1. **What do you know about python**
2. **What are the key features of Python?**
3. **What is PVM (Python Virtual Machine)?**
4. **What is the main difference between an interpreter and a compiler?**
5. **What is Variable?**
6. **What are local variables and global variables in Python?**
7. **What is identifiers?**
8. **What is 'self 'in Python?**
9. **What are the reserved keywords in python ?**
10. **What is ‘init’ Keyword In Python?**
11. **What is pyhton path ?**
12. **What is type-Conversion in python?**
13. **What is diffrance between .py files and .pyc files ?**
14. **Is indentation required in python?**
15. **What is namespace in python?**
16. **What does this mean: args, *kwargs?**
17. **What are the parameters and arguments?**
18. **How do any() and all() work?**
19. **What is difference between IS operator and == operator?**
20. **What are packing operators in Python?**
21. **What are unpacking operators in Python?**
22. **Explain about Ternary Operator or Conditional operator and how used?**
23. **pip ?**
24. **What is pep 8?**
25. **What is module & package ? Name some commonly used built-in modules in Python?**
26. **Explain os and sys modules ?**
27. **Difference Between Modules and Packages in Python**
28. **What is pickling and unpickling?**
29. **What are the built-in data types in Python?**
30. **list vs array ?**
31. **What is the difference between list & Dict?**
32. **List Comprehension vs Dictionary Comprehension in python?**
33. **What’s the difference between list and tuple?**
34. **What is range and Xrange() ?**
35. **What is the difference between set and frozen set?**
36. **Explain function list, tuple and dict ?**
37. **Explain some methods of the list.**
38. **`Extend()` ?**
39. **What is the difference between append and extend?**
40. **Explain some methods of string?**
41. **What’s the negative indexing in lists?**
42. **What is slicing in Python?**
43. **What are docstrings in Python?**
44. **How slicing work string manupulation ?**
45. **How to get list of all the key in dict ?**
46. **How can you sort a dictionary by key, alphabetically?**
47. **In Dict duplicate key is not allowed. But values can be duplicated. If we trying to add new entry (Key-value pair) with duplicate key what will be happened?**
48. **Can you cancatenate Two tuple ?**
49. **How is memory managed in Python?**
50. **What is Garbage Collection ?**
51. **Memory allocation in Python ?**

52. What is heap and stack memory in Python?
53. What is heap space and stack space in Python
54. What is a function in Python?
55. What is a recursive function?
56. Explain Break, Continue and Pass Statement?
57. What is the lambda function?
58. What's the difference normal function and lambda function?
59. What is Generator explain with example ?
60. What are iterators?
61. Difrance between Iterator and Generator
62. Normal Functions vs Generator Functions?
63. What is Decorators and give example ?
64. How is exception handling performed in Python?
65. What are diffrent way to read and wirte python file ?
66. What is Open & Open Statment ?
67. Abstraction in python?
68. Abstraction vs Encapsulation
69. What is Inheritance In python ?
70. What is multithreading and multiprocessing in python?
71. Thread ?
72. What is GIL. explain ?
73. What Is Yield In Python?
74. How do delete a given file in Python?
75. What is shallow and deep copy?
76. Explain Pandas and Numpy?
77. Diffrance between seriese and vactors and list python ?
78. Explain about map () , reduce() ,filter() function ?

##

1. What do you know about python ?

- Python is a popular programming language, similar to the English language.
- Python is an **object-oriented**, **interpreted**,**high-level** programming language.

[Object-Oriented Language : allows you to define classes to create objects. such as strings, lists, dictionaries,**]**

[Python is an interpreted language : source code of a Python program is converted into bytecode then executed by Pvm.**]**

[High-level : it's easy for humans to understand.**]**

It is used for

Web Applications, Machine Learning, Software, system scripting, Software development.

2. What are the key features of Python?

Free and Open-Source , Easy to Read ,Easy to Code

1. **Dynamic Memory Allocation** : [internally managed by the Heap data structure. "Heap" memory, also known as "dynamic" memory is an alternative to local stack memory.]
2. **Interpreted Language:** Python code is executed line by line at a time.
3. **Integrated language:** we can easily integrate Python with other languages like C, C++, etc.
4. **Easy to Debug** : You can trace error.
5. **Python is dynamically typed.** It check the types of variables at run-time.
   ex: x=111 and then x="rupesh" without error.
   Some dynamically typed languages are Python, JavaScript, Ruby, etc.

- *Statically typed languages check the types of variables at compile-time.*
- *Some statically typed languages are C++, C, Java, etc..,*

---

3. What is PVM (Python Virtual Machine)?

- Python Virtual Machine (PVM) .
- **The role of PVM is to convert the byte code instructions into machine code.**
- so the computer can execute those machine code instructions and display the output.
- Interpreter converts the byte code into machine code and sends that machine code to the computer processor for execution.

4. What is the main difference between an interpreter and a compiler?

- Interpreter translates one statement at a time into machine code.
- Compiler translates the entire code at a time into machine code.

5. What is Variable?

- Variable is a container to store values.
- Variable is the name given to a memory allocations in a program is used to hold a value.
- Ex: A = 234, B = "Mango" where A & B variable

6. What are local variables and global variables in Python?

- **global variables :**
- it is Variables declared outside a function or in global space. i/e called.
- These variables can be accessed by any function in the program.
- **local variables :**

- Any variable declared inside a function is known a local variable.
- This variable is present in the local space and not in the global space.
- **ex:** a = 2 ---->> Global Variable

  def add( ):

  b = 3 ---->> Local Variable

Local Variable	**Global Variable**
It is decleared inside function.	It is decleared outside function.
This variable is present in the local space and not in the global space.	These variables can be accessed by any function in the program.
It is stored on the stack unless specified .	It is stored on a fixed location decided by compiler.
If is not initialized , a garbage value is stored.	If it is not initialized zero is stored is as default.
def add( ): b = 3 ---->> Local Variable	**ex:** a = 2 -->> Global Variable

7. What is identifiers?

- we want to give an entity a name, that's called identifier.
- identifiers are the name given to variables, classes, methods, etc
- identifier is only used to identify an entity uniquely in a program at time of execution.
- Identifiers are case-sensitive and Whitespaces are not allowed.
- We cannot use special symbols like !, @, #, $, and so on.

8. What is 'self 'in Python?

- The 'self' prameter is a refrence to the current instance of the class.
- it is used to accesss variable that belongs to the class.
- ex :

```
class Person :
    def__init__(self,name,age):
        self.name = name
        self.age = age

    def info(self):
        print(f"My name is {self.name}. I am {self.age} years old."}
c = Person("Rupesh",30)
c.info()
```

output :
My name is Rupesh. I am 30 years old.

9. What are the reserved keywords in python ?

- Reserved words in python known as Keyword's.
- It cannot be used as variable names, function names, or any other identifiers.
- find reserved keywords in python : **help("keywords")**
- Print reserved keywords in python : **import keyword >>> print(keyword.kwlist)**
- like True,False,Break,Continue,if,is,None,and,import,for,elif,else,lambda etc.

10. What is 'init' Keyword In Python?

- __init__ is a method or constructor in Python. you can used as a file or as function both.
- like: __init_.py file and __intit() function.
- This method is automatically called to allocate memory when a new object/ instance of a class is created.
- All classes have the __init__ method.
- ex:

```
class Person:
def __init__(self,name):
    self.name = name
#sample menthode or constructor
def say_hi(self):
    print("hi,my name is',self.name)
p = Person("Rupesh")
p.say_hi()
```

o/p : hi, my name is Rupesh

11.What is pyhton path ?

- It is an environment variable which is used when a module is imported.
- PYTHONPATH is also looked up to check for the presence of the imported modules in various directories.
- The interpreter uses it to determine which module to load.
- pythonpath varibale hold a string with name of various directories that need to be added the **sys.path** directory list by python.

12. What is type-Conversion in python?

type conversion convert of one data type into another.
There are two types of conversion: implicit and explicit.

I. **implicit type conversion :**

- Python interpreter automatically converts one data type to another without any user involvement.

II. **explicit type conversion :**

- data type is manually changed by the user as per their requirement.
- there is a risk of data loss since we are forcing an expression to be changed in some specific data type.
- **type-Conversion like : int () ,float() ,str() ,eval () , oct() , etc.**

13. what is diffrance between .py files and .pyc files ?

ans :

- .py files contain source code of the program.
- .pyc files contain byte code of your program.
- python complies .py files and save as .pyc files.
- the .pyc files contain complied bytecode of python source files, this code run by pyhon virtual machine.

14. Is indentation required in python?

- yes indentation required in python .
- It specifies a block of code.
- All code within loops, classes, functions, etc is specified within an indented block.
- It is usually done using four space characters.
- If your code is not indented necessarily, it will not execute accurately and will throw errors as well.

15. What is namespace in python?

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

16. What does this mean: args, *kwargs?

- when you are not clear how many argument you need to pass a particular function, then we use *args and **kwargs.

- *args : represent varied number of arguments. it is used to add together the values of multiple arguments.

- **kwargs : represent arbitrary number of argument. that are pass in function . **kwargs are stored in the dict.

- Special Symbols Used for passing arguments:-
  *args (Non-Keyword Arguments)
  **kwargs (Keyword Arguments)

*args **ex :**	**kwargs **ex :**
def sum(*args): total = 0 for a in args : total = tatal + a print(total) sum(1,2,3,4,5) output : 15	def show(**kwargs): print(kwargs) show(A = 1, B = 2, C = 3) Output : {'A' : 1, 'B' : 2 ,'C' : 3}
The function used for this process is **pickle.dump()**	The function used for this process is **pickle.load()**
Pickling, also called serialization, involves converting a Python object into a series of bytes which can be written out to a file.	Unpicking, or de-serialization, does the opposite–it converts a series of bytes into the Python object it represents.

17.What are the parameters and arguments?

1. **Parameters** : names listed in the function definition.
2. **Arguments** : values passed to the function while call on.

- different types of arguments in Python: mainly 4 type

1. Positional Arguments:

normal arguments that we define in user-defined functions i.e. called Positional Arguments.

2. Default Arguments:

we can provide the value to the arguments in the function definition itself as default value.

When the user didn't pass the value, the function will consider the default value.

3. Keyword Arguments :

we can specify the name of the arguments while call on the function and assign values to them.

keyword arguments help us to avoid ordering which is mandatory in positional arguments.

4. Arbitrary Arguments :

It use collect a bunch of values at a time when we don't know the number of arguments that

function will get.

We * and ** operators in the function definition to collect the arguments.

18.How do any() and all() work?

- **Any :** takes a sequence and returns true if any element in the sequence is true.
- **All :** returns true only if all elements in the sequence are true.

Any [second item (True) and will return True.]	**All** [first item (False) and will return False]
print (any([False, True, False, False])) o/p : True	print (all([False, True, True, False])) o/p : False

19. What is difference between IS operator and == operator?

- **Is operator** : checks identity
- is >> keyword is used to test if two variables refer to the same object.
- returns True if the two objects are the same object.
- == **operator** : checks equality.
- == used to compare two values or expressions.
- It is used to compare numbers, strings, Boolean values, variables, objects, arrays, or functions. The result is TRUE if the expressions are equal and FALSE otherwise. .
- Example:

```
L1 = [10,20,30,40]
L2 = [10,20,30,40]
Print (L1 is L2) # False
Print (L1 = = L2) # True
L3 = L1
Print (L1 is L3) #True
Print (L1 == L3) #True
```

20.What are packing operators in Python?

- packing operators are used to collect multiple arguments in functions. They are known as arbitrary arguments.

21. What are unpacking operators in Python?

- unpacking operators in Python is * and **
- * unpacking operator is used to assign multiple values to different values at a time from sequence data types.
- like list , tuple , range

22. Explain about Ternary Operator or Conditional operator and how used?

- it is to show the conditional statements.
- This consists of the true or false values with a statement that has to be evaluated for it.
- so it looks similar to an "if-else" condition block.
- Syntax: **[if_true] if [expression] else [if_false]**
- **ex:** x < y else
- The expression gets evaluated like if x<y else y, in this case if x < y is true then the value is returned as big=x and if it is incorrect then big=y will be sent as a result.

23. pip ?

pip is pakage manager in python .

pip install < pakage_name >

24. What is pep 8?

- PEP stands for Python Enhancement Proposal.
- It is a set of rules that specify how to format Python code for maximum readability.

25. What is module & package ? Name some commonly used built-in modules in Python?

Module :

- The module is a simple Python file.
- that contains collections of functions and global variables and with having a .py extension file.
- ex: import < module_name > like numpy,pandas etc.

Package:

- It is simple directory having collection of modules.
- ex: from my_package.abc import a
- from pyspark.sql.types import *

- Some of the commonly used built-in modules are __main__ ,datetime ,zipfile, os , sys etc.

---

26. Explain os and sys modules ?

- ***The os module is responsible*** for the interaction between the program and the operating system, and provides an interface for accessing the bottom layer of the operating system.
- ***The sys module is responsible*** for the interaction between the program and the Python interpreter and provides a series of functions and variables for manipulating the runtime environment of python.

27. Difference Between Modules and Packages in Python

Modules	Packages
The module is a simple Python file that contains collections of functions and global variables and with having a .py extension file.	The package is a simple directory having collections of modules.
A module is a single file (or files) that are imported under one import and used.	A package is a collection of modules in directories that give a package hierarchy . In package also contains sub-packages inside it.
Ex: import datetime	Ex : from my_package.abc import a

28. What is pickling and unpickling?

pickling : python object sequence into byte code.
unpickling : byte code converted into python object sequence.

pickling	**unpickling**
Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using dump function, this process is called pickling.	While the process of retrieving original Python objects from the stored string representation is called unpickling.
The function used for this process is **pickle.dump()**	The function used for this process is **pickle.load()**

Pickling, also called serialization, involves converting a Python object into a series of bytes which can be written out to a file.	Unpicking, or de-serialization, does the opposite–it converts a series of bytes into the Python object it represents.

29. What are the built-in data types in Python?

Mutable **such as** list,dict,set	**Immutable** **such as** str,number,boolen,frozenset
If the value can change, the object is called mutable.	If the value cannot change, the object is called immutable.
list[ ] collection which is ordered.	String , tuple( ), boolen ( )
dict{ } is a collection which is ordered**	int( ) , float( ) , complex( )
set( ) collection which is unordered.	frozenset( )

Sequence Types: list[ ] ,range() ,tuple( )
collection data types : *List , Tuple ,Set , Dictionary*

30. list vs array ?

list	array
list can store diffrent values of data types.	it can only consist of value of same data type.
it is cannot handle direct arthematic operation.	it is can be handle direct arthematic operation.
it consume large memory .	it consume less memory .
it is suitable for storing longer sequence of data items.	it is suitable for storing shorter sequence of data items.

31.What is the difference between list & Dict?

list [ ]	**Dict { }**
list is mutable object	Dict is an mutable object. key-value pair.
list in order of the elements In list duplicate objects are allowed.	Dict unorderd of element. In Dict duplicate keys are not allowed. But values can be duplicated.

List slower than dict when we lookup in 100 items.	dictionary is 6.6 times faster than a list when we lookup in 100 items.
Lst = ['a','b',10,20,5]	Dict = {1:'A',2:'B',3:'C',4:'D'}

32. List Comprehension vs Dictionary Comprehension in python?

List comprehensions	**Dict comprehensions**
List Comprehension is methode offers a shoter syntax when you want to create a new list based on the values of an existing list.	Dict Comprehension is methode for transforming one dict into antother dict based on {key:value pair} in iterable.
it is in square braket [ ]	it is in curly braket { }
Syntax of List Comprehension: NewList = [expression(element) for element in oldList if condition]	Syntax for Dict Comprehension: {key: value for (key, value) in iterable}
Example: List = [i for i in range (11) if i % 2 = = 0] Print (list)	keys =['a','b','c','d','e'] values = [1,2,3,4,5] myDict = {k:v for (k,v) i zip (keys, values)} print (myDict)

33.What's the difference between list and tuple?

Both list and tuple are used to store the collection of objects.

list [ ]	**tuple ( )**
list is mutable object	tuple is an immutable object. read only.
Lists consume more memory.	tuple consume less memory.
Lists are allocated in two blocks: 1. all the Python object information. 2.variable sized block for the data. that way list slower than tuple.	Tuple is stored in a single block. It doesn't require extra space to store new objects. that way tuple faster than list.
Lst = ['a','b',10,20,5]	Tpl = ('a','b',10,20,5)

34. What is range and Xrange() ?

range()	Xrange()
Return a list of integers. python 2 & 3	Return a genrator object. in python 2.0
Execution speed is slower.	Execution speed is faster.
Takes more memory.	Takes less memory .
All arithmetic operations can be performed as it returns a list.	All arithmetic operations cannot be performed as genrator object.
r =range(1,6)	x =xrange(1,6)

35. What is the difference between set and frozen set?

SET	FROZENSET
Set is an unordered and unindexed collection of unique elements. Set is mutable.	Frozenset is an unordered and unindexed collection of unique elements. Frozenset is immutable.
Set is represented by curly braces like this {} or you can use set() **Ex:** S = {10,20,30,40} S.add (50) Print(S) # {40,10,50,20,30}	**Frozensets are represented by the built-in function which is frozenset().** **Ex:** s = {10,20,30,40} fs = frozenset(s) fs.add (50) Print(fs) # Attribute Error: 'frozen set' Object has no attribute 'add'
We can add new elements to the set as it ismutable	As Frozen set is immutable, add, remove such type of terminology is not applicable

36. explain function list, tuple and dict ?

Function of list :	Function of tuple :	Function of Dict :	Function of set :
sort() : sort list into accending order.	**len () :** total length of tuple.	**clear () :** Removes all element from dict.	**clear():** Removes all the elements from the set
append () : Add single element in end of current list.	**sum () :** Return sum of all items in the tuple .	**copy () :** Return copy dict.	**copy():** Returns a copy of the set

extend () : add multiple element in list	**min () :** It returns an item from the tuple with minimum value	**get () :** Return value of specified key .	**discard():** Remove the specified item
index () : Return 1st Appearance of specifc values.	**index () :** Return 1st Appearance of specifc values.	**pop() :** Removes elment with specified key .	**remove():** Removes the specified element
max() : It returns an item from the list with max value.	**max () :** It returns an item from the tuple with max value.	**update () :** update dict with specified key-value pair.	**update():** Update the set with another set, or any other iterable
len(list): It gives the total length of the list		**items() :** Return key value pair.	**union():** Return a set containing the union of sets
list(): It returns covert tuple into list .	**tuple () :** Convert list into tuple.	**values() :** Return values in dict	**pop():** Removes an element from the set

37 .Explain some methods of the list.

- append( ) : adds the element to the end of the list.
  ex: a.append(3)
- pop( ) : remove last element from the list. show remove element.
  ex: a.pop() or usning index a.pop(1)
- sort ( ) : method used to sort the list in ascending or descending order.
  a.sort() --> acending and a.sort(reverse=True) --> descending.
- reverse( ) : the method is used to reverse the list elements.
  ex: a.reverse()

38. extend () ?

- extend() method adds specified list elements (or any iterable) to the end of the current list.
- iterable like (**list, set, tuple, etc**.)
- Syntax : **list.extend(iterable)**

39. What is the difference between append and extend?

- append adds a value to a list. >>> a.append(6)
- extend adds values in another list to a list. >>> b.extend([4,5])

40. Explain some methods of string?

1. split : It returns the list. ex: a.split(',')
2. join : used to combine the list of string objects.

41. What's the negative indexing in list?

- Normal indexing of the list starts from 0.
- But negative indexing access index from the end of the list.
- start of the negative indexing is -1,-2,-3,-4 etc.
- ex: a[-1]

42. What is slicing in Python?

- Slicing accepts three arguments. start index, end index,increment step separated by a colon.
- slicing Support list, tuple, and str data types.
- syntax : variable[start : end : step]

43. What are docstrings in Python?

Ans:

- A documentation string or docstring is a multiline string used to document a specific code segment.
- Docstrings are not actually comments, but, they are documentation strings.
- These docstrings are within triple quotes.
- They are not assigned to any variable and therefore, at times, serve the purpose of comments as well.
- **ex:**

```
"""
Using docstring as a comment.
This code divides 2 numbers
"""
x=8
y=4
z=x/y
print(z)
```

44. how slicing work string manupulation ?

syntax : variable[start : end : step]

```
s = "HelloWorld "
print(s[:])   output : HelloWorld
print(s[ : :-1]) output : dlroWolleH
```

45. how to get list of all the key in dict ?

dct = { 'A' : 1 , 'B' : 2 , 'C' : 3 }
case 1 : by using list
all_keys = list(dct)
print(all_keys) >>>>>> o/p : ['A','B','C']

case 2 : by using iterable unpacking operator :
all_keys = [*dct]
print(all_keys) >>>>>> o/p : ['A','B','C']

case 3 : using keys() function
all_keys = dict.keys()
print(all_keys) >>>>>> o/p : ['A','B','C']

46. How can you sort a dictionary by key, alphabetically?

- You can't "sort" a dictionary because dictionaries don't have order but you can return a sorted list of tuples which has the keys and values that are in the dictionary.
- ex:

```
d = {'c':3, 'd':4, 'b':2, 'a':1}
sorted(d.items())
```

- o/p : [('a', 1), ('b', 2), ('c', 3), ('d', 4)]

47. In Dict duplicate key is not allowed. But values can be duplicated. If we trying to add new entry(Key-value pair) with duplicate key what will be happened?

In Dict duplicate keys are not allowed. But values can be duplicated. If we trying to add new entry(Key-value pair) with duplicate key then old value will be replaced with new value.

Example:

```
D = {}
```

48. Can you cancatenate Two tuple ?

yes ,

```
t1 = (1,2,3)
t2 =(4,5,6)
t1 = t1 + t2
print(t1) >>>>> o/p (1,2,3,4,5,6)
```

49. How is memory managed in Python?

- Memory management in python is managed by Python private heap space
- contaning all Python objects and data structures are located in a private heap.
- The programmer does not have access to this private heap.
- The python interpreter takes care of this option.
- The allocation of heap space for Python objects is done by Python's memory manager.
- The core API gives access to some tools for the programmer to code.
- Python also has an inbuilt garbage collector, which recycles all the unused memory.
- so it can be made available to the heap space.
- The gc module defines functions to enable /disable garbage collector:
- **gc.enable()** -Enables automatic garbage collection.
- **gc.disable()** - Disables automatic garbage collection.

50. What is Garbage Collection ?

- When a value in memory is no longer referenced by a vaiable, intepreter automatically removes it from memory.
- This process is known as garbage collection.
- **gc.enable()** -Enables automatic garbage collection.
- **gc.disable()** - Disables automatic garbage collection.

51. Memory allocation in Python ?

- There are two parts of memory: Stack memory and Heap memory
- The function calls and the references are stored in the stack memory, whereas all the value objects are stored in the heap memory.

1. Work of Stack Memory :

- At the time of allocation, the memory size is known to the compiler and when there is a function call that happens, then its variable gets memory allocated on the stack.
- It is the memory that is needed inside a particular function.
- When the function is called, it will be added to the program call's stack.
- Variable initialization inside the function is temporarily stored in the function call stack, where it will be deleted after the completion of the function.
- This memory allocation onto a contiguous memory is handled by the compiler, developers do not have to worry about it.
- Example: def func():

```
a = 10
b = ""
c = {}
```

2. Work of Heap Memory :

- Heap memory allocation is done when memory is allocated at the time of execution of a program written by the programmer.
- As there is a pile of memory space available in the process of allocation and de – allocation,
- that's why it is known as heap memory.
- The variable which is required globally in the program is stored in heap memory.
- ex : This memory for 5 integers is allocated in the heap. arr = [0] * 5

52. What is heap and stack memory in Python?

- function call and refernces are stored in the stack memory where asa all the value and object are stored in heap memory.

53. What is heap space and stack space in Python

- Heap space : in Python primarily maintains all objects and data structures.
- stack space : contains all the references to the objects in heap space.

54. What is a function in Python?

- A function is a block of code that is executed only when it is called.
- To define a Python function, the def keyword is used.
- If the function returning something, they need a return keyword.
-
```
def Newfunc():
        print("Hi, Welcome to Edureka")
Newfunc() #calling the function
```

- o/p : Hi, Welcome to Edureka

55. What is a recursive function?

- The function calling itself is called a recursive function.
- It is always made up of 2 portions, the **base case** and the **recursive case**.
- The base case is the condition to stop the recursion.
- The recursive case is the part where the function calls on itself.
- There are three types of recursion : Head Recursion, Tail Recursion, Body Recursion
- ex:

```
def factorial(x):
```

```
    """This is a recursive function
    to find the factorial of an integer"""

    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))
num = 3
print("The factorial of", num, "is", factorial(num))   >>>> 1*2*3 = >>> 6
```

56. Explain Break, Continue and Pass Statement?

Break	Continue	Pass
Allow loop termination when some condition is met and the control is transferred to next statement.	Allow to skipping some part of loop when some specific condition is met and control is transferred to beginning of loop.	this basically null operation. nothing happen when this is executed.

for i in range(1,10):
if i == 5:
break
print(i) o/p :
1,2,3,4 >>> (break stop)
	for i in range(1,10):
	if i == 5:
	continue
	print(i) o/p :
	1,2,3,4,6,7,8,9 >>> (continue 5 skip)
		for i in range(1,10):
		if i == 5:
		pass
		print(i)
		o/p : 1,2,3,4,5,6,7,8,9 >> nothing happen when is executed.

• break : is used to terminate the running loop.	• continue : used to skip the execution of the remaining code.	**pass keyword** is used to mention an empty block in code.

57. What is the lambda function?

- lambda function is a small anonymous function.
- It is single-line function.
- lambda function take number of arguments but have only one expression.
- Good for performing short data manipulations.
- It is used with filter() , List Comprehension,map(), etc
- Syntax : lambda arguments : expression

- Filter out all odd numbers using filter() and lambda function

58. What's the difference normal function and lambda function?

Both returns same output value. Only object returned are different.

Normal function	Lambda function
Execution time is relatively slower than lambda function.	**Execution time is relatively Faster than Normal function.**
Defined using the keyword def	**Defined using the keyword lambda**
It contain any statements like break,continue,pass ,etc in program.	lambda function take number of arguments but have only one expression
Defining a function: def add(a,b): return a+b print(add(4,5))	Defining a lambda add = lambda x, y : x + y print(add(4,5))
Function definition does have include a "return" statement.	Lambda definition does not include a "return" statement.

59. What is Generator explain with example ?

- A Python generator is a function that produces a sequence of items.
- generator are iterators which can execute only once.
- generator used yield keyword.
- it is mostly used in loop to generate an iterators an iterator by returning all the values in loop without affecting the iteration of the loop.
- The code of the generator only executes when the next function is applied to the generator object.
- **Advantages of using Generators :**
- Without Generators in Python, producing iterables is extremely difficult and lengthy.
- Generators easy to implement as they automatically implement __iter__(), __next__(),
- Memory is saved as the items are produced when required.

ex: square of number	o/p:

def my_gen(n):
for i in range(1, n + 1):
yield i * i
a = my_gen(4)
	print(next(a)) >>> 1
	print(next(a)) >>> 4
	print(next(a)) >>> 9
	print(next(a)) >>> 16

60. What are iterators?

- iterator is an object which contain a countable number of values.
- it is used to iterate over object like list , tuple, sets , dict .
- It uses the next() method for iteration.
- every iterator is not a generator.
- __iter__(): The iter() method is called for the initialization of an iterator. This returns an iterator object.
- __next__(): The next method returns the next value for the iterable.

#ex:
string = "GFG"
a = iter(string)
	print(next(a)) >>> G
	print(next(a)) >>> F
	print(next(a)) >>> G

61. Difrance between Generator and Iterator

Iterator	**Generator**
Class is used to implement an iterator used in iter an dnext keyword.	Function is used to implement a generator. yeild keyword used.
Local Variables are not used here.	All the local variables before the yield function are stored.
Iterators are used mostly to iterate or convert other object to an iterator using other function.	Generators are mostly used in loop to generate an iterator by returning all values in in loop without affecting iteration of the loop.
Iterators uses iter() and next function.	Generators uses yield keywords.
Every Iterators not a generator. ex : iter_list = iter(['A','B','C']) print(next(iter_list)) >> A print(next(iter_list)) >> B print(next(iter_list)) >> C	Every generator not an Iterator. ex: def sqr(n): for i in range(1,n+1): yeild i * i a = sqr(3) print(next(a)) >> 1 print(next(a)) >> 4 print(next(a)) >> 9

62. Normal Functions vs Generator Functions?

- Generators in Python are created just like how you create normal functions using

the 'def' keyword. But, Generator functions make use of the yield keyword return.

- Generator functions are run when the next() function is called .
- **Consider the following example to understand it better:**

Yield { Generator}	**Return {** Normal function }
Return A Generator object.	Return result of caller Statement.
A Generator function can have only one yield Statement.	Normal function can have multiple return Statement.
Return an intermediate result to the caller by g Generator.	Used when function return final value to its caller.
Yield statement can run multiple times. However, the Generator can be used only once to yield values.	Return statement can be executed only once. However, the normal function can be reused any number of time.
Code after yield statement are executed in sub- sequent	Code after return statement cannot be Executed.

63. What is Decorators and give example ?

- ***Decorators is a function that takes another function as its argument,new functionality to an existing object without modifying its structure then returns another function.***
- ***In Decorators,functions are taken as the argument into another function and then called inside the wrapper function.***
- You'll use a decorator when you need to change the behavior of a function without modifying the function itself.
- Decorators are used to modify the behaviour of function or class.

ex:

```
def decorator_func(func):
        def wrapper_func():
                print("wrapper_func Worked")
                return func()
        print("decorator_func worked")
        return wrapper_func

@decorator_func
def display():
    print('displayworked')
display()
```

output :
decorator_fun worked
wrapper_func Worked
Show worked

64. How is exception handling performed in Python?

- Python provides 3 words to handle exceptions, **try**, **except** and **finally**.

1. Try : this block will handle exceptional error to occure.
2. Except : Here you can handle the error.
3. Else : if there is no exception then this block will be execited.
4. finally : finally block always gets executed either exception is genrated or not.

- The syntax is:

```
try:
    # Some Code !
except:
    # optinal block
    # handling of exception (if required)
else :
    # some code ...
    #execute if no exception
finally:
    # some code ... (alway executed)
```

65.what are diffrent way to read and wirte python file ?

- syntax of open file function :

file_object = open("filename","mode")

- **type of mode :**

1. Read only ('r') : open txt file for reading .
2. Read and write ("r+') : open txt file for reading and writting.
3. write only ('w') : open file for writing only.
4. write and read ('w+') : open the file reading and writing.
5. append only ('a') : open file for writing only.
6. append and read ('a+') : open file for reading and writing.

66. What is Open & Open Statment ?

both are used in file handling.
with : when you used it then automatically closed file .

With the "With" statement , you get better syntax and exception handling.

'open' Statment	'with ' Statment
f = open("rupesh.txt") content = f.read() print(content) f.close()	with open ("rupesh.txt") content = f.read() as f : print(content)

67. Abstraction in python?

- abstraction that works in the object-oriented concept.
- Abstraction is used to hide the internal functionality of the function from the users.
- defined as a process of handling complexity by hiding unnecessary information from the user.
- ex:
  When we use the TV remote to increase the volume.
  We don't know how pressing a key increases the volume of the TV.
  We only know to press the "+" button to increase the volume.
- Syntax: abc module to create the abstract base class.
- from abc import ABC
- class ClassName(ABC):
-

68. Abstraction vs Encapsulation

Abstraction	Encapsulation
it is work on design level.	it is work on application level.
it is implemented to hide unnesarry data and withdrowing relevent data.	it is mechanism of hiding code and data together from outside world .
it focuses on outside vieving ex: shifting the car .	it is focus on internal working ex : production of car
it is supported in java witj interface and the abstract class	it is supported using eg. private and secure access modification systems.

69.What is Inheritance In python ?

In Inheritance, the child class acquires the properties and can access all the data members and functions defined in the prent class. A chlid can also provide its specific implementation to the function of the parent class.

In python, a drived class can inherit base by just mentioning th base in the bracket after the drived class name.

Class A(B):

ex:

```
class A :
   def display(self):
      print("A Display")

class B(A):
   def show(self):
      print("B Show")

d = B()
d.show()
d.dispaly()
```

Output :

B Show

A Display

70. What is multithreading and multiprocessing in python?

Multithreading :

- it is just multiprocessing. it way of achieving multitasking.
- defined as the ability of a processor to execute multiple threads concurrently.
- In which Multiple threads can exist within one process where:

1. Each thread contains its own register set and local variables (stored in stack).
2. All threads of a process share global variables (stored in heap) and the program code.

Multiprocessing

- Multiprocessing in Python is a built-in package that allows the system to run multiple processes simultaneously. It will enable the breaking of applications into smaller threads that can run independently.
- In multiprocessing:
  multiple threads at the same time run across multiple cores.
  Multiprocessing can be classified such as symmetric or asymmetric.

71. Thread ?

A thread is an entity within a process that can be scheduled for execution.
it is the smallest unit of processing that can be performed in an OS.

72. What is GIL. explain ?

- The Python Global Interpreter Lock or GIL, in simple words, is a mutex (or a lock) that allows only one thread to hold the control of the Python interpreter.
- This means that only one thread can be in a state of execution at any point in time.

73. What Is Yield In Python?

similar to a return statement used for returning values or objects in Python.
yield is a keyword that is used like return, except the function will return a generator.

74. How do delete a given file in Python?

```
import os
file = r'./test.txt'\
if os.path.exists(file):\
   os.remove(file)\
   print('delete success' )\
else :\
   print(f'no such Error! Hyperlink reference not valid')
```

75. What is shallow and deep copy?

Ans:

- *Shallow Copy:* it creates the exact copy as the original without changing references of the objects. changing one object will affect the other.
- *Deep Copy:* it copies the values of the original object recursively into the new object.

76. Explain Pandas and Numpy?

NumPy is a library for Python that adds support for large, multi-dimensional arrays and matrices,along with a large collection of high-level mathematical functions to operate on these arrays.

Pandas is a high-level data manipulation tool that is built on the NumPy package. The key data structure in Pandas is called the DataFrame. DataFrames are incredibly powerful as they allow you to store and manipulate tabular data in rows of observations and columns.

77. diffrance between seriese and vactors and list python ?

seriese	vactors	list
Series is a one-dimensional labeled array capable of holding data of any type. (integer, string, float, python objects, etc.).	vector is **a one-dimensional array of lists**. which has the ability to resize automatically after insertion or deletion of elements.	***List*** is a collection which is ordered and **Mutable**. list used to store the collection of objects.
The axis labels are collectively called index. Series will always contain data of the same type	Vector is thread safe. Random access of elements is possible	List is not thread safe. Random access of elements is not possible.
Series is not a list internally but a NumPy array.	It has contiguous memory. Vector may have a default size.	While it has non-contiguous memory. List does not have default size.

which is both faster and smaller (memory wise) than a python list.	ex:
	vector v;
	v.insert(5);
	v.delete();
		Lst = ['a','b',10,20,5]

78. Explain about map () , reduce() ,filter() function ?

map ( )	reduce ( )	filter ( )
function returns a map object. of the results after applying the given function to each item of a given iterable (list, tuple etc.)	it is used to **apply a particular function passed in its argument to all of the list elements** . This function is defined in "functools" module.	filter() function is used to get filtered elements.

Syntax : map(fun, iter) Parameters : fun : It is a function to which map passes each element of given iterable. iter : It is a iterable which is to be mapped	it stores the intermediate result and only returns the final summation value. reduce(fun, seq) takes function as 1st and sequence as 2nd argument. **syntax:** **functools.reduce(function, iterable)**	This function takes two arguments, first is a function and the second is iterable. **syntax:** **filter (function, iterable)**

num = (1, 2, 3, 4)
result = map(addition, num)
print(list(result))
result = map(lambda x: x + x, num)
print(list(result))
Output : [2, 4, 6, 8]
	from functools import reduce
	
	num = [1, 2, 3, 4]
	a = reduce(lambda x, y: x + y, nums)
	print(a)
	Output: 10 [6]
		def filterdata(x):
		if x>5:
		return x
		r = filter(filterdata,(1,2,6))
		print(list(r))
		Output: [6]