Welcome to this CoGrammar lecture:

# Instance, Static and Class Methods

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

## Software Engineering Session Housekeeping

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

## Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and throughout the session, should you wish to ask any follow-up questions.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: Questions

## Safeguarding & Welfare

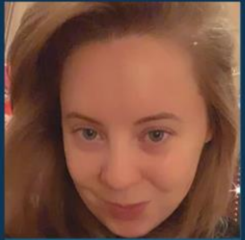
We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Rafiq Manan



Charlotte Witcher

Nurhaan Snyman

Ronald Munodawafa

Tevin Pitts

**Scan to report a safeguarding concern**

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

# Skills Bootcamp Progression Overview

To be eligible for a certificate of completion, students must fulfil three specific criteria. These criteria ensure a high standard of achievement and alignment with the requirements for the successful completion of a Skills Bootcamp.

## ✓ Criterion 1 - Meeting Initial Requirements

**Criterion 1 involves specific achievements within the first two weeks of the program. To meet this criterion, students need to:**

- Attend a minimum of 7-8 hours per week of guided learning (lectures, workshops, or mentor calls) within the initial two-week period, for a total minimum of 15 guided learning hours (GLH), by no later than 15 September 2024.
- Successfully complete the Initial Assessment by the end of the first 14 days, by no later than 15 September 2024.

CoGrammar

# Skills Bootcamp Progression Overview

## ✓ Criterion 2 - Demonstrating Mid-Course Progress

Criterion 2 involves demonstrating meaningful progress through the successful completion of tasks within the first half of the bootcamp.
To meet this criterion, students should:

- Complete 42 guided learning hours and the first half of the assigned tasks by the end of week 7, no later than 20 October 2024.

CoGrammar

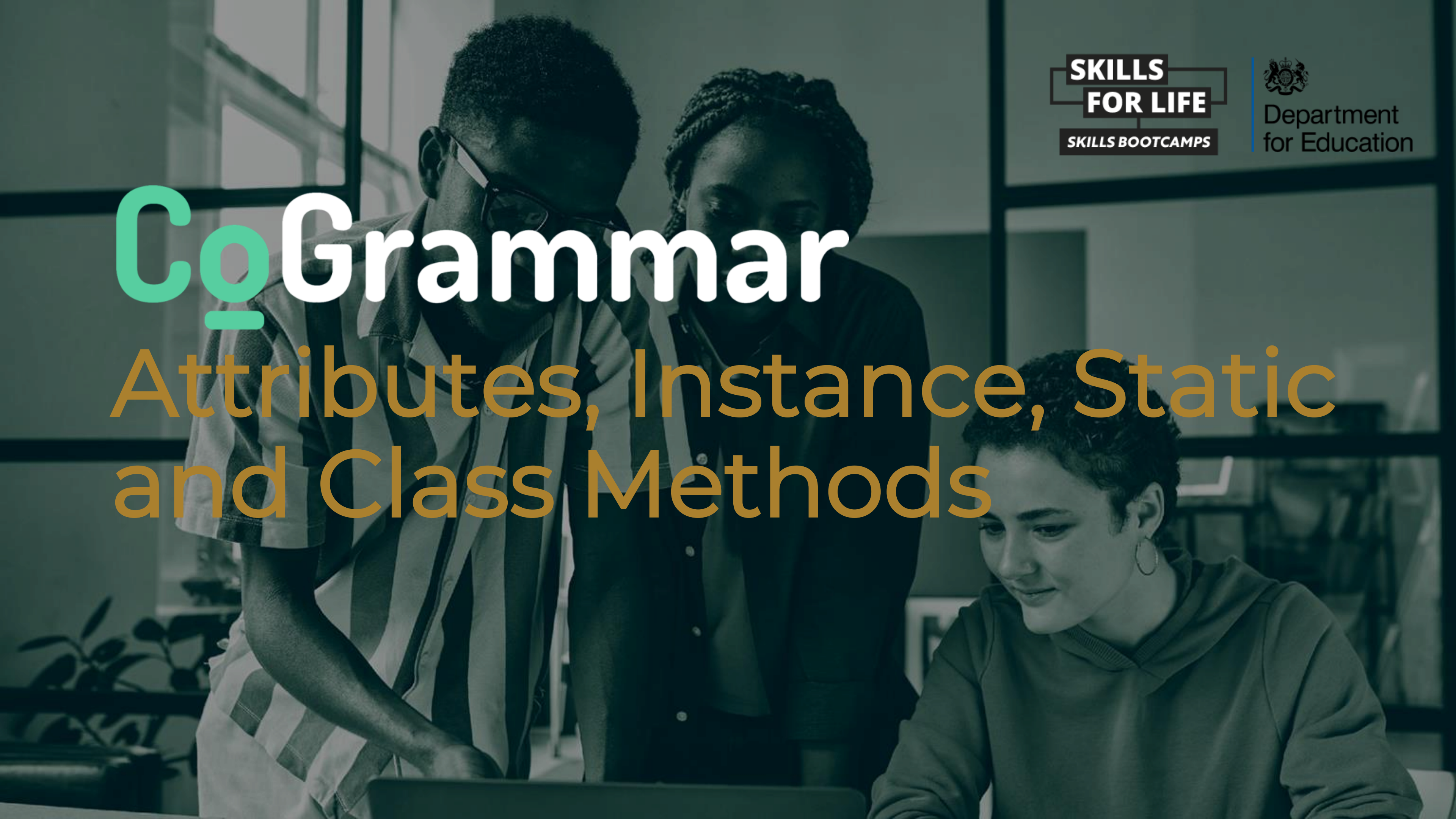
# Skills Bootcamp Progression Overview

## ✓ Criterion 3 - Demonstrating Post-Course Progress

Criterion 3 involves showcasing students' progress after completing the course.
To meet this criterion, students should:

- Complete all mandatory tasks before the bootcamp's end date. This includes any necessary resubmissions, no later than 22 December 2024.
- Achieve at least 84 guided learning hours by the end of the bootcamp, 22 December 2024.

CoGrammar



CoGrammar

# Attributes, Instance, Static and Class Methods

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

## Poll

1. **Which of the following statements best describes your understanding of instance, static, and class methods in Python?**

A. I am completely new to these concepts.

B. I understand instance methods but not static or class methods.

C. I can explain the difference between static and class methods, but I need more practice using them.

D. I feel confident using and explaining all three types of methods.

## Poll

**2. How often have you used instance, static, or class methods in your Python projects?**

A. Never used them before.

B. Rarely, I’m not confident in when to use each type.

C. Sometimes, I use instance methods mostly but struggle with static and class methods.

D. Frequently, I use all three types effectively in my code.

# Learning Outcomes

- Explain the concept of class attributes and how they differ from instance attributes.
- Define and access class attributes.
- Describe how class attributes are shared among all instances of a class.
- Explain the concept of instance attributes and how they differ from class attributes.
- Define and access instance attributes.
- Describe the purpose and use of static methods in Python.

CoGrammar

## Learning Outcomes

- Define and call static methods using the @staticmethod decorator.
- Differentiate between static methods and instance methods, particularly that static methods do not access or modify class or instance attributes.
- Describe the purpose and use of class methods in Python.
- Define and call class methods using the @classmethod decorator.

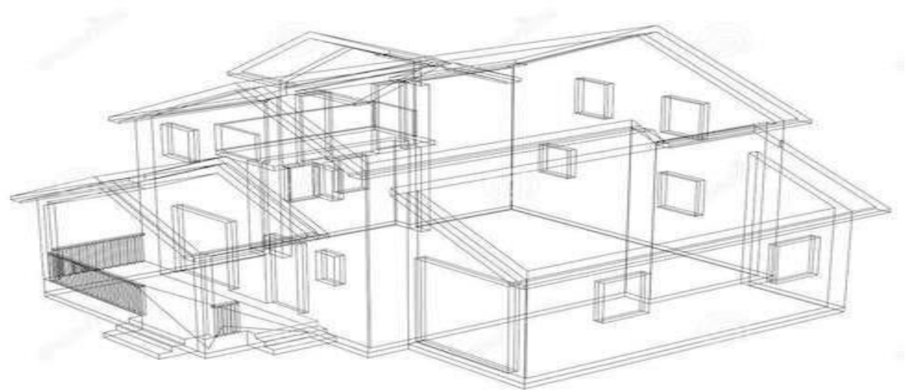
CoGrammar

## Classes

CoGrammar

## Classes - Recap

A class is a blueprint or template for creating objects. It defines the attributes and methods that all objects of that class will have.



CoGrammar

# Attributes

- Attributes are values that define the characteristics associated with an object.
- They define the state of an object and provide information about its current condition.
- For a class named ‘House’, some relevant attributes could be:
    - number_of_bedrooms
    - year_built

CoGrammar

## Methods (Behaviours)

- Methods, also known as behaviours, define the actions or behaviours that objects can perform
- They encapsulate the functionality of objects and allow them to interact with each other and the outside  world.
- For a class named ‘House’, some relevant method could be:
    - set_location(): Allows updating the location of the house

CoGrammar

## Instance Methods

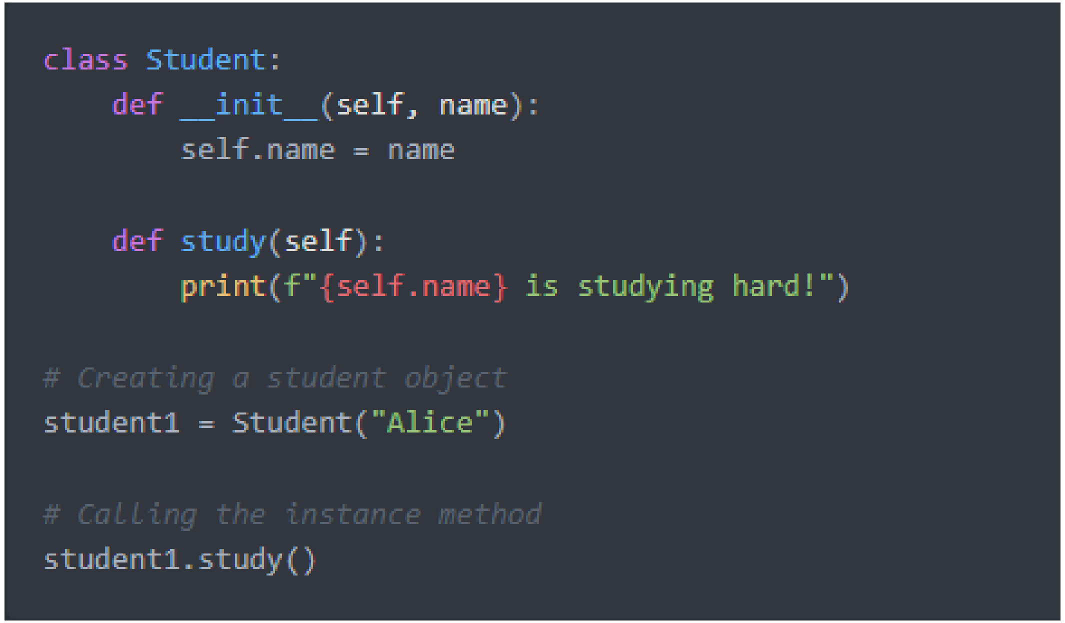
CoGrammar

## What are Instance Methods?

- Instance methods are like actions or behaviours that specific objects can perform. They have access to the object's data and are defined within the class. By using instance methods, we can model how objects interact and behave in our programs, making object-oriented programming a powerful way to structure our code.

CoGrammar

## Instance Methods - Example



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

    def study(self):
        print(f"{self.name} is studying hard!")

# Creating a student object
student1 = Student("Alice")

# Calling the instance method
student1.study()
```

CoGrammar

# Instance Methods – Real Life

- Instance Methods are like individual roles in a team.
- *Imagine a football team where each player has a specific job based on their position—strikers, defenders, and goalkeepers.*
- *Each player (or instance) has their own unique skills and responsibilities (or attributes), and when they act, they are using their individual abilities.*
- *Just like how each football player has their own role, instance methods work with the specific data (attributes) of individual objects.*

CoGrammar

## Static Methods

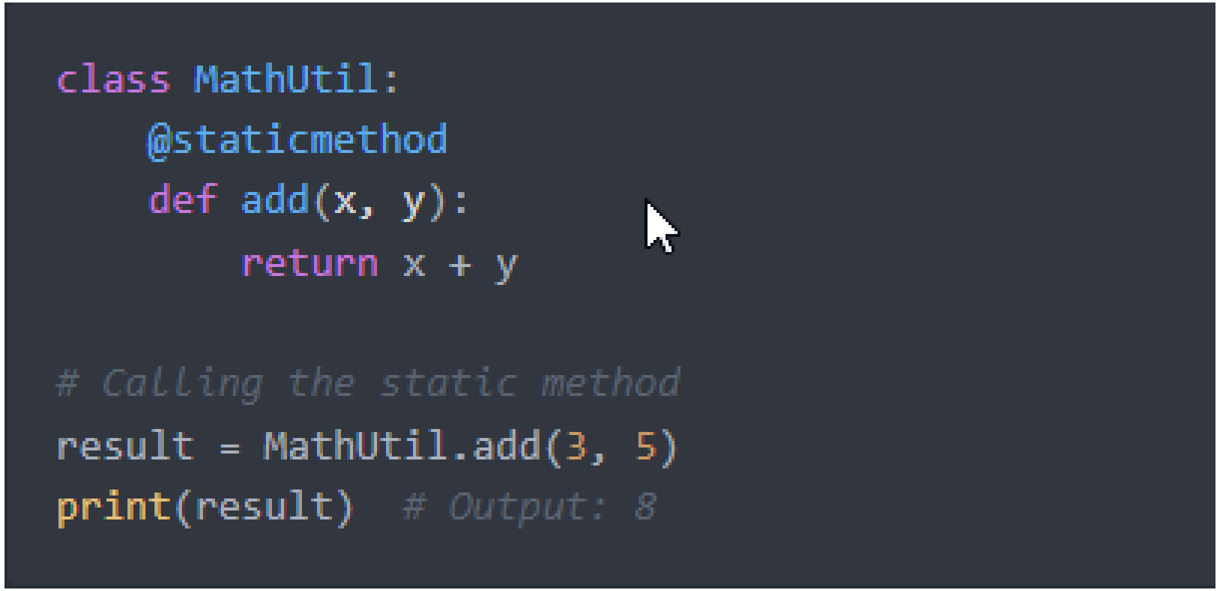
CoGrammar

# What are Static Methods?

- Static methods are like standalone functions that are associated with a class. They're useful for organising utility functions that are related to the class but don't depend on individual object instances. By using static methods, we can group together related functionalities within a class and make our code more organised and modular.

CoGrammar

## Static Methods - Example



```python
class MathUtil:
    @staticmethod
    def add(x, y):
        return x + y

# Calling the static method
result = MathUtil.add(3, 5)
print(result)  # Output: 8
```

CoGrammar

## Static Methods – Real Life

- Static Methods can be compared to general team rules.
- *These rules, such as "no player can handle the ball except the goalkeeper," apply to everyone equally.*
- *They don't depend on any particular player's skills.*
- *In programming, static methods are similar—they perform functions that are relevant to the class but do not depend on the specific data of any object.*

CoGrammar

## Class Methods

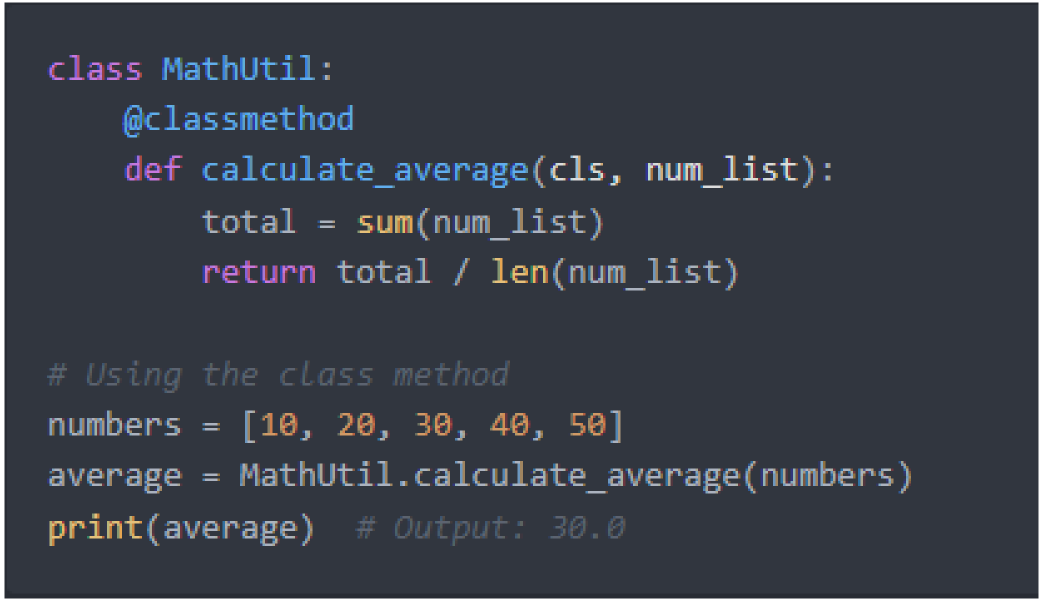
CoGrammar

# What are Class Methods?

- In object-oriented programming, a class method is a function that belongs to the class itself, rather than any specific object created from the class.
- It operates on the class as a whole, allowing modifications to class attributes or other class-wide functionalities.

CoGrammar

# Class Methods - Example



```python
class MathUtil:
    @classmethod
    def calculate_average(cls, num_list):
        total = sum(num_list)
        return total / len(num_list)

# Using the class method
numbers = [10, 20, 30, 40, 50]
average = MathUtil.calculate_average(numbers)
print(average)  # Output: 30.0
```

CoGrammar

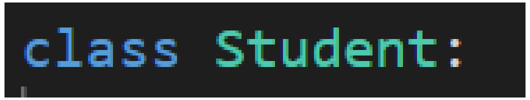
## Class Methods – Real Life

- Class Methods are like team-wide strategies.
- *Imagine the coach setting a formation or deciding the game plan.*
- *This strategy applies to the whole team, but not necessarily to each player individually.*
- *In Python, class methods operate on the class itself, and can modify class-wide attributes—much like a game plan that affects the entire team.*

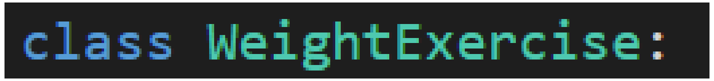
## Best Practices

## Naming Conventions

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



```python
class Student:
```

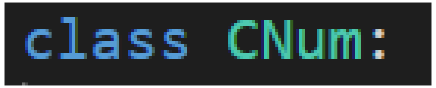


```python
class WeightExercise:
```

## Naming Conventions...

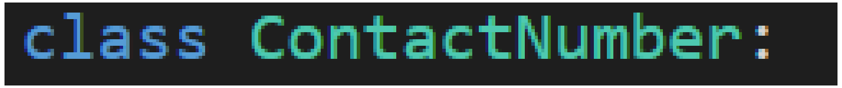
- Give your classes meaningful and descriptive names
- Other developers should already have an idea what your class is for from the name.

BAD



```
class CNum:
```

GOOD

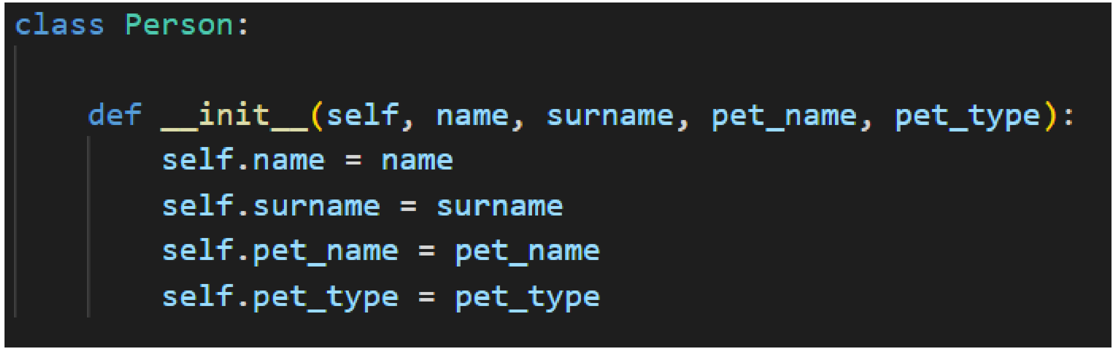


```
class ContactNumber:
```

CoGrammar

## Single Responsibility

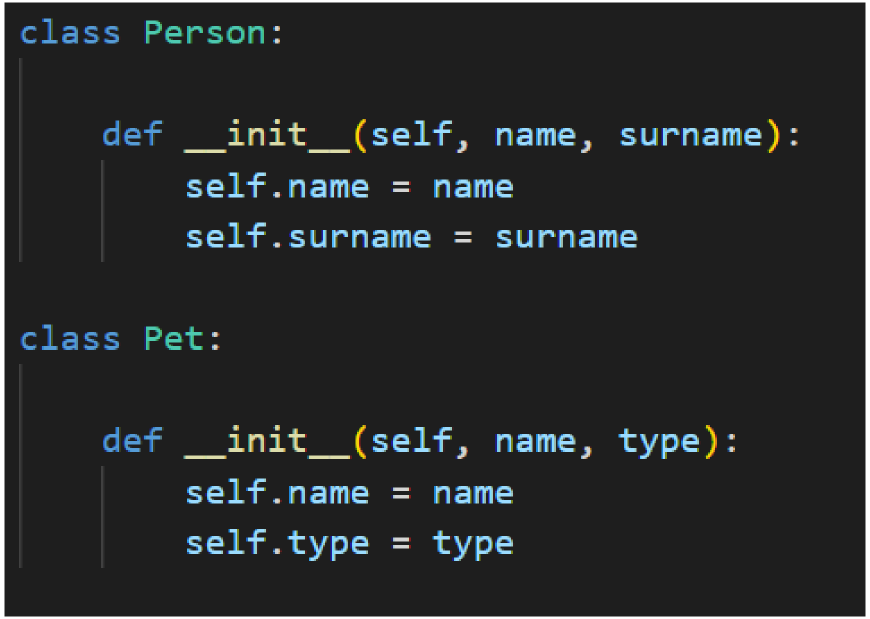
- Make sure your classes represent a single idea.
- If we have a person class where the person can have a pet, we don't want to add all the pet attributes to the person class. We will rather create a new pet class.



```python
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
```

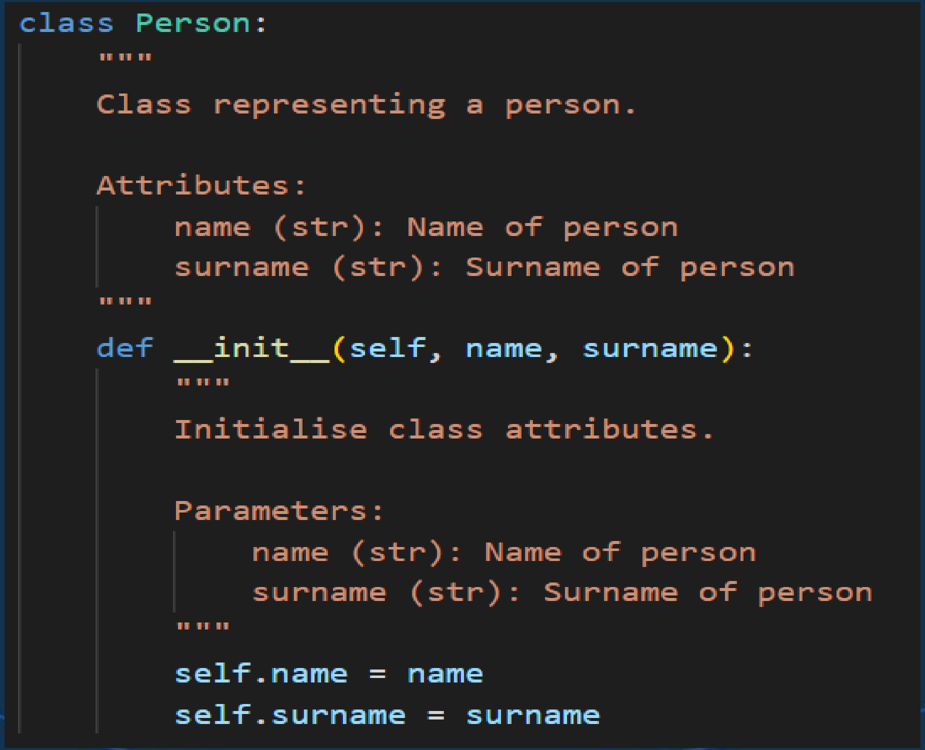
CoGrammar

## Docstrings

- We can document our classes and class methods using docstrings in the same manner we used them with functions.
- Our class docstrings will contain a short description of the class  and it’s attributes.
- A method docstring will contain a short description of the methods followed by it’s parameters and what will be returned.

CoGrammar

# Docstrings...



```
class Person:
    """
    Class representing a person.

    Attributes:
        name (str): Name of person
        surname (str): Surname of person
    """
    def __init__(self, name, surname):
        """
        Initialise class attributes.

        Parameters:
            name (str): Name of person
            surname (str): Surname of person
        """
        self.name = name
        self.surname = surname
```

## Let’s take a short break

CoGrammar

## Demo Time!

CoGrammar

## Poll

1. **Which of the following best describes the difference between instance methods, static methods, and class methods?**

A. Instance methods can access and modify both class and instance attributes; static and class methods cannot.

B. Static methods operate on class attributes, while class methods operate on instance attributes.

C. Instance methods access and modify instance attributes, class methods operate on the class itself, and static methods do not access or modify attributes.

D. All methods can modify class and instance attributes equally.

## Poll

**2. After the lesson, how confident are you in explaining the difference between class attributes and instance attributes in Python?**

A. I’m still unsure about the difference.

B. I can somewhat explain the difference but may need more practice.

C. I am confident in explaining and demonstrating the difference with examples.

D. I fully understand and can teach others the difference.

CoGrammar

## Conclusion and Recap

- Class Attributes:
  - Shared among all instances.
  - Defined and accessed at the class level.
- Instance Attributes:
  - Unique to each instance.
  - Defined and accessed through individual objects.

CoGrammar

## Conclusion and Recap

- Static Methods:
  - Defined with @staticmethod.
  - Do not modify class or instance attributes.
- Class Methods:
  - Defined with @classmethod.
  - Operate on the class itself, not individual instances.

CoGrammar

## Conclusion and Recap

- Understanding how class attributes, instance attributes, static methods, and class methods interact is crucial for building efficient and organised object-oriented programs in Python.
- By mastering these concepts, you can design more flexible and reusable code that takes full advantage of Python's OOP features.

## Learner Challenge – Option 1

- *Create a Class with Shared and Unique Attributes*:

1. Define a class Car with the following:
   - A class attribute num_wheels (set to 4 for all cars).
   - An instance attribute color to store the color of each car.
2. Create two instances of the Car class with different colours.
3. Print the number of wheels for each car (using the class attribute), and the unique colour for each instance (using the instance attribute).

**Questions to Reflect:**

- What happens when you modify the class attribute for one instance?
- Can you access the class attribute from the instance?

CoGrammar

## Learner Challenge – Option 2

- *Static vs Class Method Use Case*

1. Define a class MathOperations with:
   - A static method add(a, b) that adds two numbers and returns the result.
   - A class method change_base(cls, base) that updates a class attribute base and demonstrates using it.
2. Use the add method without creating an instance of the class.
3. Change the base using the class method, and show how it affects all instances.

**Questions to Reflect:**

- Why can the static method be called without an instance?
- How does the class method interact with the class attribute?

CoGrammar

## Tasks To Complete

- T09 – OOP Classes

## Questions and Answers

CoGrammar

# Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar