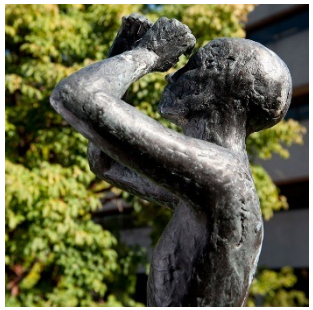


Lecture 1 – Introducing OOP

CS2513

Cathal Hoare

A TRADITION OF INDEPENDENT THINKING

UCC

University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Lecture Contents

Introductions

Object Oriented Programming (OOP) Concepts

OOP and Software Development

Who is Cathal

- Cathal Hoare
- cathal.hoare@ucc.ie
- or Message on Canvas
- Room: WGB G.67

What We Will Cover

- Further features of the Python Programming Language
  - Classes and Object Oriented Programming with Python, including:
    - OOP Design
    - Encapsulation
    - Inheritance
    - Polymorphism
  - Develop using a series of Python Libraries to develop Graphical User Interfaces (tkinter), PyGame, System libraries, regular expressions, etc.
  - Generators and Special Methods

When and Where

- Lectures:
  - Tuesday 12-1 in G.01 WGB
  - Thursday 2-3 in G.01 WGB
- Labs:
  - To be announced
- Notes, assignments etc: Canvas CS2513 Module

Assessment

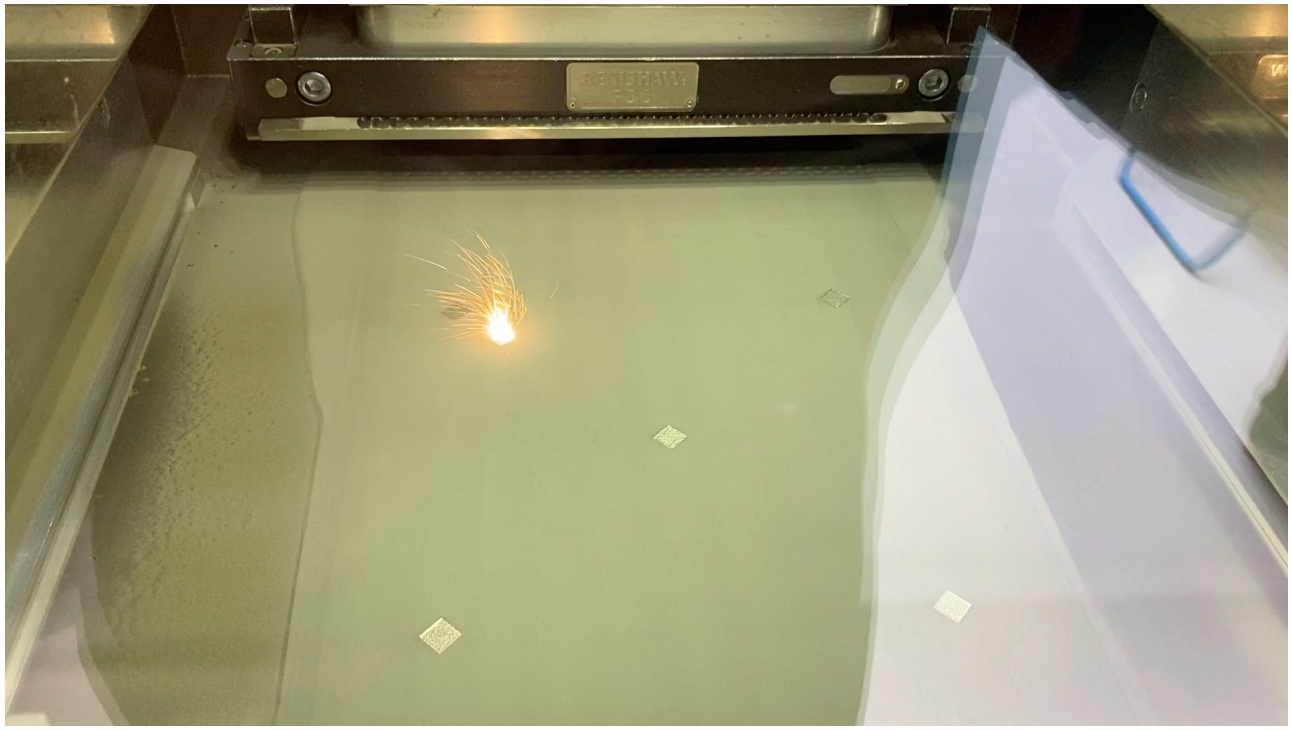
- Formal Exam – Winter Exams - worth 80% of total mark.
- CA – 20% - 2 Assignments each worth 5% of the total mark and a class test worth 10% (late October).
- A sample exam paper and solution will be provided for both exam and test.
- There will be tutorials as required (in lieu of labs) and also some worksheets to prepare you for the assignments. While not assessed, you are expected to complete them. Feedback will be given.

Plagiarism

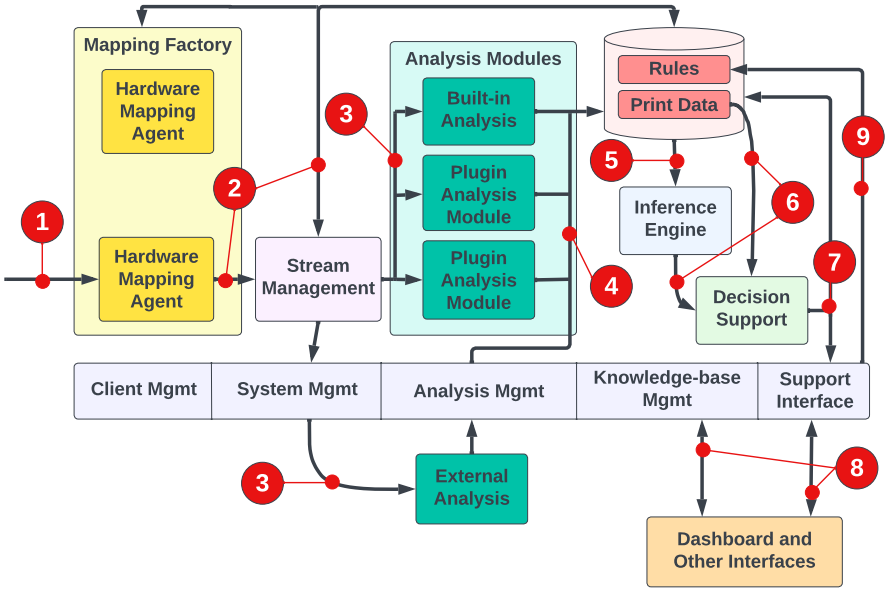
1. Plagiarism is presenting someone else's work as your own. It is a violation of UCC Policy and there are strict and severe penalties.
2. You must read and comply with the UCC Policy on Plagiarism www.ucc.ie/en/exams/procedures-regulations/
3. The Policy applies to *all* work submitted, including software.
4. You can expect that your work will be checked for evidence of plagiarism or collusion.
5. In some circumstances it may be acceptable to reuse a small amount of work by others, but *only* if you provide explicit acknowledgement and justification.
6. If in doubt ask your module lecturer *prior* to submission. Better safe than sorry!

Once code gets complicated…

Once code gets complicated…

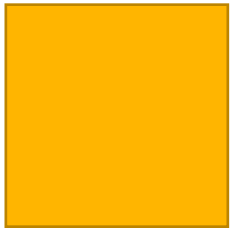


A Complicated Architecture



1. Data from hardware
2. Hardware specific parser
3. Data Stream Architecture
4. Multiple Analysis Techniques
5. Detected Anomalies
6. Reasoning Engine
7. Operator Recommendations
8. Mobile Dashboards
9. Learning Feedback Loop

Once code gets complicated…



- How do we leverage others' (reuse, extend and otherwise interact) code?
- How do we package code so that others can reuse it?
- How do we produce code so that it is easily maintained?
- How to create libraries of code for yourself and others and ensure it is working correctly and robustly?
- How do we organise large projects and allow several teammates to work together?

Intro to OOP

- One way (a popular way) is to use Object Oriented
- Programming (OOP) Object Oriented Programming is a programming paradigm or style of programming.
- It is often characterised by three concepts:

  Encapsulation, Inheritance and Composition

State

A Light has State: on or off

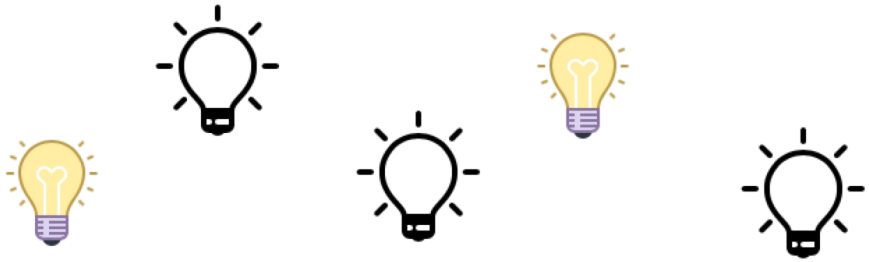
State and Actions/Behaviours

A Light has State: on or off

And we can change its state - by switching it on or off with a switch

We group state and actions/behaviours in a class. This is called encapsulation.

Classes and Objects



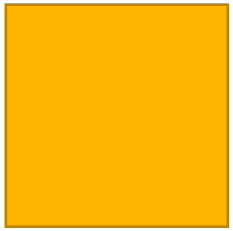
- There are many lights - we represent the light with a class. The attributes that describe the light and the actions that can be carried out on the light are encoded in the class.

- One class can describe each light. Each light object is an instance of the class.

Encapsulation

Remember Friends?

Using Encapsulation: Safer Code - Fewer Errors



Ideally, we change the light's state with a method

The developer who develops the class will understand what is needed to change the state of the light and can encode this in the method

Other developers will not know the requirements of the light as well as they are developing their own code. The light class is an opaque box to them.

Inheritance

A regular light

An led light

A regular light has already been implemented

But now we have a new type of light that requires some extra attributes and methods to model it

Inheritance

A regular light

An led light

We could start from scratch, or we could reuse the functionality of the original light - using what we can as is, overriding any existing methods that need to be specialised, and add any new attributes and methods

This is called Inheritance

Inheritance

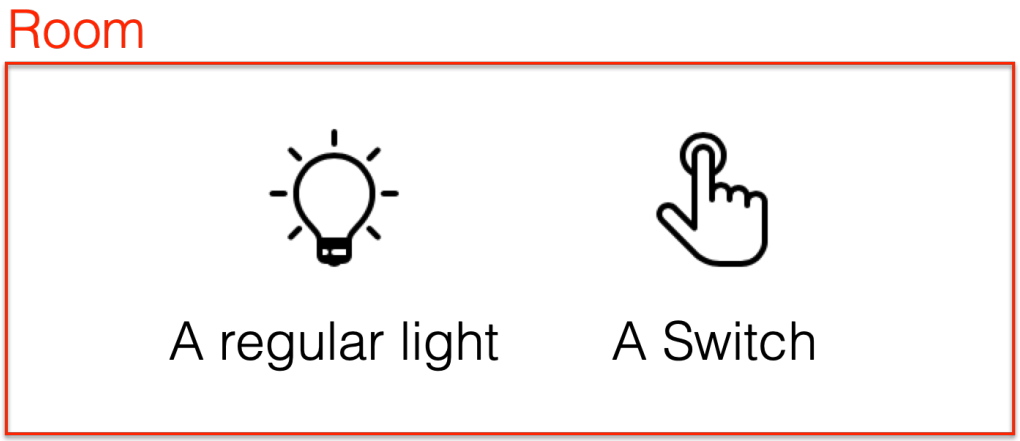
A regular light

An led light

We reduce the amount of code we need to write (write once)

We fix any bugs in the original class just once

Composition



We can model complex objects

Say we have a room, with a light, and a light switch

Composition

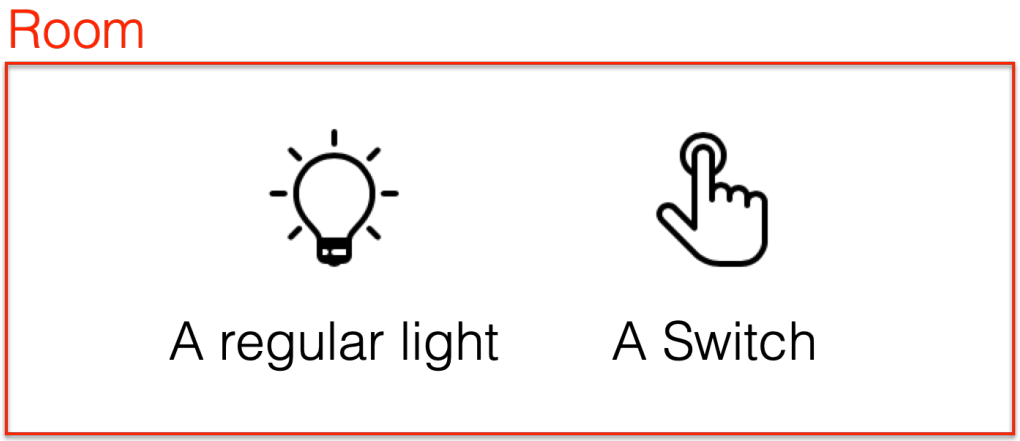
Room

A regular light

A Switch

Rather than write a single class, we write a class, Room, that is composed of the methods and actions required for a room, but reuses the classes we have already written.

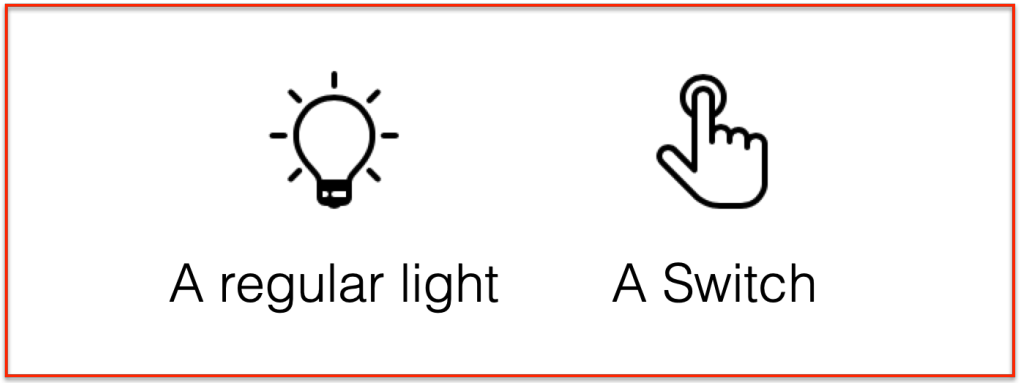
Composition



This is useful in the same was inheritance was - less code written, fewer bugs, and any bugs fixed just once.

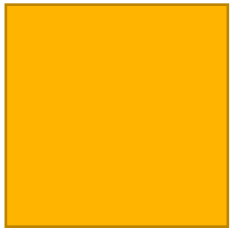
Composition

Room



Rather than write a single class, we write a class, Room, that is composed of the methods and actions required for a room, but reuses the classes we have already written.

Python vs. Pure OO Languages



- Languages such as Java (you will learn this later in the year) are pure OO languages. They enforce the principles of OO strictly.
- Python doesn't require OO to be used.
- Python also emphasises freedom to use the language in any way. Developers implement encapsulation based on trust.
- Python is more OO in some ways than Java, allowing the user of Multiple Inheritance and other features.

Advantages of OOP

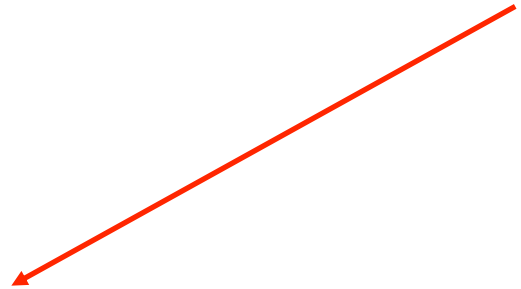
- Admits parallel code development for groups of developers
- Helps us write cleaner code that has fewer errors
  - And when errors occur, our code is easier to fix
- Helps us to reuse code
- Allows us to use related code organisation and design techniques

Advantages from Industry Perspective

- Consider the costs involved in developing a couple software project:
  - Building and Fittings
  - Equipment
  - Utilities
  - Developers

Advantages from Industry Perspective

- Consider the costs involved in developing a couple software project:
  - Building and Fittings
  - Equipment
  - Utilities
  - Developers



100 Developers at 30k to 70k per annum is an expensive resource

Advantages from Industry Perspective

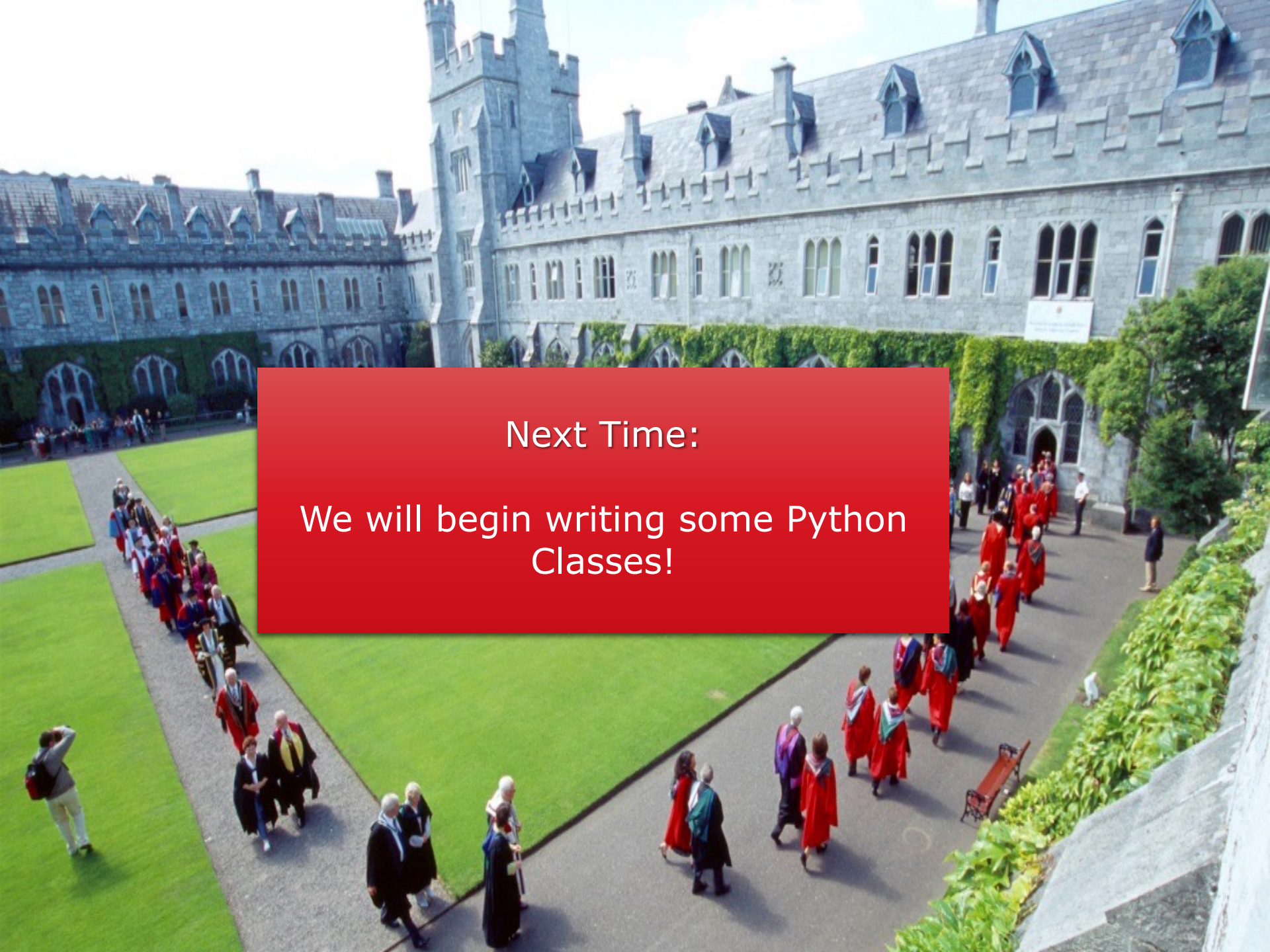
- Consider the costs involved in developing a couple software project:
  - Building and Fittings
  - Equipment
  - Utilities
  - Developers

Want to:

- maximise the time spent developing new code in parallel
- Reduce time spent fixing bugs (and the number of bugs!)

Disadvantages of OOP

- As an individual, it can be difficult to see the advantages of OOP
- There is more code with OOP, and OOP Design takes more effort
  - Not ideal for prototyping or developing a one off project
- But no one is saying every piece of code that we write in Python has to be object oriented


Next Time:
We will begin writing some Python Classes!