add → append

PYTHON

Python → Name?
(Circus Name)
→ Statistically programming language

Language fundamental:

- General purpose high level programming Language.
- Guido Van Rossam in 1989
- Date of Birth Feb 20th 1991

Python:

Ex1: print("Hello World") — // Hello World //

Ex2: To print the sum of 2 numbers.

```
a = 10
b = 20
print("The sum:", (a+b))
```

- Almost all programming features from different languages

1. Functional Programming feature from c
2. OOP's feature from c++
3. Scripting Language feature from perl & shell
4. Modular programming feature from Modula-3

Where we can use python.

1. For developing Desktop Appl^n
2. Web Appl^n
3. Database Appl^n : Games
4. Network programming
5. Developing Games
6. Data Analysis Appl^n
7. ML
8. AI appl^n
9. IOT appl^n

- In which company Python used:

PwC
Google
Netflix
DropBox
Youtube
Nasa
ISRO

Feature of python. (What are the key features)

1. simple and easy to learn
2. Freeware and Open source
3. High Level programming Language
4. Platform Independent
5. Portability
6. Dynamically typed
7. Both Procedure oriented & Object oriented
8. Interpreted
9. Extensible
10. Embedded
11. Extensive Library

Limitations

- Not using for mobile appl^n.
- No version Backword compatibility.

Version

- python 1.0V in Jan 1994
- Python 2.0V in Oct 2000
- python 3.0V in Dec 2008

Current version

- python 3.6.1
- python 2.7.13
- python 3.9.6 (current)

Q.1 What is python?

- Python is a high-level, interpretted, interactive object-oriented scripting language
- Python is designed to be highly readable.
- It uses english keywords frequently where as other languages use punctuation & it has fewer syntactical constructions than other language.

Q2 ⇒ How do you gets a **list of all the keys** in dictionary?

- For this, we use the function keys()

```
mydict = {'a':1, 'b':2, 'c':3, 'e':5}
→ mydict.keys()
['a', 'b', 'c', 'e'] ←
```

Q.3 ⇒ What is **Slicing?**

- Slicing is a techniques that allows us to retrive only a part of a list, Tuple or string.
- Slicing operator [ ]

```
a = (1, 2, 3, 4, 5)
    0  1  2  3  4
a[2:4]
(3,4)
```

(4-1) = 3

```
b = 'Hello'
    0 1 2 3 4
   -5-4-3-2-1
[ :-1]
Hell
```

Q.4 ⇒ How will you cheack if all character in a string are alphanumeric?

Alphanumeric → isalnum()

isalnum()

Q.5 ⇒ How will you capatalised 1st Letter of string?

- simple using method : capitalize()
- a = 'nilam'. capitalize() // 'Nilam'

swapcase()
string = "NILAM"
print(string.swapcase())
string = "nilam"
print(string.swapcase())

Q.6 ⇒ With python how do you find out directory you are in?

- function/ method getcwd() & import it from os

find directory
import os
os.getcwd()

change directory
os.chdir()
os.getcwd()

- import os
  os.getcwd()
  // 'C:\\Users\\Admin\\AppData\\Local\\Program'
  type(os.getcwd)
  <class 'builtin_function_or_method'>

Q.7 ⇒ How do you reverse a List?

- Using reverse() method
  a.reverse()
- Also slicing from Right to left? : a[::-1]

a = [1,2,3,4]
- a.reverse()
  a // [4,3,2,1]
- a[::-1]
  a // [4,3,2,1]

Q.8 ⇒ Why do we need break & continue?

for var in range(len(my_list)):
  # inside loop
  if condition:
    continue
  # exit for loop

while expression:
  if condition:
    break
  # inside loop
  # while loop exist

- Both Break & Continue are statement that control flow in python loops.
- Break stops the current loop from executing further and transfers the control to the next block.
- Continue jumps to the next iteration of loop without exhausting it.

Q.9 ⇒ In one line, show us how you'll get max alphabetical character ? from string ?

- For this we will simply use the max fn.

```
• max('NeelAmolArya')
// 'y'
• min('abkdjzX')
// 'X'
• max('abkdjzX')
// 'z'
```

- ASCII value.

min ⇒ A:65 - Z:90
max ⇒ a:97 - z:122

A-Z → kuch fn
a-z → max fn

Note →

'max' ==> find only lower case Alphabets
'min' ==> find only Upper case Alphabets.

Q.10 ⇒ How will you remove a duplicate element from List?

- We can turn it into a set to do that.

```
list = [10,20,30,10,20,30,50]
List
    // [10,20,30,10,20,30,50]
set(List)
    // {10,20,50,30}
setitems = set(List)
setitems
    // (10,20,50,30)
```

printing list

Q.11 ⇒ What are supported datatypes in python?

- 6 standard datatypes:

① Numbers
② String
③ List
④ Tuple
⑤ Dictionary
⑥ set

List:
- collection of items of different datatypes which can be changed at runtime.
- Mutable
- Allow to store mutable & immutable obj.
- Ordered collection
- store Duplicates & Heterogenous elements
- store similar ele.
- we can process elements using indexing & slicing.

Tuple:
- collection of items of diff^{n} datatypes which cannot be changed.
- Only has Read-only access to the collections
- Immutable
- Ordered collection
- store Duplicates & Heterogenous elements
- we can process elements through indexing and slicing
- modification not present
- Allow to store Mutable & Immutable obj.

Immutable - Obj whose value can't change after once created. (cannot Remove or Add)
bool, int, float, tuple, str, frozenset

mutable - Whose value can changed once created.
list, set, dict (can Remove or Add)

Set:
- collection of items of similar data types.
- Mutable
- Modify After creating
- Collection Unordered
- not an Ordered collection
- List can't be element of set
- pre-defined obj.

Dictionary ⇒

- collection of items with (key value pairs)
- (ordered) collection.
- store elements using keys
- keys are also obj
- keys & values can be heterogenous.
- key must be unique.
- (Mutable)

{ }

Q.12 ⇒ What is [PIP, software] in Python?

- PIP is a acronym for python installer package which provides a seamless interface to install various python module.

- It is a command line tool which can search for packages over the internet and install them without any user interaction.

Q.13 ⇒ What is a [pickling and unpickling ?] unpickling

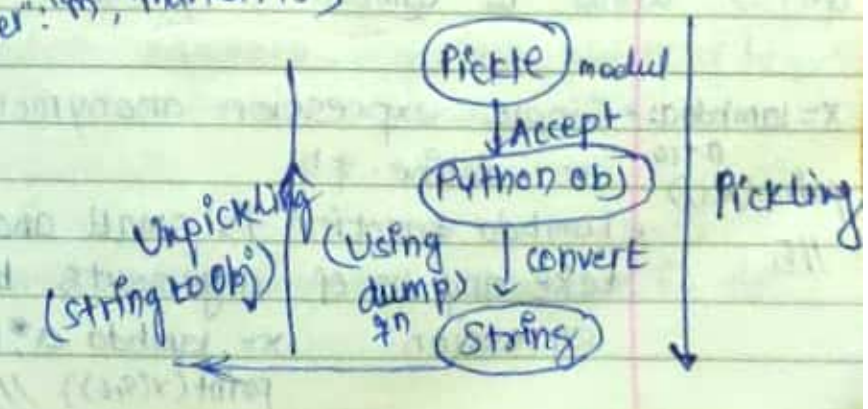
(Pickle)

- Pickling module accepts any python object and converts it into a string representation and dumps it into a file by using dump function

The process of retrieving origional python objects from the stored string representation is called Unpickling.

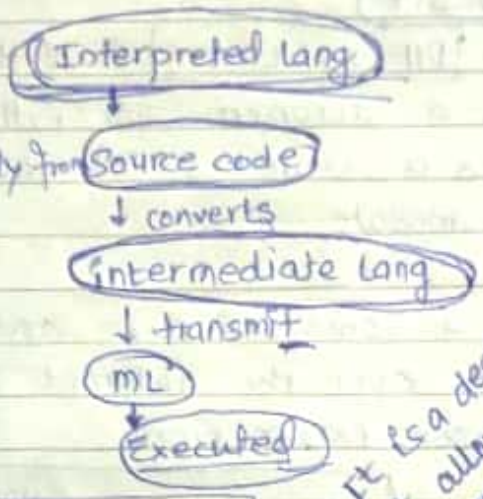
(Pickle obj into bin)

```
import pickle
f=open("data.txt","wb")
dct={"name":"Ravi","age":23,"Gender":"M","marks":75}
pickle.dump(dct,f)
f.close()
```



Q.14 ⇒ How python is **Interpreted?**

- Python Language is an interpreted lang.
- Python program is run directly from the source code
- It converts the source code that is written by the programmer into an intermediate lang. which is again transmitted into ML that has to be executed.



Q.15 ⇒ What are **Python Decorators**? ⇒

- Specific change that we make in python syntax to alter function easily.

It is a design pattern in python that allows a user to add new functionality to an existing obj without modifying its structure. before the def of fn

```
def plus_one(num):
    return num+1
add_one = plus_one
add_one(5)
```
1/16

Q.16 ⇒ What is the diffn between List and Tuple?

- List is mutable while Tuple is immutable.

Q.17 ⇒ What is Lambda in python?

```
x = lambda a: a+10
print(x(5))
```
1/15

- Single expression anonymous fn often used as inline fn.
- Lambda function is small anonymous fn, it can take any no. of arguments but only have one expression.

```
x = lambda a,b : a*b
print(x(5,6))
```
1/30

Q.18 ⇒ Why Lambda forms in python does not have statements?

- It is used to make new fn obj and the return them at runtime.

Q.19 ⇒ What is pass in python?

- Pass means, no operation python statement.
- It is place holder in compound statement, where there should be a blank left and nothing has to return their.

```
for x in [0,1]:
    pass
```

Ex: def myfunction():
    pass o/p: no longe o/p

Q.20 ⇒ What are Iterators?

- An Iterators is an obj. that contain countable no. of values
- Used to iterate a group of elements, containers like list.
- methods: __iter__(), __next__().

```
mytuple=(10,20)
myIT=iter(my tuple)
print(next(myIT))
```

Ex: mytuple=(10,20,30,40)
myit=iter(mytuple)
print(next(myit))
print(next(myit))
print(next(myit))
//10, 20 30

Q.21 ⇒ What are generator in python?

- Way of implementing Iterators are known as generator

Q.22 ⇒ How to delete file in python

- By using command. ⇒ os.remove(filename)
  or os.unlink(filename)

Q.23 ⇒ How to generate random numbers in Python?

- To generate random numbers in python, need to

```
import command
import random
random.random()
```

This returns a random floating point number in the range [0,1)

Q.26 → What is Exception

• An unwanted and unexpected event that disturbs normal flow of program

• Types → ① Syntax error ② Runtime error

Identifiers:

Name to Python program is called as Identifier

• Alphabet Symbols: (Either Upper OR Lower case)
• Should not start with digit
• Are case sensitive
• Can not use reserved words (Eg: def = 10 ✗)

• 123total ✗
• total123 ✓
• _abc_abc_ ✓
• Javashare ✓
• Ca$h ✗
• def ✗
• if ✗

• Alphabet Sym
• total123
• case sensitive

✓ _add ==> Private
✓ __add ==> strongly private
✓ _add_ ==> Magic method

Constructor

__init__

constructors are generally used for fulfilling an obj. The task of constructor is to analize to the data member of the class when an obj is created.

• 2 types
1) defaults 2) parameterised

```
def __init__(self):
    # Body of the constructor
```

① default constructor

```
class xyz:
    def __init__(self)
        self.xy = "xyz")
```

② parameterised constru.

```
class addition:
    first=0
    second=0
    Third=0
def __init__(self,f,s):
    self.first=f
    self.second=s
```

Reserved word:

- 33 Reserved Word :

-True, False, None
-and, or, not, is
- if, elif, else
- while, for, break, continue, return, in, yeild
- try, except, finally, raise, assert
- import, from, as, class, def, pass, global, nonlocal, lambda, del, with

Data Types:

- 14 Data Types
- Standard data types / Supported

Numeric	Tuple
string	set
List	Dictionary

1. Int
2. Float
3. complex
4. bool
5. str
6. Bytes
7. Bytes Array
8. range
9. List
10. tuple (immutable)
11. set
12. Frozenset (immutable)
13. dict
14. None

Inbuilt Function:

1. type() : type of variable
2. id() : address of variable
3. print() : print value

Representation of values:

1. Decimal form
   - digits are 0 to 9

   Eg: a = 10

② Binary Form (Base 2)

- Allowed digit : 0 & 1
- Value should be prefixed with 0b or 0B

Eg: a = 0B1111
a = 0B123
a = b1111

③ Octal form (Base-8)

- Allowed digit : 0 to 7
- prefixed with 0o or 0O (zero + letter (small/capi))

Eg: a = 0o123
a = 0o786

④ Hexadecimal form (Base-16)

- Allowed digits are: 0 to 9, a-f (Upper & lower)
- Prefixed with 0x to 0X

Eg: a = 0XFACE
a = 0XBeer

Base Conversion:

① bin(): To convert any base to binary.

Eg: bin(15): '0b1111'
bin(0011): '0b1001'

② oct(): To convert from any base to octal

Eg: oct(10): '0o12'
oct(0B1111): '0o17'

③ hex(): To convert from any base to hexa decimal

Eg: hex(100): '0x64'
hex(0B1111111): '0x3f'

Float Data Types:

- Eg: f=1.234
  type(f) // float //
- Eg: f=1.2e3
  print(f) //1200.0

$1.2e3$
1.2×10^3

Complex Data Type:

Eg: 5+5j

$a+bj$ $\quad j=\sqrt{-1} \quad \therefore j^2=-1$

a → Real, b → Imaginary part

Boolean Data Types:

b = True
type(b) // bool

Eg: a=10
b=20
c=a<b
print(c)
// True

True + True = 2
True - False = 1

Str Type:

s1 = 'daya'
s1 = """daya"""
s1 = """daya
reddy"""

List : Ordered, mutable, heterogenous collection of elements (Duplicates are allowed)

List = [10, 10.5, 'durga', True, 10]

Ex: List = [10,20,30]
List.append("durga")
List // [10,2030, 'durga']
list.remove(20)
List // [10,30, 'durga']
list2 = list*2
list2 // [10,30,'durga',10,30,'durga']

Tuple: Ordered, heterogenous, Immutable
(Duplicate Also Allowed)

Eg:
```
t=[10,20,30,40)
t.append("daya")   // Attribute error
t.remove(10)       // Attribute error
```

Set: Group of values without duplicates when order is not important then we go for set data type

```
s={100, 0, 10, 200, 10, 'durga'}
```

Dict: Represent a group of values as key-value pairs then we should go for dict type

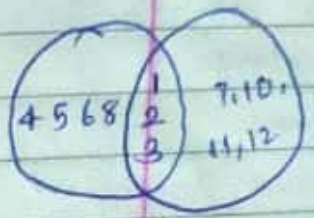
```
d={101:'durga', 102:'ravi', 103:'shiva'}

d[101]='sunny'
d      // {101:'sunny', 102:'ravi', 103:'shiva'}
```

strigs:

- Group of character
- strigs are Arrays

Set:



Datatype	Description	M/I	Example
int	we can use to represent whole or integral numbers.	Immutable	a=10 type(a) // int
Float	we can use to represent the decimal / floating point numbers	Immutable	b=10.5 type(b) // float

complex	we can use to represent the complex number.	Immutable	c=10+5j
			type(c) // complex
			c.real // 10
			c.imag // 5.0

Bool	we can use to represent Logical values (True/False)	Immutable	Flag=True Flag1=False type(Flag) // bool
str	To represent sequence of character	Immutable	s=durga type(s) // string
bytes	To represent a sequence of byte value from 0-255	Immutable	list=[1,2,3,4] b=bytes(list) type(b) // bytes
bytearray	To represent a sequence of byte values from 0-255	mutable	List=[1,2,3,4] ba=bytearray(list) type(ba) // bytearray
range	To represent a range of values from 0-255	Mutable	r=range(10) r1=range(10,0) r2=range(0,10,2)
list	To represent an ordered collection of objects	mutable	l=[10,11,12,13,14,15] type(l) // list
tuple	-"- Ordered	Immutable	t=(1,2,3,4)
set	To represent an Unordered	Mutable	s={1,2,3,4,5,6}
Frozenset	To represent an unordered collection of unique objects	Immutable	s={1,2,3,'daya',100} fs=frozenset(s)
dict	To represent a group of key value pairs	mutable	d={101:'durga', 100:'daya', 103:'rani'} type(d) // dict

Escape character

(9)
\n - New line
\t - Horizontal tab
\r - carriage return
\b - backspace
\f - formfeed
\v - vertical tab
\' - single quote
\" - Double —"—
\\ - Back slash symbol.

Q. How to Remove whitespaces from a string in python?

- strip([str])

```
string = "  java  "
string2 = "   java  "
string3 = "    java   "

print(string)
print(string2)
print(string3)
print ("After stripping all have placed in a sequence:")
print (string.strip())
print (string2.strip())
print (string3.strip())
```

- %
```
  java
   java
    java
```

After stripping all have placed in a sequence.

```
java
java
java
```

Operators

① Arithmatic operators

+ : Addition
− : Substraction
* : Multiplication
/ : Division
% : Modulo operator
// : Floor Division
** : Power Operator

② Relational operator

>, >=, <, <=

② Equality Operator

==, !=

④ Logical operator

and, or, not

⑤ Bitwise operator

Left Right

&, |, ^, ~, <<, >>

⑥ Assignment Operator

+=, -=, *=, /=, %=, //=, **=,
&=, |=, ^=, >>=, <<=

Special operators

① Identity Operators
- i) is
- ii) is not

② Membership operators
- i) in
- ii) not in

Operator Precedence

() → paranthesis
** → Exponential
~, - → Bitwise complement, unary minus operator
*, /, %, // → Multiplication, division, modulo, Floor division
+, - → addition, subtraction
<<, >> → Left and Right shift
& → bitwise And

^ → Bitwise X-OR

| → Bitwise OR

>, >=, <, <=, ==, != → Relational or comparision Ope.

=, +=, -=, *= → Assignment operators

is, is not → Identity operators

in, not in → Membership operators

not → Logical not

and → Logical and

or → Logical or

SET

- Duplicates are not Allowed
- Indexing & slicing Not Allowed
- Mathamatical Operation: union, intersection, difference

- Important function:

① add(x)

```
s={10,20,30}
s.add(40);
print(s) // {40,10,20,30}
```

② update(x,y,z)

```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5))
print(s)
// {0,1,2,3,40,10,50,20,60,30}
```

Q. What is the diff^n bet^n Add & update in set?

- Add() - To add individual item to set
- Update() - To add multiple items to set

③ copy():

- Returns copy of set

```
s={10,20,30}
s1=s.copy()
print(s1)
// {10,20,30}
```

* ④ pop()

- It removes & returns some random element from the set

```
s={40,10,30,20}
print(s) // {40,10,30,20}
print(s.pop()) // 40
print(s) // {10,20,30}
```

- // It can be access after pop()

⑤ remove(x)

- It removes specific elements of set

```
s={40,10,30,20}
s.remove(30)
print(s)
// {40,10,20}
s.remove(50) ---> keyerror
```

- // It cannot be accessed after removing

⑥ discard(x)

- It remove specific element from set

```
s={10,20,30}
s.discard(10)
print(s)
// {20,30}
s.discard(50)
print(s) ..... // {20,30}
```

⑦ clear()

- To remove all elements of set

```
s={10,20,30}
print(s) ..... // {10,20,30}
s.clear()
print(s) ..... // set()
```

Mathamatical Operation on the set:-

① Union()

- return all elements prasent in both set

```
x.union(y)
x|y
```

Eg:

```
x={10,20,30,40}
y={20,60,50,60}
print(x.union(y))
// {10,20,30,40,50,60}
print x|y
// {10,20,30,40,50,60}
```

② intersection()

- common elements present in both x & y
- x.intersection(y) or x&y

```
x={10,20,30,40}
y={30,40,50,60}
print(x.intersection(y)) --- // {40,30}
print(x&y) ------ // {40,30}
```

③ difference()

- element present in x but not in y
- x.difference(y) or x-y

```
x={10,20,30,40}
y={30,40,50,60}
print(x.difference(y)) - // {10,20}
print(x-y) --- // {10,20}
print(y-x) --- // {50,60}
```

④ Symmetric difference()

- present in either x or y but not in both
- (x.symmetric_difference(y)) or (x^y)

```
x={10,20,30,40}
y={30,40,50,60}
print(x.symmetric_difference(y))
prin x^y  // {10,50,20,60}
```

Q. Write a program eliminate duplicates present in list?

⇒

```
l=eval(input("Enter list of values"))   // Enter list of values:
s=set(l)                                [10,20,30,10,20,40]
print(s)                                // {40,10,20,30}
```

⇒

```
l=eval(input("Enter List of Values"))
ll=[]
for x in l:
    if x not in ll:
        ll.append(x)
print(ll)
// Enter the list of values : [10,20,30,10,20,40]
// [10,20,30,40]
```

Q. Write a program to print a different vowels prasent in the given word?

```
w = input("Enter word to search for vowels:")
s = set(w)
v = {'a','e','i','o','u'}
d = s.intersection(v)
print("The different vowel prasent in", w, "are", d)
```

output: Enter word to search for vowels: durga
The different vowel prasent in durga are {'u','a'}

Function:- A fn is a block of code which only runs when it is ~~created~~ called - A fn can return data as a result.

① Built in function:
A fn which are coming along with python software
Eg: id(), type(), input(), eval()....

② User Define fn:
The fn which are devoleped by programmer

```
def function_name(parameter):
    """ doc string """
    ....
    ....
    return value
```

creating function
we can use 2 keywords
1) def (mandatory)
2) return (optional)

• Parameter- parameters are input to the function.

```
def wish(name):
    print("Hello", name, "Good morning")
wish("Daya")   // Hello Daya Good morning
wish("Ravi")   // Hello Ravi Good morning
```

Q. Write a function to take number as input & print its square

```
def squareIt(number):
    print("The square of", number, "is", number*number)
squareIt(4)   // The square of 4 is 16
squareIt(5)   // The square of 5 is 25
```

- Return statement

fn can take input value as parameter & execute bussiness logic & returns o/p to the caller with return statement.

Q. Write a fn to accept 2 numbers as i/p and return sum.

```
def add(x,y):
    return x+y
result = add(10,20)
print("The sum is", result)
print("The sum is", add(100,200))
 // The sum is 30
// The sum is 300.
```

Cheack weather the given number is even or odd?

```
def even_odd(num):
    if num%2==0:
        print(num, "is Even number")
    else:
        print(num, "is Odd number")
even_odd(10)
even_odd(15)

// 10 is Even number
// 15 is odd number.
```

Q. Write a function to find Factorial of given number?

```
def fact(num):
    result = 1
    while num >= 1:
        result = result * num
        num = num - 1
    return result
for i in range(1,5):
    print("The Factorial of", i, "is:", fact(i))

// The factorial of 1 is 1
                    2 is 2
                    3 is 6
                    4 is 24
                    5 is 120
```

* Returning Multiple values from a fn:

Ex:

```
def sum_sub(a,b):
    sum = a+b
    sub = a-b
    return sum, sub
x, Y = sum_sub(100, 50)
print("The sum is :", x)        // The sum is: 150
print("The sum is :", Y)        // The sub is: 50
```

Ex2:

```
def calc(a,b):
    sum = a+b
    sub = a-b
    mul = a*b
    div = a/b
    return sum, sub, mul, div
t = calc(100,50)
print("The Results are")
for i in t:
    print(i)
// The results are
150
50
5000
2.0
```

• Types of Arguments:

1) Positional Arg: passed to fn in correct positional order

```
def sub(a,b):
    print(a-b)
sub(100,200)
sub(200,100)
```

2) Keyword arguments: pass arg values by keyword. i.e by parameter name.

```
def wish(name,msg):
    print("Hello",name,msg)
wish(name="Durga", msg="Good Morning")
wish(msg="Good Morning", name="Durga")

// Hello Durga Good Morning
// Hello Durga Good Morning
```

3) Default Arguments:

Sometimes we can provide default values for our positional arguments.

```
def wish(name="Guest"):
    print("Hello", name, "Good Morning")
wish("Durga")
wish()
// Hello Durga Good Morning
// Hello Guest Good Morning
```

4) Variable length arguments: (*)

def f1(*n)

Eg:

```
def sum(*n):
    total = 0
    for n1 in n:
        total = total + n1
    print("The sum =", total)
sum()          // The sum = 0
sum(10)        // The sum = 10
sum(10,20)     // The sum = 30
sum(10,20,30,40) // The sum = 100
```

① Global var: outer / global space

The variable declare before the main function

② ~~Valid~~ Local var: Inside / local space

The variable declare Before the main function.

- Function Aliasing

for the existing f^n we can give another name

Function Decorators:

- Decorator is a fn which can take a fn as arguments & extend its functionality & returns modified fn with extended functionality.

Generators:

- Generator is a function which is responsible to generate a sequence of values.
- We can write generator functions just like ordinary fn but it uses yeild keyword to return values.

Eg:

```
def mygen():
    yeild 'A'
    yeild 'B'
    yeild 'C'
g = mygen()
print(type(g))      // <class 'generator'>
print(next(g))      // A
print(next(g))      // B
print(next(g))      // C
```

Eg: To generate first n num.

```
def firstn(num):
    n=1
    while n<=num:
        yeild n
        n=n+1
values = firstn(5)
for x in values:
    print(x)
```

op: 1 2 3 4 5

Input And Output Statements:

- Reading Dynamic input from the keyword:
  ① Raw_input() ② input()

① Raw_input(): This statement Always reads the data from the keyboard in the format of String Format.
- we have to convert string type to our required type by using corresponding type casting method.

Eg:
```
x=raw_input("Enter First Number:")
print(type(x)) -----(It willbe Always print str type only for any i/p)
```

② input(): Used to read data directly in our required format.
- we are not required to perform type casting.

Eg:
```
x=input("Enter Value)
type(x) ------(//10-int, "daya"-str, 10.5-float, True-bool)
```

Q. Write a program to read 2 numbers from the keyboard & print sum.

```
x=input("Enter a first number:")
y=input("Enter a second number:")
p=int(x)
q=int(y)
print("The sum is:",p+q)
```

O/P:
Enter 1st nu.=100
=200
The sum is 300

OR

```
x=int(input("Enter First Number:"))
y=int(input("Enter Second Number:"))
print("The sum:",x+y)
```

OR

```
print("The Sum:",int(input("Enter First Nu.:"))+int(input("Enter Second Nu.:")))
```

Command Line Arguments:

- Arguments is not Array it is a list. It is available sys module
- The Argu. which are passing at command line execution time called comaand line Arg.

Tuples

- same as list but it's immutable.
- Data is fixed and it never changes

- Tuple Creation:

① t=() : empty tuple
② t=(10,) : single valued tuple, comma should be optional
③ t=10,20,30 / t=(10,20,30) : multivalued tuple & ( ) are optional
④ By using tuple() function:

```
list=[10,20,30]
t=tuple(list)
print(t)
```

- Accessing elements of tuple:

① By using indexing:

```
t=(10,20,30,40,50,60)
print t[1] //20
print t[-1] // 60
```

② By using slice operator

```
t=(10,20,30,40,50,60)
print(t[2:5]) //30,40,50)
print(t[::2]) //(10,30,50)
```

- Mathamatical Operators for Tuple:

① concatenation operator(+)

```
t1=(10,20)
t2=(30,40)
t3=t1+t2
print(t3)
//(10,20,30,40)
```

② Multiplication Operator / Repetitive

```
t1=(10,20,30)
t2=t1*3
print(t2)
// (10,20,30,10,20,30,10,20,30)
```

• Important fn in Tuple:

① len()

```
t=(10,20,30,40)
print(len(t))
//4
```

② count()

```
t=(10,20,10,10,20)
print(t.count(10))
//3
```

③ index

```
t=(10,20,10,10,20)
print(t.index(10))
//0
```

④ sorted()

```
t=(40,10,30,20)
t1=sorted(t)
print(t1)
print(t)
//[10,20,30,40]
//(40,10,30,20)
```

⑤ min() and max() fn

```
t=(40,20,30,10)
print(min(t))
print(max(t))
//10
//40
```

⑥ cmp():

```
t1=(10,20,30)
t2=(40,50,60)
t3=(10,20,30)
print(cmp(t1,t2)) //-1 {No comma
print(cmp(t1,t3)) //0 {comma
print(cmp(t2,t3)) //+1 {No comma
```

• Diff^{n} bet^{n} List & Tuple

List	Tuple
1) Group of comma separeted values within square brackets eg: l=[10,20,30,40]	1) Group of comma separated values & Bracket are optional Eg: t=(10,20,30,40)/t=10,20,30,4
2) List object are mutable. we can perform many changes	2) Tuple objects are Immutab we can't perform any change
3) IF the content is not fixed and keep on changging then we should go for List. Eg: l[1]=70	3) Tuple objects are Immutable. once we create can't change it's content t[1]=70 — ValueError
4) List object can not used as keys for Dictionaries because key should be Hashable and Immutable.	4) Tuple objects can be used as keys for Dictionaries because keys should be Hashable and Immutable.
→ List is Array → heterogeneous elements → Diff^{n} data types, → mutable → Allows Duplicates	

Exception Handling :

- every programming 2 types of Error

① Syntax Error	② Runtime Error
• Programmer is responsible to correct these syntax errors. Once all syntax error are corrected then only program execution will be started.	• Also known as Exception • While executing the program if something goes wrong because of end user input or programming logic or memory problems then we will get Runtime errors.

Q. What is Exception?

- An unwanted & unexpected event that disturb normal flow of program.

- ZeroDivisionError
- TypeError
- ValueError
- FileNotFoundError
- EOFError
- SleepingError
- TyrePlunctured Error

(Types of Errors)

- Types of Exception
  2-types

① Pre-defined Error ② User Defined Error

Python Logging:

• Logging the Exception:

It is highly recommended to store complete application flow and exceptions information to a file

Q Write python program to write exception information to log file.

```
import logging
Logging.basicConfig(Filename='mylog.txt', Level=Logging.INFO)
logging.info("A New Request came:")
try:
    x=int(input("Enter First Number:"))
    y=int(input("Enter second Number:"))
    print(x/y)
except ZeroDivisionError as msg:
    print("cannot divide with zero")
    Logging.exception(msg)
except ValueError as msg:
    print("Enter only int values:")
    Logging.exception(msg)
Logging.info("Request Processing Completed")
```

op

Enter First Number:	10	10	10
second	2	0	ten
	5.0	cannot divide with zero	Enter only int values

OOPS:

Q. what is class?

- In python every thing is an object. To create object we required some model or Plan or Blue print which is nothing but a class.

class define by class keyword

Ex:
```
class student:
    """This is student class with required data"""
print(student.__doc__)
help(Student)
```

- 3 types of variable are allowed.

① Instance Variable (object Level Variable)
② Static Variable (class Level Variable)
③ Local Variable (Method Level variable)

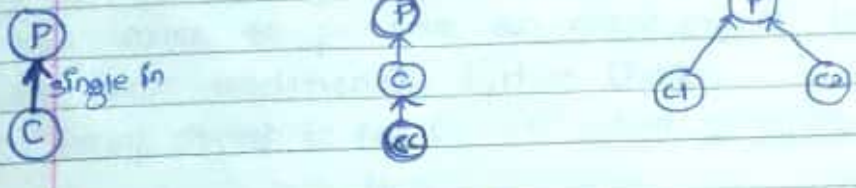
Q. Diff^n betⁿ method & constructor

Method	Constructor
① Name of method can be any name	① constructor name should be always __init__
② Method will be executed if we call that method	② constructor will be executed automatically at the time of object creation.
③ Per object method can be called any number of times	③ Per object, constructor should be executed only once
④ Inside method we can write bussiness logic	④ Inside constructor we have to declare & initialize instance variable.

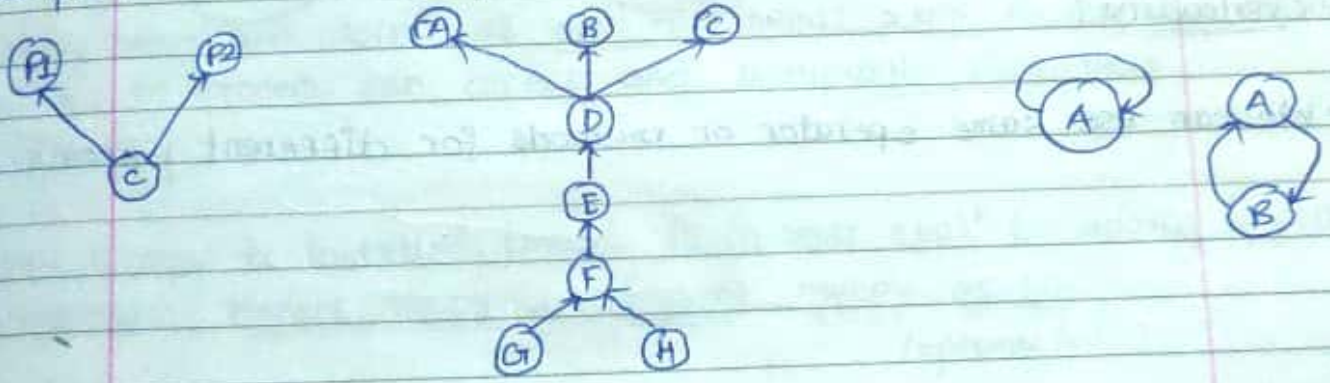
Types of Inheritance :

1. Single Inheritance
2. Multi-Level Inheritance
3. Hierarchical Inheritance
4. Multiple
5. Hybrid
6. Cyclic

1. Single Inheritance 2. Multilevel In. 3. Hierarchical



4. Multiple Inhe. 5. Hybrid Inhe. 6. cyclic Inhe.



Polymorphism :

Poly → many Morphs → forms ⇒ Many forms

Ex:

- + operator acts as concatenation ('Durga' 'Raja') & Arithmatic addn (10+20)
- * operator acts as multiplication & Repetition operator

- Related to polymorphism 4 topics are imp.

① Duck Typing Philosophy of python
② Overloading
  a) Operator
  b) method
  c) Constructor

Overloading :

- We can use same operator or methods for different purposes.

Numpy:

- python library
- used for working with Array
- Numerical Python
- It also has functions for working in domain of linear algebra, fourier transform, matrices.

Q. Why use Numpy?

- In python we have lists that serve the purpose of arrays. but they are slow to process
- Numpy aims to provide an array object that is upto 50x faster than traditional Python lists.
- The Array object in Numpy is called ndarray

Q. Why is Numpy Faster than Lists?

- Numpy array are stored at one contineous place in memory unlike lists, so process can access and manipulate them very efficiently

- Once Numpy is installed, import it in your appl^n by adding import keyword: import numpy or import numpy as np.
  (optional name)

- Diamensions in Array:

① 0-D Arrays:

- 0-D Arrays, or scalars, are the elements in Arrays. Each value of array is 0-D Array.

Ex:
```
import numpy as np
arr = np.array(42)
print(arr)
```
//42

② 1-D Array:

- An Array that has 0-D Arrays as it's elements is called Unidiamensional or 1-D Arrays.

```
import numpy as np
arr = np.array([1,2,3,4,5])
print(arr)
// [1, 2 3 4 5]
```

③ 2-D Array:

```
import numpy as np
arr = np.array([[1,2,3],[4,5,6]])
print(arr)
// [[1 2 3]
   [4,5 6]]
```

④ 3-D Array:

```
import numpy as np
arr=np.array([[[1,2,3],[4,5,6]],[[1,2,3],[4,5,6]]])
print(arr)
// [[[1 2 3]
     [4 5 6]]
    [[1 2 3]
     [4 5 6]]]
```

Pandas:

- Pandas is a python library used for working with Data set
- Used to Analyze data.
- It has fn Analyzing, cleaning, exploring, manipulating data.

Q. Why use Pandas?

- Pandas Allow us to Analyze big data & make conclusion based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.

Q. What can Pandas do?

- Pandas gives answers about the data. Like:
  - Is there corelation between two or more columns?
  - what is average value?
  - Max value?
  - Min value?

Q. what is pandas series?

- Pandas series is like a ~~pandas~~ column in table.
- It is 1-D Array holding Data of Any type

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.series(a)
print(myvar)
```

o/p:

```
0  1
1  7
2  2
dtype : int64
```

- Lables:

If nothing else specified, the values are labled with their index number, first value has 0 index, second value has index 1

Dataframes: It is a 2-D data structure, like a 2-D Array or table with rows and columns.

Scipy:

Q. What is scipy?

- Scipy is a scientific computation library that uses Numpy underneath.
- Scipy stands for Scientific Python.
- It provide more utility functions for optimization, stands and signal processing.
- Like Numpy, Scipy is open source so we can use it freely.

Q. Why Scipy Use?

- Scipy has optimized and added fn that are frequently used in Numpy & Data Science

Q. which lang scipy written in?

- Scipy is predominantly written in python, but few segment are written in C

Matplotlib:

Q. What is Matplotlib?

- It is a low level graph plotting library in python that serves as a visualization utility

Matplotlib Pyplot:

- Most of the matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias.

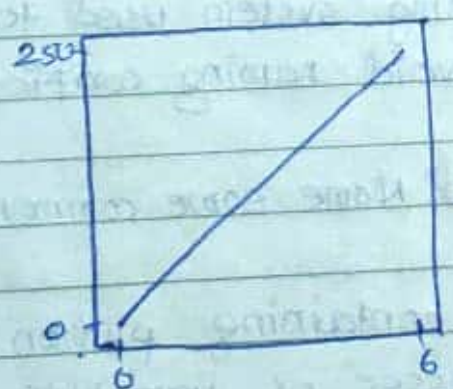
```
import matplotlib.pyplot as plt
```

Ex. Draw a line in a diagram from position (0,0) to position (6,250)

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])
plt.plot(xpoints, ypoints)
plt.show()
```

o/p:



THAI IANT

TOP Interview Questions :-

① Diff" bet" List & Tuple.

List	Tuple
- List are mutable	- Tuple are immutable
- they can be edited	- they cannot be edited
- List are slower Than Tuple	- Tuple are faster than list

② What type of language is python?

→ Python is a scripting language but in general sense, it is a general-purpose pro. lan.

③ What an interpreted langu. ?

→ An interpreted lang. is any programming lang ~~that~~ which is not-machine level code before runtime

④ What is peps?

→ PEP- stand for Python Enhancement Proposal. It is a set of Rules that specify how to format python code for max readability.

⑤ What is namespace?

→ A namespace is a naming system used to make sure that names are unique to avoid naming conflicts.

⑥ What are Python module? Name some commonly used built in modules in python?

→ Python module are file containing python code. The code can either be functions, classes or variables A python module is a .py file containing executable code.

some of the commonly used Built in modules

- os
- sys
- math
- Random
- data time
- JSON

*

Q. What are Global & Local variable?

Global Variable:

Variable declared outside a function or in global space are called global variable. These variable can access by any function in the program.

Local variable:

Any variable declare inside a function. and the variable present in Local space.

```
a=2          ---> # Global Variables
def add():
b=3          ---- # Local. Var
c=a+b
print(c)
add()
// 5
```

shallow copy()

- A shallow copy create a new object which stores the reference of the origional elements.

So, a shallow copy doesn't create a copy of nested objects, instead it just copies the reference of nested objects. This means, a copy process does not recurse or create copies of nested objects itself.

Ex: Create a copy using shallow copy

```
import copy
old_list = [[1,2,3],[4,5,6],[7,8,9]]
new_list = copy.copy(old_list)

print("Old List:", old_list)
print("New List:", new_list)
```

op: Old List: [[1,2,3],[4,5,6],[7,8,9]]
New List: [[1,2,3],[4,5,6],[7,8,9]]

In above program we created a nested list and then shallow copy it using copy() method.

This mean it will create new and independent object with same content. To verify this we print the both old_list and new_list.

Ex: Added new object using shallow copy.

```
import copy

old_list = [[1,2,3], [4,5,6], [7,8,9]]
new_list = copy.copy(old_list)

old_list[1][1] = 'AA'
print("Old list:", old_list)
print("New list:", new_list)
```

changes are done in shallow. Added 'AA' in both

o/p:

```
Old list: [[1,2,3], [4,'AA',6], [7,8,9]]
New list: [[1,2,3], [4,'AA',6], [7,8,9]]
```

Ex: Adding [4,4,4] to the old list using shallow.

```
import copy

old_list = [[1,1,1], [2,2,2], [3,3,3]]
new_list = copy.copy(old_list)
→ append.old_list([4,4,4])
print("Old list:", old_list)
print("New list:", new_list)
```

changes are done

o/p:

```
Old list: [[1,1,1], [2,2,2], [3,3,3] [4,4,4]]
new list: [[1,1,1], [2,2,2], [3,3,3] [4,4,4]]
```

Deep Copy()

A deep copy creates a new object and recursively adds the copyes of nested objects present in the original elements

Ex: Copying a list using deepcopy()

```
import copy
old_list = [[1,1,1], [2,2,2], [3,3,3]]
new_list = copy.deepcopy(old_list)
print("Old list:", old_list)
print("New list:", new_list)
```

o/p⇒

```
Old list = [[1,1,1], [2,2,2], [3,3,3]]
New list = [[1,1,1], [2,2,2], [2,3,3]]
```

Ex: Adding a New nested obj using Deep copy.

```
import copy
old_list = [[1,1,1], [2,2,2], [3,3,3]]
new_list = copy.deepcopy(old_list)
print("old list:")
old_list[1][2] = 'BB'
print('Old list:', old_list)
print("New list:", new_list)
```

changes are done in old_L
But not in New_List

o/p:

```
old_list = [[1,1,1], [2,2,'BB'], [3,3,3]]
New_List = [[1,1,1], [2,2,2], [3,3,3]]
```

Shallow copy	Deep copy
Shallow copy reflects changes made to the new/copied object in the original object	Deep copy does not reflect changes made to the new/copied object in the original object.
OR	
In shallow copy, pointer points to the same copy of object of class	In deep copy, it creates copy of each class obj. inside of the class.

Q. What is the diff^{n} betn Python Array and List

- Array and list in python, have the same way of storing data. but array can hold only a single data type elements. whereas list can hold any datatype elements.

Ex:
```
import array as arr
My_Array = arr.array('i', [1,2,3,4])
My_List = [1, 'abc', 1.20]
print(My_Array)
print(My_List)
// array('i', [1,2,3,4])[1, 'abc', 1.2]
```

single data type held

Any datatype element.

Q. What is __init__?

⇒ __init__ is a method or constructor in python. This method is automatically called to allocate memory when a new obj of class is created. all classes have the __init__ method

```
class Employee:
def __init__(self, name, age, salary);
self.name = name
self.age = age
self.salary = 2000
E1 = Employee("XYZ", 23, 2000)  // E1 is object of class
print(E1.name)                  // __init__ allocate memory for E1
print(E1.age)
print(E1.salary)

// XYZ
   23
   2000
```

Q. What is self? ; (It is hold the obj addr.)

- self is a word which explicit the obj
- self is pointer which refers to object of class
- self is not compulsory → we use any word → But we need pointer → self is variable

⇒ Self is an object of class. In python, this is explicitly included at the first parameter

Q. What is mean by Parameter / Arguments ← both r same.

- Information that are passed into fn.

OR

- A parameter is the variable listed inside the paranthesis in the fn defn.

*Q. difference bet" List, Tuple & set

List []	Tuple ()	Set {}
• Mutable	• Immutable	- Mutable
- It is ordered collection of items.	- It is ordered collection of items	- Unordered collection of item.
- Items of list can be replaced or changed.	- Items of tuple can not be replaced or changed	- Items in ~~tuple~~ set can not be changed or replaced.
- Different Data type	- Different Data types	- Similar Data types
Mylist = ["apple", "cherry"]	Mytuple = ("apple", "cherry")	myset = {"apple", "cherry"}

Q. Why Tuple is Faster than List?

- Tuple are stored in a single block of memory. ~~tuples~~ are immutable so it dosen't required extra space to store new object.
- List are ~~stored~~ in allocated in two blocks: the fixed one with all the python object information and a variable sized both for the data.

Q. Why fn is imp in Python
→ ~~Recursion~~ in python.
→ Reusability

*Q. What are function python?

- A fn is a block of code which is executed only when it is called. to define a python function, the def keyword is used.

```
def newfunc():
    print("Hi")
newfunc()
```
calling fn

Q. What does [: : -1] do? ~~Reverse~~

It is used to reverse the order of Array or a sequence.

- Types of fn in python
  1. Built in fn
  2. Lambda fn
  3. Recursion fn
  4. User defined.

* Q. What are Iterators?

- Iterators are objects which can be traversed through or iterated upon

Q. Generator ?

(yeilding is a crucial fn appln where memory is a constraint)

- Function that return an iterable set of items.
- Also we can use yeild() function

Q. How will you capitalized First letter of string?

"nitam"

string name. capitalized() nitam.capitalize()

// Nitam

Q. Decorator?

- Specific change that we make in python syntax to alter the function easily without changing it's structure

#Ex: def decoratorfun():
return another_fun

Q. Filter ?

- filter method filters the given iterable with the help of a function that test each element in the iterable to be True or not.
- (filters some values Based on conditional Logic) Ex: list(filter(lambda x: x>5, range(8)))

Q. Arguments?

- Information can be passed into functions as arguments.
- Argument are specified after the function name, inside the parantheses we can add many arguments as u want just separate them with comma

```
def my_function(fname);
    print(fname + "Refsnes")
myfunction("Emil")
my_function("Linus")
```

O/p: Emil Refsnes
Linus Refsnes

Q. Why Python is a Interpreted language?

- Because it goes through an interpreter, which turns code you write into the language understood by your computer processor.

Q. What type of Lang? programming or scripting?

- Python is capable of scripting, but in general sense, it is considered as a general-purpose programming language.

Q. What an interpreted language?

- An interpreted Lang. is any programming lang. which is not in machine level code before runtime.

Q. What is a PEP8?

PEP stand for Python Enhancement Proposal. It is a set of rules that specify how to format python code for maximum rediability.

(PEP8 is a coding conversion that lets us write more readble code. it is a set of recommendations)

Q- what is namespace in python?

- A namespace is a naming system used to make sure that names are unique to avoid naming conflicts.

Q. What are Python modules? Name some commonly used built-in modules in python?

- Python modules are files containing python code this code either be functions, classes or variables. A python module is a .py file containing executable code

Built modules are
os, sys, math, random, datetime, JSON

Q.

- Global variable - Declared in outside the function and Global space these variable can accessed by any function in the program.

- Local Variable - Declared in inside the function and Local space. and not in global space

```
a=2 ---------- # Global (Outside)
def add()
b=3 }          # Local (Inside)
c=a+b }
print(c)
add()
```

115

Recursion: When a fn makes a call to itself, it is termed recursion. But, then in order for it to avoid forming an infinite loop, we must have base condition.

```
def facto(n);
if n==1 : return1
return n*facto(n-1)
facto(4)
```

1124

Q. What type of conversion are in python?

- Type conversion refers to the conversion of one datatype into another.

- int() — any datatype into integer type
- float() — " " float type
- ord() — converts char into interger
- hex() — converts int to hexadecimal 0x(12) / 0X(12)
- oct() — " " Octal 0o(12) / 0O(12)
- tuple() — This fn used to convert to a tuple
- set() — This fn ~~used to~~ returns the type after converting set
- list() — This fn returns used to any data type to list type.
- dict() — Used to convert a tuple of order(key, value) into a dictionary.
- str() — Used to convert into into a string.
- complex — (real, imag): This fn converts real number to complex number. (real, imag)

Constructor:

~~Constructor initialize the objects and~~

Constructor are generally used for ~~initiali~~ instanting an object. The task of constructors is to initialize to the data members of the class when an object of class is created.

init() is called the constructor and is always called ~~the constructor~~ when obj. is created.

```
def __init__(self)
    # Body of the constructor.
```

*Q.

Lambda function: Lambda fn is an anonymous fn. this fn can have any no. of parameters, but, can have just one statement.

```
a= lambda a,b: a+b
print(a(5,6))  // 11
```

Q. How do you write **comments** in python?

- Comments in python start with (#) character. However, alternatively at times, commenting is done using docstrings. (string enclosed within triple quotes)

Q. What is the purpose of (is), (not) and (in) operator?

is: returns true when 2 operands are true ("a" is 'a')

not: Returns the inverse of boolean value.

in: checks if some elements is present in some sequence.

Q. What is the usage of help() & dir() fn in python?

Help() fn: is used to display the documentation string and also facilitates you to see the help related to modules, keywords, attributes, etc.

Dir() fn: the dir() fn is used to display the defined symbols.

Q. Explain split(), sub(), subn() methods of "re" module in python.

- To modify strings, python "re" module providing 3 methods.

- **split()** - uses a regex (Regular Expression) pattern to "split" a given string into list.
- **sub()** - finds all substrings where the regx pattern matches and then replace them with a different string
- **subn()** - It is similar to sub() and also returns the new string along with the no. of replacement.

Q. How to add values to python Array?

- Elements can be added to an array using append() extend() and insert(i,x) functions.

```
a=arr.array('d',[1.1, 2.1, 3.1])
a.append(3.4)
print(a)
a.extend([4.5, 6.3, 6.8])
print(a)
a.insert(2,3.8)        # i (index no), x (item)
print(a)
```

o/p:

```
array('d',[1.1,2.1,3.1,3.4])
array('d',[1.1,2.1,3.1,3.4,
          4.5,6.3,6.8])
array('d',[1.1,2.1,3.8,3.1,3.4,
          4.5,6.3,6.8])
```

* Q. How to remove values to a python array?

• Array element can be removed using pop() or remove() method

```
a=arr.array('d',[1.1,2.2,3.8,3.1,3.7,1.2,4.6])
print(a.pop())      // 4.6 => // poped item can be access
prfnt(a.pop(3))     // 3.1
a.remove(1.1)       => //Removed item can not access
print(a)            array('d',[2.2,3.8,3.7,1.2])
```

Q. Packages: collection of modules.

Q. What are python Libraries/modules? Names?

• Labraries are the collection of python modules.

Numpy, Pandas, Matplotlib, Scikit-learn

(np) (pd) (plt) // import numpy as np

alias

import numpy as np

Q. What is split() used for?

split() method used to separate a given string.

```
a= "python program"
print(a.split())   // ['python', 'program']
```

Q. Explain how can you access module in python?

A module is a file containing/consisting of python code
A module can define functions, classes and variables.
A module also having a runable code

Explain How can a access a module in python from 6

Ex: This is an .py file

```
def print_func(par):
    print "Hello", par
    return
```

Q. char() fn Python

The char() method returns a string representing character whose Unicode code point is an integer.

```
char(num)
num: integer value
```

Ex:

```
number = [17, 38, 79]   # char builtin fn.
For number in numbers:
    letter = char(num)   #convert ASCII to character
    print("character of ASCII value.", number, "is", letter)
```

o/p:

character of ASCII value 17 is
character of ASCII value 38 is &
character of ASCII value 79 is O

* Q. ENUM(): enumerate function

- Enum is a class in python for creating enumerations, which are a set of symbolic names bound to unique, constant value. by using type()

```
Ex: import enum
    class Days(enum.Enum):
        sun=1
        mon=2
        Tue=3
    print("The enum member as a string is:", end=" ")
    print(Days.Mon)
```

```
print ("The enum member as a repr is :",end =" ")
print (Days.Sun)
print ("The type of enum member is : ",end=" ")
print (type(Days.Mon))
print ("The name of enum member is : ",end =" ")
print ( Days.Tue.name)
```

op:

The enum member as a string is : Days.Mon
he enum member as a repr is :
The type of enum member is :
The name of enum member is : Tue

Q. **Reduce():** •reduce() fn accepts a fn & a sequence and returns a single value calculated.

reduce() is a function that implements a mathamatical techniques called folding or reduction. reduce() is useful when you need to apply a fn to an iterable and reduce it to a single cumulative value. apply

```
From functools import reduce
numbers = [0, 1, 2, 3+4]
reduce (my_add, numbers)  // my_add & numbers are arguments.
```

o/p:

0+1 = 1
1+2 = 3
3+3 = 6
6+4 = 10, //10

Q. **MAP / map()** •map fn execute a specific fn for each item in an iterable. the item is send to the fn as a parameter

•map(fn, iterables)

map() applies a fn to on all the atoms of iterator given as i/p
An iterators are, list, tuple, set, dict, string an iterable
map object.

it return a new list we

Ex: Using map with Tuple

```
def myMapFunc(n):
    return n.upper()
my_tuple = ('java', 'python', 'c++', 'c')
updated_list = map(myMapFunc, my_tuple)
print(updated_list)
print(list(updated_list))
```

o/p: ['JAVA', 'PYTHON', 'C++', 'C']

The o/p we get is a tuple back with all the values in it are converted to uppercase.

Q. **Pass()**: Pass statement is used as placeholder for future code.

It is a placeholder in a compound statement. If we want to create empty class or fn, this pass keyword helps to pass the control without error.

```
class student:
    pass ------ // passing class
class student:
def fnfo():
    pass ---------- // pass fn
```

Q. **docstring**

The python docstring is a string literal that occurse as the first statement in a module, fn, class or method defn. it provides a convenet way to associate the documentation.

```
# one line docstring
def hello():
    """ A function to greet"""
    return "hello"
```

* Q. **Inheritance** (It allows us to define a class that inherits all the methods & properties from another class)

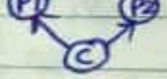
- Inheritance allows one class to gain all the members of another class.
- The class from which we are inheriting is called super class and the class that is inherited is called a derived / child class.
  - Parent class - Base class
  - Child class - Derived class

① Single Inheritance- Where a derived class acquires the member of a single super class.

② Multi-Level Inheritance - A derived class d1 in inherited from base class b1, and d2 are inherited from b2 (base2)

③ Hierarchical Inheritance - From one base class we can inherit any number of child classes.

④ Multiple Inheritance - A derived class is inherited from more than one base class.



Q. How are **classes** created in Python?

- Class: It is like an obj constructor or 'BLUEPRINT' for creating obj
- Class in python is created using the class keyword.

```
class Employee :                    <- constructor
    def __init__(self, name):       <- Argument / parameter
        self.name = name
E1 = Employee ("abc")
print (E1.name)

// abc
```

(class)

Q. Does Python support Multiple Inheritance?

- Multiple Inheritance means that a class can be derived from more than one parent classes.
- python does support multiple inheritance, unlike java.

* Q. What is polymorphism in Python?

- The literal meaning of polymorphism is the condition of occurrence in different form.
- It refers to the use of a single type entity to represent different types in different scenarios

(string) (int)

- + ⇒ concatenation & Arithmatic
- * ⇒ multiplication & Repetition

#Polymorphism in addition operator:

```
num1 = 1
num2 = 2
print(num1+num2)
//3
```

```
str1 = "Python"
str2 = " Program"
print(str1 + " " + str2)
// Python Program
```

* Q. Define Encapsulation in Python? (Wrapping data & methods that work on data within one unit)

- Encapsulation means binding the code & the data together. A (hiding properties data code & data in single unit) python class in an example of encapsulation.

- Q. How do you do Data Abstraction in Python?
- Data Abstrastraction is providing only the required details and hiding the implementation from the word. It can be acheived in python by using interfaces and abstract class.

Q. How to create empty class in python?

- An empty class is a class that does not have any code defined within it's block. It can be created using class pass keyword. pass is a null statement.

```
class a :
    §abc ; abc ; defgh ; pers ; pass
obj(a)
obj = a()                    o/p:
obj.name = "xyz"             // Name = xyz
print("Name = ", obj.name
```

Q. What does an object() do?

Real time entity with an identity, 9th. identity — unique id

- It returns a featureless object that is a base for all classes. Also, it does not take any parameter.

Q. What are the different types of operator?

1. Arithmatic Operators ⇒ +(Add), -(sub), *(Mul), /(Div), %(modulo), //(Floor Div), **(power operator)
2. Relational Operators ⇒ >, <, >=, <=
3. Equality operators ⇒ ==, != (=: Assign)
4. Logical operators ⇒ And, OR, Not.
5. Bitwise operators ⇒ &, |, ^, ~, <<, >>
6. Assignment operator ⇒ +=, -=, *=, /=, %=, //=, **=, &=, |=, ^=, >>=, <<=

Special Operators

1) Identity Ope
   1. is
   2. is not
2) Membership ope
   1. in
   2. not in

Q. Overloading

1) Methods or functions must have the same name and different signature

2) Exaple of compile time polymorphism

3) In this method inheritance, may be or may not be required

4) Performance betn methods within the class

5) Used to in order to add more to the behaviour of methods

6) there is no need of more than one class

Ex:

```
def add (datatype, *args):
    if datatype == 'int';
        answer = 0
    if datatype == 'str';
        answer = ''
    for x in args;
        answer = answer + x
    print (answer)
add ('int', 5, 6)
add ('str', 'Hi', 'Geeks')
// 11
  Hi Geeks
```

Overriding

1) Methods or functions must have the same name and same signature

2) It is example of Run time polymorphism

3) Inheritance, Always required

4) Is Done betn parent class & child class methods.

5) Used in order to change the behaviour of exist methods.

6) there is need of at least of two classes.

Ex:

```
class A:
    def fun1(self):
        print("feature_1 of class 'A'")
    def fun2(self):
        print('feature_2 of class A')
class B(A):
    def fun1(self):
        print("modified feature_1 of
              class A by class B')
    def func3(self):
        print('feature_3 of class B')
obj = B()
obj.fun1()
// modified feature1 of
   class A by class B.
```

Q. Example of [shallow & Deep copy]

```
from copy import copy, deepcopy
list_1 = [1, 2, [3, 5], 4]
# shallow copy
list_2 = copy(list_1)                    # reference to the obj
list_2[3] = 7
list_2[2].append(6)
list_2   // [1, 2, [3, 5, 6], 7]
list_1   // [1, 2, [3, 5, 6], 4]
# deep copy
list3 = deepcopy(list_1)
list_3[3] = 8
list_3[2].append(7)
list_3   // [1, 2, [3, 5, 6, 7], 8]
list_1   // [1, 2, [3, 5, 6], 4]
```

append → new_list

append → old_list

copied items from new_list

append → new_list

Not copied items from newlist

Not Append → old_list

we recursively insert

Q. What is the diff^n bet^n [xrange & range]

xrange → python2
range → python3

- Both are similar in functionality.
- Both generate sequence of integer.
- range() → returns python list
- xrange() → returns an xrange obj

Ex:

```
for i in xrange(10):          # num: from 0 to 9
    print i                   # o/p: 0 1 2 3 4 5 6 7 8 9
for i in xrange(1,10):        # num: from 1 to 9
    print i                   # o/p: 1 2 3 4 5 6 7 8 9
for i in xrange(1,10,2):      # skip by 2 for nxt
    print i                   # o/p: 1 3 5 7 9
```

declare comments : #char

Alphanumeric : isalnum()

Capitalize first letter : capitalize() //Ex: "Nilam"
"nilam".capitalize()
// "Nilam"

How to capitalize full string: swapcase() → Capital to small, Small to capital

Ex: string = "NILAM"
print(string.swapcase())
// "nilam

string = "nilam"
print(string.swapcase())
// "NILAM"

String into lowercase : .lower()

string into upper case : .upper()

string = 'Ayushi'
print(string.lower())
// ayushi
string = 'ayushi'
print(string.upper())
// 'Ayushi'

How to findout which directory you are currently in?

import os
os.getcwd()

# change the current working directory : os.chdir("enter")
os.getcwd()

How do you insert an object at a given index in python?

```
>>>a=[1,2,4]
>>>a.insert(2,3)
>>>a
// [1,2,3,4]
```

How do you reverse list?

- reverse() method : a.reverse()
- slicing : [a[::-1]]

How will you convert a list into a string?

join()

```
nums=['one', 'two', 'three', 'four', 'five']
s=' '.join(nums)
s
// 'one two three four five'
```

How will you convert remove a duplicate element from a list?

- We can turn it into a set to do that (Different types)

Ex: list = [1, 2, 3, 1, 2, 4]
set(list)
// {1, 2, 3, 4}

list :-> Ordered
set :-> Unordered
(Similar data type)

How would you convert a string into an int in python?

- string contain only numeric characters, convert it into an integer using the int() function

```
int('277')
// 277
```

int("string")
// Error

How do you take input in python?

The input() fn takes as an arguments, the text to be displayed for task:

```
a=input('Enter a number')
// Enter a number
```

How to calculate the length of a string?

len()

Ex: len('Ayushi Daya')
// 11

* # Explain List comprehension

• Way to declare a list in one line of code. Lets take look at one such example.

[i for i in range(1,11,2)] // [1,3,5,7,9]

• [i*2 for i in range(1,11,2)] // [2,6,10,14,18]

How do you get all values from a python dictionary?

all keys from dictionary : .keys() method.
values from dictionary : .values() method

What if you want to toggle case for a python string?
(conversion upper to lower)

• swapcase()

'AyuShi'.swapcase()
// aYUsHI

How many arguments can the range() fn take?

• range() fn in python can take up to 3 arguments.

✓ one argument
list(range(5))
// [0,1,2,3,4]

✓ Two argument.
list(range(2,7))
// [2,3,4,5,6]

✓ Three arguments
list(range(2,9,2))
// [2,4,6,8]

How would u randomize the content of list?

• import the fn shuffle() from the module random.

from random import shuffle
shuffle(mylist)
mylist

How do you remove whitespaces: isstrip() method

Ex: 'Nilam_'.isstrip()
// 'Nilam'

How do we execute python?

- Python files first compile to bitecode, then the host execute them

Explain parameter-passing mechanism?

- To pass it's parameter to a fn, python uses pass by reference if u change a parameter within a fn, the change reflects in the calling fn. this is it's default behaviour. however when we pass literal arguments like strings, numbers or tuple, they pass by value because they are immutable

filter: → (This fn lets us keep the values that satisfy some conditional logic) filters in some values based on conditional logic

```
list(filter(lambda x: x>5, range(8)))  // [6,7]
```

* map(): map applies a fn to every element in an iterable.

```
list(map(lambda x: x**2, range(8)))
// [0,1,4,9,16,25,36,49]
```

Reduce(): Reduce, repeatedly reduces a sequence pair-wise until we reach a single value

```
from functools import reduce
reduce(lambda x,y: x-y, [1,2,3,4,5])
// -13
```

delete / Remove	del()	remove()
	list = [3,4,5,6,7]	list = [3,4,5,7]
	del list[3]	list.remove(5)
	list // [3,4,5,7]	list // [3,4,7]

How do you write a file for writing?

```
file = open('tabs.txt', 'w') // writing the file.
file.close() //closing file
```

- 'w' - write
- 'r' - read
- 'wr' / 'rw' - read & write.

diff^n bet^n append() & extend()

append() ⇒ Add an element to the end of list
extend() ⇒ Add another list to the end of list.

Explain try, raise & finally

Three keyword use in exception handling. we put risky code under the try block, use the raise statement to explicity raise an error. & use the finally block to put code that we want to execute anyway.

What good is recursion?

- Need to put in less efforts
- smaller code than that by loops.
- Easier to understood code.

type() : type of objects we're working with ⇒ type()

isinstance() : this takes in two arguments- a value and a ty
if the value is of the value is of the kind of the specified type, it returns True. Else, it returns False

issubclass() : This takes two classes as arguments. If the fi
one inherits from the second. it returns True, Else, if returns False.

What is Monkey Patching?

- It refers to modyfying a class or module at Runtime.
  It extends python code at Runtime.

Ex:

```
from pkg.module import Myclass
def sayhello(self):
    print("Hello")
Myclass.sayhello = sayhello
```

What is the dogpile effect?

- The dogpile effects occurs when cache expires and websites are hit by numerous request the same time. It is triggered because we allowed more than one request to execute the expensive query.
  # prevented using semaphore lock

Q. diff^n bet^n script & program?

• script	• program
• Scripts are often interpreted from sourcecode or bytecode whereas the applications they control are traditionally compile to native machine code.	• The program has an executable from that the computer can use directly to executes the instructions.
• A "script" is code that acts upon some system in an external or independent manner. It can be removed or disabled.	• A "program" is code that constitutes a system.

Q. What do you mean by *args & **kwargs?

*args:

- In case when we don't know how many arguments will be passed to a fn like when we want to passed a list, or Tuples of values, we use *args.

```
def func(*args):
    for i in args:
        print(i)
func(3,2,1,4,7)
```

o/p: 3 2 1 4 7

**kwargs

It takes keyword arguments when we don't know how many there will be.

```
def func(**kwargs);
    for i in kwargs;
        print(i, kwargs[i])
    func(a=1, b=2, c=7)
```

o/p: a 1
b 2
c 7

- The words *args & **kwargs are convention, we can use anything in their place.

Q. What is closure in python?

- A closure in python is said to be occure when a nested fn references value in it's enclosing scope the whole point is there that it remembers the value

```
def A(x);
    def B();
        print(x)
    return B
A(7)()
```

117

Q. What is the iterator protocol?

. The iterator protocol for python declare that we must make use of two functions to build an iterator - iter() and next().

iter() ==> To create an iterator

next() ==> To iterate to the next element

EX:

```
a = iter([2,4,6,8,10])
next(a)
//2
next(a)  //4
next(a)  //6
next(a)  //8
next(a) //8
next(a) --- // Error: (most recent call last)
```

Q. What is frozen set?

set: set is a collection of item, where there cannot be any duplicate a set is a unordered

frozenset: It is a immutable. this means we cannot change values. This also makes it eligible to be used as a key for dictionary.

Ex: set:

```
myset = {1,3,2,2}
myset
//{1,2,3}
```

frozenset:

```
myset = frozenset([1,3,2,2])
myset
//frozenset({1,2,3})
```

Q. What is JSON? Describe in brief how you'd convert JSON data into python data?

- JSON stands for JavaScript object Notation. It is a highly popular data format, and it stores data into NoSQL databases, JSON is built on 2 structures:
  1) A collection of <name, value> pairs
  2) An ordered list of values

Q. Optionally, what statements can you put under a try-except block?

- we have 2 of those:

else: To run a peace of code when the try block doesn't create an exception.

finally: To execute some peace of code regardless of weather there is an a exception.

```
try;
print("Hello")
except;
print("sorry")
else;~~else:~~
print("Oh then")
finally;
print("Bye")
```

o/p:

// Hello
Oh then
Bye

Q. What is Tkinter?

GUI → Graphical User Interface

- Tkinter is a famous python library with which you can craft a (GUI). It provides support for different GUI tools and widgets like buttons, labels, text boxes, radio buttons, and more. These tools and widgets have attributes like dimensions, colors, fronts, colors and more.

```
import tkinter
top = tkinter.TK()
```

Q. GUI - Graphical User Interface (It is a user interface that includes graphical elements, such as windows, icons & buttons)

Q. Explain different ways to create an empty Numpy array?

- First method

```
import numpy
numpy.array([])
// array([], dtype = float64)
```

- Second method

```
numpy.empty(shape=(0,0))
//array([], shape=(0,0), dtype=float64)
```

Q. How is multithreading achieved in python?

- A thread is a lightweight process, and multithreading allows us to execute multiple threads at once. As you know, Python is a multithreaded language. It has a multi-threading package.

Q. What is tuple unpacking?

- A namedtuple will let us access a tuple's elements using a name/label. We use the fn namedtuple() for this, and import it from collections.

max():

This fn is used to compute the maximum of the values passed in it's arguments & lexicographically largest value if strings are passed as arguments.

Ex:- # printing the maximum of 4, 12, 43.3, 19, 100
print ("Maximum of 4, 12, 43.3, 19 and 100 is: ", end=" ")
print (max (4, 12, 43.3, 19, 100))
// Maximum of 4, 12, 43.3, 19 and 100 is: 100

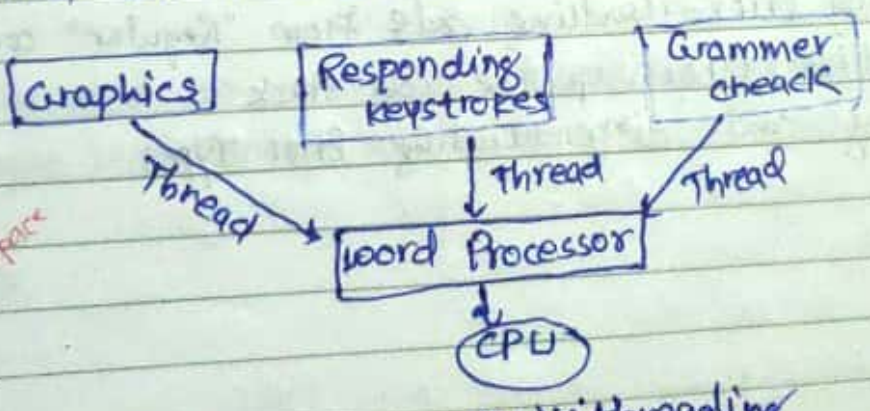
min():

This fn is used to compute the minimum of the values passed in it's argument and lexicographycally smallest value if strings are passed as arguments.

Ex: print ("Minimum of 4, 12, 43, 3, 19 and 100 is: ", end=" ")
print (min (4, 12, 43.3, 19, 100))
// Minimum of 4, 12, 43.3, 19 and 100 is: 4

Multithreading:

It is a technique when multiple threads (created) are spawned by a process to do different tasks, at about the same time, just one after the other. This gives the illusion that the threads are running in parallel, But they are actually run in a concurrent manner.

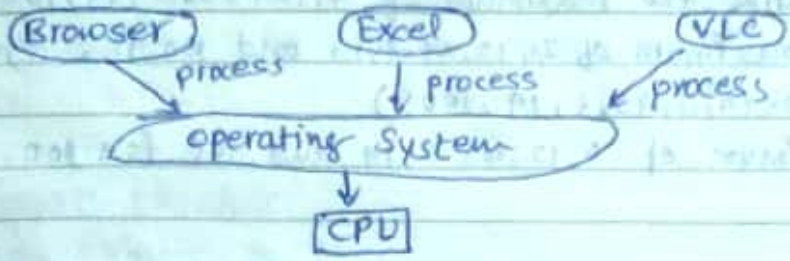


multithreading.

we can't number of task in only one single process space

Multitasking:

- multitasking is a when CPU is provided to execute multiple task at a time.
- Multitasking involves often CPU switching bet^n the tasks, so that users can collaborate with each program together.



multitasking

Q. Advantages of Inheritance?

- Provides code reusability, readability, scalability
- It reduces code repetition.
- We can place all the standard methods and attributes in parent class. These are accessible by the child class derived from it.
- By dividing the code into multiple classes, the applications work better & error identification is easy.

Q. What are the Advantages of Exception Handling?

1) separating Error-Handling code from "Regular" code
2) Propagation Errors up the call stack
3) Grouping and differentiating Error Types

Q. What is zip function in python

- The zip() fn returns zip object, which is an iterator of tuples where the first item in each passed iterator is paired together, and then the second item in each passed iterator are paired together etc.

- If passed iterators have different lengths, the iterator with the least items decides the length of the new iterator.

zip (iterator1, iterator2, iterator3, ...)

iterator1, 2, 3 :> Iterator object that will be joined together.

Q. What is mean by object?

- Python is an object Oriented Programming Language.
- An object is simply a collection of data (variables) and methods (functions) that can acts on those data.
- Silly class is a ~~ob~~ BLUEPRINT for that object.
- example: An integer variable belongs to integer class. An object is a real-life entity.
- Python treated as an object, including variable, functio list, tuple, dictionary, set etc.
- Instance

Q. **Access Specifier**

- Various object oriented language like C++, JAVA, Python control access modifications which are used to restrict access to the variable and methods of the class.

- Most programming language having 3 forms of access modifiers, which are public, private and Protected in a class.

- **Public Access Modifier**

The members of the class that are declared public are easily accessible from any part of the program. All data members and member functions of a class are public by default.

```
# Program to illustrate public access modifier in a class
                    can be access anywhere # Global Var
class Geek:
    # constructor
    def __init__(self, name, age):
        # Public Data members
        self.geekName = name      # public Data Member
        self.geekAge = age
    # Public member function      # Method of class
    def displayAge(self):
        # Accessing public data member
        print("Age:", self.geekAge)
# creating object of class
obj = Geek("R2J", 20)
# accessing public data members
print("Name:", obj.geekName)
```

#calling public member funct" of the class
obj.displayAge()

o/p: Name: R2J
Age: 20

• Protected Access Modifier

The members of the class that are declared protected are only accessible to a class derived from it. Data member of a class are declared protected by adding a single underscore '_' symbol before the data member of the class.

ex: (_name, _roll, _branch)

• Private Access Modifier

The members of a class that are declared private are accessible within the class only, private access modifier Data members of a class are declared private by adding a double underscore '__' symbol before the data member of that class

ex: (__name, __roll, __branch)

Q. What are function like Random ?

- Python has a built-in module that you can use to make random numbers.

- The Random module has a set of methods.

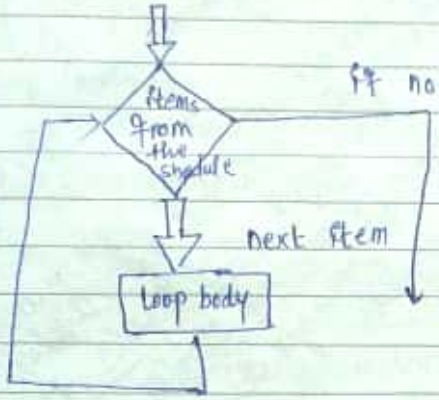
Method	
random()	Returns a random float number between 0 and 1
uniform()	Returns a random float number between two given parameters
sample()	Returns a given sample of a sequence
shuffle()	Takes a sequence and returns the sequence in a random order.
choice()	Returns a random element from the given sequence
choices()	Returns a list with a random selection from the given sequence
seed()	Initialize the random number generator.
setstate()	Restores the internal states of the random number grater.
getrandbits()	Returns a random number representing the random bits
gauss()	Returns a random float number based on the Gaussian distribution (used in probability theories.

Q. What is mean by for loop

→ for loop is used to iterate the statements.

→ frequently used to traverse the data structure like list, tuple or dict.

for iterating_var in sequence:



fn

Built in fn

→ Absolute
absolute = = abs(num)

→ All fn (list, tuple, Dict)
all = = all(iterable)

→ Ascii ... (string, list)
asci = ascii(object)

→ binary = = bin(num) ... (parameter)

→ bool = = bool([values]) ... (parameter)

→ dictionary = = dict()

→ enumerator = = enumerate(iterable, start=0)

→ eval = eval(expression, global = none, local = none)

→ filter = = filter(function, iterator)

→ getatt() = = getattr(object, name)

→ help = = help()

→ length = = len(s)

→ max = max()

→ min = min()

→ Recursion

Q. What are the core default modules available in python?

- email ⇒ helps to parse, handle & generate email message
- string ⇒ Contains fn to process standard python string
- sqlite3 ⇒ Provide methods to work with SQL database
- xml ⇒ Enables xml support
- logging ⇒ Adds support for Log classes & methods
- traceback ⇒ Allows to extract and print stack trace details

Q. Python program to print all prime numbers in an interval

• Python program to display all the prime numbers within an interval.

```
lower = 900
upper = 1000

print("Prime numbers between", lower, "and", upper, "are:")

for num in range(lower, upper + 1):
    # all prime numbers are greater than 1
    if num > 1:
        for i in range(2, num):
            if (num % i) == 0:
                break
        else:
            print(num)
```

Output:

Prime numbers between 900 and 1000 are:

907
911
919
929
937
941
947
953
967
971
977
983
991
997

Q. Python program to find the fractional of a Number

```
# Python program to find the factorial of a number provided by user.

# change the value for a different result.
num = 7
# To take input from the user
# num = int(input("Enter a number : "))

factorial = 1

# check if the number is negative, positive or zero.
if num < 0:
    print("sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        factorial = factorial * i
    print("The factorial of", num, "is", factorial)
```

output:

The factorial of 7 is 5040

Q. Display the multiplication Table:

```
# Multiplication table (from 1 to 10) in python

num = 12
# To take input from the user
# num = int(input("Display multiplication table of? "))

# Iterate 10 items from i = 1 to 10

for i in range(1, 11):
    print(num, 'x', i, '=', num*i)
```

O/p:

12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120

Q. Print the fibonacci sequence.

```
# Program to display the Fibonacci sequence upto n-th term

nterms = int(input("How many terms? "))

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
    print("Please enter a positive integer")
elif nterms == 1:
    print("Fibonaci sequence upto", nterms, ":")
    print(n1)
else:
    print("Fibonacci sequence:")
    while count < nterms:
        print(n1)
        nth = n1 + n2
        # update values
        n1 = n2
        n2 = nth
        count += 1
```

o/p:

```
How many terms? 7
Fibonacci sequence:
0
1
1
2
3
5
8
```

Python Program to check Armstrong Number

- Tuple
- Tuple
- Tuple

- A positive integer is called an Armstrong number of order [n] if

$abcd\ldots = a^n + b^n + c^n + d^n + \ldots$

- In case of an Armstrong number of 3 digits, the sum of cube of each digit is equal to the number itself.

for example: $153 = 1*1*1 + 5*5*5 + 3*3*3$
// 153 is Armstrong number

- Check Armstrong number (for 3 digits)

```
# take input from the user
num = int(input("Enter a number: "))

# initialize sum
sum = 0

# find the sum of the cube of each digit
temp = num
while temp > 0:
    digit = temp % 10
    sum += digit**3
    temp //= 10

# display the result
if num == sum:
    print(num, "is an Armstrong number")
else:
    print(num, "is not an Armstrong number")
```

o/p:

```
Enter a number: 663
663 is not an Armstrong number

Enter a number: 407
407 is an Armstrong number
```