PYTHON:

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.
* Python file saved in the extension of .py

Ex: basics.py (basics is the filename) (py is extension file)

* Print statement is :

print(“welcome to python”);

* Comments:

# print(“hello”) (single line comment)

“”” print (“hello”)

Print(“world”) “”” (multi line comment)

* Variable:

these are used for storing data values

1. Ex: x = 5

y = "John"

z=’ john’

**for output:**

print(x) # 5

print(y) # john (In String denotation double quotes and single

quotes are gives same output)

print (z) # john

**for get type:**

print (type(x)) # <class,int>

1. Ex :x, y, z = "Orange", ‘Banana’, 5

print(x) # orange

print(type(x)) #<class,str>  
print(y) # banana  
print(z) #5

* Global Variable:

the variable can be used every where,both inside and outside the functions

EX 1: x="python"

def fun():

print ("welcome to " + x) # (welcome to python)

fun()

Ex 2: x="easy" (Global variable)

def fun():

x="python" (Local variable)

print ("welcome to" + x) # (welcome to python)

fun()

print ("python is" + x) #( python is easy)

Note: its calls local variable first after that calls global variable

* DATA Types:

The variables stores the different data types they are

|  |
| --- |
|  |
| 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 |

|  |  |
| --- | --- |
| DATA TYPE | EXAMPLE |
| STRING | X=”python” |
| INT | X=10 |
| FLOAT | X=10.5 |
| COMPLEX | X=1+1j |
| LIST | X=[“mom”,’dad’,”bro”] |
| TUPLE | X=(“mom”,”dad”,”bro”) |
| SET | X={“mom”,”dad”,”bro”} |
| DICTIONARY | X={“name”:”prassu”,”age”:22} |
| RANGE | Range(6) |
| FROZEN SET | X=frozenset ({"apple", "banana", "cherry"}) |
| BOOL | X=true |
| BYTES | X=b”hello” |
| NONE | X=none |

CASTING: Covert into specific type

1. x= int(2) #2
2. x= int(2.5) #2
3. x= float(3.6) #3.6
4. x= float(2) # 2.0
5. x=str(“prassu”) # prassu

STRING: It is a sequence of characters

1.slicing

2.modify

3.conatenate

4.format

5.string methods

1.slicing:

x = "welcome to python"

print(x[:7]) # welcome

print(x[12:17]) #python

2.modify:

\* UPPER CASE(convert into uppercase)

x=”hello”

Print(x.upper()) # HELLO

\*LOWER CASE(convert into lowercase)

x=”HELLO”

Print(x.lower()) # hello

\*Strip method( it is used to remove the whitespace before or end of the sentence)

X= “ welcome to python “

Print(x.strip()) #welcome to python

\*Replace method (replace the value)

x=”Happy”

print(x.replace(“H”, “h”))

3. concatenate : ( combination of two Strings)

X=”prasanna”

Y= “ kumar”

Z=x+y

Print(Z) # prasanna kumar

* BOOLEAN: Booleans represent one of two values: True or False.

EX:

1. print(10>9) #TRUE
2. print(10<9) #FALSE
3. def myFunction() :

return True

if myFunction():

print("YES!")

else:

print("NO!") #YES

* OPERATORS:

These are used to perform arthematic operations on variables and values.

1.Arthematic operators

2. Assignment operators

3.Comparison operators

4.Logical operators

5.Identity operators

6.Membership operators

7.Bitwise operators

1 Arthematic operators:

|  |  |
| --- | --- |
| **operators** | **example** |
| addition | x+y |
| subtraction | x-y |
| multiplication | x\*y |
| Division | x/y(Quotient) |
| Modulus | x%y (Remainder) |
| Exponential | X\*\*y |
| Floor division | x//y |

2. Assignment operators:

|  |  |
| --- | --- |
| OPERATORS | EXAMPLE |
| = | X=5 |
| += | X+=5 x=x+5 |
| -= | x-=5 x=x-5 |
| \*= | X\*=5 x=x\*5 |
| %= | X%=5 x=x%5 |
| /= | x/=5 x=x/5 |
| //= | x//=5 x=x//5 |
| \*\*= | X\*\*=5 x=x\*\*5 |
| &= | X&=5 x=x&5 (AND) |
| |= | X|=5 x=x|5 (OR) |
| ^= | X^=5 x=x^5 (X-OR) |
| >>= | x>>=5 x=x>>5 |
| <<= | X<<=5 x=x<<5 |

3. Comparision operator:

|  |  |
| --- | --- |
| OPERATORS | EXAMPLES |
| == | X==Y |
| != | X!=Y |
| > | X>Y |
| < | X<Y |
| >= | X>=Y |
| <= | X<=Y |

4. Logical operators:

|  |  |
| --- | --- |
| OPERATORS | EXAMPLES |
| and | Returns true if both statements are true  X<5 & x<6 (x=3) |
| or | Returns true if any one of the statements is true  X<5 & x>9 (x=3) |
| not | Reverse the result,if input is true then result is false |

List:

1. Lists are used to store multiple items in a single variable.
2. These are Written in a square brackets only []

Ex 1: Accessing List

1)list =[“mom”,”dad”,2]

Index[ 0,1,2]

Print(list) # mom,dad,2

Print(list[0]) # mom

Print (list[-1]) # 2

print(list[0:]) #mom,dad,2

print(list[1:]) # dad,2

print(list[0:2]) # mom,dad

note: search start at 0(included) and end at index2(not included)

Ex2:

Q: newone=[“mom”,”dad”,2,”sis”,”prassu”,”prem”]

A: newone[2]=”bro” (for change single value)

print (newone) # mom,dad,bro,sis,prassu,prem

newone[0:2]=”suryam”,”ram” (for change multiple values)

print(newone) # suryam,ram,2,prassu,prem

newone.append(“sai”) ( for add items to list)

print(newone) # mom,dad,2,sis,prassu,prem,sai

newone.insert(2,”sai”) (for change item at particular index)

print(newone) # mom,dad,sai,sis,prassu prem

newone.remove (2) (for remove value in the list)

print (newone) # mom,dad,sis,prassu,prem

newone.pop( 0) (for removeing the value at index value 0)

print(newone) #dad,sis,prassu,prem

Ex3: (print the list one by one)

Q: list=[“roshini”,”rohini”,”sai”,”swamy”]

Using for loop

1. For i in list:

Print(i) # roshini

Rohini

Sai

Swamy

1. [print[i] for i in list]

using while loop

while i < len(list):

print (list[i])

i=i+1

Ex4: (list comprehension)

1. list=[“roshini”,”rohini”,”sai”]

newlist=[]

for i in list:

if “h” in i:

newlist.append(i) (for the values which having “h” those

values are written )

print(newlist) #roshini,rohini

2.newlist=[i in i list if “h” in i]

print(newlist) #roshini,rohini

3.newlist=list.copy()

Print(newlist) #roshini,rohini,sai

Ex5: sort the values

list=[23,20,29,18,5]

list.sort() ( for ascending order)

Print(list) #5,18,20,23,29

list=[23,20,29,18,5]

list.sort(reverse= True) ( for descending order)

Print(list) # 29,23,20,18,5

Ex6: joining of two lists

list 1=[“a”,”b”,”c”]

list 2=[1,2,3]

list 3=list 1+list 2

print(list 3) # a,b,c,1,2,3

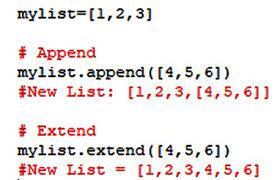
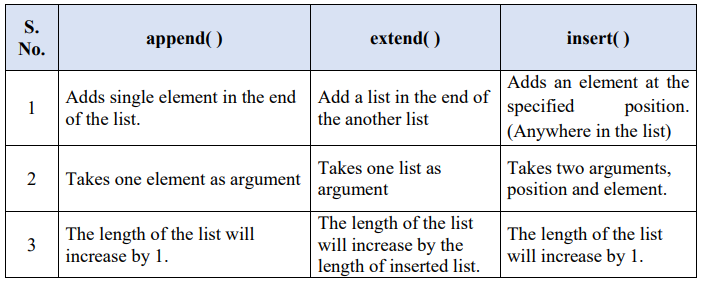
for x in list2;

list1.append(x)

print(list1) #a,b,c,1,2,3

list1.extend(list2)

print(list1) #a,b,c,1,2,3

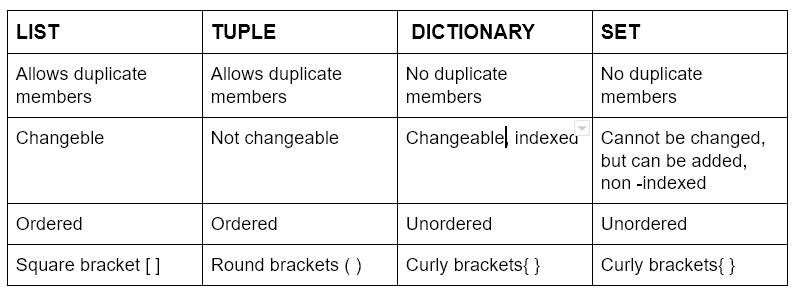


Types of methods in list :

append(),clear(),copy(),count(),extend(),index(),insert(),pop(),remove(),reverse(),sort().

Types of methods in Tuples:

Count(),index()



DICTIONARY:

It is pair of keys and values

Methods in dictionary:

Clear(),copy(),fromkeys(),get(),items(),keys(),values(),pop(),popitem(),setdefault(),update()

Ex 1: list={

“name” :”prassu”,

“age” :22

“designation” : “GET”

}

Print(list)

#(“name”:”prassu”,”age”:22,”designation”:”GET”)

X=list(“name”)

Print(x) # “prassu”

X=list.get(“name”)

Print(x) #”prassu”

X=list.keys()

Print(x) # “name”,”age”,”designation”

X=list.values()

Print(x) # “prassu”,22,”GET”

list[“age”]=24

Or

List.update({“age”:24})

Print(list) #(“name”:”prassu”,”age”:24,”designation”:”GET”)

List.pop(“age”)

Print(list) #(“name”:”prassu”,”designation”:”GET”)

X=list.items()

Print(x) #(“name”:”prassu”,”age”:24,”designation”:”GET”)

LOOPS: a loop is a sequence of instructions that is continually repeated until a certain condition is reached.

1.If loop

Ex:

1. (if condition)

A=20

B=30

If (A<B)

Print(“a is lessthan b “)

1. (if else cond)

A=20

B=30

If(A>B):

Print(“a is greaterthan b”)

Else:

Print(“less”)

1. (if –elif-else cond)

A=20

B=30

If(A<B):

Print(“a is lessthan b”)

Elif(A==B):

Print(“a isequal to b”)

Else:

Print(“b is greaterthan a”)

2.while loop

i=1  
while i<4:  
 print(i)  
 i+=1

output:

1

2

3

3. for loop: for loop is used for iterating over a sequence

names=["prassu","prem","suryam","ram"]  
for i in names:  
 print(i)

Ex2: break loop is used for break the stament after iteration passed

names=["prassu","prem","suryam","ram"]  
for i in names:  
 if i=="suryam":  
 break  
 print(i)

output:

prassu

prem

Ex3: continue statement we can stop the current iteration of the loop, and continue with the next:

names=["prassu","prem","suryam","ram"]  
for i in names:  
 if i=="prem":  
 continue  
 print(i)

output:

prassu

suryam

ram

EX4: A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop"

colors={"red","blue","black"}  
cars={"audi","bmw","skoda"}  
for x in colors:  
 for y in cars:  
 print(x,y)

output:

red bmw

red audi

red skoda

blue bmw

blue audi

blue skoda

black bmw

black audi

black skoda

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.

def name():  
 print("hello")  
name()

output:

hello

calling a function with arguments

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

Name,age

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

Prassu,22

Using print statement:

def person(name,age):  
 print(name,age)  
person("prassu","22")  
person("prem","26")

output:

prassu 22,prem 26

def table(x):  
 return 10\*x;  
print(table(3))  
print(table(5))

output:

30

50

Recursion: it means the function calls itself

def recursion(a):  
 if (a>0):  
 num= a + recursion(a-1)  
 print(num)  
 else:  
 num=0  
 return num  
 print("result is:")  
recursion(3)

output:

1

3

6

ARRAY:

Array is an collection of values morethan one element at a time

names=["prassu","prem","bro"]  
#for access the element at particular index  
x= names[0]  
print("1st :" +x)  
# for modifying the array  
names[0]= "prassu"  
print(names)

# for finding the length of array

x=len(names)  
print(x)

# for add elements to array

names.append(“ram”)

Print(names)

# for remove the element in array

names.pop(1)

Print(names)

# for sort the elements in array

names.sort()

Print(names)

# for reverse the elements in a array

names.sort(reverse=True)

Print(names)

# for extends method

Cars=[“audi”,”benz”]

names.extends(cars)

print(names)

OOPS Concept

Object oriented programming language

* It is used for reusability of code
  1. Class and object

Object : anything that has state/properties and behavior is called an object.

Ex: chair,bottle…….

Class :it is a blueprint to create objects

Syntax: class className:

#attributes(properties/state)

Id=””

Color=””

Height=””

#functions(methods/behaviour)

Wash()

Setcap()

Constructor: it gets called and initializes all the attributes or properties with default values

* It is also a method/function, but a special method is given by python itself.

(\_\_init\_\_(self))

* Types of constructor:

1. Default constructor
2. Parameterized constructor

Constructor with parameters

Class(static) variables:

1. the variables mentioned in classs

2. The values of variables are always fix

3. it accessed without create an object

Instance(dynamic) variables:

1.can change the value of variables with help of arguments

INHERIRANCE:

* We can create child class from parent class
* Child classes will inherit all the properties and functions from the parent class
* Provides reusuability of code
* In inheritance child class always execute the other methods of parent class but parent class not execute the child class methods.

Inheritance has 2 types: they are

1. Single level( parent-child) # class A
2. Multi level(grand parent-parent-child) #Class B(A)
3. multiple level: in one class inherits the both classes at a time #Class C(A,B)

4.Hierarchical inheritance: multiples class from single parent class

5.Hybrid inheritance: it consists of multiple types of inheritance

class games(): #parent class

name=”prem”  
 def cric(self):  
 print("playing cricket")  
  
 def ball(self):  
 print("playing ball")  
class indoor(games): #child class  
 def tennis(self):  
 print("playing tennis")  
  
 def shuttle(self):  
 print("playing shuttle")  
  
Indoor=indoor()  
Indoor.tennis()  
Indoor.cric()

print(games.name)

output:

playing tennis

playing cricket

prem

ENCAPSULATION:

* Process of wrapping code and data together into a single unit

ACCESS MODIFIERS:

1.public :it can be accessible from any part of the program

2.protected: cannot be accessed outside the class but can be accessed from within class and its sub(child) classes

3.private:it can be accessible only within the class

POLYMORPHISM: it means having many forms

Same name but different forms/functionalities

1.method overloading: same name but different parameters

2.method overriding: child class has the same method as a declared in parent class then its called as overriding

The output of parent class is override by the output of child class

NOTE: Method overloading concept is not in python

# METHOD OVERRIDING

class Employee:  
 def message(self):  
 print('This message is from Emp')  
class Department(Employee):  
 def message(self):  
 print('This Department is inherited from Emp')  
emp = Employee()  
emp.message()  
print('------------')  
dept = Department()  
dept.message()

OUTPUT:

This message is from Emp

------------

This Department is inherited from Emp

\

1. abstraction

Data abstraction is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved with either abstract classes orinterfaces (which you will learn more about in the next chapter)

DATE Format:

import datetime  
x=datetime.datetime.now()  
print(x)

output:

2023-06-27 15:48:11.308399

Strftime() method

import datetime  
x=datetime.datetime(2023,9,23)  
print(x.strftime("%Y"))

output:(In place of %Y)

Y= 2023(full year)

y=23(half of full year)

A=Saturday(full name of day)

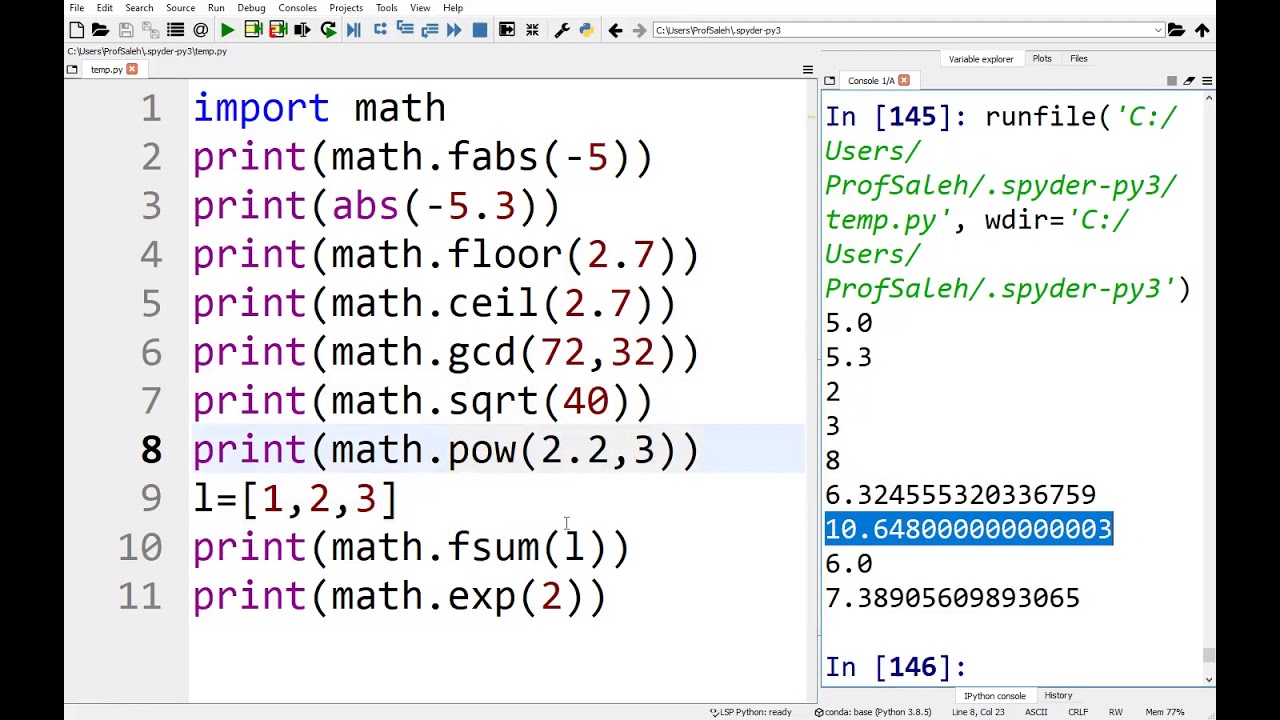
a=sat(half name of day)

B=September(full name of month)

b=sept(half name of month)

x=23/09/2023 (normal date format)

Mathematical functions:



EXCEPTION HANDLING:

When an error occurs, or exception as we call it, Python will normally stop and generate an error message.

The try block lets you test a block of code for errors.

The except block lets you handle the error.

The else block lets you execute code when there is no error.

The finally block lets you execute code, regardless of the result of the try- and except blocks

try:  
 x=10  
 y=9  
 if x>z:  
 print("greater")  
 else:  
 print("lower")  
  
except:  
 print("yes")  
finally:  
 print("block is executed")

output:

yes

block is executed

PYTHON USER INPUT FORMAT:

In this ask the enter value

username=input("enter value:")  
print("username :"+username)

OUTPUT:

enter value: prassu

username : prassu

Exception handling:

1.syntax error

2.type error

3.name error

4. index error

5.key error

6.attribute error

7.IO error

8.import error

9.zero division error

Self method:

class person:  
 def \_\_init\_\_(self, name, age, dept):  
 self.name = name  
 self.age = age  
 self.dept = dept  
 def fun2(some):  
 print("name: "+some.name)  
p1=person("prassu",22,"it")  
p2=person("bro",23,"ca")  
p1.fun2()  
p2.fun2()

OUTPUT:

name: prassu

name: bro

MULTI THREADING:

When multiple processors are running on a program in Python 3, each processor runs simultaneously to carry out its own tasks..

import time # import time module  
import threading  
from threading import \*  
  
  
def cal\_sqre(num): # define a square calculating function  
 print(" Calculate the square root of the given number")  
 for n in num: # Use for loop  
 time.sleep(0.3) # at each iteration it waits for 0.3 time  
 print(' Square is : ', n \* n)  
  
  
def cal\_cube(num): # define a cube calculating function  
 print(" Calculate the cube of the given number")  
 for n in num: # for loop  
 time.sleep(0.3) # at each iteration it waits for 0.3 time  
 print(" Cube is : ", n \* n \* n)  
  
  
ar = [4, 5, 6] # given array  
  
t = time.time() # get total time to execute the functions  
# cal\_cube(ar)  
# cal\_sqre(ar)  
th1 = threading.Thread(target=cal\_sqre, args=(ar,))  
th2 = threading.Thread(target=cal\_cube, args=(ar,))  
th1.start()  
th2.start() # to start the function  
th1.join()  
th2.join() # to stop the function  
print(" Multi threading executed ")

output:

Calculate the square root of the given number

Calculate the cube of the given number

Square is : 16

Cube is : 64

Square is : 25

Cube is : 125

Square is : 36

Cube is : 216

Multi threading executed

LAMBDA EXPRESSIONS:

It is a small anonymous function

It can take many of arguments,but have only one expression

Syntax:

lambda arguments : expression

1.ADD 10 to argument a

x = lambda a: a + 10  
print(x(6))

functions:

1.MAP():it returns an iterator corresponding to transformed collection. Each property apply to all the iterators in the list

Syntax:

Map(expression,sequence)

y=[2,4,6]  
  
print(list(map(lambda x:x\*\*2,y)))

OUTPUT:

[4, 16, 36]

2.FILTER(): it filter the data from existing list, for example we input somedata and filter the even numbers from that data

y=[0,1,2,3,5,4,6]  
  
print(list(filter(lambda x:x%2==0,y)))

Output:

[0, 2, 4, 6]

3.REDUCE():

from functools import reduce  
numbers = [1, 2, 3]  
print(reduce(lambda x, y: x + y, numbers))

Output:

6

[Python Programs - Python Programming Example (geeksforgeeks.org)](https://www.geeksforgeeks.org/python-programming-examples/?ref=shm)