Python Programming

Unit 05 – Lecture 02: Members, Class Attributes, GC, Abstract Classes

Tofik Ali

School of Computer Science, UPES Dehradun

February 14, 2026

Repository: `https://github.com/tali7c/Python-Programming`

Quick Links

Core Concepts   Demo   Interactive   Summary

Agenda

Learning Outcomes

- Use public/private naming conventions in Python classes

Learning Outcomes

- Use public/private naming conventions in Python classes
- Inspect built-in class attributes like `__dict__` and `__doc__`

Learning Outcomes

- Use public/private naming conventions in Python classes
- Inspect built-in class attributes like `__dict__` and `__doc__`
- Explain garbage collection at a high level

Learning Outcomes

- Use public/private naming conventions in Python classes
- Inspect built-in class attributes like `__dict__` and `__doc__`
- Explain garbage collection at a high level
- Create abstract classes using the `abc` module

Public vs Private in Python

- Python uses conventions (not strict access control)

Public vs Private in Python

- Python uses conventions (not strict access control)
- `name`: public

Public vs Private in Python

- Python uses conventions (not strict access control)
- `name`: public
- `_name`: internal ("protected" by convention)

Public vs Private in Python

- Python uses conventions (not strict access control)
- `name`: public
- `_name`: internal ("protected" by convention)
- `__name`: name-mangled (harder to access accidentally)

Built-in Class Attributes

- `__dict__`: attributes dictionary

Built-in Class Attributes

- `__dict__`: attributes dictionary
- `__doc__`: documentation string

Built-in Class Attributes

- `__dict__`: attributes dictionary
- `__doc__`: documentation string
- `__class__`: object's class

Built-in Class Attributes

- `__dict__`: attributes dictionary
- `__doc__`: documentation string
- `__class__`: object's class
- `__module__`: module name where class is defined

Garbage Collection (GC)

- Automatically frees memory of unreachable objects

Garbage Collection (GC)

- Automatically frees memory of unreachable objects
- Python uses reference counting + cycle detection

Garbage Collection (GC)

- Automatically frees memory of unreachable objects
- Python uses reference counting + cycle detection
- Usually you do not manually free memory

Abstract Classes

- Abstract class defines an interface (methods to implement)

```python
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass
```

Abstract Classes

- Abstract class defines an interface (methods to implement)
- Prevents creating incomplete objects

```python
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass
```

Demo: Abstract Shape

- File: `demo/abstract_shape_demo.py`

Demo: Abstract Shape

- File: `demo/abstract_shape_demo.py`
- Implements:

Demo: Abstract Shape

- File: `demo/abstract_shape_demo.py`
- Implements:
  - abstract base class `Shape`

Demo: Abstract Shape

- File: `demo/abstract_shape_demo.py`
- Implements:
    - abstract base class `Shape`
    - derived `Circle` and `Rectangle`

Checkpoint 1

Question: What does name mangling mean for an attribute like `__balance`?

Checkpoint 2

Question: Why do we use abstract classes?

Think-Pair-Share

Discuss:

- Create an abstract class `Vehicle` with method `start()`.
- What subclasses can implement it?

Key Takeaways

- "Private" in Python is mostly convention (_ and __)

Key Takeaways

- "Private" in Python is mostly convention (_ and __)
- Built-in attributes help introspection (`__dict__`, `__doc__`)

Key Takeaways

- “Private” in Python is mostly convention (`_` and `__`)
- Built-in attributes help introspection (`__dict__`, `__doc__`)
- GC manages memory automatically

Key Takeaways

- "Private" in Python is mostly convention (`_` and `__`)
- Built-in attributes help introspection (`__dict__`, `__doc__`)
- GC manages memory automatically
- Abstract classes define required methods for subclasses

Exit Question

Which module is used for abstract base classes in Python?