OOP

Object Oriented Programming

Programming paradigm to design and develop programs based on Class/Object - as fundamental building block of your program

* Advantages:

Easy to understand
Clear modular structure of programs
Data hiding and protection
Extensible and maintainable program structure
Higher level reusability of Data and functions
Enables to develop GUI based and large scale software systems

* Basic OOP concept:

Encapsulation - wrapping or packing of data and functions together into a single unit (capsule / container)

Class - Container for logical related data and methods
Basic building block of OOP
Template for creating object
Provides encapsulation and data hiding

Data hiding - Protection of data
- Private data in class - not accessible to outside code and functions

- Constructor - special method used to initialize Object (when created)
  - have special name in Python
  - Allocates size of object

Default Constructor

- If you don't define constructor for a class - default (parameterless) constructor is automatically created by python runtime environment
- Initializes all instance variables to default values

0 - for Numeric
Null - for Object
False - for Booleans

- Garbage Collection
- PVM provides automated memory management
- Programmer does not have to delete unused objects
- Garbage Collector (component of PVM) deletes objects - when it determines that they are no longer being used by object variables
- This process is Garbage Collection
- After Garbage Collection, objects without any reference are removed from heap memory
- Garbage collection performs cleaning activity only on heap memory and it tries to keep as much as possible free space in heap memory

Page No.
Date :

* 2 types of Variable in Class

1.) Normal / Instance Variable
- Seperate values or memory allocation for every object or instance
- Normal data members and methods defined in class

2.) Class / Static Variable
- Common value shared by all objects of same class
- Only one memory allocation shared by all objects
- are called Class-level data members
- Makes your program memory efficient (saves memory)

* 3 types of Methods

1.) Instance Method
- Purpose is to set or get details about instances (objects)
- Most common type of method used in a Python class
- One default parameter: self, which points to an instance of class
- Any method you create inside a class is an Instance method unless you specially specify Python otherwise
- An instance method knows it's instance

2) Class Method

- Purpose is to set or get details of class
- Can't access or modify specific instance data
- Bound to class instead of their objects
- In order to define a class method, you have to specify that it is a class method with the help of @classmethod decorator
- One default parameter: cls, which points to the class
- Without creating an instance of class, you can call the class method with - Class_name.Method_name()
- A class method knows it's class
- Most common use is for creating Factory methods (methods that return a class object for different use cases)

3) Static Method

- Cannot and do not need to access the class data
- Are self sufficient and can work on their own
- Since they are not attached to any class attribute, they cannot set or get the instance or class state
- In order to define a static method, you can use @staticmethod decorator
- We do not need to pass any special or default parameters
- Static method doesn't know it's class or instance
- Used for creating Utility functions (to accomplish routine programming tasks)

Date :

. Inner Class
You can create object of inner class inside the outer class
You can create object of inner class outside the outer class provided you use outer class name to call it

* Inheritance
- An important aspect of OOP
- In inheritance, the child class acquires the properties and can access all the data members and functions defined in parent class
- A child class can also provide it's specific implementation to the functions of parent class
- Provides code reusability to program
- Define new classes that are built upon existing classes

Syntax: class ParentClassName:

    class ChildClassName1 (ParentClassName):
        pass

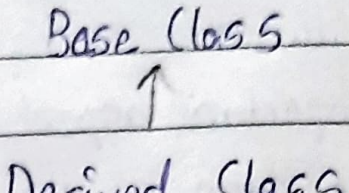
    class ChildClassName2 (ChildClassName1):
        pass

Advantages: Reusability
Reduce redundancy of data/code
Represents real life concepts/problem domain
Extensibility

3 types

i) Single level Inheritance

When a child class inherits from only one parent class
(1 parent class - 1 child class)



Syntax:

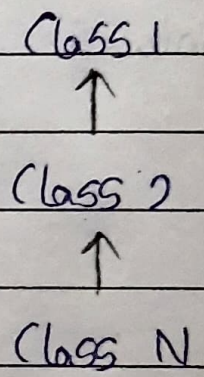
```
class derived-class(base class):
  <class-suite>
```

ii) Multilevel Inheritance

When a derived class inherits from another derived class
(1 parent class - 1 child class - Sub class child)

Syntax:

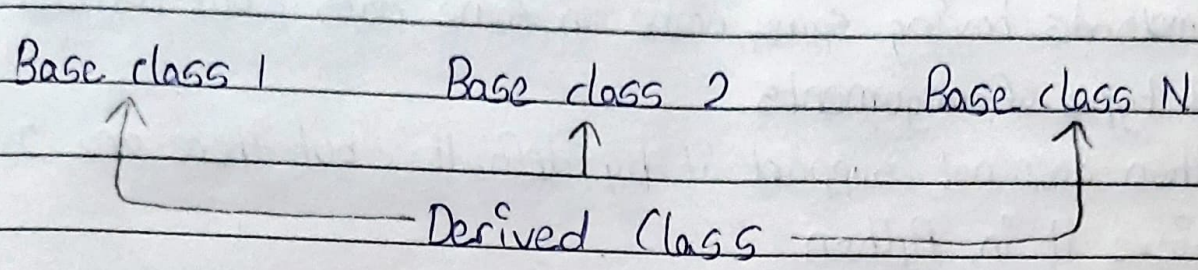
```
class class1:
    <class suite>
class class2(class1):
    <class suite>
class class3(class2):
    <class suite>
    .
    .
    .
```



Date :

iii) Multiple Inheritance

When a child class inherits from multiple parent classes



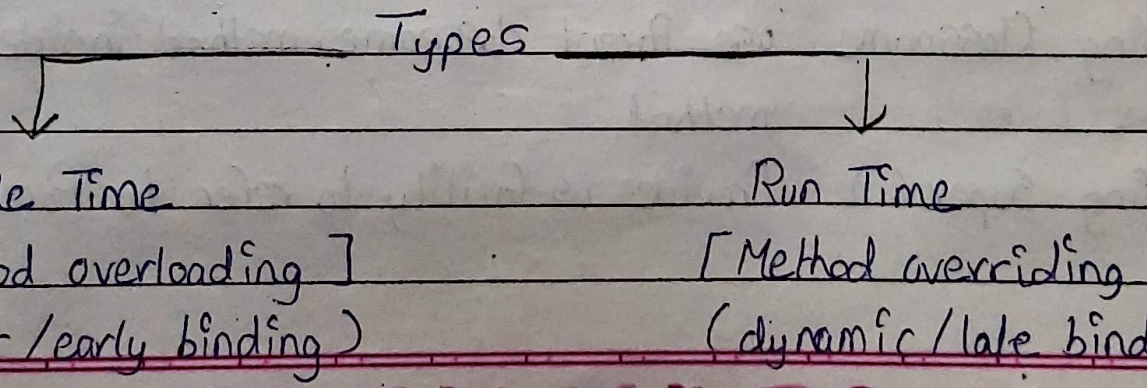
Syntax:

```
class Base1:
    <class suite>
class Base 2:
    <class suite>
    :
class Base N:
    <class suite>

class Derived (Base1, Base2, ... BaseN):
    <class suite>
```

* Polymorphism

Same method name but different implementations



1) Method Overloading - compile time polymorphism

- 2 methods having same name in same class, but different number and type of arguments
- Python does not support it by default, but there are 2 ways to achieve it in Python

Method 1 (Not the most efficient method):
We can use the arguments to make the same function work differently i.e as per the arguments

Method 2 (efficient)
By using Multiple Dispatch Decorator

2) Method overriding - an time polymorphism

- Allows a subclass or class child to provide a specific implementation of a method that is already provided by one of it's supper classes or parent classes
- Calling Parent's method within the overriden method by 2 ways

Using Classname: use Parent classname.method inside overriden method

Using Super(): Provides us facility to refer to Parent class

Date :

In Method overriding, Parent and child class - methods have same name with same prototypes
While compiling the program with overridden method calls, Compiler is not able to resolve such method calls
Hence overridden methods are resolved with correct Implementation at run time by PVM

- Diamond Problem
Occurs when 2 classes have a common ancestor and other class has both those classes as base classes

- Class Decorators

@staticmethod - used to declare a method as class/static method
@property - for creating getter method
@methodname.setter - for creating setter method
@final - to prohibit inheritance and method overriding
@abstractmethod - to create abstract method

* Exception
- Is an abnormal condition that occurs in program at run-time
- A run time error
- Abnormally terminates the program or disrupts normal flow of program
Reasons:
Due to wrong user input
Logical mistake

Q) Difference between Syntax error and Exception

→

Syntax error: As the name suggests this error is caused by wrong syntax in code.
It leads to termination of program

Exceptions: Raised when program is syntactically correct but the code resulted in error
Error doesn't stop the execution of program, but it changes normal flow of program
Base class for all the exceptions in Python

* Exception Handling
Mechanism to handle run time errors in well defined way
Advantages:
- Avoid abnormal termination of program
- To maintain normal flow of application
- To handle possible error conditions
- Robust and Reliable software

* Assertion
Is a sanity check that you can turn on or off when you are done with your testing of program
Are carried out by assert statement

Date :

Q) Assertion vs Exception

→

Assertion are intended to be used solely as a means of detecting programming errors (bugs)
Exception indicate other kinds of error or "exceptional condition"

Q) Raise vs Assert

→

Raise is typically used when you have detected an error condition or some condition does not satisfy
Assert is similar but the exception is only raised if a condition is met
Both have a different philosophy

Raise - raise an exception
Assert - raise an exception if given condition is met
try - execute some code that might raise an exception and if so, catch it

Q) Error vs Exception

→

Errors cannot be handled, while Python exceptions can be handled at the run time
An error can be a syntax error, while there can be many types of exceptions

* Type of Errors

1) Syntax error
- Parsing error
- Caused when the parser detects a syntactic issue in code.

2) Out of memory error
- Memory errors are dependent on system RAM and are related to heap
- Caused due to:
  Using a 32 bit Python architecture
  loading a very large data file
  Running a ML Model

3) Recursion error
- Related to stack and occurs when you call function
- Transpires when too many methods one inside another is executed viz. limited by size of stack

4) Exception

Date :

* Etending and Embedding Python

Extending : Invoking C function from Python
Embedding : Embedding Python code into C program

Extending is more useful
Advantages
Improve time Complexity
Parallilize your code (Multitasking like task)
Move slow parts of code to a faster language

* File Stream Handling

. Files
- A permanent storage area containing any kind of data stored on secondary storage devices
- Managed by Operating system through File system applications
- Big data files are managed by Distributed File system
- File data is transferred to RAM

. Streams
- A memory area in RAM with contiguous arrays of bytes
- Unidirectional
- Has 2 ends. One attached to Python program and other to any I/O device

2 types of Streams

- Input Stream - Data flows from device to Python program
  - Used to read data from file

- Output Stream - Data flows from Python program to device
  - Used to write data to file

* Opening a File in Python

2 types of files can be handled in Python

- Text files
- Binary files

open() function is used

Syntax: File_object = open("File_name", "Access_mode")

Opening modes:

r	Read mode	
w	Write mode	
a	Append mode	
t	Text mode	Accompanied with r, w, a (Default mode)
b	Binary mode	
r+	Read and write	
w+	Write and Read	
a+	Append and Read	

Q) Difference between r+, w+, a+

→

All allow reading and writing

For

r+, an error occurs if file does not exist and the exisisting data is not discarded but can be overwritten

w+, the exisisting data is always discarded

a+, the exisisting data is not discarded and cannot be overwritten

* Closing a file in Python

close() method is used