Python is a high-level, general-purpose, and very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry.

PRINT:

File Edit Format Run Options Window Help

```
print("Sundarraj")
```

```
============= RESTART: C:\Users\SUMAHALI\Downloads\Python\Print.py =========
Sundarraj
>>>
```

VARIABLES:

```
a=10
b=20
c=a+b
print(c)
```

```
=========== RESTART: C:\Users\SUMAHALI\Downloads\Python\Variables.py =========
30
>>>
```

STRING:

```
name="GoldFish"
print(name[0:4])     # print index from 0 to 3
```

```
========= RESTART: C:/Users/SUMAHALI/Downloads/Python/String.py ======
Gold
```

```
name="SundarRaj"
print(name[6:])      # print from index 6
```

```
Raj
```

```
name="SundarRaj"
print(name[:6])      # print from 0 to 5 / print upto index 5
```

```
Sundar
```

```
name="Sundarraj"
greeting="Good Morning"
print("Hello {}, {} How are you".format(name, greeting))     # String formatting
```

```
Hello Sundarraj, Good Morning How are you
```

CASTING:

```
a="10"  # 10 is stored as a string
b="40"  # 40 is stored as a string
c=a+b
print("withoutCasting: "+ c)

d=int("10") # convert a string into integer
e=int("40")
f=d+e
print("With Casting: ",f)
```

```
>>>
============ RESTART: C:\Users\SUMAHALI\Downloads\Python\Casting.py ============
withoutCasting: 1040
With Casting:  50
>>>
```

USER INPUT:

```
a=input() # input automatically take the input as string
b=input() # for example a =10, b=20
c=a+b     # then c = 1020
print(c)

#you have to convert the str into int

d =int(input())
e =int(input())
f=d+e
print(f)
```

```
====== RESTART: C:\Users\SUMAHALI\Downloads\Python\Get Input from User.py ====
12
12
1212
33
12
45
>>>
```

USER INPUT INDICATION:

```
a=int(input("Enter value for A "))  # sentence inside the input is displayed in output page
b=int(input("Enter value for B "))
c=a+b
print(a,"+", b, " = ", c)   # , is used to add the another element in print statement
```

```
====== RESTART: C:\Users\SUMAHALI\Downloads\Python\UserInputIndication.py ====
Enter value for A 40
Enter value for B 23
40 + 23  =  63
>>>
```

IF – ELSE:

```
name="Sundarraj"

if(name=="Sundarraj"):
    print("He is a Tester")
else:
    print("He is not a Tester")
```

```
============= RESTART: C:\Users\SUMAHALI\Downloads\Python\IfElse.py ==========
He is a Tester
>>>
```

ELIF:

```
mark = int(input("Enter your Mark: "))
if(mark<35):
    print("Fail")
elif(mark>35 and mark<70):
    print("Pass")
else:
    print("First Class")
```

```
============== RESTART: C:\Users\SUMAHALI\Downloads\Python\Elif.py ========
Enter your Mark: 70
First Class
>>>
```

AND OPERATOR:

```
num=int(input("Enter a Number: "))

if(num%3==0 and num%5==0):
    print("The number is divisible by 3 and 5")
else:
    print("The number is Not divisible by 3 and 5")
```

```
========== RESTART: C:\Users\SUMAHALI\Downloads\Python\AndOperator.py ========
Enter a Number: 44
The number is Not divisible by 3 and 5
>>>
```

OR OPERATOR:

```
age=int(input("Enter your Age: "))
salary=int(input("Enter your Salary: "))

if(age>40 or salary>45000):
    print("You are eligible for Loan")
else:
    print("You are not eligible for Loan")
```

```
=========== RESTART: C:\Users\SUMAHALI\Downloads\Python\OrOperator.py ======
Enter your Age: 21
Enter your Salary: 26000
You are not eligible for Loan
>>>
```

FOR LOOP:

```
#simple for loop

for i in "Sundarraj":
    print(i)
```

```
============ RESTART: C:\Users\SUMAHALI\Downloads\Python\ForLoop.py ==========
S
u
n
d
a
r
r
a
j
```

```
#for loop in range

print() #this is to leave a blank line between output

for j in range(5):
    print(j)
```

```

0
1
2
3
4
```

```
for k in range(1,10):   # it is starts from 1 and end at 9 (not 10)
    print(k)
```

```
1
2
3
4
5
6
7
8
9
```

NESTED FOR LOOP:

```
for i in range(6):
    for j in range(i):
        print("*", end="")  # end="" --> not move the cursor to next line
    print()
```

```
========= RESTART: C:\Users\SUMAHALI\Downloads\Python\NestedForLoop.py =======

*
**
***
****
*****
```

COMMENTS:

```
# single line comment

# This is
# Multi line
# Comment
```

FOR EACH LOOP:

```
names=["Sundarraj", "Karthik", "Anjith", "Prem", "pavan"]   # list

for name in names:  # for each loop, name -- variable
    print("Hello " + name)
```

```
========== RESTART: C:/Users/SUMAHALI/Downloads/Python/ForEachLoop.py =========
Hello Sundarraj
Hello Karthik
Hello Anjith
Hello Prem
Hello pavan
```

WHILE LOOP:

```
i=1
while(i<=10):
    print(i)
    i+=1
```

```
======= RESTART: C:/Users/SUMAHALI/Downloads/Python/WhileLoop.py =====
1
2
3
4
5
6
7
8
9
10
>>>
```

COLLECTION:

LIST:

```
a=[1,2,3,4,5]
a.append(6) #append() -- add the element in the list
print("append is used",a)

print()
#-------------------------------------------------------------------------

b=[9,8,7,6,5]
b.insert(1,20)  #insert(0,20) -- insert the value 20 at index 1
print("insert is used",b)

print()
#-------------------------------------------------------------------------

c=[12,23,43,56,76]
c[1]=5  # Replace the element at index 1 with 5
print("replace is used",c)

print()
#-------------------------------------------------------------------------

d=[1,4,7,9,2]
d.pop(2)    # pop(2) -- remove the element at the index 2
print("pop is used",d)

print()
#-------------------------------------------------------------------------

e=[1,2,3,4,5]
f=[6,7,8,9,10]
e.extend(f) # extend() -- join the list f with list e
print("extend is used",e)
```

```
===== RESTART: C:/Users/SUMAHALI/Downloads/Python/CollectionList.py ====
append is used [1, 2, 3, 4, 5, 6]

insert is used [9, 20, 8, 7, 6, 5]

replace is used [12, 5, 43, 56, 76]

pop is used [1, 4, 9, 2]

extend is used [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>>
```

TUPLE:

```
# Allow duplicates
# Any type of data can be stored
# We cannot modify the tuple item -- we cannot add or remove

a=(1,2,3,4,5)
print(a)

print()
#-----------------------------------------------------------------

b=(10,20,30,40,50)
c=list(b)    # convert tuple into list
# now we can add, insert, extend and pop with list c
print("Tuple is converted into list",c)
```

```
========= RESTART: C:/Users/SUMAHALI/Downloads/Python/Tuple.py =======
(1, 2, 3, 4, 5)

Tuple is converted into list [10, 20, 30, 40, 50]
>>>
```

SET:

```
# Do not allow duplicate, Duplicate values will be remove.
# Any type of data can be stored
# We cannot modify the set item but we can add or remove items
# Sets are unordered
# add(), update(), remove(), pop()

a={1,2,3,4,1}
print("Duplicate value is removed", a)

print()
#-------------------------------------------------------------------------

b={10,20,30,40}
b.add(50)    # element 50 is added to the set
print("Element is added", b)

print()
#-------------------------------------------------------------------------

b.remove(30) # element 30 is removed
print("Element is removed",b)
```

```
========== RESTART: C:/Users/SUMAHALI/Downloads/Python/Set.py ==========
Duplicate value is removed {1, 2, 3, 4}

Element is added {40, 10, 50, 20, 30}

Element is removed {40, 10, 50, 20}
>>>
```

DICTIONARY:

```
# Do not allow duplicate
# Any type of data can be stored
# Key:Value pair

a={"Name":"sundarraj",
   "Age":22,
   "location":"Karur"
   }
print(a)

print()
#-------------------------------------------------------------------------

print(a.keys()) # print the keys

print()
#-------------------------------------------------------------------------

print(a.values())    # print the values

print()
#-------------------------------------------------------------------------

a["location"]="Chennai" #update the values
print(a)

    # Or
a.update({"location":"bangalore"})
print()
print(a)

print()
#-------------------------------------------------------------------------

a.pop("location")    #remove the key and value
print(a)
        #Or
print()
del a["Age"]
print(a)


print()
#-------------------------------------------------------------------------

a.clear()
print(a)
```

```
======= RESTART: C:/Users/SUMAHALI/Downloads/Python/Dictionary.py ====
{'Name': 'sundarraj', 'Age': 22, 'location': 'Karur'}

dict_keys(['Name', 'Age', 'location'])

dict_values(['sundarraj', 22, 'Karur'])

{'Name': 'sundarraj', 'Age': 22, 'location': 'Chennai'}

{'Name': 'sundarraj', 'Age': 22, 'location': 'bangalore'}

{'Name': 'sundarraj', 'Age': 22}

{'Name': 'sundarraj'}

{}
>>>
```

FUNCTION:

```
# function syntax
def add():  #function
    a=int(input("Enter the value for A: "))
    b=int(input("Enter the value for B: "))
    print("the addition of A and B is: ", a+b)

add()   #calling the function

print()
#-----------------------------------------------------------------
#function with argument and parameter

def greeting(name): # name -- variable
    print("Good Morning",name)

greeting("Sundarraj")   # passing a string

print()
#-----------------------------------------------------------------

def oddOrEven(num):
    if(num%2==0):
        print("Even")
    else:
        print("Odd")

oddOrEven(10)
```

```
======== RESTART: C:/Users/SUMAHALI/Downloads/Python/Function.py ======
Enter the value for A: 22
Enter the value for B: 3
the addition of A and B is:  25

Good Morning Sundarraj

Even
```

RETURN STATEMENT:

```
def greeting():
    return "Hello Sundarraj"

greeting()  #   This will not print the statement inside the method

print(greeting())   # This will print
```

```
==== RESTART: C:/Users/SUMAHALI/Downloads/Python/ReturnStatement.py ===
Hello Sundarraj
```

CLASS AND OBJECT:

```
class calculation:  # class
    num1=3  #variables
    num2=5
    def add():  #function
        print("Addition a+b")
    def sub():
        print("Subtraction a-b")

a=calculation   # a -- object of class calculation
s=calculation   # s -- object of class calculation

a.add()     # invoke add function using object a
s.sub()     # invoke sub fucction using object s
```

```
===== RESTART: C:/Users/SUMAHALI/Downloads/Python/ClassAndObject.py ===
Addition a+b
Subtraction a-b
```

```
class mobile:   # class
    price=""    # variables
    size=""
    ram=""

samsung=mobile()    # objects
vivo=mobile()
oppo=mobile()

samsung.price="25,000"  # set values to variables
samsung.size="6 inch"
samsung.ram="8GB"

vivo.price="20,000"
vivo.size="5.7 inch"
vivo.ram="6GB"

oppo.price="23,000"
oppo.size="5.5 inch"
oppo.ram="12GB"

print("Samsung Price:",samsung.price)
print("Vivo Size:", vivo.size)
print("Oppo RAM:", oppo.ram)
```

```
Samsung Price: 25,000
Vivo Size: 5.7 inch
Oppo RAM: 12GB
>>>
```

CONSTRUCTOR:

```
# def __init__(self): -- syntax of constructor

class laptop:
    def __init__(self): # constructor
        # self represent the curren object, like "this" keyword in java
        self.price=0
        self.ram=""

    def info(self): #function
        print("Price:", self.price)
        print("RAM:", self.ram)

dell=laptop()   # object

dell.price=50000
dell.ram="8GB"

dell.info()
```

```
====== RESTART: C:/Users/SUMAHALI/Downloads/Python/Constructor.py =====
Price: 50000
RAM: 8GB
```

```
class animal:
    def __init__(self, color):  # constructor with parameter "color"
        self.color=color
    def display(self):
        print("Color of animal is:", self.color)

horse=animal("White")   # passing parameter
lion=animal("Orange")
dog=animal("Black")

horse.display()
lion.display()
dog.display()
```

```
Color of animal is: White
Color of animal is: Orange
Color of animal is: Black
>>>
```

INSTANCE VARIABLE:

```
# Variable which represent the curren objectr is called Instance variable
# variable used inside constructor is called instance variable

class laptop:
    def __init__(self,price,ram): # constructor
        self.price=price    # Instance variable
        self.ram=ram

    def info(self): #function
        print("Price:", self.price)
        print("RAM:", self.ram)

dell=laptop(65000, "8GB")   # object
hp=laptop(55000,"16GB")

dell.info()
hp.info()
```

```
==== RESTART: C:/Users/SUMAHALI/Downloads/Python/InstanceVariable.py ===
Price: 65000
RAM: 8GB
Price: 55000
RAM: 16GB
```

CLASS VARIABLE:

```
# Variable inside the class is called class variable
# variable is common to all the object the class

class laptop:

    displaySize="15.6 inch" # class variable

    def __init__(self,price,ram):
        self.price=price
        self.ram=ram

    def info(self):
        print("Price:", self.price)
        print("RAM:", self.ram)
        print("Display",self.displaySize)

lenovo=laptop(60000, "16GB")
acer=laptop(45000,"8GB")

lenovo.info()
print()
acer.info()
```

```
===== RESTART: C:/Users/SUMAHALI/Downloads/Python/ClassVariable.py ====
Price: 60000
RAM: 16GB
Display 15.6 inch

Price: 45000
RAM: 8GB
Display 15.6 inch
```

INSTANCE FUNCTION:

```
# whenever we use the self keyword inside the function is called instance method

class Car:

    fuel="petrol"

    def __init__(self):
        self.brand=""
        self.airbag=4

    def setAirBag(self,airbag):     # instance method
        self.airbag=airbag

    def getAirBag(self):
        print(self.airbag)

tata=Car()

tata.setAirBag(6)
tata.getAirBag()
```

```
===== RESTART: C:/Users/SUMAHALI/Downloads/Python/InstanceMethod.py ====
6
```

CLASS FUNCTION:

```
# Class method comes with cls argument

class Car:

    fuel="petrol"

    def __init__(self):
        self.brand=""
        self.airbag=4

    @classmethod
    def changeFuel(cls):    # class method
        cls.fuel="electric power"   # change fuel type from petrol to electric power
        print(cls.fuel)


Car.changeFuel()
```

```
====== RESTART: C:/Users/SUMAHALI/Downloads/Python/ClassMethod.py =====
electric power
```

STATIC METHOD:

```
# Method which won't use instance and class variables is called Static Method

class Car:

    fuel="petrol"

    def __init__(self):
        self.brand=""
        self.airbag=4

    @staticmethod       # decorator
    def info():         # Static Method
        print("This is the example of Static Method")

ford=Car()
ford.info()
```

```
====== RESTART: C:/Users/SUMAHALI/Downloads/Python/StaticMethod.py ===
This is the example of Static Method
```

SINGLE LEVEL INHERITANCE:

```
class testing:  # super class
    def info(self):
        print("To ensure the quality of the product")

class apiTesting(testing):  # sub class / api class inherits testing class
    def uses(self):
        print("To test the api")

a1=apiTesting() # create object of apiTesting class
a1.uses()   # call function inside the apiTesting class
a1.info()   # call function inside the testing class
```

```
=== RESTART: C:/Users/SUMAHALI/Downloads/Python/SingleInheritance.py ==
To test the api
To ensure the quality of the product
```

MULTIPLE INHERITANCE:

```
class mobile:
    def smartphone(self):
        print("We can call")

class computer:
    def laptop(self):
        print("we can edit")

class tablet(mobile, computer): # tablet class inherits both mobile and computer class
    def spec(self):
        print("compact to use")


tab=tablet()         # object of tablet class
tab.spec()           # call function inside tablet class
tab.smartphone()     # call function inside mobile class
tab.laptop()         # call function inside computer class
```

```
== RESTART: C:/Users/SUMAHALI/Downloads/Python/MultipleInheritance.py ==
compact to use
We can call
we can edit
```

MULTI LEVEL INHERITANCE:

```
class Principal:
    def dutyOfPrincipal(self):
        print("Report to Management")

class HoD(Principal):   # inherits Principal class
    def dutyOfHoD(self):
        print("Report to Principal")

class Professor(HoD):   # inherits HoD class
    def dutyOfProfessor(self):
        print("Report to HoD")

class Student(Professor):   # inherits Professor class
    def contact(self):
        print("Student can contact professor, HoD as well as Principal")

s1=Student()             # object of Student class

s1.contact()             # function inside Student class
s1.dutyOfProfessor()     # function inside Professor class
s1.dutyOfHoD()           # function inside HoD class
s1.dutyOfPrincipal()     # function inside Principal class
```

```
= RESTART: C:/Users/SUMAHALI/Downloads/Python/MultiLevelInheritance.py =
Student can contact professor, HoD as well as Principal
Report to HoD
Report to Principal
Report to Management
```

HIERARCHICAL INHERITANCE:

```
# all sub classes inherits same super class

class college:
    def purpose(self):
        print("Going to study")

class Student1(college):    # inherits College class
    def std1(self):
        print("Student 1")

class Student2(college):    # inherits College class
    def std2(self):
        print("Student 2")

class Student3(college):    # inherits College class
    def std3(self):
        print("Student 3")

s1=Student1()
s1.purpose()

s2=Student2()
s2.purpose()

s3=Student3()
s3.purpose()
```

```
= RESTART: C:/Users/SUMAHALI/Downloads/Python/HierarchicalInheritance.py
Going to study
Going to study
Going to study
```

SUPER KEYWORD:

```
# super keyword is used to access the function or variable in inherited another class
class A:

    def Greeting(self):
        print("Hello A")

class B(A):     # class B inherits A

    def Greeting(self):
        super().Greeting()  # super Keyword
        print("Hello B")

b1=B()      # object of class B
b1.Greeting()
```

```
====== RESTART: C:/Users/SUMAHALI/Downloads/Python/SuperKeyWord.py ===
Hello A
Hello B
```

ENCAPSULATION:

```
#<<<<<<<<<  ACCESS MODIFIERS <<<<<<<<<

# __ double underscore denotes "PRIVATE"
# _  single underscore denotes "PROTECTED"
# Without any underscore denotes "PUBLIC"

class Company:

    __companyName="Capgemini"   # Private variable

    def CompanyName(self):  # public function to access the private variable
        print(self.__companyName)

c1=Company()
c1.CompanyName()
```

```
===== RESTART: C:/Users/SUMAHALI/Downloads/Python/Encapsulation.py ====
Capgemini
```

EXCEPTION HANDLING:

```
try:    # try block
    a=int(input())
    b=int(input())
    print(a+b)
except Exception as e:  # e to show the actual mistake
    print("Give only Integer Value",e)
```

```
=== RESTART: C:/Users/SUMAHALI/Downloads/Python/ExceptionHandling.py ==
11
hii
Give only Integer Value invalid literal for int() with base 10: 'hii'
```

```
try:    # try block
    a=int(input())
    b=int(input())
    print(a+b)

except ValueError as e:  # value error (Int, str, char)
    print("Give only Integer Value",e)

except TypeError as e:  # type error (+, -, *, /)
    print("Check the operation",e)
```

```
== RESTART: C:/Users/SUMAHALI/Downloads/Python/DifferentExceptions.py ==
11
hii
Give only Integer Value invalid literal for int() with base 10: 'hii'
```

```
try:    # try block
    a=int(input())
    b=int(input())
    print(a+b)

except ValueError as e:  # value error (Int, str, char)
    print("Give only Integer Value",e)

except TypeError as e:  # type error (+, -, *, /)
    print("Check the operation",e)

except Exception as e:  # It handle all types of error
    print("Check the operation",e)

finally:    # It will execute even exception was occurred or not
    print("It will execute even exception was occurred or not")
```

```
=== RESTART: C:/Users/SUMAHALI/Downloads/Python/FinallyInException.py ==
12
22
34
It will execute even exception was occurred or not
>>>
=== RESTART: C:/Users/SUMAHALI/Downloads/Python/FinallyInException.py ==
43
Hii
Give only Integer Value invalid literal for int() with base 10: 'Hii'
It will execute even exception was occurred or not
```

FILE HANDLING:

```
# place the text file and this python file in same location

file=open("FileHandling.txt")   # open() -- open the file
content=file.read()     # read() -- read the file
print(content)      # print the content present in the file
```

```
====== RESTART: C:/Users/SUMAHALI/Downloads/Python/FileHandling.py =====
Sundarraj
Software Engineer
```

```
# write() -- replace the content already present in the file
# close() the file after wrote the file
# place the text file and this python file in same location

file=open("FileHandling.txt","w")   # w -- write mode
content=file.write("Capgemini")
file.close()


# after closing the file you should open to read the updated file
file=open("FileHandling.txt","r")   # r -- read mode
content=file.read()
print(content)
```

```
======= RESTART: C:/Users/SUMAHALI/Downloads/Python/WriteFile.py =====
Capgemini
```

```
# r+ -- we can read or write, no need to change manually w & r
file=open("FileHandling.txt","r+")  # r+ -- read or write mode
content=file.read()
print(content)
```

```
# a -- append with already existing data, write() not replace the old content
file=open("FileHandling.txt","a")  # a -- append with already exixting
file.write("Sundarraj\n")
file.write("Software Engineer")
file.close()
file=open("FileHandling.txt","r")   # r -- read mode
c=file.read()

print(c)
```

```
    Capgemini
    Sundarraj
    Software Engineer
>>>
```

```
# readLine() -- print a single line in the file
```