

CoGrammar

Week 11 – Open Class 2





Software Engineering Lecture Housekeeping

 The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
 (FBV: Mutual Respect.)

- No question is daft or silly ask them!
- There are Q&A sessions midway and at the end of the session, should you
 wish to ask any follow-up questions. Moderators are going to be
 answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes.
 You can submit these questions here: <u>Open Class Questions</u>

Software Engineering Lecture Housekeeping cont.

- For all non-academic questions, please submit a query:
 www.hyperiondev.com/support
- Report a safeguarding incident:
 www.hyperiondev.com/safeguardreporting
- We would love your feedback on lectures: Feedback on Lectures

Progression Criteria

⊘ Criterion 1: Initial Requirements

• Complete 15 hours of Guided Learning Hours and the first four tasks within two weeks.

⊘ Criterion 2: Mid-Course Progress

- Software Engineering: Finish 14 tasks by week 8.
- Data Science: Finish 13 tasks by week 8.

- Complete all mandatory tasks by 24th March 2024.
- Record an Invitation to Interview within 4 weeks of course completion, or by 30th March 2024.
- Achieve 112 GLH by 24th March 2024.

Record a Final Job Outcome within 12 weeks of graduation, or by 23rd September 2024.

Unlock Prestigious Co-Certification Opportunities

New Partnerships Unveiled!

• University of Manchester & Imperial College London join our circle along with The University of Nottingham Online.

Exclusive Opportunity:

- Co-certification spots awarded on a first-come basis.
- Meet the criteria early to gain eligibility for the co-certification.

Key Deadlines:

- 11 March 2024: 112 Guided Learning Hours & 'Build Your Brand' tasks completion.
- 18 March 2024: Record interview invitation or self-employment.
- **15 July 2024**: Submit verified job offer or new contract.



Lecture Objectives

- Review common best practices when working with classes and objects.
- 2. Apply best practices to your code.
- 3. Open Floor Q&A

OOP: Best Practices



Naming Convention

- Python classes use the CamelCase naming convention.
- Each word within the class name will start with a capital letter.
- E.g. Student, WeightExercise

class Student:

class WeightExercise:



Use Meaningful Names

- Give your classes meaningful and descriptive names.
- Other developers should already have an idea what your class is used for when they read the class name.

Not Good

class CNum:

Good

class ContactNumber:

Docstrings

 A form of documenting your classes and methods. Think of these as user manuals for each function.

```
class Pet:
   Represents a virtual pet.
   - pet_name (str): The name of the pet.
   - pet_type (str): The type or breed of the pet.
   def __init__(self, pet_name: str, pet_type: str) -> None:
       Initialize a Pet object with a name and type.
       Parameters:
       - pet name (str): The name of the pet.
       - pet type (str): The type or breed of the pet.
       Returns: None
       self.pet_name = pet_name
       self.pet_type = pet_type
```

Type Annotations

• Type hints make your code more understandable by offering developers to see what types of arguments a class or method expects while giving them an idea of what will be returned.

```
class Pet:
    def __init__(self, pet_name: str, pet_type: str) -> None:
        self.pet_name = pet_name
        self.pet_type = pet_type

def display_pet(self) -> str:
        return f"{self.pet_name} the {self.pet_type}"
```

Encapsulation

- Encapsulate data within your classes and limit access to the data through methods.
- We want to hide the internal state and implementation of the object from the outside world.
- Access Modifiers
 - Public: Attributes and methods used for accessibility from outside the class.
 - Protected: Attributes and methods used for accessibility within the class and its subclasses.
 - **Private:** Attributes and methods used for accessibility only within the class itself.

Encapsulation (Continue)

- In python we cannot enforce encapsulation but we can use naming convention to show which properties and attributes should be accessible outside of the class.
- We can add a single underscore at the start of a variable name to signify that it should be protected. E.g. _name or _grades
- We can add 2 underscores at the start of a variable name to signify that it should be private. E.g. _name or _grades

Encapsulation

• We can now add getter and setter methods to our class to control access to data.

```
class Student:

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

   def name(self):
        return self.__name

   def set_name(self, name):
        self.__name = name
```

Single Responsibility

- Make sure your classes represent a single idea.
- If we have a person class that can have a pet, we won't add all the pet attributes to the person class. We will rather create a new class.

```
class Person:

def __init__(self, name, surname, pet_name, pet_type):
    self.name = name
    self.surname = surname
    self.pet_name = pet_name
    self.pet_type = pet_type
```

Single Responsibility

```
class Person:
   def __init__(self, name, surname):
        self.name = name
        self.surname = surname
class Pet:
   def __init__(self, name, type):
        self.name = name
        self.type = type
```

Wrapping Up

Classes

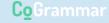
A class is a blueprint for creating objects. Objects are instances of a class, and they encapsulate attributes and the methods that operate on those attributes.

Attributes and Methods of a Class

Attributes are variables that store data within a class or an object. Methods are functions within a class that operate on its data.

Access Modifiers

- **Public:** Attributes and methods used for accessibility from outside the class. (ie. self.public_variable)
- **Protected:** Attributes and methods used for accessibility within the class and its subclasses. (ie. self._protected_variable)
- Private: Attributes and methods used for accessibility only within the class itself.
 (ie. self._private_variable)



CoGrammar

Thank you for joining

