Python Programming
Unit 05 – Lecture 01 Notes
OOP Basics, Classes, Objects, Constructors

Tofik Ali

February 14, 2026

Contents

1 Lecture Overview

Object-Oriented Programming (OOP) helps you design programs by modeling entities as **objects** that combine:

- **data** (attributes),
- **behavior** (methods).

This lecture introduces classes, objects, constructors, and class vs instance variables.

2 Core Concepts

2.1 Class and Object

Class: blueprint (definition).
Object: instance created from the class.

2.2 Defining a Class

```
class Student:
    def __init__(self, name, sapid):
        self.name = name
        self.sapid = sapid

    def display(self):
        print(self.name, self.sapid)
```

2.3 What is `self`?

`self` refers to the current object. When you call `s.display()`, Python internally calls `Student.display(s)`.

2.4 Constructor: `__init__`

`__init__` initializes a new object.

```
s1 = Student("Asha", "5001")
```

2.5 Special Methods (Dunder Methods)

Special methods start and end with double underscores. Common examples:

- `__init__`: constructor
- `__str__`: string representation (used by `print`)
- `__repr__`: developer representation

```
class Student:
    def __str__(self):
        return f"{self.name} ({self.sapid})"
```

2.6 Class Variables vs Instance Variables

Class variable is shared among all objects:

```
class Student:
    college = "UPES"
```

Instance variables belong to each object:

```
def __init__(self, name):
    self.name = name
```

3 Demo Walkthrough

File: `demo/student_class_demo.py`

Observe:

- multiple objects share the class variable `college`,
- each object has its own `name`, `sapid`, and marks list,
- `__str__` makes printing objects easy.

4 Interactive Checkpoints (with Solutions)

Checkpoint 1 Solution

Question: purpose of `self`?

Answer: It refers to the current object and allows methods to access the object's data.

Checkpoint 2 Solution

Question: how many copies of a class variable exist?

Answer: One shared copy in the class (objects reference it).

5 Practice Exercises (with Solutions)

Exercise 1: Rectangle Class

Task: Create a `Rectangle` class with `length`, `width` and a method `area()`.

Solution:

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

r = Rectangle(4, 3)
print(r.area())
```

Exercise 2: Add `__str__`

Task: Add `__str__` to print `"Rectangle(4,3)"`.

Solution:

```
def __str__(self):
    return f"Rectangle({self.length},{self.width})"
```

6 Exit Question (with Solution)

Question: special method used as constructor?
Answer: `__init__`