead of self:

class Person:  
  def \_\_init\_\_(mysillyobject, name, age):  
    mysillyobject.name = name  
    mysillyobject.age = age  
  
  def myfunc(abc):  
    print("Hello my name is " + abc.name)  
  
p1 = Person("John", 36)  
p1.myfunc()

# **Python Inheritance**

## Python Inheritance

Inheritance allows us to define a class that inherits all the methods and properties from another class.

**Parent class** is the class being inherited from, also called base class.

**Child class** is the class that inherits from another class, also called derived class.

### **Example**

Create a class named Person, with firstname and lastname properties, and a printname method:

class Person:  
  def \_\_init\_\_(self, fname, lname):  
    self.firstname = fname  
    self.lastname = lname  
  
  def printname(self):  
    print(self.firstname, self.lastname)

class Student(Person): # Inheritance  
  pass

#Use the Person class to create an object, and then execute the printname method:  
  
x = Person("John", "Doe")  
x.printname()

## Use the super() Function

Python also has a super() function that will make the child class inherit all the methods and properties from its parent:

### **Example**

class Student(Person):  
  def \_\_init\_\_(self, fname, lname):  
    super().\_\_init\_\_(fname, lname)