**Object Oriented programming**

**Notes**

1. **Why do you need oops?**

* To **create a new datatype**, we need to create a class in which we must define the properties of the new datatype is to create.
* To bring **real-world scenarios to virtually** such as an object behaving differently in different situations or cloning a function or method with multiple functionality oops is required.
* By creating a class, we can restrict access to methods that can be accessible only by the respective class object

1. **Advantage of oops?**

* **Reusability** - Allows reuse of code [ inheritance]
* **Modularity** – breaking the code into small pieces. Which makes it easy to modify and maintain.
* **Encapsulation** – hiding of data (private variable and exposing only necessary functionality.
* **Scalability** – The project grows in complexity and adds more classes and objects, hierarchies without breaking the existing functionality.
* **Abstraction** –
* We can set the access level to the variable the variable inside the class.
* Ease of maintenance

1. **Disadvantage of oops?**

* Increased complexity – can be used for only huge tasks but it might be overkilled for simple processes.
* Slower Performance – compared to procedural language.
* High memory consumption – storing objects requires more memory compared to traditional data structures
* Learning curve – concepts like inheritance, polymorphism, and encapsulation can be challenging to beginners.

1. Class?

* Class is a template, design, or blueprint of an object is called class.
* Inside the class you will define all the properties of the object
* A class can have n number of objects.
* In an empty class there are 27 magic methods that will be automatically loaded into the class. They are called magic methods or dunder methods means “double underscores”
* SYNTAX:

class class\_name:

#attributes ----- >states (variable) and behaviour(methods)

1. Object?

* The instance of the class which has its own states and behaviours is called an object.
* SYNTAX:  
   object\_references = class\_name(arguments)

1. Attribute classification

Class variable

States (variables)

Instance variable

Attributes

Magic methods or

Dunder methods

Normal Instance method (self)

Instance method

Static method

Class method

Behaviour (method)

* States – variable defined inside the class is called as states of the object.
* Behaviours- method define inside the class is the called as behaviours of the object
* Class variable – variable defined inside the class but outside the method is called class variable.
* Instance variable – to create a instance variable an constructor is required. A dunder method \_\_init\_\_ acts as a constructor in python.
* Whenever class is invoked init will be called automatically or indirectly.

1. Four pillars of OOPS?

* Encapsulation
* Inheritance
* Polymorphism
* Abstraction

1. Encapsulation?
   * Binding data into a class is called encapsulation.
   * It should also follow the rules of python bean class.
   * Rules of **python bean class**
     1. All instance variables should be private.
     2. All getter and setter method should be public.
     3. All private variables must have getter and setter methods.
2. Package and modules.

* Package - Folder which contains sub package & modules are called packages.
* Modules – python file ends with .py contains class, objects and reusable function are called modules.

1. Access Specifiers?
   * Attributes of the class can be used within the class outside of the class, within the modules, within the package or outside of the package.
   * There are three types of Access Specifiers.
     1. Public – no underscore before attribute
     2. Protected – single underscore before attribute
     3. Private – double underscore before attribute
2. Public access specifiers?
   * Attributes define with no underscore prefix to it called public access specifiers
   * public attribute can be accessed
     1. within the class
     2. outside the class within the module.
     3. outside the class in another module within the package.
     4. outside the package in another package within main package.
3. Protected access specifiers?
   * Attributes define with single underscore prefix to it called public access specifiers.
   * Protected attribute can be accessed
     1. within the class
     2. outside the class within the module.
     3. outside the class in another module within the package.
     4. Protected members cannot be accessed outside the package without inheritance but can be accessed by the object of the class which has **Is-a-relationship (inheritance)**
4. Private access specifiers?
   * Attributes define with single underscore prefix to it called public access specifiers.
   * Private attribute can be accessed
     1. Only within the class