PYTHON: Python is an object-oriented interpreted high-level general purpose programming language. It was created by Guido-Van-Rossum

FEATURES OF PYTHON:

* Simple and easy to learn
* free and open source
* Dynamically typed programming language
* Object-Oriented language
* Support GUI

APPLICATIONS:

* Console applications
* Desktop applications
* We based
* database
* AI and ML
* Gaming
* Networking

DATATYPES:

* Numeric
* Integer (-20,10,1)
* Float (20.3,9.3,520.123)
* Complex (3+4j,3-4j)
* Dictionary
* Boolean
* True (1)
* False (0)
* Set
* Sequence Type
* Strings
* List
* Tuple

VERSION:  
To know the version of python in your system.

import sys

print (sys. version. split () [0])

output: 3.11.3

COMMENTS:

They are two types of comments in python

* Single-line
* We use #-symbol
* Multi-line
* We use Triple single or Double Quotes

1. \n 🡪 is used for newline
2. \t 🡪 is used for tab space
3. \b🡪 deletes the previous character
4. type ()🡪 is used to know the type of the data
5. bin ()🡪 is used to get binary value
6. oct ()🡪 is used to get octal value
7. hex ()🡪 is used to get hexadecimal value
8. ord ()🡪character changes to Ascii value
9. chd ()🡪 Ascii value changes to Character

VARIBALE:

Name of the memory location / data stored in the container is called variable

* Variable cannot start with number
* Variable name cannot have a space instead we use underscore (\_)

INPUT:

The function asks the user for the data that gets stored in a variable is called input.

* SYNTAX: variable\_name = input ()

MULTI-LINE INPUT:

When we want to take multiple inputs from the user. we did not need to write input function for each value. We use (map function).

* SYNTAX:
* Variable\_names = map (datatype, input (). split (““))

Eg:

a, b, c = map (int, input (). split (“”))

KEYWORDS:

They are 35 keywords in python programming language

import keyword

print (keyword. kwlist) # prints all the keywords

OPERATORS:

* Arithmetic Operators [ +, -, \*, /, %, \*\*]
* Comparison Operator [==, !=, >, <, <=,>=]
* Assignment Operator [=, +=, -=, \*=, /=, %=, //=, \*\*+]
* Logical Operator [ and, or, not]
* Bitwise Operator [ &, |, ^, ~, <<,>>]
* Membership Operator [in, not in]
* Identity Operator [is, is not]

STRINGS:

String is a sequence collection or group of characters enclosed between Single, Double, Triple quotes.

* String is an immutable datatype

HELLO

0 1 2 3 4 🡪 Positive index

-5, -4, -3, -2, -1 🡪 Negative index

STRING METHODS:

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

CONDITONAL STATEMENTS:

In python they are 3 conditional statements

* if
* if-else
* if-elif-else

if: It is also called one-way selection statements. It executes only if the condition is true otherwise skips

SYNTAX:

if condition:

block of statements

if-else: It is also called Two-way selection statements. If the condition is true it executes true block otherwise else block executes

SYNTAX:

if condition:

block of statements

else:

block of statements

if-elif-else: It is also called muti-way selection statements. whenever we want to check more than one condition we use if-elif-else

SYNTAX:

if condition:

block of statements

elif condition:

block of statements

else:

block of statements

LOOPS:

In python they are two types of loops

* while:
* It is executed repeatedly until the given is false

SYNTAX:

while condition:

block of statements

increment (otherwise infinite loop occur)

* for:
* It is used when you how many to loop or sequence data

SYNTAX:

for variable\_name in range (start, stop, step size):

block of statements

FUNCTIONS:

* It is a block of statements to perform specific task
* Function starts with def keyword
* Functions end with colon (:)
* Every function should have a name not equal to keyword

SYNTAX:

def variable\_name(parameters):

block of statements

variable\_name () # function call

LIST:

* List is declared by using [] brackets
* List is an ordered, indexed, mutable
* List allows duplicate values

LIST METHODS:

<https://www.w3schools.com/python/python_ref_list.asp>

SET:

* + Set is declared by using Curly braces {}
  + Set is an unordered, unindexed, mutable
  + Set does not allow duplicates

SET METHODS:

<https://www.w3schools.com/python/python_ref_set.asp>

TUPLE:

* Tuple is declared by using parenthesis ()
* Tuple ordered, indexed, immutable
* Tuple allows duplicate values

TUPLE METHODS:

<https://www.w3schools.com/python/python_ref_tuple.asp>

DICTIONARY:

* Dictionary is declared by using curly braces
* Dictionary is ordered, indexed, mutable
* Dictionary does not allow duplicate values
* Key and value are called item

DICTIONARY METHODS:

[https://www.w3schools.com/python/python\_ref\_dictionary.asp](PYTHON.docx)

FILE HANDLING:

They are two types of files

* Text files
* Binary files

To open any file, we use open ()

SYNTAX:

Variable\_name = open (“filename.txt”, “Access mode”)

Variable\_name. close ()

MODES FOR TEXT FILES:

r 🡪 reading r+ 🡪 read & write

w 🡪 writing w+ 🡪 read & write

a 🡪 appending a+ 🡪 read & append

read mode methods:

* In read mode executes only if the file is exists otherwise error raised
* They are 3 modes in read modes
* read () 🡪 It is used to read entire data in the file

syntax:

* variable\_name. read (bit positions)
* bit positions are optional which is used to print given no. of bits
* readline () 🡪 It is used to print /display content in the first line

syntax:

* variable\_name. readline ()
* readlines () 🡪 It is used to print entire content in the file

syntax:

* variable\_name. readlines ()

write mode methods:

* In write mode, if the file exists the file is opened and cursor starts at beginning and data will be overwritten
* If file does not exist then new file is created with given name
* They are 2 modes in write mode
* write () 🡪 used to write single line
* syntax:

variable\_name. write (“data”)

* writelines (): 🡪 It is used to write multiple lines
* syntax:

variable\_name. writelines (“

data

“)

append mode:

* To append & write the data
* File exist 🡪 cursor points at the end of the content in file
* File doesn’t exist 🡪 new file will be created and cursor starts at the beginning

OOPS:

* Class: Collection of objects. A class contains the blueprints from which objects are being created.
* Class has methods and attributes

| |

Functions variables given in constructor

* Class is declared by using “class” keyword
* Syntax:

class Classname:

pass

In the above example a class named Classname is created by using class keyword

Object:

object is the instance /physical existence of the class

It has state, behaviors, identity

| | |

Attributes methods unique name

Inheritance:

* Derived class /child class /sub class inherit the properties of Base class/ parent class/super class
* But Base class cannot inherit properties of derived class
* They are 5 types
* Single inheritance
* Multi-level inheritance
* Multiple inheritance
* Hybrid inheritance
* Hierarchical inheritance

Single inheritance:

There would be one parent/base class and one derived/child class.

The derived class inherit properties of Base class

Eg:

class Parent:

def display (self):

print (“parent”)

class Child (Parent):

def show (self):

print (“child”)

obj = Child () #Output:

obj. display () parent

obj. show () child

multi-level inheritance:

Acquiring the properties from level wise. When a child class inherit properties from multiple parent classes

Eg:

class Grandparent:

def gdisplay (self):

print (“Grandfather”)

class Parent (GrandParent):

def pdisplay(self):

print(“Parent”)

class Child (Parent): # acquire properties of both grandparent and parent class

def cdisplay (self):

print (“Child”)

obj= Child () # object is created

obj. gdisplay ()

obj. pdisplay ()

obj. cdisplay ()

output:

Grandfather

Parent

Child

NUMPY:

NumPy is a library for the python programming language, adding support for large multi-dimensional arrays and matrices

VERSION:

To know the version of the numpy, we use “\_\_version\_\_”

import numpy as np

print (np. \_\_version\_\_)

output = ‘1.23.2’

* Numpy is faster than list due to less byte read of memory
* Contiguous memory allocation

CREATING ARRAYS:

1.Using array method

0-D 🡪 if one element is present is called zero dimension

Eg:

import numpy as np

a = np. array ((10))

print(a)

output = 10

1-D 🡪 collections of zero dimensions will form one dimension

Eg:

import numpy as np

a = np. array ([10,20,30,40])

print(a)

output = [10,20,30]

2-D 🡪 it is collections of elements in a grid -like structure with rows

Or

Collections of 1-Dimensions will form 2-D

Eg:

import numpy as np

a = np. array ([[1,2,3,4], [5,6,7,8]])

print(a)

output = [ [1,2,3,4]

[5,6,7,8]]

3-D 🡪 collections of 2-Dimensions are called 3-Dimensions

Eg:

import numpy as np

a = np. array ([[[1,2,3,4], [5,6,7,8]], [[9,10,11,12], [13,14,15,16]]])

print(a)

output= [[[1,2,3,4]

[5,6,7,8]]

[[9,10,11,12]

[13,14,15,16]]]

* asarray () with nditer ():

SYNTAX:

np. asarray (inputdata, dtype, order)

| |

array variable\_name ‘F’🡪Colum major

‘C’ 🡪 row major

Eg:

a = [[10,20,30], [40,50,60]]

b = np. asarray (a, data= int, order= C)

print(b) # normal 2-D array matrix vastundhi

# If you use asarray you use “nditer ()”

for i in nditer(b):

print(i)

output:10

20

30

40

50

60

3.frombuffer ():

* from converting string to buffer and then consider data
* it is used to convert only string

SYNTAX:

np. frombuffer (input data, dtype= ‘S!’, count = 10, offset=3)

| | |

variable\_name enni values ekkada nunchi

ravalo mention start chayali

Eg:

import numpy as np

a = b”my name is vijay”

b = np. frombuffer (a, dtype=’S1’)

print(b)

output:

[b'm' b'y' b' ' b'n' b'a' b'm' b'e' b' ' b'i' b's' b' ' b'v' b'i' b'j'

b'a' b'y']

​

* fromiter ():

To convert any sequence data into 1-D array we use

fromiter ()

SYNTAX:

np. fromiter (input data array name, dtype=””)

Eg:

import numpy as np

a = 'vijay'

b= np. fromiter (a, dtype='S1')

print(b)

output:

[b'v', b'i', b'j', b'a', b'y']

Array methods/ Properties:

1. To know the dimensions of an array we use

“Variable\_name. ndim “

1. To know the shape of an array we use

“Variable\_name. shape”

* To know the type of an array we use

“Variable\_name. dtype”

* To know the size of an element we use

“Variable\_name. itemsize”

* To know the total size of array we use

“Variable\_name. nbytes”

* To know no-of elements present in the array we use

“Variable\_name. size”

* To repeat an array for few times we use

“np. repeat (data array name, no. of times)”

TYPES OF MATRICES:

* All Zero Matrix:
* np. zeros ((row\_no, col\_no))

* All one’s matrix:
* np. ones ((row\_no, col\_no))
* Matrix with given number:

It takes two parameters 🡪 1. shape, 2.value

* np. full ((row, Colum), number)
* Matrix with random decimal values:
* np. random. rand (row, colum)
* matrix with random integer values:
* np. random. randint (End value, size = (row, colum))
* Identity Matrix:
* np. identity(rows)
* np.eye(rows)

NUMERICAL RANGES: