PYTHON DOCUMENTATION

PYTHON SYLLABUS

INTRODUCTION

- ✓ History
- ✓ Features
- ✓ working with Python
- ✓ Basic Syntax
- ✓ Variable and Data Types
- ✓ Operator

Input-Output

- ✓ Printing on screen
- ✓ Reading data from keyboard

Conditional Statements

- ✓ If
- ✓ If- else

- ✓ Nested if-else

Looping

- ✓ For
- ✓ While
- ✓ Nested loops

Control Statements

- ✓ Break
- ✓ Continue
- ✓ Pass

String Manipulation

- ✓ Accessing Strings
- ✓ Basic Operations
- ✓ String slices
- ✓ Function and Methods

Lists

- ✓ Introduction

- ✓ Accessing list
- ✓ Operations
- ✓ Working with lists
- ✓ Function and Methods

Tuple

- ✓ Introduction
- ✓ Accessing tuples
- ✓ Operations
- ✓ Working
- ✓ Functions and Methods

Dictionaries

- ✓ Introduction
- ✓ Accessing values in dictionaries
- ✓ working with dictionaries
- ✓ Properties
- ✓ Functions

Functions

- ✓ Defining a function

- ✓ Calling a function
- ✓ Types of functions
- ✓ Function Arguments
- ✓ Anonymous functions
- ✓ Global and local variables

Modules

- ✓ importing module

Exception Handling

- ✓ Exception
- ✓ Exception handling
- ✓ except clause

File handling

- ✔File write
- ✔File read
- ✔File create
- ✔File write

Directory management

- ✔Directory create
- ✔Directory delete
- ✔Directory move

MULTI THREADING

- ✔ Execute thread

DATE AND TIME

- ✔ Read date and time

VOICE

- ✔ Text to speech

SPEAKER

- ✔ Play sound

USB STORAGE

- ✔ Store data s in pendrive or external harddisk

SERIAL

- ✔ Send data via Serial port

BOOT

- ✔ Auto run

OPENCV

- ✔ Image write
- ✔ Video write
- ✔ Live video stream
- ✔ Face detection
- ✔ Color detection
- ✔ Object detection

INTRODUCTION

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

It is used for:

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

1. Web development – Web framework like Django and Flask are based on Python. They help you write server side code which helps you manage database, write backend programming logic, mapping urls etc.

2. Machine learning – There are many machine learning applications written in Python. Machine learning is a way to write a logic so that a

machine can learn and solve a particular problem on its own. For example, products recommendation in websites like Amazon, Flipkart, eBay etc. is a machine learning algorithm that recognises user's interest. Face recognition and Voice recognition in your phone is another example of machine learning.

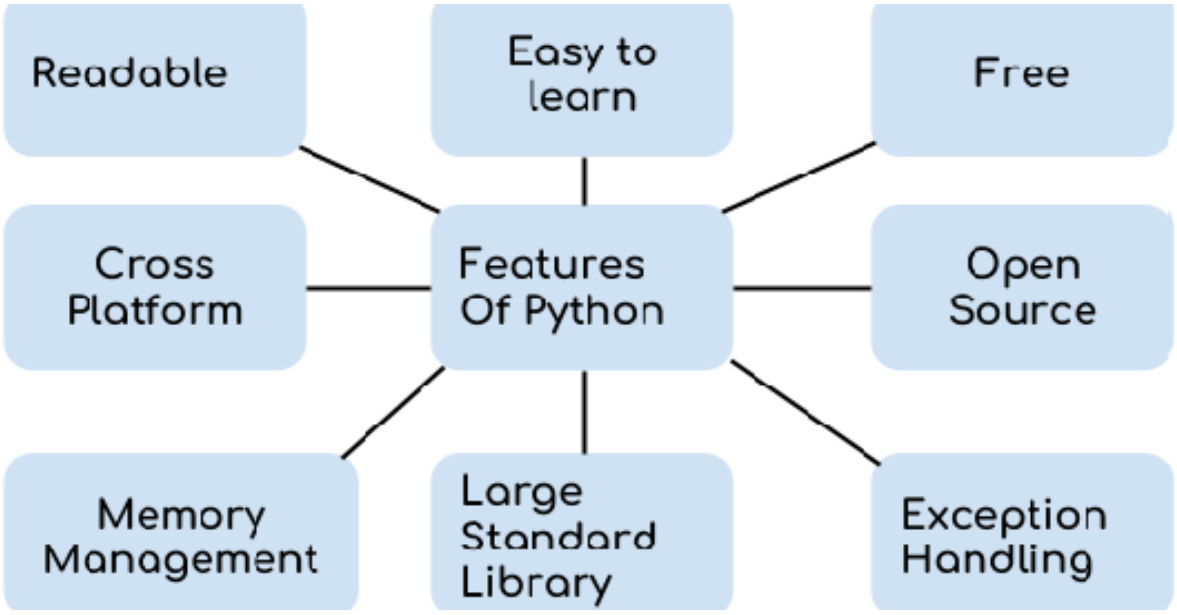
3. Data Analysis – Data analysis and data visualisation in form of charts can also be developed using Python.

4. Scripting – Scripting is writing small programs to automate simple tasks such as sending automated response emails etc. Such type of applications can also be written in Python programming language.

5. Game development – You can develop games using Python.

6. You can develop **embedded applications** in Python.

7. **Desktop applications** – You can develop desktop application in Python using library like TKinter or QT.



- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written.

1. **Readable:** Python is a very readable language.

2. **Easy to Learn:** Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn.

3. **Cross platform:** Python is available and can run on various operating systems such as Mac, Windows, Linux, Unix etc. This makes it a cross platform and portable language.

4. **Open Source:** Python is a open source programming language.

5. **Large standard library:** Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.

6. **Free:** Python is free to download and use. This means you can download it for free and use it in your application.

7. **Supports exception handling:** If you are new, you may wonder what is an exception? An exception is an event that can occur during program exception and can disrupt the normal flow of program. Python supports exception handling which means

we can write less error prone code and can test various scenarios that can cause an exception later on.

8. **Advanced features:** Supports generators and list comprehensions. We will cover these features later.

9. **Automatic memory management:** Python supports automatic memory management which means the memory is cleared and freed automatically. You do not have to bother clearing the memory.

PYTHON INSTALLATION

Go to website link

https://www.python.org/downloads/windows/

go to "**Note that Python 3.9.0 cannot be used on Windows 7 or earlier"**

Click Download Windows x86-64 executable installer

BASIC SYNTAX

```
print ("Hello, World!")

x= 5
y= "John"
print(x)
print(y)
```

Basic syntax:

```
print("Hello, World!")
```

VARIABLE AND DATA TYPES:

```
a = 5
print(a, "is of type", type(a))

a = 2.0
print(a, "is of type", type(a))

a = 2.22
print(a, "is of type", type(a))

a = ‘c’
print(a, "is of type", type(a))

a = “welcome”
print(a, "is of type", type(a))
```

OPERATOR

- ✔ Arithmetic operators
- ✔ Assignment operators
- ✔ Comparison operators
- ✔ Logical operators
- ✔ Identity operators
- ✔ Membership operators
- ✔ Bitwise operators

Arithmetic operators

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

Example:

```
x = 5
y = 3
print(x + y)
```

```
x = 5
y = 3
print(x - y)
```

```
x = 5
y = 3
print(x * y)
```

```
x = 12
y = 3
print(x / y)
```

```
x = 5
y = 2
print(x % y)
x = 2
y = 5
```

```
print(x ** y) #same as 2*2*2*2*2
```

```
x = 15

y = 2

print(x // y)
```

Assignment operators

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 Operators

Python Comparison 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

Example:

```
x = 5

y = 3

print(x == y)
```

```
x = 5

y = 3

print(x != y)
```

```
x = 5
```

```
y = 3

print(x > y)
```

```
x = 5

y = 3

print(x < y)
```

```
x = 5

y = 3

print(x >= y)
```

```
x = 5

y = 3

print(x <= y)
```

Python Logical Operators:

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)

```
x = 5
print(x > 3 and x < 10)
```

```
x = 5
print(x > 3 or x < 4)
```

returns True because one of the conditions are true (5 is greater than 3, but 5 is not less than 4)

```
x = 5
print (not(x > 3 and x < 10))
```

returns False because not is used to reverse the result

Python Identity Operators

Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the 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

Using is

```
x = ["apple", "banana"]

y = ["apple", "banana"]

z = x

print(x is z)

# returns True because z is the same object as x

print(x is y)

# returns False because x is not the same object as y, even if they have the same content
```

```
print(x == y)

# to demonstrate the difference betweeen "is" and "==": this comparison returns True because x is equal to y
```

Using is not

```
x = ["apple", "banana"]

y = ["apple", "banana"]

z = x

print(x is not z)

# returns False because z is the same object as x

print(x is not y)

# returns True because x is not the same object as y, even if they have the same content

print(x != y)

# to demonstrate the difference betweeen "is not" and "!=": this comparison returns False because x is equal to y
```

Python Membership Operators

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

```
x = ["apple", "banana"]

y = ["apple", "banana"]

z = x

print(x is z)\

# returns True because z is the same object as x

print(x is y)

# returns False because x is not the same object as y, even if they have the same content

print(x == y)


# to demonstrate the difference betweeen "is" and "==": this comparison returns True because x is equal to y
```

Python bitwise operators

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

```
a = 60            # 60 = 0011 1100
b = 13            # 13 = 0000 1101
c = 0

c = a & b;        # 12 = 0000 1100
print (c)

c = a | b;        # 61 = 0011 1101
print (c)

c = a ^ b;        # 49 = 0011 0001
print (c)

c = ~a;           # -61 = 1100 0011
```

```
print (c)

c = a << 2;       # 240 = 1111 0000
print (c)

c = a >> 2;       # 15 = 0000 1111
print (c)
```

INPUT-OUTPUT

```
a = 5
print('The value of a is', a)
```

```
num = input('Enter a number: ')
Enter a number: 10
```

CONDITIONAL STATEMENTS

Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b

- Greater than: a > b
- Greater than or equal to: a >= b

EXAMPLE

```
a=10
b=20
if(a>b):
        print(“ a is greater”)
else:
        print(“b is greater)



a=5
b=2

if(a > b):
        print( “ a is greater “)
elif(a<b):
        print( “ b is less “)
else:
        print( “ a equal to b )




a=5
b=2
c=3

if( a>b ):
        print(“a is greater “)
        if(c<a):
                print(“ c is less “)
        elif(c>a):
                print(“ a is greater “)
        else:
                print(“ c is equal”)

elif( a<b ):
        print(“a is less    “)

else:
        print(“ a is equal”)
```

```
a = 8
b = 2
if(a!=b):
        print(“true”)
```

LOOPING

```
digits = [0, 1, 5]

for i in digits:
    print(i)
else:
    print("No items left.")


marks = {'James': 90, 'Jules': 55, 'Arthur': 77}

for student in marks:
    if student == student_name:
        print(marks[student])
        break

for i in range (0,5,1):
        print(i)
```

```
c = “I know the human being”

for i in c:
        print(c)

list2 = [“san”,”man”,”van”]

for i in list2:
        print(i)
```

```
while(1):
        print(“hello earth”)


while(0):
        print(“hello earth”)


a=0
while(1):
        a++;
        if(a == 10):
                print(“loop breaked”)
                break

Nested loops:

While(1):
        Print(“welcome to loop”)
        While(1):
                Print(“welcome to nested loop”)
```

CONTROL STATEMENTS (BREAK, CONTINUE, PASS)

```
for I in “hello world”:
        print(I )
        if( I == ‘w’):
                break

for c in "ToolsQA":
  if c == "Q":
    pass
  else:
    print(c)
for c in "ToolsQA":

    if c == "Q":
        continue
    print(c)
```

STRINGS

'hello' is the same as "hello".

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

```
    print("Hello")
    print('Hello')
```

Assign String to a Variable

```
a = "Hello"
print(a)
```

Multiline Strings

```
a = "Hello"
print(a)

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

Print(a)
```

Strings are Arrays

```
a = "Hello, World!"
print(a[1])
```

Looping Through a String

```
a = "Hello, World!"
print(a[1])

for x in "banana":
        print(x)
```

String Length

```
for x in "banana":
        print(x)
print(len(x))



a = "Hello, World!"
         print(len(a))
```

Check String

```


txt = "The best things in life are free!"
if "free" in txt:
   print("Yes, 'free' is present.")
```

Check if NOT

```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

Python - Slicing Strings

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

Upper Case

```
a = "Hello, World!"
print(a.upper())
```

Lower Case

```
b = a = "Hello, World!"
print(a.lower())
```

Replace String

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

Split String

```
a = "Hello, World!"
print(a.split(","))
```

String Concatenation

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

LISTS

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

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

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

List Length

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

List Items - Data Types

```
mylist = ["apple", "banana", "cherry"]
print(type(mylist))
```

Python - Access List Items

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

Range of Indexes

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])
```

CHANGE ITEM VALUE

```
thislist = ["apple", "banana", "cherry"]

thislist[1] = "blackcurrant"

print(thislist)


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

thislist[1:2] = ["blackcurrant", "watermelon"]

print(thislist)
```

Remove Specified Item

```
thislist = ["apple", "banana", "cherry"]

thislist.remove("banana")

print(thislist)

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

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

thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

Loop Through a List

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

```
thislist = ["apple", "banana", "cherry"]
i = 0
while i < len(thislist):
  print(thislist[i])
  i = i + 1
```

Sort List Alphabet order

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

Sort Descending

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

Copy a List

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

Python - Join Lists

```
print(mylist)
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

TUPLE

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

Access Tuple Items

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

Range of Indexes

```
print(thistuple[1])

thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
```

```
print(thistuple[2:5])
```

Change Tuple Values

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"

x = tuple(y)
print(x)
```

Update values

```
thistuple = ("apple", "banana", "cherry")
thistuple.append("orange") # This will raise an error
print(thistuple)
```

Loop Tuple

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

Join Two Tuples

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

PYTHON SETS

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

Length of a Set

```
Print(len(thisset))
```

Access Items

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)


thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)

thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)
```

Remove Item

```
thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)
```

Loop Items

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

Join sets

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)
```

intersection

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.intersection(y)

print(z)
```

PYTHON DICTIONARIES

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

```
thisdict = {
  "username": "Ram",
  "password": "12345",
  "year": 1964
}
print(thisdict)


print(thisdict["username"])

print(len(thisdict))

print(type(thisdict))
```

ACCESS ITEM

x = thisdict["password "]

x = thisdict.get("password ")

get keys and values

x = thisdict.keys()

x = thisdict.values()

x = thisdict.items()

Change Values

thisdict["year"] = 2018

thisdict.update({"year": 2020})

Adding Items

```
thisdict["color"] = "red"
print(thisdict)
```

Removing Items

```
thisdict.pop("password ")

del thisdict["password "]
```

Loop dictionary

```
for x in thisdict:
  print(x)

for x in thisdict.values():
  print(x)

for x in thisdict.keys():
  print(x)

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

Copy Dictionaries

```
mydict = thisdict.copy()
print(mydict)
```

FUNCTIONS

Creating a Function

```
def my_function():
    print("Hello from a function")
```

Types of functions

- ✔Without argument without return type
- ✔Without argument with return type
- ✔With argument without return type
- ✔With argument with return type

WITHOUT ARGUMENT WITHOUT RETURN TYPE

```
def  add():
        a=1
        b=2
        c=a+b
        print(c)
add()
```

WITHOUT ARGUMENT WITH RETURN TYPE

```
def  add():
```

```
        a=1
        b=2
        c=a+b

        return(c)

d = add()
print(c)
```

WITH ARGUMENT WITHOUT RETURN TYPE

```
def  add (a,b):
        c= a+b
        print(c)

 add(8,4)
```

WITH ARGUMENT WITH RETURN TYPE

```
def  add(a,b):
        c=a+b
        return(c)


d = add(8,4)
print(d)
```

PYTHON MODULES

Create a Module To create a module just save the code you want in a file with the file extension .py:

Save below code in a file named `addition.py`

```
def  add(a,b):
        c=a+b
        return(c)
```

Use a Module

```
import   addition

c  = add(5,2)

from addition  import*
```

EXCEPTION HANDLING.

```
import os

try:
        os.makedir(“videos”)
except:
        print(“already exist”)
```

FILE HANDLING

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

```
file1 = open("MyFile.txt","x")

file1.close()



file1 = open("myfile.txt","w")
file1.write("Today \n")
file1.close()

file1 = open("myfile.txt","a")
file1.write("happy day \n")
file1.close()

file1 = open("myfile.txt","r")
print (file1.readlines() )
 file1.close()

file1 = open("myfile.txt","r")
print (file1.read() )
file1.close()
```

DIRECTORY MANAGEMENT

Get Current Directory:

```
import os
os.getcwd()
```

List Directories and Files

```
os.listdir()
```

Making a New Directory

```
os.mkdir('test')
```

Renaming a Directory or a File

```
os.rename('test','new_one')
```

Removing Directory or File

```
os.remove('old.txt')

os.rmdir('new_one')
```

MULTI THREADING

A process is basically the program in execution. When you start an application in your computer (like a browser or text editor), the operating system creates a process.

```
def task1():
    while(True):
        print(“hi”)

def task2():
    while(True):
        print(“lie”)

   def task3():
    while(True):
        print(“lie”)


if __name__ == "__main__":
    try:
        t1 = threading.Thread(target=task1)
    except:
        t1 = threading.Thread(target=task1)
        time.sleep(2)
    try:
        t2 = threading.Thread(target=task2)
    except:
        t2 = threading.Thread(target=task2)
        time.sleep(2)

    t1.start()
    t2.start()
    t1.join()
    t2.join()
    print("Done!")
```

DATE AND TIME

```
#pip install datetime
import datetime
x = datetime.datetime.now()
print(x)
```

TEXT to VOICE

```
pip install text-to-speech

from text_to_speech import speak
speak(str(“hai “), "en")
```

SPEAKER (for linux,raspberry pi)

```
import os
os.system('/usr/bin/omxplayer -o both /home/pi/Desktop/project/1.wav &')
```

SPEAKER (FOR WINDOWS)

```
#Pip  install  playsound

from playsound import playsound
playsound('4.wav')
```

USB DETECT

```
import os

from glob import glob
from subprocess import check_output, CalledProcessError

def get_usb_devices():
    sdb_devices = map(os.path.realpath, glob('/sys/block/sd*'))
    usb_devices = (dev for dev in sdb_devices
        if 'usb' in dev.split('/')[5])
    return dict((os.path.basename(dev), dev) for dev in usb_devices)

#############################################################################
def get_mount_points(devices=None):
    devices = devices or get_usb_devices()  # if devices are None:
get_usb_devices
    output = check_output(['mount']).splitlines()
    output = [tmp.decode('UTF-8') for tmp in output]

    def is_usb(path):
        return any(dev in path for dev in devices)
    usb_info = (line for line in output if is_usb(line.split()[0]))
    return [(info.split()[0], info.split()[2]) for info in usb_info]

#############################################################################

def get_usb_name():

    usb_detect = get_mount_points()
    usb_name = usb_detect
    print(usb_name)
    print(type(usb_detect))
    usb_name_length = len(usb_name)
    print(usb_name_length)

    return(usb_name,usb_name_length)\


#############################################################################
def usb_name_split(usb_name):

    usb_name_get = str(usb_name)

    print(usb_name_get)
```

```
    usb_name_get = map(str.strip,usb_name_get.split(','))
    usb_name_get = list(usb_name_get)

    usb_name_get_length = len(usb_name_get)
    usb_name = usb_name_get[1]
    print(usb_name)


    usb_name = map(str.strip,usb_name.split("'"))

    usb_name = list(usb_name)
    usb_name = usb_name[1]
    print(len(usb_name))
    return (usb_name)

usb_name,length = get_usb_name()

if(length == 0):
        print("pendrive_not_insert")
else:
        drive_name = usb_name_split(usb_name)
        print("drive_name")
        print(drive_name)
```

SERIAL

```
import serial

import time

#pip install pyserial

import serial
import time

#serial initial

ser = serial.Serial(
port ='/dev/ttyS0',
baudrate = 9600,
parity=serial.PARITY_NONE,
stopbits=serial.STOPBITS_ONE,
bytesize=serial.EIGHTBITS,
timeout=1,
rtscts=0
 )
```

```
#serial write

ser.write(b'AT+CMGF=1\r')

time.sleep(0.5)

#serial read

rcv_sms_res = ser.readline().decode().strip('\n').strip('\r')
print(rcv_sms_res)
time.sleep(0.5)

#serial write

ser.write(str.encode('hi \r'))
time.sleep(2)
```

BOOT(LINUX)

```
Method 1: rc.local

Go to terminal and type sudo nano /etc/rc.local and press enter

#Editing rc.local – add lines to end

sudo python /home/pi/sample.py &

exit(0)


Method 2: .bashrc

Go to terminal and type sudo nano /home/pi/.bashrc and press enter

echo Running at boot

sudo python /home/pi/sample.py
```

OPEN CV

```
pip install opencv-python
pip install opencv-contrib-python
```

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human.

```
#Video_show

import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)

while True:
    # Capture frame-by-frame

    ret, frame = cap.read()
    cv.imshow('frame', gray)
    if cv.waitKey(1) == ord('q'):
        break

cap.release()
cv.destroyAllWindows()


#Playing Video from file

import numpy as np
import cv2 as cv
cap = cv.VideoCapture('vtest.avi')

while cap.isOpened():

    ret, frame = cap.read()
    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    cv.imshow('frame', gray)
    if cv.waitKey(1) == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

```
#Saving a Video
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)

# Define the codec and create VideoWriter object

fourcc = cv.VideoWriter_fourcc(*'XVID')
out = cv.VideoWriter('output.avi', fourcc, 20.0, (640,  480))

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    out.write(frame)

    cv.imshow('frame', frame)

    if cv.waitKey(1) == ord('q'):
        break

# Release everything if job is finished
cap.release()
out.release()
cv.destroyAllWindows()
```