**Questions:**

what is python and its features?

what is object,identity, type?

How memory allocation happend?

What are numbers?

What are build in types in numbers?

Does float values return a exact values in arthmatic operations?

How to import a package and what are the ways to import packages?

what is tuble?

how to declare a tuble with the single element?

Does tuble allow item assigment and indexing?

Ans: it will not allow and through an error

what is a list?

difference b/w list and tuble

Ans: Immutable and immutable. Faster and slower as tuble is unchangable.

List will mentioned in square brackets and tuble in round brackets.

What is set?

What is remove between remove and discard in set?

What is dict? What can be the type key?

How to use pop and set default item in dict?

What are the conditional statements and explain with an example?

What are the diff types of loops and explain syntax of while and for loop with else?

What is the usage of break and continue?

How to create function in python?

What is the usage of \*args and \*\*kwargs?

What is return statement?

What is oops and does python support OOPS?

What is class and how you create a class?

What is \_\_ int\_\_ method?

What is pass keyword?

How to see name space of the object?

What is instance object and what is self?

What are class variables?

What is difference b/w instace,class and static method?

How to Call a init method through a class method?

Usage of day time module?

What is generators and how to fetch value from generator outside?

What are iterators, create a user defined iterators?

Does Python support inheritance and explain the diff. types of inheritance?

What is super and explain with the usage?

What is method over writing?

What do you mean by MRO?

What are abstract classes and how to implemented?

**List:**

accpeting values from the input

fname = input("enter the name")

age = input("enter the age")

= int(input("enter the age"))

list.pop() remove() count() copy() reverse() copy() Sort() del

list are odered

list and dic are unodered.

=============================================================================================

**SET:{}**

Unordered

can't store duplicate values

set.add() it can take only hasble types. It cant take mutable types.

set1.pop()

print(set1.pop())

set.remove()

set.discard()

remove will throw an error

discard will not thrown an error.

Dictionary: {}

dict = {key: value, key:value}

unorder

key is always unique.

dict. keys() values() items()

dict.get(key)

dict2.pop()

POP()

Popitem() **it will remove the last element in dict in the version of python 3.7 but in version 2 randomly removes it.**

Setdefault()

Clear()

Len(dict2)

Dict2.pop(1000, ‘not present’)

Del

Dict2 =

+=======================================================================

Conditionals statements

a==b

a!=b

a>b

a>\_b

a<b

if a<b:

print("a greater than b")

else:

print (" b greater than a")

print("a greater than b") if a>b else print("b greater than a")

Condtionals:

1. if

2. elseif (elif)

3. else

Loops:

While

for

then using break and continue statements.

**For loop:**

Print(help(print))

Range(9)

Print(range(9)) which gives the value (0 to 8)

You can use else condition in for loop statement:

for x in range(0, 9):

print(x, end = '\t')

for y in range(0,9,2):

print(y, end = ' ', sep=':')

for x in range(0,9):

print(x)

else:

print("for loop done")

print(x)

list2 = ['apple', 'orange', 'banana', 'graphes']

for fruit in list2:

print(fruit)

str1='apple'

for char in str1:

print(char)

dict2 = {1:'apple', 2:'banana', 3:'mango'}

for key in dict2.keys():

print(key)

for key,values in dict2.items():

print(key, values, sep=':')

for fruit in list2:

if fruit == 'orange':

continue

print(fruit)

for fruit in list2:

if fruit == 'orange':

break

print(fruit)

**Questions:**

1. **list1 = [12,13,19,21,26] find odd and even numbers**
2. **find factorial numbers**
3. **str1 = (“Hello World HieE”)**
4. **input the string from the user and count the number of voles**
5. **take the input a string from the user and reverse the string using slicing and without slicing.**
6. **Give a number state whether is prime numbers or not**

str1 = ("Hello World HieE")

count=0

for char in str1:

if char.isupper():

count+=1

print(count)

str1 = "Hello WOrld hie"

vowels =('a', 'e', 'i', 'o', 'u')

for char in str1:

if char in vowels:

count+=1

print(count)

list1 = [12,13,19,21,26]

add\_list[]

even\_list[]

for val in list1:

if val%2==0:

even\_list.append()

else:

odd\_list.append()

print(even\_list)

print(odd\_list)

num=5

fact=1

for i in range(i, num+1):

fact = fact\*i

print(fact)

**Functions:**

**Find the second max number from list(3,5,6,6,4)**

**Strings:**

str1 = "Hello World"

print(type(str1))

print(str1+str1, end='\t')

print(str1+' '+str1, end='\t')

print(str1\*2)

print(str1\*3)

print(len(str1))

print(str1[6])

print(str1[0:8:2])

print(str1[::])

print(str1[4:14])

print(len(str1[-11:-5]))

print(str1[::-1])

print(str1[-11:-5:-2])

print(str1.split('l'))

print(str1.upper())

str2 = “Hello World abc skd qwerty xyz”

print(str2.split())

print(str2.split(‘ ‘, maxsplit=2)

str2 = " Vignesh Pandiyan "

print(str2.strip()) “delete the spaces of both the side”

print(str2.lstrip()) “delete spaces from left”

print(str2.rstrip()) “delete spaces from right”

print("my name is {} and coming from {}" .format("vignesh", "Madurai"))

str3 = Vignesh

print(str3.captialize())

str5 = "Hello world"

#print("the position of l is :{}" .formart(str5.find("l")))

print(str5.find("l")) “find the position of “l” in index format”

print(str5.count("o")) “it will count how many times “o” used in

print(“the position of l is :{}” .format(“str5.find(“l”)))

**membership operators:**

in and not “it will consider as a Boolean statement like true or false”

print("wo" in str5)

print("qwerty" not in str5)

**Generators:**

**it generates the values when users need.**

**Iterators:**

tuple1 = ("iphone", "apple")

iter\_tuple = iter(tuple1)

print(type(iter\_tuple))

print(next(iter\_tuple))

print(next(iter\_tuple))

str1="hello"

str2="world"

print(int.\_\_add\_\_(4,5))

print(str.\_\_add\_\_(str1,str2))

class MyIter:

def \_\_iter\_\_(self):

self.a = 1

return self

def \_\_next\_\_(self):

if self.a <=5:

x=self.a

self.a+=1

print("inside next method")

return x

else:

print("condition failed")

raise stopIteration

mynumbers = MyIter()

MyIter = iter(mynumbers)

print(next(mynumbers))

print(next(mynumbers))

print(next(mynumbers))

print(next(mynumbers))

print(next(mynumbers))

print(next(mynumbers))

OOPS:

Class: is a blueprint or structure of all methods

**Self – instance variable itself**

**int**

**Class can be accessable by instance as well as class.**

**List1 = [10, 12, 18, 20, 18, 6, 20]**

**Find the second highest score**

**Methods:**

**Instance method:**

It will use the argument called as “self”

Used to access the instance variable

**Class method:**

Class have the **decorator** “@classmethod”

It will use the argument called as “cls”

Used to access the class variable

**Static method:**

**Not connecting to any class or instance “self or cls”**

have the **decorator** “@staticmethod”

logically make some sense

**Inheritance:**

Inheritance allows us to define a class that inherits all the methods and properties from another class.

**Parent class** is the class being inherited from, also called base class.

**Child class** is the class that inherits from another class, also called derived class.

**Types:**

Single

Multiple

Multilevel

Hybrid

Hierarical

**Method over writing:**

**Method resolution order: MRO**

It shows the order in which over written methods are being in inheritated classes.

Print(help(ch))

**Abstract method:**

**File handling:**

File1 = open

*Missing topics:*

*Readbinary and writebinary*

*Multithreading, Lambda, function,*

*Contest manager*

**Reduce function:**

reduce(function, sequence)

1. First and second element in sequence is taken
2. Function is applied on these two elements
3. Same is carried forward to all the elements till here are no elements in sequence.
4. Final result is returned.

from functools import reduce

list1 = [1, 2, 3, 4, 5]

result = reduce(lambda x,y : x + y, list1)

result1 = reduce(lambda x,y : x if x > y else y, list1)

print(result)

print(result1)

**#Zip function**

tuple1 = (11,12,13)

list1 = [100,200,300,400,500]

list2 = ["orw", "vv", "ee", "ww"]

for i, m, n in zip(tuple1, list1, list2):

print(i,m, n)

#collections package

# from collections import deque

# list1 = deque([1,2,3,4])

# list1.append(6)

# list1.appendleft(-9)

# print(list1)

# print(list1.pop())

# print(list1.popleft())

**#OrderedDict**

from collections import OrderedDict

od = OrderedDict()

od['a'] = 1

od['b'] = 2

od['c'] = 3

od['d'] = 4

print (od)

for key, value in od.items():

print(key, value)

print("\nAfter deleting")

od.pop('c')

print(od)

**#defaultdict**

**# from collections import defaultdict**

**# vehicle = defaultdict(lambda: 'Train')**

**# vehicle['vicky'] = 'SkodA'**

**# vehicle['vinod'] = 'Benz'**

**# print(vehicle)**

**# print(vehicle['vicky'])**

**# print(vehicle['sasi'])**

**#Counter**

**from collections import Counter**

**list1 = [11,22,33,44,11,22,33,77]**

**print(Counter(list1))**

Home work:

Std\_list = [student1, 2…., student10]

Marks = [10, 12,78,90,43,50,60,34,78,25]

Find top 5 students names

Find least 5 students names

Find names b/w 25 to 75

Line1 = “Today is Tuesday”

Line2 = “Tomorrow is Wednesday”

Find common words from both lines

Find only unique lines from both lines.

**#namedtuple**

#Like dictionaries they contain keys

#that are hashed to a particular value. But on contrary,

#it supports both access from key value and iteration,

#the functionality that dictionaries lack.

from collections import namedtuple

student = namedtuple('student',['name', 'age', 'DOB'])

S = student('Vicky', '29', '18FEB1990')

print("The student age using index is : ", end = "")

print(S[1])

print("The student name using keyname is : ", end = "")

print(S.name)

print("The student DOB using getattr() is : ", end = "")

print(getattr(S, 'DOB'))

**#frozenset**

#Sets are mutable but frozen sets are immutable,

#both are unorder and can't accept duplicate values

Fav\_subject = ["OS", "DBMS", "Python"]

f\_subject = frozenset(Fav\_subject)

print(f\_subject)

student = {"name": "Vicky", "age": "29", "gender": "Male"}

key = frozenset(student)

print('The frozen set is :', key)