1. **What is decorator? Types of decorator?**

A A tool that allows programmers to modify the behavior of function or class. Decorators allow us to wrap another function in order to extend the behavior of the wrapped function, without permanently modifying it.

a) Function decorator
b) Class decorator

2. **What is monkey patching?**

A The term monkey patch refers to dynamic (or run-time) modifications of a class or module.
e.g

```
 class A:
     def monk1(self):
         print('In monk function 1')
 def monk2(self):
     print('In monk function 2')
 A.monk1 = monk2                     #patching

 obj = A()
 obj.monk1()

O/P: In monk function 1
```

3 **What is pip?**

A pip(preferred installer program) is the standard package manager for Python. It allows you to install and manage additional packages that are not part of the Python standard library.

4 **From where PIP installs the library?**

A Pip installs libraries and packages from Python Package Index (PyPi) using a requirement specifier. Requirement specifier contains project name and may contain version specifier as well

5. **What is meant by PyPi?**

A Python Package Index (PyPi) helps you find and install software developed and shared by the Python community. PyPI hosts an extensive collection of packages that include development frameworks, tools, and libraries.

6 **What is lambda function in python?**

A In Python, a lambda function is a single-line function declared with no name, which can have any number of arguments, but it can only have one expression. Such a function is capable of behaving similarly to a regular function declared using the Python's def keyword.
For e.g

```

remainder = lambda num: num % 2

print(remainder(5))

O/P: 1
```

7 **Characteristics of lambda function**

A

- A lambda function can take any number of arguments, but they contain only a single expression. An expression is a piece of code executed by the lambda function, which may or may not return any value.
- Lambda functions can be used to return function objects.
- Syntactically, lambda functions are restricted to only a single expression

8 **Why do we use Lambda function?**

A Lambda functions are used when you need a function for a short period of time. This is commonly used when you want to pass a function as an argument to higher-order functions, that is, functions that take other functions as their arguments.

```
def testfunc(num):
    return lambda x : x * num

result1 = testfunc(10)

print(result1(9))

O/P: 90
```

9 **What is namespacing?**

A A namespace is a collection of currently defined symbolic names along with information about the object that each name references. You can think of a namespace as a dictionary in which the keys are the object names and the values are the objects themselves.

10 **Types of namespace?**

A **1. Built-in** - contains the names of all of Python's built-in objects. These are available at all times when Python is running.
2. Global - contains any names defined at the level of the main program. Python creates the global namespace when the main program body starts, and it remains in existence until the interpreter terminates.
3. Local - the interpreter creates a new namespace whenever a function executes. That namespace is local to the function and remains in existence until the function terminates.

11 **Scope in python?**

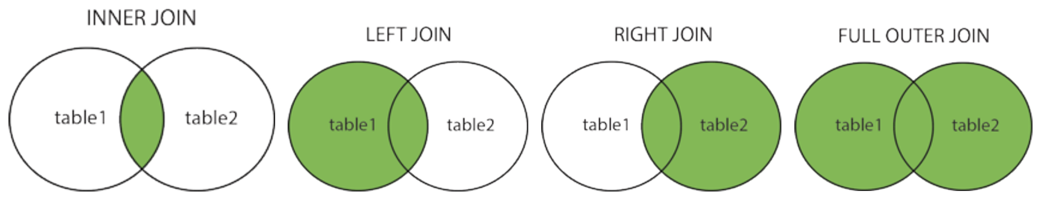
A The scope of a name is the region of a program in which that name has meaning. The interpreter determines this at runtime based on where the name definition occurs and where in the code the name is referenced.

12 **What are the types of scope in python?**

A

1. **Local: -** The Variables which are defined in the function are a local scope of the variable.
2. **Enclosing or non- local: -** Nonlocal Variable is the variable that is defined in the nested function. It means the variable can be neither in the local scope nor in the global scope.
3. **Global: -** The Variable which can be read from anywhere in the program is known as a global scope. These variables can be accessed inside and outside the function. When we want to use the same variable in the rest of the program, we declare it as global.
4. **Built-in: -** The Variable which can be read from anywhere in the program is known as a global scope. These variables can be accessed inside and outside the function. When we want to use the same variable in the rest of the program, we declare it as global.

13. **What is LEGB rule?**

A	Whenever a code refers to a name defined multiple times, it searches for that variable name by **LEGB** (Local-Enclosing-Global-Builtin) rule. The interpreter searches for a name from the inside out, looking in the **l**ocal, **e**nclosing, **g**lobal, and finally the **b**uilt-in scope
14	**What is the difference between module, package and library in python?**
A	**Module:** The **module** is a simple Python file that contains collections of functions and global variables and having a .py extension file. It is an executable file and to organize all the modules we have the concept called Package in Python. **Package:** The **package** is a simple directory having collections of modules. This directory contains Python modules and also having __init__.py file by which the interpreter interprets it as a Package. The package is simply a namespace. The package also contains sub-packages inside it. **Library:** The **library** is having a collection of related functionality of codes that allows you to perform many tasks without writing your code. It is a reusable chunk of code that we can use by importing it in our program; we can just use it by importing that library and calling the method of that library with period (.).
15	**OOP's questions**
A	Refer following link: https://www.edureka.co/blog/interview-questions/oops-interview-questions/
16	**What is join in database? Types of join?**
A	A JOIN clause is used to combine rows from two or more tables, based on a related column between them. Types of Join: • **(INNER) JOIN**: Returns records that have matching values in both tables • **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table • **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table • **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table  INNER JOIN table1 table2 LEFT JOIN table1 table2 RIGHT JOIN table1 table2 FULL OUTER JOIN table1 table2
17	**What is pass by value and pass by reference in python?**
A	In python arguments are always passed by object reference. If immutable arguments are passed then the passing acts like call by value.
18	**What Is meant by property in python?**
A	Python programming provides us with a built-in @property decorator which makes usage of getter and setters much easier in OOP. Syntax: `property(fget=None, fset=None, fdel=None, doc=None)` where,

	fget is function to get value of the attribute fset is function to set value of the attribute fdel is function to delete the attribute doc is a string(like a comment)
19	**What are the different data structures in python?**
A	Numeric: - int, float, complex, byte, bytearray None Sequence: - range, list, tuple, str Mappings:- set, dictionary, frozenset Boolean
20	**Explain each datatype with functionality**
A	https://www.programiz.com/python-programming/variables-datatypes#:~:text=They%20are%20defined%20as%20int,belongs%20to%20a%20particular%20class.&text=Integers%20can%20be%20of%20any,limited%20by%20the%20memory%20available.
21	**Why list is mutable?**
A	In a list, elements can be modified, individual values can be replaced and the order of elements can be changed. That means list are dynamic and thus it allow adding, replacing and deleting its elements.
22	**What is the difference between mutability and immutability?**
A	

Mutable	Immutable
mutable objects can change their state or contents	immutable objects can't change their state or content.
type list**,** dict**,** set	**int, float, bool, string, unicode, tuple**
Mutable objects are great to use when you need to change the size of the object	Immutable objects are quicker to access and are expensive to **change** because it involves the creation of a copy.

23	**What is hashing mechanism?**
A	Hash method in Python is a module that is used to return the hash value of an object. In programming, the hash method is used to return integer values that are used to compare dictionary keys using a dictionary look up feature. When used, it calls for the __hash__ () of an object which is set by default during the creation of the object by the user. Syntax `hash(object)`
24	**What is meant by dictionary lookup?**
A	Fetching values based on keys from a dictionary, is known as key dictionary look up
25	**What is the difference between == and is operator related to function?**
A	

Identity operator (is)	Equality operator(==)
Python is operator checks whether two variables point to the same object in memory	compares the value or **equality** of two objects

```
list1 = []
list2 = []
```

```
list3=list1

if (list1 == list2):
        print("True")
else:
        print("False")
if (list1 is list2):
        print("True")
else:
        print("False")
if (list1 is list3):
        print("True")
else:
        print("False")
list3 = list3 + list2
if (list1 is list3):
        print("True")
else:
        print("False")
```

Output:-

```
True
False
True
False
```

Here since id of list and list 2 are different the is operator points to memory and shows false whereas == shows value equality and hence shows True

26 **How iterator can be used to generate the generator?**

A We can create a generator by using the keyword **yield** statement. Python generators are an easy and simple way of creating iterators and are mainly used to declare a function that behaves like an iterator. The generator is a function which we can iterate over one value at a time most probably in a day to day life, every programmer will use iterable objects like lists, strings, and Dict, etc.

```
def generator():

   print("program working sucessfully")

   yield 'x'

   yield 'y'

   yield 'z'

generator()

OutPut:

<generator object generator at 0x000000CF81D07390>
```

27 **Difference between xrange and range?**

A

range	xrange
range() returns the list	xrange() returns the xrange object
range() is faster if iterating over the same sequence multiple times.	Faster in implementation than range as it evaluates only generators
can be used in python3 and python2	Used only in python2

28 **Difference between (try/except/finally) and (try/except/else)?**

A

- Try: This block will test the excepted error to occur
- Except: Here you can handle the error
- Else: If there is no exception then this block will be executed
- Finally: Finally block always gets executed either exception is generated or not

Try and except:-

First try clause is executed i.e. the code between try and except clause.

- If there is no exception, then only try clause will run, except clause will not get executed.
- If any exception occurs, the try clause will be skipped and except clause will run.
- If any exception occurs, but the except clause within the code doesn't handle it, it is passed on to the outer try statements. If the exception is left unhandled, then the execution stops.
- A try statement can have more than one except clause.

Else:-

- The code enters the else block only if the try clause does not raise an exception. Else block will execute only when no exception occur.

Finally:-

Python provides a keyword finally, which is **always executed** after try and except blocks. The finally block always executes after normal termination of try block or after try block terminates due to some exception.

29 **How to write our own exception in python?**

A

Users can define custom exceptions by creating a new class. This exception class has to be derived, either directly or indirectly, from the built-in `Exception` class. Most of the built-in exceptions are also derived from this class.

E.g:

```
class Error(Exception):
    """Base class for other exceptions"""
    pass

class ValueTooSmallError(Error):
    """Raised when the input value is too small"""
    pass

class ValueTooLargeError(Error):
    """Raised when the input value is too large"""
    pass
```

```python
# you need to guess this number
number = 10

# user guesses a number until he/she gets it right
while True:
    try:
        i_num = int(input("Enter a number: "))
        if i_num < number:
            raise ValueTooSmallError
        elif i_num > number:
            raise ValueTooLargeError
        break
    except ValueTooSmallError:
        print("This value is too small, try again!")
        print()
    except ValueTooLargeError:
        print("This value is too large, try again!")
        print()

print("Congratulations! You guessed it correctly.")
```

Here is a sample run of this program.

```
Enter a number: 12
This value is too large, try again!

Enter a number: 0
This value is too small, try again!

Enter a number: 8
This value is too small, try again!

Enter a number: 10
Congratulations! You guessed it correctly.
```

30	**difference between .pyc and .py**
A	• .py files contain the source code of a program. Whereas, .pyc file contains the bytecode of your program. We get bytecode after compilation of .py file (source code). .pyc files are not created for all the files that you run. It is only created for the files that you import.

	• Before executing a python program python interpreter checks for the compiled files. If the file is present, the virtual machine executes it. If not found, it checks for .py file. If found, compiles it to .pyc file and then python virtual machine executes it. • Having .pyc file saves you the compilation time.
31	**How memory is managed in python?**
A	Memory management in Python involves a private heap containing all Python objects and data structures. Interpreter takes care of Python heap and that the programmer has no access to it. - The allocation of heap space for Python objects is done by Python memory manager. The core API of Python provides some tools for the programmer to code reliable and more robust program. - Python also has a build-in garbage collector which recycles all the unused memory. When an object is no longer referenced by the program, the heap space it occupies can be freed. The garbage collector determines objects which are no longer referenced by the program frees the occupied memory and make it available to the heap space. - The gc module defines functions to enable /disable garbage collector.
32	**Difference between append and extend?**
A	

Append()	Extend()
adds an element to a list	concatenates the first list with another list
adds its argument as a single element to the end of a list, the length of the list itself will increase by one	iterates over its argument adding each element to the list, extending the list
e.g my_list = [‘Hello', 'Everyone'] my_list.append('How are you') print my_list O/P: [' Hello ', ' Everyone ', ' How are you']	e.g my_list = ['List', 'for'] another_list = [6, 0, 4, 1] my_list.extend(another_list) print my_list O/P: ['List', 'for', 6, 0, 4, 1]

33	
A	
34	
A	
35	
A	
36	
A	
37	
A	
38	
A	
