5CS037 - Concepts and Technologies of AI

Prepared by:

Prakash Gautam/Sachin Kafle

Lecturer/Tutor at HCK

Agenda

- Module Leader and Lecturer/Tutor Introduction
- Module Introduction
- Week 1 lecture Coverage
  - ❑ Introduction to Python OOP
  - ❑ Magic Methods
  - ❑ Operator Overloading
  - ❑ Features of OOP: Inheritance/Composition

Python OOP

Why Object Oriented Programming?

Object Oriented Programming

Python is a multi-paradigm programming language. It supports different programming approaches.
One of the popular approaches to solve a programming problem is by creating objects. This is known as Object-Oriented Programming (OOP).

An object has two characteristics:
attributes
behavior

What are attributes and what are behaviors?

Attributes(Appearances):

1. Color
2. Mileage
3. Cc
4. Weight

Methods (Behavior):

1. Move_forward
2. Turn_Left
3. Turn_right
4. Stop

```
Class Bike:
        color = “Black”
        mileage = 40kmpl
        def move_forward(self):
                print(“{} bike is moving forward”.format(self.color))



#making objects
```

```
Class Bike:
        color = “Black”
        mileage = 40kmpl
        def move_forward(self):
                print(“{} bike is moving forward”.format(self.color))



#making objects

Suzuki = Bike()
Suzuki.move_forward()
```

```
Class Bike:

        def __init__(self, color, mileage):
                self.color = color
                self.mileage = mileage

        def move_forward(self):
                print("{} bike is moving forward".format(self.color))


#making objects

Suzuki = Bike('Black", 50)
Suzuki.move_forward()          #Black bike is moving forward

Honda = Bike("Blue", 67)
Honda.move_forward              #Blue color bike is moving forward
```

Magic Methods – Operator Overloading

Magic methods are special methods that you can define to add 'magic' to your classes. They are always surrounded by double underscores, for **example**, the __init__ and __str__ magic methods.

Usage of Magic Methods – Operator Overloading

Operator Overloading means giving extended meaning beyond their predefined operational meaning.

Example of operator overloading

Operator	Magic Method
+	__add__(self, other)
–	__sub__(self, other)
*****	__mul__(self, other)
/	__truediv__(self, other)
//	__floordiv__(self, other)
%	__mod__(self, other)
**	__pow__(self, other)
>>	__rshift__(self, other)
<<	__lshift__(self, other)
&	__and__(self, other)
|	__or__(self, other)
^	__xor__(self, other)


THANKS FOR LISTENING!
ANY QUESTIONS?