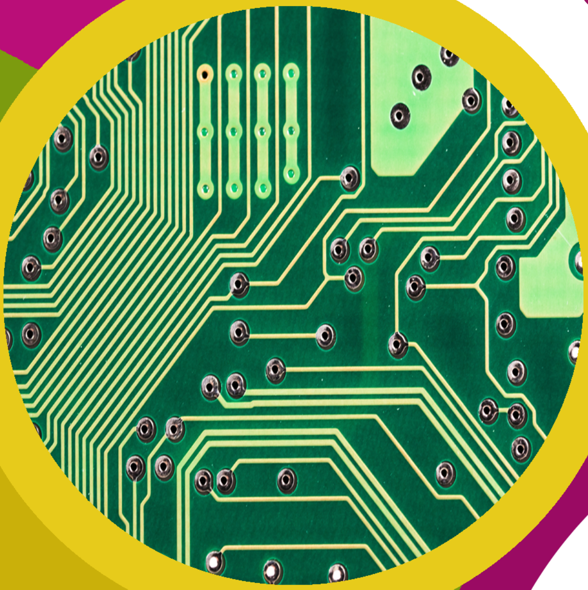
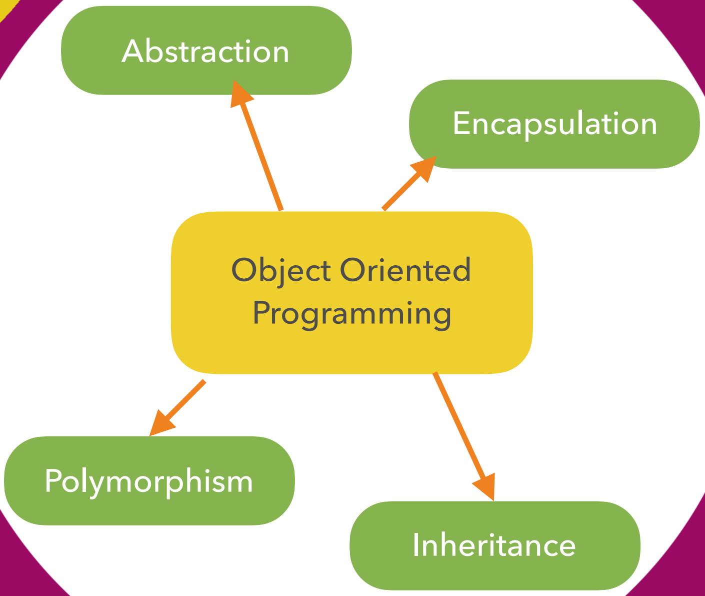
technocamps

Inspiring | Creative | Fun
Ysbrydoledig | Creadigol | Hwyl

Object Oriented GUI Session Plan





Workshop Schedule

Introduction - 10 minutes

Object Oriented Programming brief - 10 minutes

Classes, Objects and Methods - 1 hour 30 minutes

Adding a GUI - 20 minutes

Abstraction - 20 minutes

Encapsulation - 20 minutes

Inheritance - 20 minutes

Polymorphism - 20 minutes

Final Activity - 1 hour

Conclusion - 10 minutes

Post-Day Questionnaires - 10 minutes

Note: These are estimated times, these will vary between classes, schools etc. so times will need to be adjusted accordingly.

Total: 4 hours 50 minutes

Attendee Prerequisites

1. Experience of Python Programming.
2. Experience of programming GUIs in Python.

Learning Outcomes

1. Improved knowledge of classes, methods and object in Python
2. Knowledge of the object oriented programming concept abstraction
3. Knowledge of the object oriented programming concept encapsulation
4. Knowledge of the object oriented programming concept inheritance
5. Knowledge of the object oriented programming concept polymorphism

Preparation

1. Ensure all computers have Python 3 installed and ready to be used.
2. Print out Object Oriented GUI workbooks, one for each student attending workshop.
3. Each student will need a pencil for answering questions in the workbook.

Session Plan Key

In this session plan we use the following colours to differentiate the types of activities:

- **Yellow - Explain.** Teachers should explain the slide/example to the class.
- **Green - Discuss.** Teachers should start an open discussion with the class to get them to feedback some answers/ideas.
- **Purple - Activity.** Students are expected to complete an activity whether it be in their workbooks or on the computer, followed by a discussion of their solutions.
- **Green - Introduction/Conclusion.** The introduction/conclusion is also colour coded green. Teachers should hand out materials in the introduction and conclude the day and collect materials at the end.

Introduction

Begin with introductions, and a brief explanation of the Technocamps programme, before handing out pre-day questionnaires to be filled out by the students and teacher.

Activity: Python Review

Students are to write the outcomes of each piece of python in their workbooks. Ask students for their answers and compare. If there are discrepancies confirm results using python.

Explain

Object oriented programming makes use of classes, objects and methods. These classes and objects can share properties and methods. Object oriented programming has 4 key concepts:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Activity: Check previous understanding

Ask the students to write down what they think each concept means.

Activity: Collect Information

Each student has to collect the following information about their classmate(s):

- Name
- Age
- Number of Siblings
- Number of pets
- Teacher/Tutor's name

Each student should collect information for at least 2 classmates.

Explain that the information they have collected can be turned into a class and each student can have their own object.

Classes, Objects and Methods

Explain

Classes are like recipes. They are used to build objects.

To define a class in python we use the keyword class. Show an example class in python. Indicate the properties within that class.

Discuss

Students are to look back at the information they just collected.

- What would we call the class to store this information?
- What properties would the class have?
- What default values could be set, if any?

Activity: Writing Code

Students should write a class using python reflecting their decisions from the discussion.
A suitable class name would be Student. The class should have the properties: name, age, numberOfSiblings, numberOfPets, teacherName. With regards to default values if all of the class share the same teacher this would be a good default value. If all of the class are the same age this would be a good default value however make sure the class also consider that at some point this default value would not be correct. We should try to future proof our code.

Activity: Checking Understanding

Ask students to write their own description of a class in their workbooks.

Explain

An object is an instance of a class.

An object can either use the default values set by the class or it can set it's own values. Show an example of an object in python.

Activity: Writing Code

Students are to select a classmate they collected information about and create an object to hold that information.

- Which properties do we need to set ourselves?
- Which properties can use the default values, if any?

Activity: Checking Understanding

Ask students to write their own description of an object in their workbooks.

Explain

All classes have a function called __init__(), which is always executed when the class is being initiated. This __init__() function allows you to assign values to object properties, or complete other operations that are necessary to do when the object is created.

Classes, Objects and Methods

Discuss

- What could our __init__ method look like for this class?
- Do we need to set all of the values?

The __init__ method should have inputs for all of the values that we wish to set per object. For example if agreed that the teacher value would not change then there is no need for this to be in the __init__ method, however the other fields are unique to each person and should therefore be set.

Explain

Classes can also have other methods that the object can call. Think of these as instructions.

Discuss

If our class was a dog, then what can we tell the dog to do?

Activity: Writing Code

Think of 3 methods that we can use in our class? Write these into your class and test they work correctly.

Activity: Checking Understanding

Ask students to write their own description of a method in their workbook.

Adding the GUI

Discuss

Discuss the GUI code and what it does with the class. Encourage students to analyse what each line does.
The lambda function is an anonymous function not bound to an identifier. It is used in our GUI buttons to allow us to send specific data (parameters) to the method we are calling. If it was not defined as a Lambda function it would loop.
if __name__ == '__main__': ensures that the code executed is that in the main method. This is useful when we have multiple files but it good to future proof code.

Activity: Checking Understanding

Students should draw what they think the GUI will look like into their workbooks.

Activity: Adding the GUI

Students should add the GUI element from their workbook into their code and ensure all methods still work as expected. Once complete they should reflect on their drawing from the previous activity and see if it matches the actual outcome.

Discuss

Once the class have added the GUI into their program, discuss what improvements they think can be made.

- Do we know when a student has been added to the list? Have they all added that print statement?
- Can we enter an invalid date? What are the consequences of this?

Improving the GUI

Explain

Explain the importance of validation. Data should be accurate, up to date and it it's not it won't be useful. Briefly mention GDPR and the data protection act.

Discuss

Discuss some previous cases where companies have been fined for holding incorrect information.

- What are the consequences?
- What about the customer?
- Reputation?

Activity: Checking Understanding

Students should write the importance of validation in their workbooks.

Activity: Writing Code

Students should firstly add a condition to check if the name and age is present in the entry fields. If it is blank then the student should not be added.
Students should also copy the validation method from their workbooks and utilise it in the addStudent method.

Abstraction

Explain

Applying abstraction means that each object should only expose a high-level mechanism for using it.

Discuss

A coffee machine. It does a lot of stuff and makes quirky noises under the hood. But all you have to do is put in coffee and press a button.

A mobile phone. It has just a few buttons but lots of different outcomes.

Activity: Checking Understanding

Students should look back at what they initially thought abstraction meant and write a correct definition of it in their workbooks.
Students should also think of 2 more everyday objects that they see/use but don't know what happens behind the scenes. Write down how they interact with the object and the outcome.

Encapsulation

Explain

Encapsulation is achieved when each object keeps its state private, inside a class. Other objects don't have direct access to this state. Instead, they can only call a list of public functions - called methods. In python __ sets a variable to private.

```python
class Car():
    doors = 1
    colour = "N/A"
    __seats = 1

    def __init__(self,doors,colour):
        self.doors = doors
        self.colour = colour

    def start(self):
        print("Vroom, vroom")

    def printNumberSeats(self):
        print("This car has", self.__seats, "seats")

car1 = Car(4,"black")
car1.printNumberSeats()
```

The first example piece of code will print the number of seats which is set in the Car class.

In the second example we try to overwrite the number of seats by setting the object's variable __seats to 4. We do this using:
car1.__seats=4
Here the print statement still says the car has 1 seat. car1.__seats does not overwrite the variable. This is because the variable is private and can only be modified in the class itself or by calling public methods.

The final example shows a method has been written called "setNumberSeats" which takes in the object itself and the variable seats and sets the class variable __seats to be of that value. Demonstrate that having the object call this method works when trying to change the number of seats.

```python
class Car():
    doors = 1
    colour = "N/A"
    __seats = 1

    def __init__(self,doors,colour):
        self.doors = doors
        self.colour = colour

    def start(self):
        print("Vroom, vroom")

    def setNumberSeats(self,seats):
        self.__seats = seats

    def printNumberSeats(self):
        print("This car has", self.__seats, "seats")

car1 = Car(4,"black")
car1.setNumberSeats(4)
car1.printNumberSeats()
```

Inheritance

Explain

Objects are often very similar. They share common logic. But they're not entirely the same. So how do we reuse the common logic and extract the unique logic into a separate class? One way to achieve this is inheritance.
It means that you create a (child) class by deriving from another (parent) class. This way, we form a hierarchy. The child class reuses all fields and methods of the parent class (common part) and can implement its own (unique part).

Discuss

Person inheritance diagram:

- Person is the main parent class. Teacher and Student are the subclasses. They inherit everything that a person has.
- A Teacher also has a subject that they teach.
- A Student has Classes and Grades.
- The Teacher class is also the parent class of Public Teacher and Private Teacher.
- So a Public and Private Teacher inherit everything from the Teacher class, including the Person values.
- A Private Teacher also has Students and a Public Teacher has a School.

Explain

Example piece of code:

- The parent class Vehicle has now been created.
- Car is the subclass of Vehicle and inherits all of it's properties and methods.
- Here we show a Car object being created using the Vehicle __init__ method.
- The Car also uses the Vehicle's setNumberSeats method and printNumberSeats method.

Activity: Checking Understanding

In the workbooks students must try to match the subclass to the parent classes.

Polymorphism

Explain

Polymorphism gives a way to use a class exactly like its parent so there's no confusion with mixing types. But each child class keeps its own methods as they are.

This typically happens by defining a (parent) interface to be reused. It outlines a bunch of common methods. Then, each child class implements its own version of these methods.

Polymorphism is perhaps the most complex concept.

- Polymorphism means that different types respond to the same function.
- Polymorphism is very useful as it makes programming more intuitive and therefore easier.

Explain

Explain the shape diagram demonstrating polymorphism:

- All shapes can have their surface area calculated.
- All shapes can have their perimeter calculated.
- Each shape has a different implementation of each method. For example to calculate the surface area of a square it's length*width. This does not apply to a circle or a triangle therefore each specific shape needs it's own implementation.

Explain

A car and a bike are different types but they are both vehicles. Therefore we can place them in a list and call the method start().

Explain

Method overriding is another type of polymorphism. In the code we can see the method start() is defined for a vehicle and for a car and bike. We are overriding the parent class method. So when we create a vehicle the vehicle class method will be called, when we create a car the car method will be called and when we create a bike the bike method will be called.

Activity: Checking Understanding

Select two parent classes from the inheritance activity and think of some methods that would be defined in the parent (interface) class but only implemented in the child class.

Activity: Consolidate

The aim of this activity is to consolidate all of the student's learning from the session.
Students have to create a GUI Zoo program that holds the following information.

- 3 Ducks – Daffy, Donald, Daisy. Daffy is 5 years old, Donald is 3 years old and Daisy is 1. How many legs do ducks have? What sound do they make?
- 2 Giraffes – George and Gerald. George is 7 and Gerald is 10 years old. How many legs do giraffes have? What sound do they make?
- 1 Elephant – Nelly. Nelly is 12 years old. How many legs do elephants have? What sound do they make?
- Make a method that prints the following:

"Hello! My name is ____. I am a (type of animal). I have ___ legs. (Sound)"

Tips:

- Students should create 4 classes:Animal (parent class), Duck, Giraffe, Elephant.
- The animal class should have properties: name, age, legs, sound
- The duck class should set the legs to be 2.
- The giraffes should set the legs to be 4, as do the elephants.
- Each animal class should override the noise/sound method.
- For each animal an object of the correct type should be created.
- The print statement should be a method of the Animal class as all of the animals can use it.

Extension:

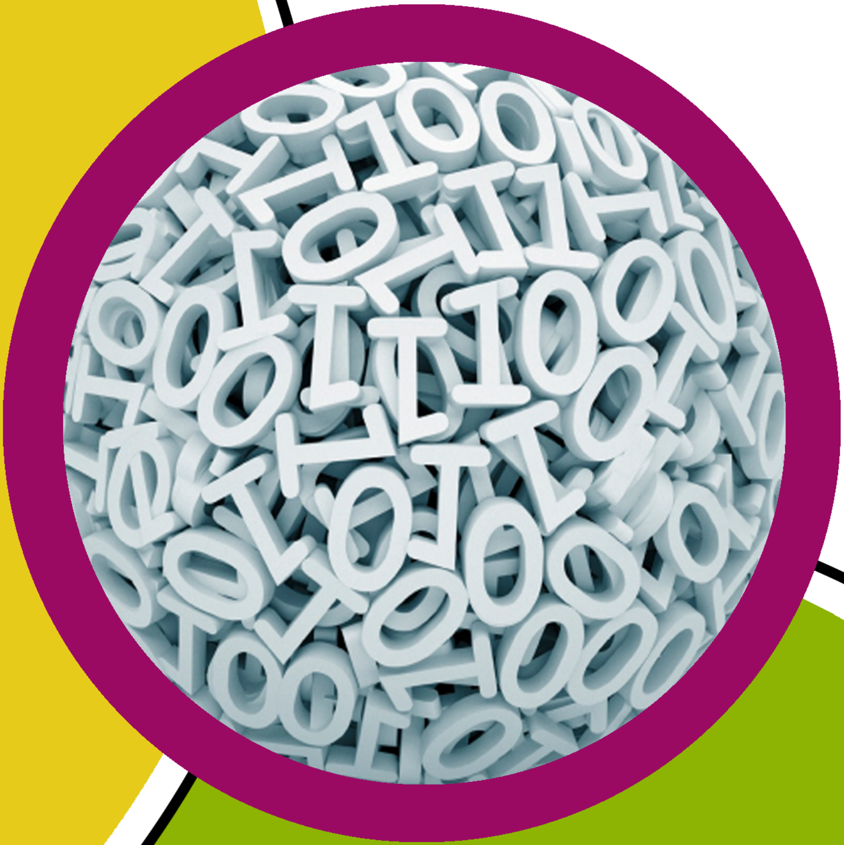
Students can add more types of animals or more methods/properties to the classes.
Get students to compare each others code and ask them to explain their code to each other in pairs. This helps them to think about the way they've written the code and also allows them to correct/help out one another.

Conclusion

End the class highlighting the main parts of object oriented programming.
The four main concepts are:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

In order to implement these we need to make use of classes, objects and methods.



technocamps

Inspiring | Creative | Fun

Ysbrydoledig | Creadigol | Hwyl

@Technocamps

Find us on Facebook