PYTHON PROGRAMMING
OOPs in python

S ROHITH ADITHYA

INTRODUCTION

- Object-oriented programming (OOP) is a method of structuring a program by bundling related properties and behaviors into individual objects. In this tutorial, you'll learn the basics of object-oriented programming in Python.
- Conceptually, objects are like the components of a system. Think of a program as a factory assembly line of sorts. At each step of the assembly line, a system component processes some material, ultimately transforming raw material into a finished product.
- An object contains data, like the raw or preprocessed materials at each step on an assembly line. In addition, the object contains behavior, like the action that each assembly line component performs.
- objects are at the center of object-oriented programming in Python.

PYTHON OOPS CONCEPTS

- OOP is a programming paradigm that structures code using objects and classes.
- A class is a blueprint for creating objects.
- Creating an object from a class is called instantiation.
- Attributes store data (e.g., name, age).Methods perform actions (e.g., walking, talking). Classes can inherit attributes and methods from other classes.
- Polymorphism allows objects of different classes to be treated uniformly.
- Encapsulation bundles data and methods into a single unit (object). OOP promotes modular, reusable, and maintainable code.
- Python supports OOP principles, making it a powerful language for software development.
- OOP in Python provides a structured way to model real-world entities using objects and classes. By understanding the basics, you can create well-organized and efficient code.

DEFINE A CLASS IN PYTHON

In Python, you define a class by using the class keyword followed by a name and a colon. Then you use .__init__() to declare which attributes each instance of the class should have

EX-class students:

```
def __init__(me,name, age):
    me.name =  name
    me.age = age
```

Here class is defined as students.

INSTANCE METHODS

- Instance methods are functions that you define inside a class and can only call on an instance of that class. Just like .__init__(), an instance method always takes self as its first parameter.

EX-class Student:

```
    species = "Human"

    def __init__(self, name, age):

        self.name = name

        self.age = age

# Instance method

    def description(self):

        return f"{self.name} is {self.age} years old"

 # instance method

    def listen(self, listen):

        return f"{self.name} listens {listen}"
```

INHERITANCE

- Inheritance is the process by which one class takes on the attributes and methods of another. Newly formed classes are called child classes, and the classes that you derive child classes from are called parent classes.
- Ex-

```
class Parent:
    hair_color = "brown"
class Child(Parent):
    hair_color=“black”
```

BASIC THINGS LEARNT IN THIS PPT.

- Define a class, which is a sort of blueprint for an object
- Instantiate a class to create an object
- Use attributes and methods to define the properties and behaviors of an object
- Use inheritance to create child classes from a parent class

CONCLUSION

- In this presentation, you learned about object-oriented programming (OOP) in Python. Most modern programming languages, such as Java, C#, and C++, follow OOP principles, so the knowledge that you gained here will be applicable no matter where our programming career takes us.

THANK YOU