CoGrammar

Introduction to Object Oriented Programming

SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

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: **Open Class Questions**

CoGrammar

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.

✅ **Criterion 3: Post-Course Progress**

- 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.

✅ **Criterion 4: Employability**

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

Lecture Objectives

1. Develop a concise understanding of OOP and articulate its application in programming.

CoGrammar

Recap: Functions

Functions

- We can use python built-in functions or we can define our own functions with their own behaviours.

Parameters Variables

- We use parameter variables to receive input to use within the function.

Function Scope

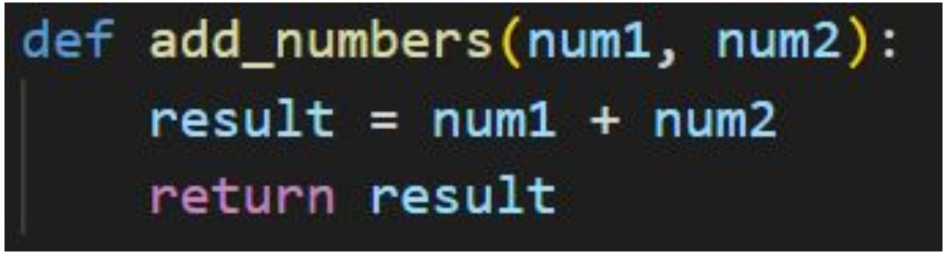
- Functions can use global variables but the main program can’t access variables within the function.

Return

- We can return data from a function using the ‘return’ keyword.

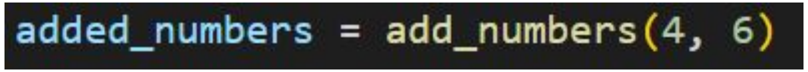
Recap: Functions

Defining a Function



```python
def add_numbers(num1, num2):
    result = num1 + num2
    return result
```

Calling a Function



```python
added_numbers = add_numbers(4, 6)
```

Poll:

Assessment

What is Object Oriented Programming ?

OOP is a way of organizing code around objects, which are self-contained modules that contain both data and instructions that operate on that data.

Why use OOP ?

- OOP promotes encapsulation by bundling data and behaviour together within objects.

- OOP promotes abstraction by focusing on essential characteristics and behaviours of objects, hiding the underlying implementation details.

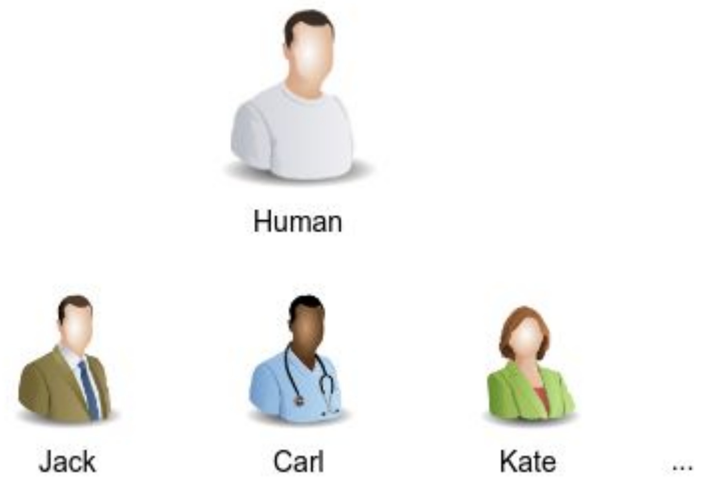
- OOP promotes code organisation into independent modules called classes. This separation of concerns allows developers to focus on specific tasks without worrying about the intricacies of other parts of the program.

- OOP reduces code duplication and simplifies development effort.

https://www.freecodecamp.org/news/python-write-to-file-open-read-append-and-other-file-handling-functions-explained/

CoGrammar

Example



What are Objects ?

- An object is a fundamental building block that represents a real-world entity or concept. It encapsulates both data and behaviour.

- Objects represent key characteristics or attributes of real world entities.

- Objects also encapsulate the actions or behaviours associated with real-world entities.

Question:

What are the benefits of using OOP in Python programming?

Objects in Python

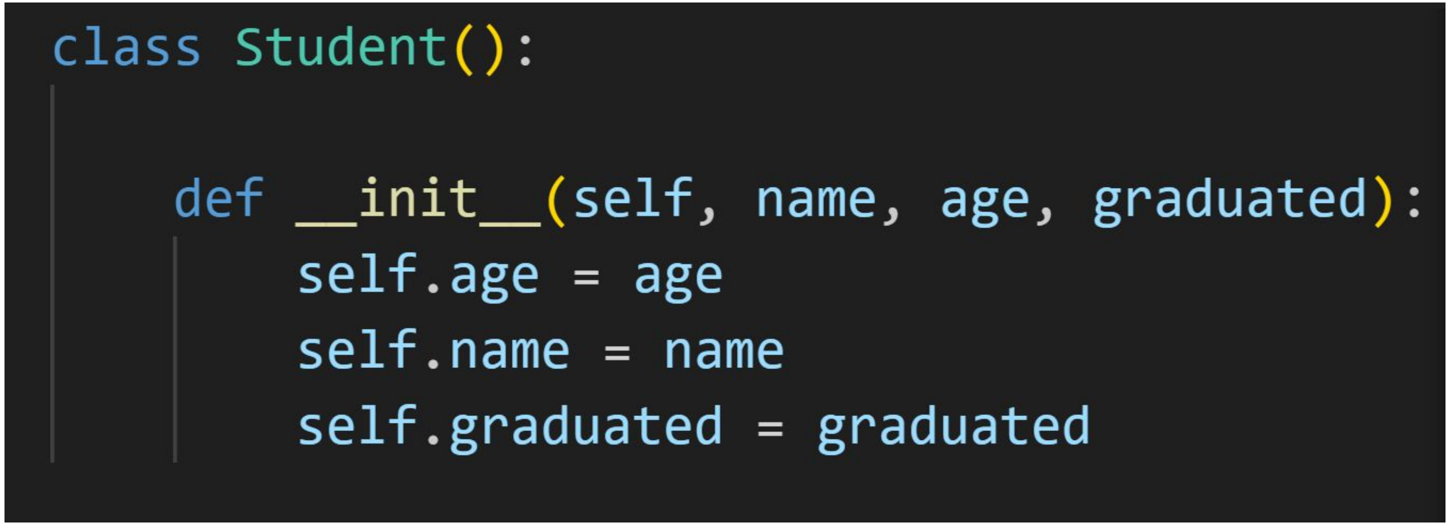
- In Python, everything is an object. Every entity, including data values and functions, are considered objects.
- They allow you to hide the internal implementation details of data and only expose methods for interacting with data.
- Without knowing it, you have actually been using objects in Python.
- For example: string.split() - this uses the split() method present in the string object.
- Imagine needing to call split(string, delimiter) - not as powerful of a notation!

Class Properties

- Class properties consist of all the methods that an objects has.
- These can be accessed using the “.” e.g. string.upper() - this calls the upper() method present in the string object.
- FUN/USEFUL FACT: You can actually see all of the properties an object using dir().

Creating a Class

- __init__ function is called when class is instantiated.



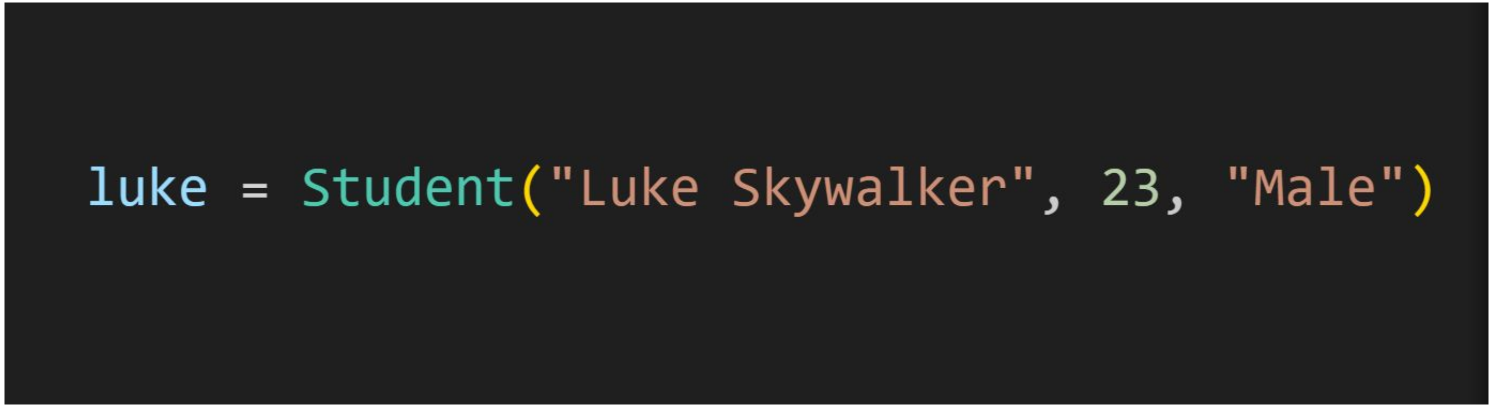
```
class Student():

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

CoGrammar

Class Instantiation

- Class takes in three values: a name, age and grade.



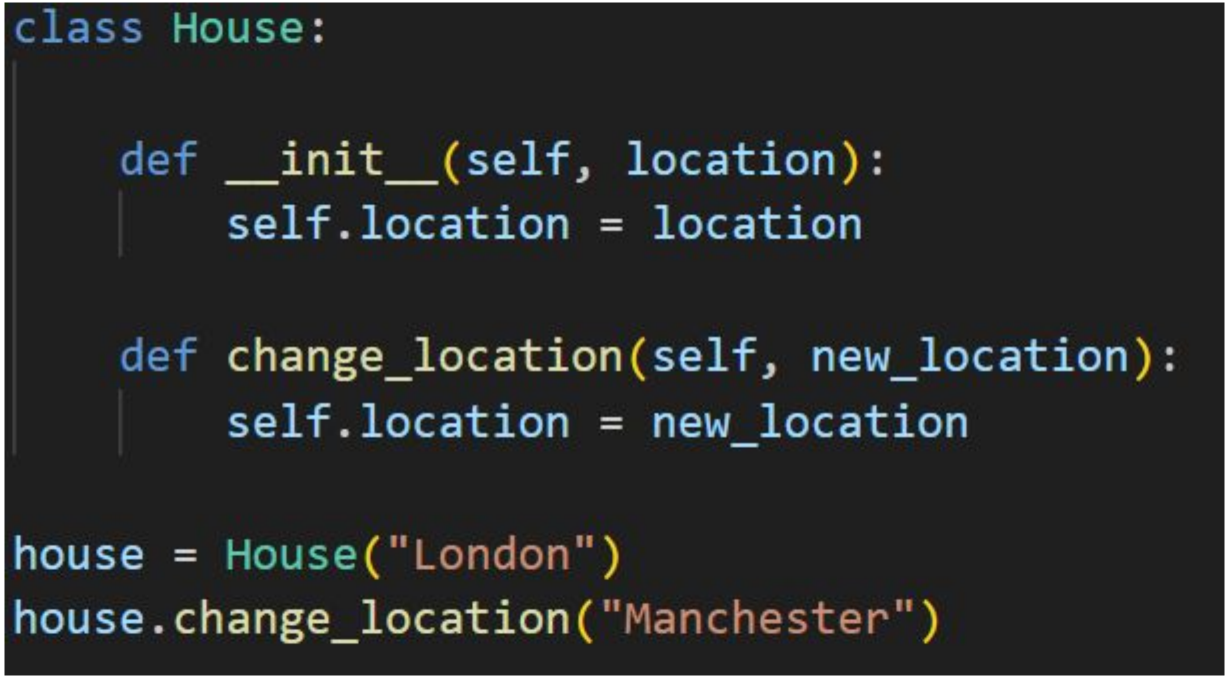
```
luke = Student("Luke Skywalker", 23, "Male")
```

Methods(Behaviours)

- Methods, define the actions or behaviors 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

Class Methods



```python
class House:

    def __init__(self, location):
        self.location = location

    def change_location(self, new_location):
        self.location = new_location

house = House("London")
house.change_location("Manchester")
```

CoGrammar

Questions around object oriented programming

Poll:

Assessment

Wrapping Up

Object Orientation in Programming

Provide flexibility when writing to files, allowing you to manipulate file content as needed.

Objects in Python

Determine how the file is accessed, whether existing data is retained or overwritten, and whether the file is created if it does not exist.

CoGrammar

Thank you for joining