Welcome to the CoGrammar

Skills Bootcamp - Classes

The session will start shortly...

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

CoGrammar

Cyber Security 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 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 Academic Sessions. You can submit these questions here: **Questions**

Cyber Security Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: [www.hyperiondev.com/support]

- We would love your **feedback** on lectures: Feedback on Lectures

- Find all the lecture **content** in you Lecture Backpack on GitHub.
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

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

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

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

CoGrammar HyperionDev

Stay Safe Series:

Mastering Online Safety One week at a Time

While the digital world can be a wonderful place to make education and learning accessible to all, it is unfortunately also a space where harmful threats like online radicalization, extremist propaganda, phishing scams, online blackmail and hackers can flourish.

As a component of this BootCamp the *Stay Safe Series* will guide you through essential measures in order to protect yourself & your community from online dangers, whether they target your privacy, personal information or even attempt to manipulate your beliefs.

Pause Before You Post:

Managing Your Digital Presence

- Impact on Reputation.
- Permanent Record.
- Privacy Concerns.
- Miscommunication.
- Influence on Others.
- Professional Implications.
- Mental Well-being.

Learning Objectives & Outcomes

- Define what a class is and understand its role in Python's OOP structure.
- Create simple classes in Python.
- Initialize class instances (objects) and understand instance variables.
- Implement methods within classes.
- Differentiate between instance and class variables.
- Discuss the 4 pillars of OOP

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS
Department for Education

CoGrammar

Skills Bootcamp - Classes

November 2024

Classes

- Imagine you're designing a library system:
    - How would you describe a book in a way that captures both common features (like title and author) and unique details for each book

CoGrammar

Polls

Please have a look at the poll notification and select an option.

What best describes a class in Python?

a. A block of code that defines an object template.
b. A variable that stores data.
c. A loop that repeats a block of code.
d. A data structure to store multiple values.

CoGrammar

Polls

Please have a look at the poll notification and select an option.

What are methods within classes used for?

a. To define the class name.
b. To initialise class instances.
c. To perform actions with class data.
d. To break up the class.

CoGrammar

Classes

- Classes are the foundation of object-oriented programming (OOP), which allows us to organize code efficiently and design more complex systems with ease.

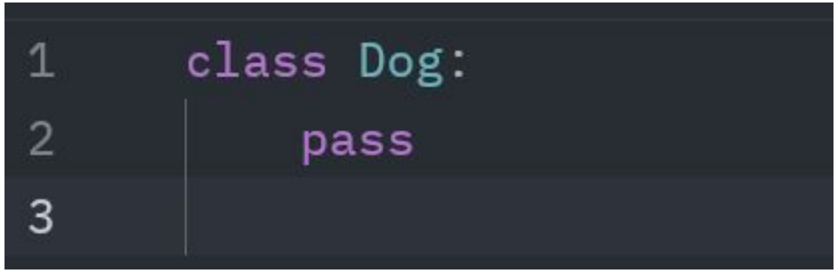
Introduction to Classes and Objects

- **Class**:
  - A blueprint for creating objects (a type or category of object).
- **Object**:
  - An instance of a class, with its own unique data and methods.
- In Python, everything is an object, including integers, strings, functions, and classes.

CoGrammar

Defining a Class

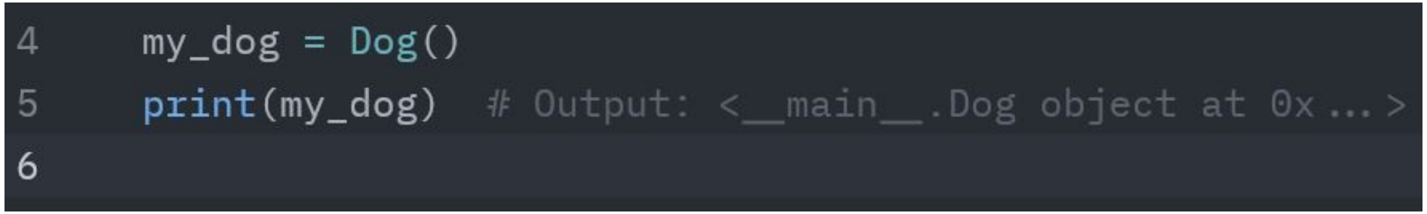
- A class is defined using the **class** keyword followed by the class name and a colon. By convention, class names are capitalized.
- In the code snippet below, Dog is a simple class with no attributes or methods yet.



```python
class Dog:
    pass

```

Creating Objects

- Objects (or instances) are created by calling the class name as if it were a function.

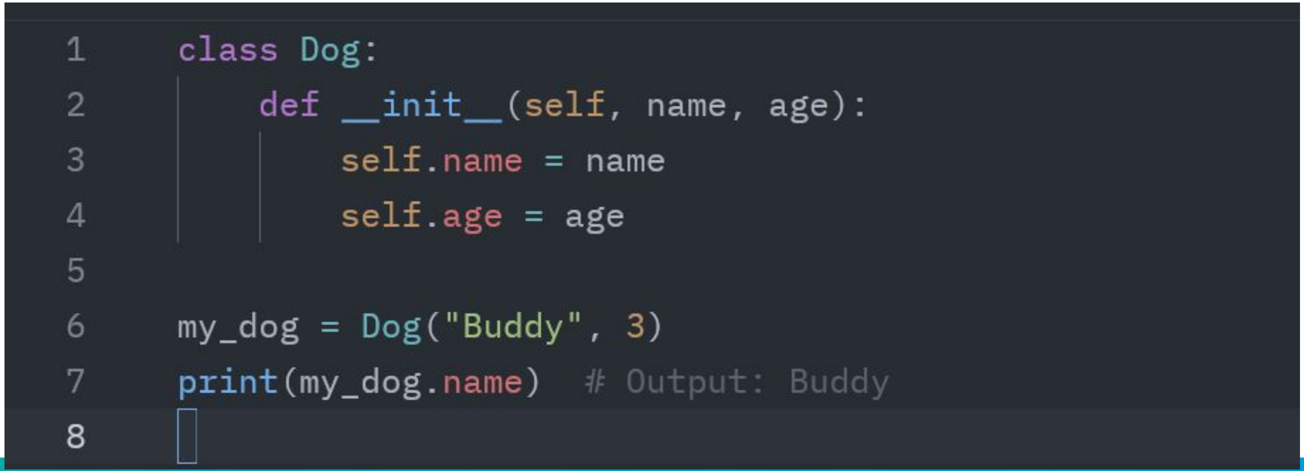


```
my_dog = Dog()
print(my_dog)  # Output: <__main__.Dog object at 0x...>
```

- Here, my_dog is an instance of the Dog class.

The __init__ Method

- The *__init__* method (also called the constructor) is a special method that initializes an object when it's created.
- The *self* parameter represents the instance being created.



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

my_dog = Dog("Buddy", 3)
print(my_dog.name)  # Output: Buddy

```

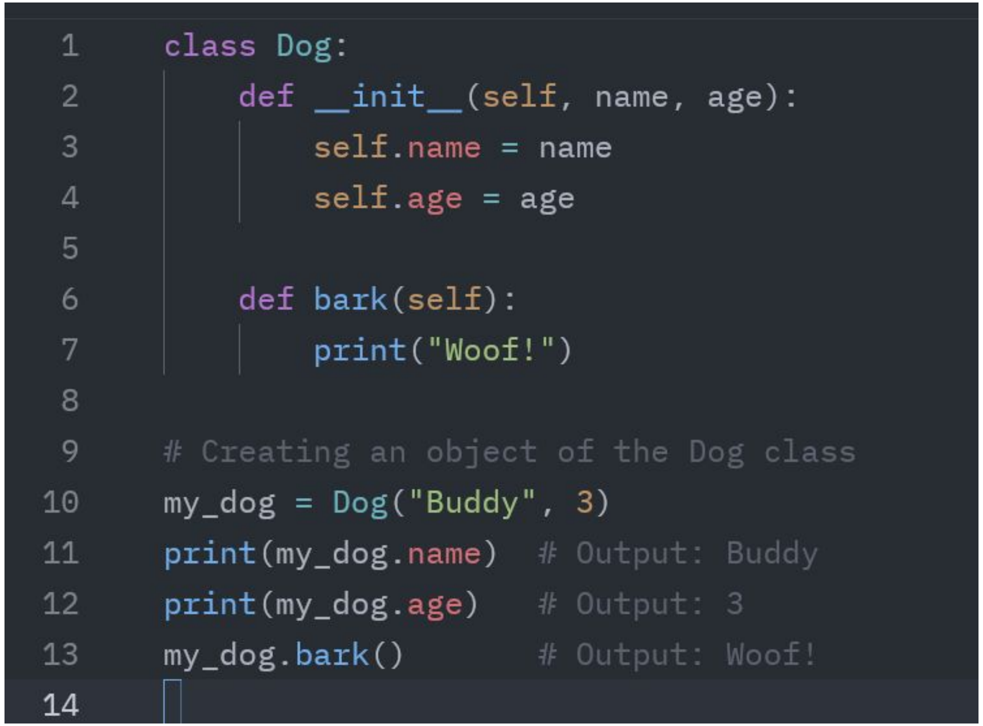
CoGrammar

Instance Variables and Methods

- **Instance variables** store data specific to each object.
- **Instance methods** are functions that operate on an instance of the class and usually access instance variables.

CoGrammar

Instance Variables and Methods



```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        print("Woof!")

# Creating an object of the Dog class
my_dog = Dog("Buddy", 3)
print(my_dog.name)  # Output: Buddy
print(my_dog.age)   # Output: 3
my_dog.bark()       # Output: Woof!
```

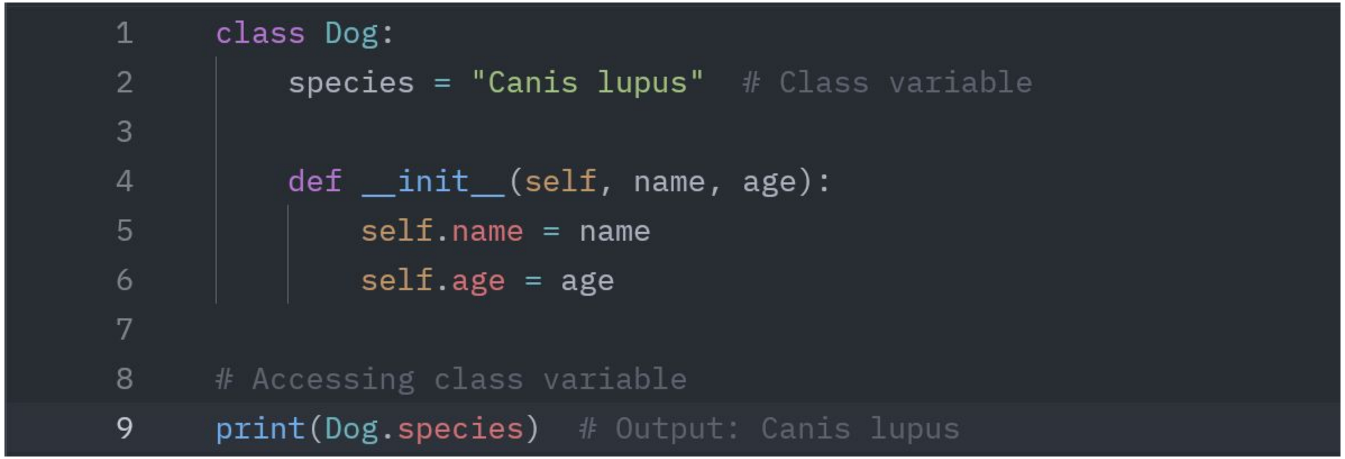
In this example:

- **__init__** is an *initializer* method called when an object is created, setting name and age as instance variables.
- **bark** is an instance method that prints "Woof!" when called on a Dog object.

CoGrammar

Class Variables and Methods

- Class variables are shared among all instances of a class.
- Class methods, defined with @classmethod, are bound to the class itself and can access class variables.



```python
class Dog:
    species = "Canis lupus"  # Class variable

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

# Accessing class variable
print(Dog.species)  # Output: Canis lupus
```

CoGrammar

Let's take a break

CoGrammar

The Four Pillars of OOP in Python

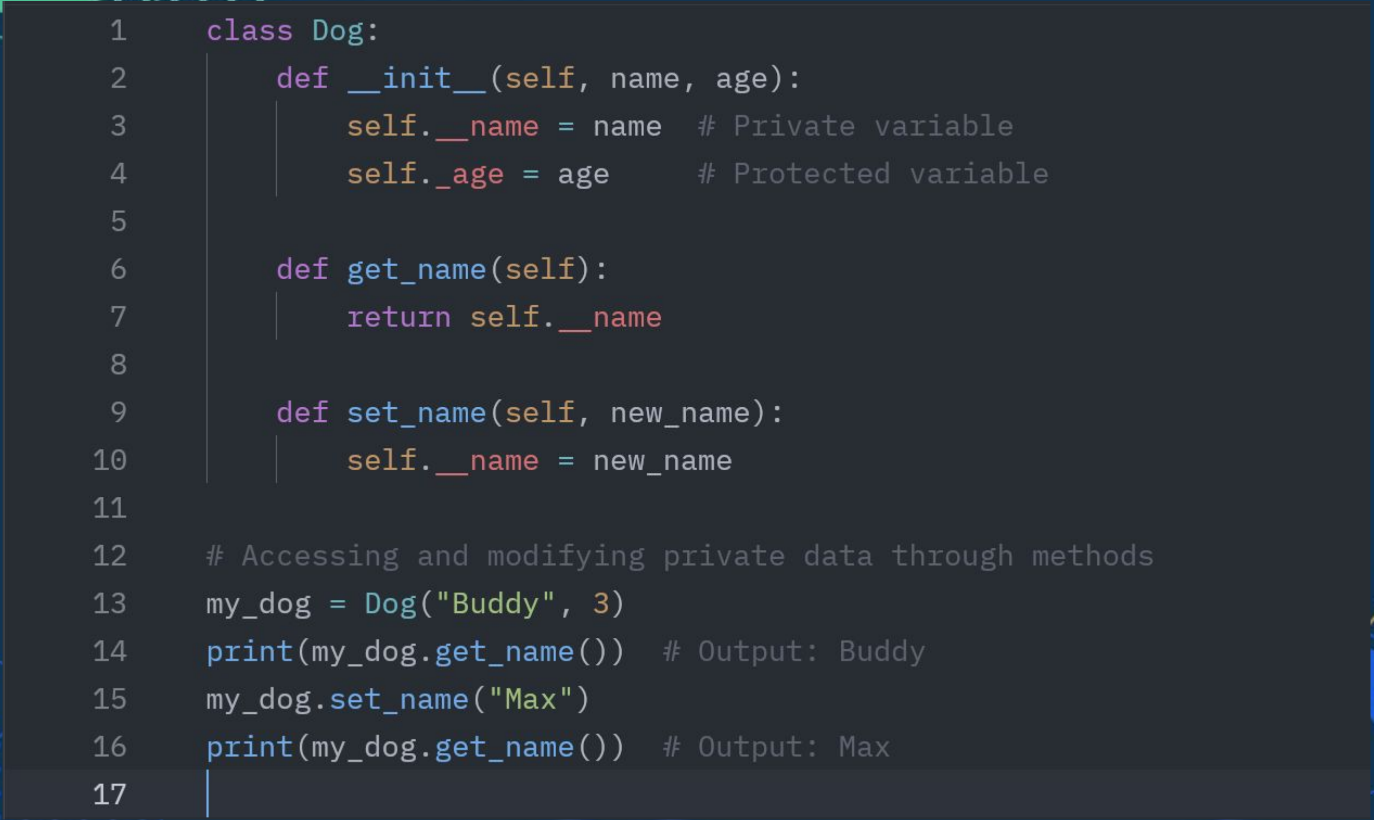
- There are four pillars of Object-oriented programming:
  - **Abstraction**: Hiding complex details.
  - **Encapsulation**: Protecting data.
  - **Inheritance**: Reusing existing code.
  - **Polymorphism**: Different forms based on the context.

Encapsulation

- **Encapsulation** is the concept of hiding data to prevent direct access.
- In Python, prefixing an attribute with an underscore _ or double underscore __ indicates a private or "protected" variable.
- Purpose:
  - Protects an object's internal state from unintended changes.
  - Provides controlled access to the object's data through methods (getters and setters).

CoGrammar

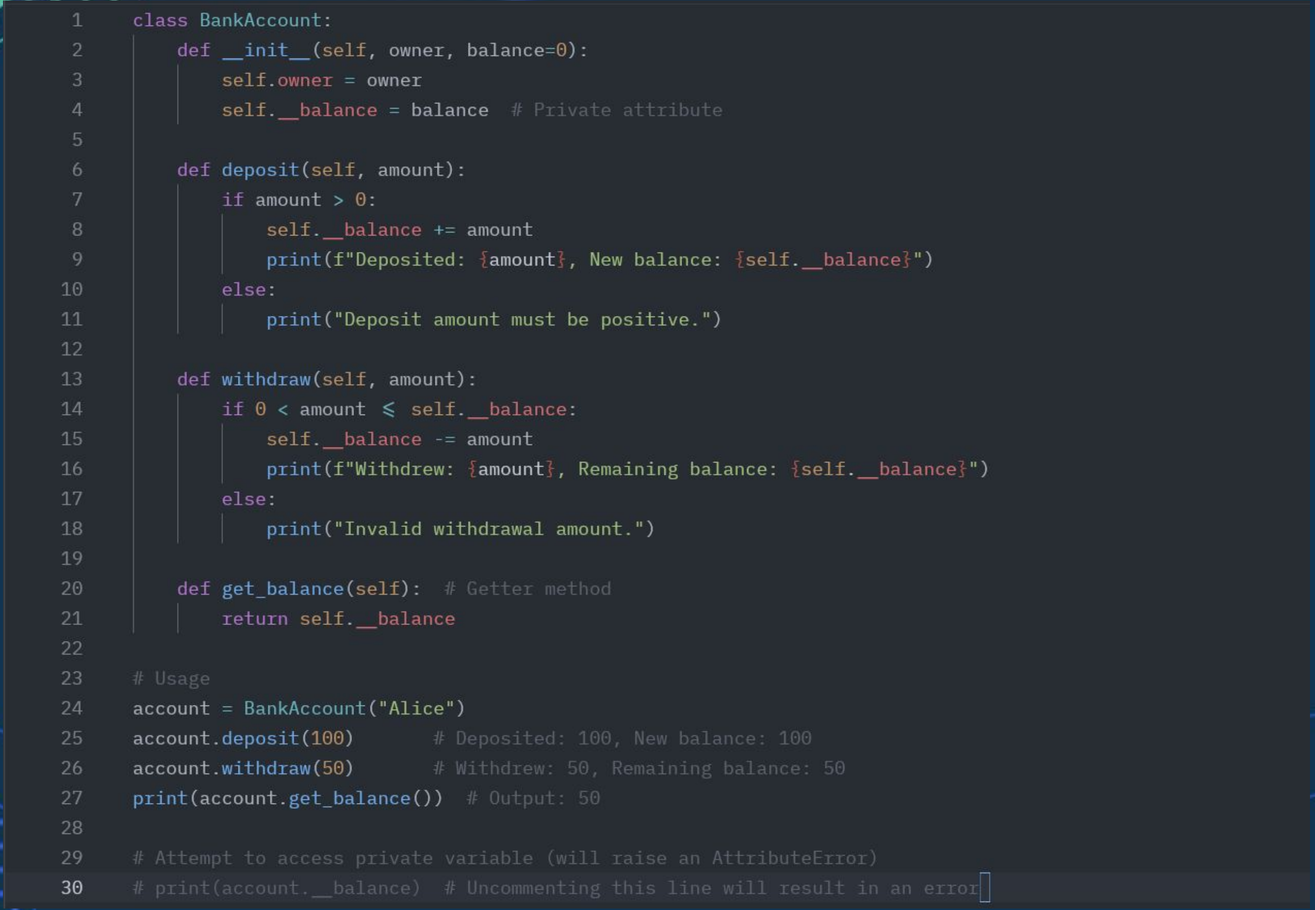
Encapsulation



```python
class Dog:
    def __init__(self, name, age):
        self.__name = name  # Private variable
        self._age = age     # Protected variable

    def get_name(self):
        return self.__name

    def set_name(self, new_name):
        self.__name = new_name

# Accessing and modifying private data through methods
my_dog = Dog("Buddy", 3)
print(my_dog.get_name())  # Output: Buddy
my_dog.set_name("Max")
print(my_dog.get_name())  # Output: Max
```

CoGrammar

Code Example



```python
class BankAccount:
    def __init__(self, owner, balance=0):
        self.owner = owner
        self.__balance = balance  # Private attribute

    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount
            print(f"Deposited: {amount}, New balance: {self.__balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if 0 < amount <= self.__balance:
            self.__balance -= amount
            print(f"Withdrew: {amount}, Remaining balance: {self.__balance}")
        else:
            print("Invalid withdrawal amount.")

    def get_balance(self):  # Getter method
        return self.__balance

# Usage
account = BankAccount("Alice")
account.deposit(100)       # Deposited: 100, New balance: 100
account.withdraw(50)       # Withdrew: 50, Remaining balance: 50
print(account.get_balance())  # Output: 50

# Attempt to access private variable (will raise an AttributeError)
# print(account.__balance)  # Uncommenting this line will result in an error
```

CoGrammar

Summary

- **Classes and Objects:** Introduced the fundamental concepts of classes as blueprints and objects as instances of those classes.
- **Instance Variables and Methods:** Explained how to define and access instance variables and create methods that operate on those instances.
- **The __init__ Method:** Highlighted the purpose of the constructor method to initialize objects with specific attributes.
- **Encapsulation:** Covered encapsulation to protect data with private and protected variables.
- Introduced the four pillars of OOP

CoGrammar

Polls

Please have a look at the poll notification and select an option.

Which of the following best describes the purpose of a class method?

a. To create an instance of the class.
b. To perform actions on class data.
c. To define class attributes.

Polls

Please have a look at the poll notification and select an option.

What is encapsulation in OOP?

a. Allowing unrestricted access to attributes.
b. Bundling data and methods into a class to protect it.
c. Using inheritance to extend class functionality.

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar