1. What Is Object-Oriented Programming?

```
object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming.
It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.
The main concept of OOPs is to bind the data and the functions that work on that together
as a single unit so that no other part of the code can access this data.
Main Concepts of Object-Oriented Programming (OOPs)
Class
Objects
Polymorphism
Encapsulation
Inheritance
Abstraction
```

2. Difference between Procedural programming and OOPs?

```
Procedural programming:
1) Programme is divided into small parts called function.
2) It follows top down approach.
3) There is no access specifier.
4) Adding new data and function is not easy.
5) it is not having the provision to hide data so it is less secure.
6) Overloading is not possible.
7) function is more important than data.
8) based on unreal world
9) e.g. C, Fortran, Pascal, Basic etc


OOPs:
1) Programme is divided into small parts called object
2) It follows bottom up approach.
3) We can specify the access local, private, protected etc.
4) Adding new data and fuction is easy.
5) it is having provision to hide the data, so more secure.
6) overloading is possible.
7) data is more important than function.
8) based on real world.
9) e.g. Java, C++, Java, Python etc.

```

3. What are the fundamental principles/features of Object-Oriented Programming?

```
below are the fundamental and features of OOPs:

1) Class:
- it is a blue print of objects.
- it consist the attributes and methods.
- it follows ClassName convection

2) Objects:
- object is an entity that has a state and behavior associated with it.
- An object consist of:
    state: it is represented by the attributes of an object. it is also reflects the properties of object.
    Behavior: it is represented by methods of an object.
    Identity: it gives a unique to an object and enables one object interact with other objects.

3) Inheritance:
- it is capability of class to inherit the other class properties in it as a child class.
types of inheritance:
i) single/simple inheritance.
ii) multilevel inheritance.
iii) multiple inheritance.

4) Polymorphism:
- it is simple meaning is many forms
- we can use the same function/method name in many class.

5) Encapsultion:
- it is to restrict the attribute/methods to modify at outside the class.

6) Abstraction:
- it is blueprint of project.
- this class is not use for implementaion.
- it is only for declaration


```

4. What is an object?

```
Objects:
- object is an entity that has a state and behavior associated with it.
- An object consist of:
    state: it is represented by the attributes of an object. it is also reflects the properties of object.
    Behavior: it is represented by methods of an object.
    Identity: it gives a unique to an object and enables one object interact with other objects.
```

5. What is a class?

In [ ]:

```
1) Class:
- it is a blue print of objects.
- it consist the attributes and methods.
- it follows ClassName convection
```

6. What is the difference between a class and an object?

```
Class
1) Class is a template for declaring and creating the object.
2) when class is created no memory is allocated.
3) the class has to declared only once
4) A class can not manipulated as they are not available in memory
5) class is a logical entity.
6) Class does not contain any value.


Object:
1) An object is an instance of a class
2) Objects are memory allocated memory space.
3) it is created many times as per requirement.
4) can be manipulated.
5) a object is physical entity.
6) object is like a variable of class
```

7. Can you call the base class method without creating an instance?

```
Yes.
by using the ClassName.base_class_method()
```

8. What is inheritance?

```
Inheritance:
- it is capability of class to inherit the other class properties in it as a child class.

```

9. What are the different types of inheritance?

```
types of inheritance:
i) single/simple inheritance.:
    single base class is inherit to child class
ii) multilevel inheritance.:
    each class is inherit to another one in sequence. we can access all varibales/attributes from all class
iii) multiple inheritance.:
    can inherit many class at once.
```

10. What is the difference between multiple and multilevel inheritances?

```
Multiple inheritance:
```

```
1) Multiple Inheritance is an Inheritance type where a class inherits from more than one base class.
2) Multiple Inheritance is not widely used because it makes the system more complex.
3) Multiple Inheritance has two class levels namely, base class and derived class.

Muiltilevel Inheritance:
1) Multilevel Inheritance is an Inheritance type that inherits from a derived class,
making that derived class a base class for a new class.
2) Multilevel Inheritance is widely used.
3) Multilevel Inheritance has three class levels namely, base class, intermediate class and derived class.
```

11. What are the limitations of inheritance?

```
1) inherited functions work slower than normal function as there is indirection.
2) Improper use of inheritance may lead to wrong solutions.
3) Often, data members in the base class are left unused which may lead to memory wastage.
4) Inheritance increases the coupling between base class and derived class.
A change in base class will affect all the child classes.
```

12. What are the superclass and subclass?

In [ ]:

```

In OOP, inheritance is one of the principles where a class(derived class) can be derive
The derived class(child class or subclass) inherits all the properties and methods of t
(parent class or superclass).The subclass adds some attributes to superclass.
How to check if a class is subclass of another?
Python provides a function issubclass() that directly tells us if a class is subclass o
Following example shows the use of issubclass() function:
Syntax: issubclass(Derived,Base) will return True/False
```

13. What is the super keyword?

In [ ]:

```
he super() function is used to give access to methods and properties of a parent or sib
The super() function returns an object that represents the parent class.
Need not to use the self keyword if super keyword is used
In Python, super() has two major use cases:
Allows us to avoid using the base class name explicitly
Working with Multiple Inheritance
```

14. What is encapsulation?

```
It is Used to restrict the access of variables and methods
Protecting data from modification of private variable/methods
To prevent accidently modification of data from outside the class
```

```
Syntax >> __variablename
__methodename
Encapsulation is one of the fundamental concepts in object-oriented programming (OOP).
It describes the idea of wrapping data and the methods that work on data within one unit.
This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data.
To prevent accidental change, an object's variable can only be changed by an object's method.
Those types of variables are known as private variable.
A class is an example of encapsulation as it encapsulates all the data that is member functions, variables, etc.
```

15. What is the name mangling and how does it work?

```
Used to access private variables and methods from outside the class

    Object._ClassName__privatevariable
    Object._ClassName__privatemethod
```

16. What is the difference between public and private access modifiers?

```
Public access modifier:
1) this modifier is applicable to top-level classes and interfaces
2) can be access from the child class of same packages.
3) can be access from non child class of same packages.
4) can be accessed from child class of outside package.
5) can be accessed from non-child class of outside package.
6) modifier is the most accessible modifier.
7) it is the recommended modifier for method.


Private access modifier:
1) this modifier is not applicable to top level classes and interfaces.
2) can not be access from the child class of same packages
3) can not access from non child class of same packages
4) can not accessed from child class of outside package.
5) can not accessed from non-child class of outside package.
6) modifier is the most restricted modifier.
7) modifier is the recommended modifier for data members.
```

17. Is Python 100 percent object-oriented?

```
No, we cannot classify Python as strictly an object-oriented programming language.
1. Although borrowing most of the features from OOP, Python isn't a 100% OOP language since it does not allow strong
encapsulation.
2. This is because its creator Guido van Rossum aimed to keep things simple and that meant not hiding data in the strictest
```

```
sense of the term.
3. Instead of encapsulation, in Python, there's a convention for data hiding wherein you can prefix the data members with
two underscores '_'.
4. Apart from this, Python supports all the basic features of OOP language.
```

18. What is data abstraction?

In [ ]:

```
Consider as a Blueprint of Classes
We can not create object of abstract class
Abstract class is only used for declaration, does not for implemation
By default python does not support the abraction. So we need to import abc module
Abstraction is used to hide the internal functionality of the function from the users.
The users only interact with the basic implementation of the function, but inner workir
User is familiar with that "what function does" but they don't know "how it does."
In Python, an abstraction is used to hide the irrelevant data/class in order to reduce
It also enhances the application efficiency
```

19. How to achieve data abstraction?

In [ ]:

```
In Python, abstraction can be achieved by using abstract classes and interfaces.
A class that consists of one or more abstract method is called the abstract class.
Abstract Base Classes
We import the ABC class from the abc module. The ABC works by decorating methods of the
We use the @abstractmethod decorator to define an abstract method or if we don't provic
the method, it automatically becomes the abstract method
An abstract base class is the common application program of the interface for a set of
It can be used by the third-party, which will provide the implementations such as with
It is also beneficial when we work with the large code-base hard to remember all the cl
```

20. What is an abstract class?

In [ ]:

```
A class that consists of one or more abstract method is called the abstract class.
Abstract methods do not contain their implementation.
Abstract class can be inherited by the subclass and abstract method gets its definitior
Abstraction classes are meant to be the blueprint of the other class.
An abstract class can be useful when we are designing large functions.
An abstract class is also helpful to provide the standard interface for different imple
Python provides the abc module to use the abstraction in the Python program.
We can import the Abstract Base Class from the abc module
```

21. Can you create an object of an abstract class?

In [ ]:

```
No, we cannot create an object of an abstract class.
Abstract class is only used for declaration, does not used for implementation
Abstract classes are meant to be the blueprint of the other class.
It is only used to hide the internal functionality of the other class functions from th
An Abstract class can contain the both method normal and abstract method.
An Abstract cannot be instantiated; we cannot create objects for the abstract class.
```

22. Differentiate between data abstraction and encapsulation

In [ ]:

```
Abstraction vs Encapsulation
Data Abstraction can be described as the technique of hiding internal details of a prog
only.
Data Encapsulation can be described as the technique of binding up of data along with i
as a single unit.
Implementation hiding is done using this technique.
Information hiding is done using this technique.
Data abstraction can be performed using Interface and an abstract class.
Data encapsulation can be performed using the different access modifiers like protected
Abstraction lets somebody see the What part of program purpose.
Encapsulation conceals or covers the How part of the program's purpose.
Abstraction is the process or method of gaining the information.
While encapsulation is the process or method to contain the information.
In abstraction, problems are solved at the design or interface level.
While in encapsulation, problems are solved at the implementation level.
Abstraction is the method of hiding the unwanted information.
Whereas encapsulation is a method to hide the data in a single entity or unit along wit
information from outside.
We can implement abstraction using abstract class and interfaces.
Whereas encapsulation can be implemented using by access modifier i.e. private, protect
In abstraction, implementation complexities are hidden using abstract classes and inter
While in encapsulation, the data is hidden using methods of getters and setters.
The objects that help to perform abstraction are encapsulated.
Whereas the objects that result in encapsulation need not be abstracted .
```

23. What is polymorphism?

In [ ]:

```
The word polymorphism means having many forms.
In programming, polymorphism means same function name (but different signatures) being
Polymorphism with class methods:
Below code shows how python can use two different class types, in the same way.
we create a for loop that iterates through a tuple of objects.
Then call the methods without being concerned about which class type each object is.
We assume that these methods actually exist in each class
```

24. What is the overloading method?

In [ ]:

```
Method Overloading:
Two methods cannot have the same name in Python; hence method overloading is a feature
operator to have different meanings.
Method Overloading is an example of Compile time polymorphism.
In this, more than one method of the same class shares the same method name having dif
Method overloading is used to add more to the behavior of methods and there is no need
for method overloading.
Note: Python does not support method overloading. We may overload the methods but can
Method Overriding:
Method overriding is an example of run time polymorphism.
In this, the specific implementation of the method that is already provided by the pare
the child class.
It is used to change the behavior of existing methods and there is a need for at least
overriding.
In method overriding, inheritance always required as it is done between parent class(s
(child class) methods.
Overloading method vs Overriding method
In the method overloading, methods or functions must have the same name and different
Whereas in the method overriding, methods or functions must have the same name and same
Method overloading is a example of compile time polymorphism.
Whereas method overriding is a example of run time polymorphism.
In the method overloading, inheritance may or may not be required.
Whereas in method overriding, inheritance always required.
Method overloading is performed between methods within the class.
Whereas method overriding is done between parent class and child class methods.
It is used in order to add more to the behavior of methods.
Whereas it is used in order to change the behavior of exist methods.
In method overloading, there is no need of more than one class.
Whereas in method overriding, there is need of at least of two classes.
```

25. What are the limitations of OOPs?

In [ ]:

```
Usually not suitable for small problems
Requires intensive testing
Takes more time to solve the problem
Requires proper planning
The programmer should think of solving a problem in terms of objects
The size of the programs created using this approach may become larger than the progra
procedure-oriented programming approach.
Software developed using this approach requires a substantial amount of pre-work and p
OOP code is difficult to understand if you do not have the corresponding class documen
In certain scenarios, these programs can consume a large amount of memory.
```