

# INTRODUCTION

()

History of python:

()

Invention of python : Guido van Rossum, in 1989

Date of Birth of python : 20 Feb 1991.

1989 —

→ 20 Feb 1991 - Friday

→

Guido van Rossum when working at NRI (national research institution) in Netherlands.

()

Python:- Python is a general purpose high level programming language.

()

Why name python:- when Guido Van Rossum began implementing python, he was also reading the published scripts from 'Monty Python's Flying Circus' BBC comedy series from the 1970s. He thought that he needed a name that was short, unique and slightly mysterious. So, he decided to call the language python.

()

Python Interpreter:-

()

Python being a scripting language. It contains a software package called Interpreter.

DR

→

A python interpreter is a computer program that converts each high level program statement into

machine code:

- An interpreter translates the command the you write out into code that the computer can understand.

## ① Features of python:-

1. Free and open source.
2. Easy to code.
3. Easy to read.
4. object-oriented language.
5. High-level language.
6. Large community support.
7. Easy to debug.

## ② Coding standard by PEP 8

1. Consistency and readability - writing more code  
using less code, also in writing  
python code using simple syntax and  
obeying all the rules of PEP 8 makes it easier  
to understand the code from others, and it makes  
writing simpler, it is also better to do

- writing readable

2. Writing the program properly - good writing  
using proper rules of writing

83

to the program becomes a good program. A  
good program is one that does what it is intended to do.

## → Creating variables

python has no command for declaring a variable.  
 A variable is created the moment you first assign a value to it.

Ex -  $x = 5$

$y = "John"$

i. print(x) → between id and value nothing  
 print(y) : will become

output =	5	• ("blue whale") trying	-x3
	John	-initialization	○

① Identifies it to write id of value instead of id.

ii. initialization. It is a good programming habit in python, we do nothing in initialization with global declaration of id then  
 . functioning

: c < d fi -x  
 ("out of memory id will") trying

: initialization id gives us fi unless no use ever like nothing ←

: c < d fi -x  
 ("out of memory id will") trying

- Nothing

• and also prints out something run nothing ○

# WORKING WITH PYTHON

## • Basic syntax:-

### (i) print statement:-

python syntax can be executed by writing directly in the command line;

Ex- `print("Hello world")`.

## • Python indentation:-

Indentation refers to the spaces at the beginning of a code line.

where in other programming languages the indentation in code is for readability only, the indentation in python is very important.

Ex- `if 5 > 2:`

`print("Five is greater than two")`

→ python will give you an error if you skip the indentation:

Ex- `If 5 > 2:`

`print("Five is greater than two")`.

## • Variable:-

Variable are containers for storing data values.

## String and string methods in python.

① string:-

# collection of characters.

new\_string = " my name is Aditya".  
 print(new\_string)

### Method

1. len().

to find length.

Ex- data = "Aditya"  
 x = len(data).

print(x) "Aditya" = 6  
 ("i") 6 = x

(X) wrong

2. lower().

to make small.

name = "Aditya" "Aditya" word in python brief.  
 x = name.lower()

print(x) "Aditya" = prob  
 ("i") prob = x

(X) wrong

3. Upper()

to make capital.

## 4. • replace()

name = "Aditya prateep Singh"  
~~name = "Aditya prateep Singh"~~  
~~x = name.replace("Aditya", "Adi")~~  
~~print(x)~~

## 5. • Capitalize()

To make first letter of word capital.  
~~(String) trim~~

## 6. • Center(?)

data = "Name".  
~~data = "Name"~~  
~~x = data.center(100)~~  
~~print(x)~~

## 7. • endswith(?)

check.

data = "Aditya Singh"  
~~data = "Aditya Singh"~~  
~~x = data.endswith("gh")~~  
~~print(x)~~

## 8. • find(?)

find anything in string you want

data = "Aditya Singh"  
~~data = "Aditya Singh"~~  
~~x = data.find("i")~~  
~~print(x)~~

## 9. • index(?)

• tells where at

find the position of character.

name = "Aditya"

x = name.index ("d")

print(x).

10.

.isalpha()

No space

check string is made of alphabet or not.

Ex = data = "string"

x = data.isalpha()

11.

.isdigit()

No space.

check string is made up of digit or not

12.

.isspace()

(comer) trim

check whether the string is empty or not.

13.

.istitle()

abottom

check whether all the word in string has capital  
(first character) or not.

14.

.count()

with search string

to count anything in string.

15.

.join(str-name)

(comer) trim

## List and List method

### ① List:-

① List is a collection which is ordered and changeable.  
It is create with using square bracket.

Ex- `names = ["Aditya", "Nitin", "Prashant", "Sumit"]`  
print(names)

Output:- `["Aditya", "Nitin", "Prashant", "Sumit"]`

Note:- List are mutable.

Ex- `names = ["Aditya", "Prashant"]`

`names[0] = "Adii".`

print(names)

Output:- `['Adii', 'Prashant']`

## Methods

① `append("?)` - add to last

To add something at end and you add  
duplicate items also.

`names = ["Adii", "Sumit"]`

`X = names.append("Prashant")`  
print(names).

② `insert(index, -)`

2. • remove ("?")

to remove something from list.

3. del

to delete list with item between si & fi

Ex -

data = ["Adi", "Seemrit"]

del data.

print(data)

output:

Error. [bcz Adi is deleted]

4. • clear()

• clears all items

to clear data from list.

5. • reverse()

• lists in reverse

to reverse the list.

6.

• sort()

to make the list in ascending order.

• sort() for making list brief at.

## Tuples and tuples method

- Tuples :- all many ghar me 2 avamne ot.
- Tuple is also a collection which is ordered and unchangeable.  
It is created with the help of round bracket.  
( ).

[ "times", "Bob" ] = Blob

Note:- we cannot change tuples OR they are immutable.

### Methods

[ tuple1 + tuple2 ] and + tuple

1. len(Tuplename).

To find length.

2. to add.

x = tup-1 + tup-2

point(x).

3. • count (" ? ")

to count something. tall ek ek ek ot

4. • index (" ? ")

to find the position of something.

## Sets and sets function (Methods)

### • Sets :-

• A set is a collection which is unordered and unindexed . can't add duplicate item.

+ we create set with curly bracket { }

Ex- Items = { "rice", "sugar", "salt" }  
point (items).

? = fruits ->

Note They are unordered.

### Methods

E : "apple"

1. add ("")

(fruits) trying

to add something in sets. (only one item)

2. update ("-", "-")

used to add multiples items in sets.

? = fruits

Ex-

names = { "Aditya", "X", "ABC" }

point (names).

names.update ( [ "PQR", "NAC", "LPU" ] )

point (names).

## Dictionary and Dictionary methods

### ① Dictionary :-

- ② A dictionary is a collection which is ~~unorderd~~, changeable & indexed. ~~has . bracket~~  
 we can create dictionary with curly bracket if then they have keys and values.

Ex - student = {

"name": "Rohan",

"class": 4,

"Rollno": 23,

"age": 9

}

print (student)

1. To print any specific value.

student = {

    " name" : upper.

    print (student["name"])

Output:-

Rohan.

2. To change the <sup>value</sup> key of any value key.

student["name"] = "xyz"

print (student)

z - tribute

3. len ( dict name)

(mark) pop() method - mark

to find length.

(mark) bring

4. To add new item.

student {  
    "  
    "y  
}

student["Marks"] = "95%"

print (student)

5. To remove something from dictionary.

(i) • pop ("—")                  ↴ some

(ii) del student ("—")

6. To delete full dictionary.

(i) del student

7. • clear()

to delete Key-value pair.

8. • copy()

To make copy of dictionary.

Ex-

student = {

"

}

(tribute) tribute

(tribute) tribute

phone = student. copy (student)

print (phone)

. Model brief of

model answer book of

tribute

"

{

"copy" = [tribute] tribute

(tribute) tribute

model answer book of

model

{

("\_\_") group (i)

("\_\_") tribute list (ii)

model answer book of

tribute list (iii)

(model)

model answer book of

# LOOPS

① while loop :-

$i = 1$

while  $i \leq 10$ :

print(i)

$i + 1$ .

If  $i == 4$ : Break.

Break.

$i = i + 1$

(16) trying

②

Break :- " ", X, "\*", num } trying

$i = 0$

while  $i \leq 10$ :

$i + 1$

If  $i == 4$ :

Break

print(i)

X

$i = 0$

while  $i \leq 10$ :

print(i)

If  $i == 4$ :

Break

$i + 1$ .

✓

③

Continue :-

$i = 0$

while  $i \leq 10$ :

$i + 1$

If  $i == 4$ :

continue.

print(i)

①

Fan :-

Ex-

for  $x$  in range(0, 10):

print( $x$ )

$x += 1$

$i = i$

:01 => i after

#

To make table.

(D) true

$i = +i$

num = int(input("enter the no:"))

for  $x$  in range(1, 11): show

$b = num * x$

print(num, "\*", x, "=" , b)

$0 = i$   
:01 => i after  
(D) true

$i = +i$

:P = -i if  
show

$i = +i$

✓

$0 = i$   
:01 => i after  
(D) true

$i = +i$

:P = -i if  
show

(i) true

X

$0 = i$   
:01 => i after  
 $i = +i$

:P = -i if  
writing

writing

(D) true

# FUNCTIONS

## ① Function:-

① A function is a block of code that performs a specific task.

## ① Types of function:-

1. Built in function.

2. User defined function.

## ① Built in function:-

1. `max(, )`

check the maximum between or from two.

Ex:

`a = 10`

`b = 20`

`c = max((a, b))`

`print(c)`

2. `sum(, )`

`add .`

Ex:

`a = 10`

`b = 10`

`c = sum(a, b)`

`print(c)`

① User define function

② def my\_function()

we make function by using def

Ex- a = int(input("enter the number"))

def my\_square():

b = a\*\*2

print(b)

my\_square().without output

-: without output

(C:\&gt;) from

work now to convert maximum with short

a = b

a = d

(C:\&gt;) max = 0

(2) today

(C:\&gt;) max = 0

bb

a = D

= 0

## ① Lambda function :-

Ex - normal function

Ex - (i)  $\text{def squ}(a): \text{return } a**2$  ] normal  
 $\text{point}(\text{squ}(2))$   
 $\text{squ}(a)$

Ex - Lambda function

Ex (ii)  $\text{squ} = \lambda a: a**2$

Ex (iii)  $\text{tot} = \lambda a, b: a+b$   
 $\text{point}(\text{tot}(2, 3))$

# FILE HANDLING

①

File handling :- This will be additional.

(Additional) thing

②

Types of file in python :-

1.

Text file (Ex..txt)

2.

Binary files (.jpg, .dat)

③

Opening a file :-

```
open ("file_name.txt", "r")
```

④

Reading a file in python :-

```
f = open ("Test.txt", "r")
```

```
myfile = f.read()
```

```
print myfile
```

```
f.close()
```

first make file named "test.txt". and then do this.

```
f = open ("Test.txt", "r")
```

```
for a in f:
```

```
print (a)
```

① write a file in python:-

```
f = open("text1.txt", "w")
```

make file in python.

```
f = open("text1.txt", "w")
f.write(" ")
f.close()
```

② append a file in python (update) :-

```
f = open("text1.txt", "w")
f = open("text1.txt", "a")
```

over write

```
f.write(" ")
f.close()
```

```
f = open("text1.txt", "a")
f.write(" ")
f.close()
```

add.

with open

```
with open("text1.txt", "r") as f:
    myfile = f.read()
    print myfile
```

③ Rename and delete a file :-

import file +

import os

old\_file\_name = ("old.txt")

new\_file\_name = ("")

f = open("old\_file\_name", "r")

(read) f.read()

(file) f.readline()

(read) f.readline()

(read) f.readline()

f.close()

("w", "text.txt")

(w+) "text.txt")

(w+) f.write()

(write)

("a", "text.txt")

(a+) "text.txt")

(append)

f.write("Hello")

(file) f.close()

(close)

# OBJECT ORIENTED PROGRAMMING

- ① python is an object oriented programming language.
- ② Class :- blueprint or template of creating objects
- ③ A class is a blueprint or template for creating objects
- ④ Object :-
- ⑤ An object is a real world entity.

class foun:

```
def printdetails(self)
```

```
print ("your name is:", self.name)
```

```
print ("your rollno is:", self.Rollno.)
```

```
object = foun()
```

```
object.name = str(input("enter your name"))
```

```
object.rollno = int(input("enter your rollno"))
```

```
object.printdetails()
```

: foun() : foun()

: (name, Rollno) : foun()

name = name, Rollno = Rollno

① oops to python:-

To map with real world scenarios, we started using objects in code.  
This is called object oriented programming.

① class:-

class is a blueprint for creating object.

~~class student:~~ class student: on this will create a object with A name = "Karan"

# object

s1 = Student()

print(s1.name)

Print of sub

(Karan) distributing sub

① \_\_init\_\_ function:- map " ) func

Carry. j2. " is or have map " ) func

# constructor:-

( constructor = twle

All class have a function called \_\_init\_\_( ), which is always executed when the class is being initiated.

# creating class

class student:

def \_\_init\_\_(self, fullname):

self.fullname = name

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

- Class & Instance attributes:-

→ class attribute < object attribute.

```
class student:
    school = "GLA"      # class attribute
    def __init__(self, school):
        self.school = school # object attribute.
```

- Methods :- Work on the object basis program

- Methods are function that belongs to object.

- static method :- no self parameter at begin

- Methods that doesn't use the self parameter (work at class level)

```
Ex:- class student:
    def college():
        print("ABC college")
```

Ex:- ~~class student:~~ ~~def college():~~ ~~print("ABC college")~~

```
@staticmethod
def hello():
    print("Hello")
```

- Decorators allow us to wrap another function in order to extend the behaviour of the wrapped function, without permanently modifying it.

- Abstraction:-

- Hiding the implementation details of a class and only showing the essential features to the user.

- To hide unnecessary things.

- Encapsulation:-

data +  
function → capsule.

- wrapping data and function into a single unit (object).

- del keyword:-

- used to delete object properties or object itself.

# del s1, mylist  
# del s1

- private (like) attributes & methods:-

- # conceptual implementation in python:-

- Private attributes & method are meant to be used only within the class and are not accessible from outside the class.

Ex

- name -> name and has batter info
- hello(self): view of following

## ○ Inheritance :-

- when one class (child / derived) derives the properties & method of another class (parent / base).

Ex- Class Car:

--- ---

(a) batter info

--- ---

(b) batter with

class ToyotaCar (Car):

-- phrasing

## ○ Types of Inheritance

1. Single Inheritance.

probable outcome

2. Multi-level Inheritance.

3. Multiple Inheritance. (i) through many inheritance

(ii) through single problem

## ○ Super method:-

Inherits without &amp; overrides

○ Super() method is used to access methods of the parent class.

(d) - bho ... D

d+D

## ○ class method

(dL - dho ... D

d - o

d &gt; p

○ A class method is bound to the class &amp; receives the class as an implicit first argument.

d &gt; D

@ getter  
@ setter

Date / /  
Page No.

Shivalal

Note:-

Static method can't access or modify class state & generally for unity.

class student:

@ classmethod # decoration  
+ college def college(cls)  
. pass

- Static method
- class method (cls)
- instance method (self)

① property :-

we use @ property decorator on any method in the class to use the method as a property.

② Polymorphism:-

③ Operator overloading

When the same operator is allowed to have different meaning according to the context.

④ Operator & Dunder functions

- a+b
- a-b
- a\*b
- a/b
- a%b

a...add...(b)

a...sub...(b)

a...mul...(b)

a...truediv...(b)

a...mod...(b)