technocamps

Llywodraeth Cymru
Welsh Government

Prifysgol Abertawe
Swansea University

Cardiff University
Prifysgol Caerdydd

Prifysgol Bangor University

Cardiff Metropolitan University | Prifysgol Metropolitan Caerdydd

University of South Wales
Prifysgol De Cymru

Cyngor Cyllido Addysg Uwch Cymru
Higher Education Funding Council for Wales
hefcw

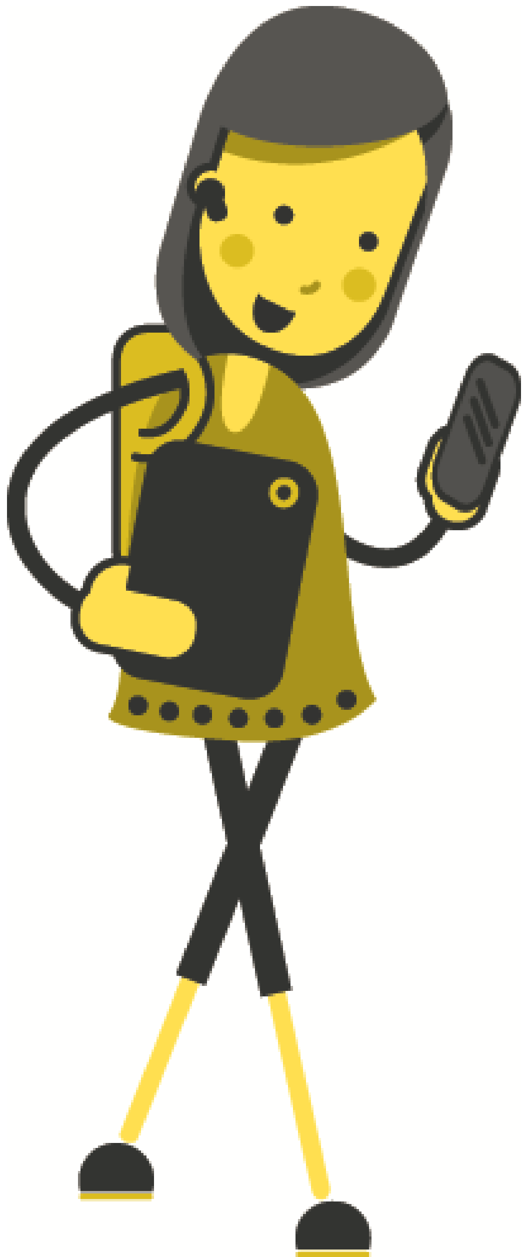
Prifysgol Cymru Y Drindod Dewi Sant
University of Wales Trinity Saint David

Prifysgol Aberystwyth University

Prifysgol Glyndŵr Wrecsam | Wrexham Glyndŵr University

institute of CODING in wales
technocamps

Text Adventure with OOP



What is OOP?

Look around the room… what do you see?

OBJECTS!

OOP stands for Object Oriented Programming.

Today we're going to learn how to program in Python using the ideas of OOP – basically we're going to think of our program as if each element was a real physical object.

What is OOP not?

Procedural programming

- The flow of **data** between functions.
- Very **hierarchical**, often one big function calling lots of others.
- Can result in **tight coupling** (undesirable dependencies) and changes to code tend to ‘ripple out’ to other code.

What is OOP?

Object Oriented Programming

- A network of cooperating objects communicating by a message system.
- Objects are peers no one is really in charge.
- Looking at an object from outside you should have no idea how it is implemented. You should be able to replace entire implementation without effecting other objects.

How is OOP?

An **object** is defined by **what it can do**, not by how it does it.

- Practically this means that it is defined by the messages it can receive and send.
- The **methods** that handle these messages make up the only interface the object has with the rest of the system.

Most objects will **require some data** in order to implement their capabilities.

How is OOP?

Object oriented programming is, as the name suggests, all about **objects**.

Classes are **blueprints** that can be used to create objects. A class defines:

- The **data (attributes)** an object will store.
- The **behaviour** an object will be capable of.
- How other objects can interact with it.

To create an Object, we must specify the blueprint (Class) for it!

Why is OOP?

Let’s say we have the class **Person**

What **attributes** does a Person have?

A *person* in real life has:

- A name
- An age
- A home address
- A list of family members
- *… And a LOT more …*

Why is OOP?

Let's say we have the class **Person**

What **attributes** does a Person have?

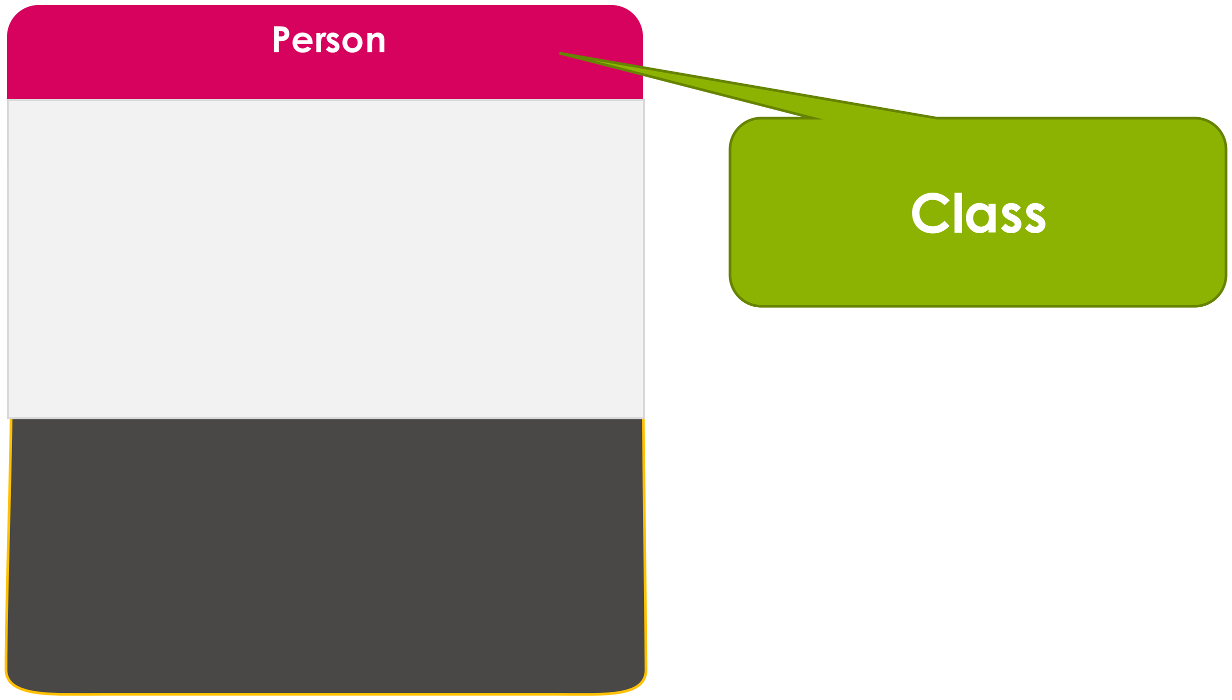
A *person* in real life has:

- **A name**
- **An age**
- **A home address**
- A list of family members
- … *And a LOT more* …

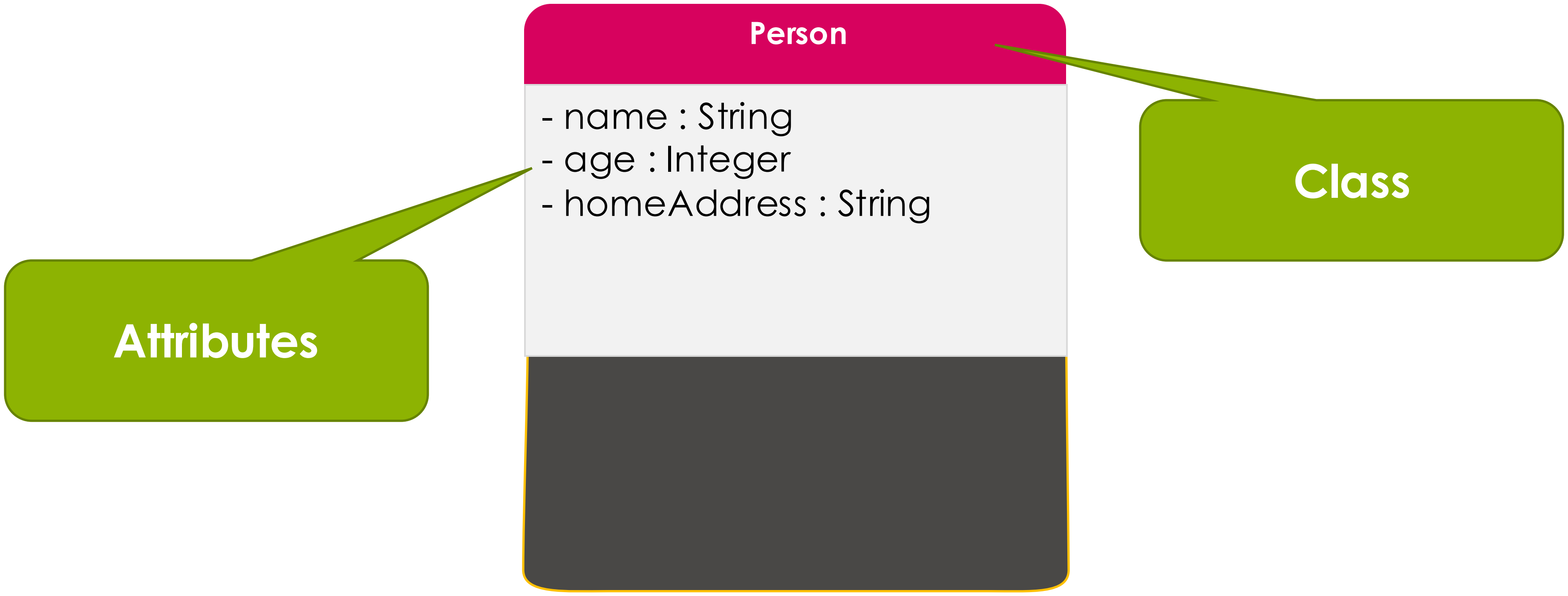
In my program, I only care about the **name**, **age**, and **home address** of the Person.

- The **name** and **home address** are String datatypes
- The age is an **Integer** datatype

Why is OOP?



Why is OOP?


Person
- name : String
- age : Integer
- homeAddress : String
Class
Attributes

Why is OOP?

What **behaviours** can a person have?

A person can:

- Speak
- *Give us their name*
- *Give us their age*
- Walk
- *Give us their address*
- Run
- Sleep
- … *Again, LOTS more ...*

Why is OOP?

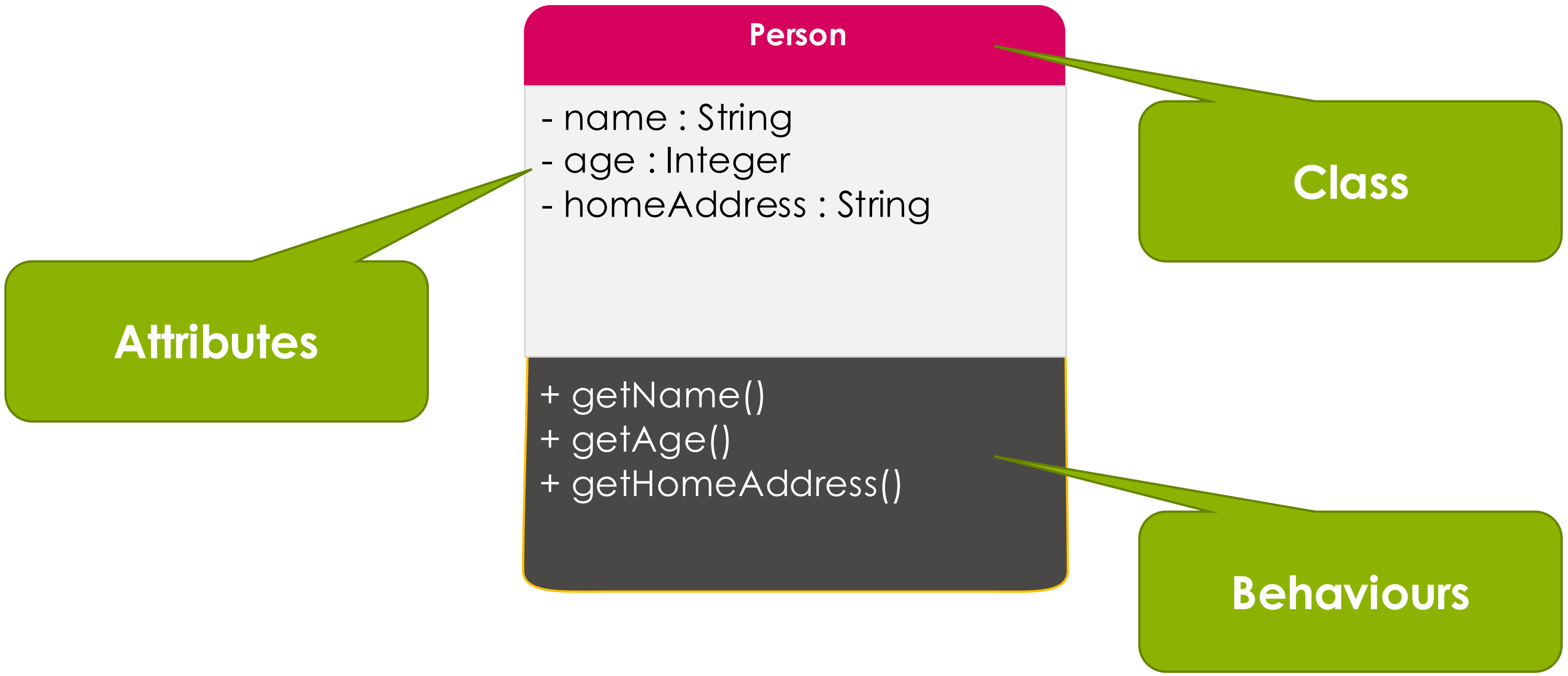
What **behaviours** can a person have?

A person can:

- Speak
- ***Give us their name***
- ***Give us their age***
- Walk
- ***Give us their address***
- Run
- Sleep
- *… Again, LOTS more …*

My program is only concerned with the bold **behaviours**

Why is OOP?


Person
- name : String
- age : Integer
- homeAddress : String
+ getName()
+ getAge()
+ getHomeAddress()
Class
Attributes
Behaviours

Let’s Start our Text Adventure!

Think of the style you’d like your text adventure game to be; should it be an adventure, a horror, educational?

This game is entirely your own, we’re just going to guide you on making it!

Where is the game set?

Who are the characters?

What is the goal?

Making our world!

When making a game it's a good idea to keep notes of your game before it becomes too big to keep track of.

As we have no graphical element (at least at this stage) for our text adventure, it is a good idea to make ourselves a map.

Using notepad, create a map that looks something like this:

```
y | x    0            1            2
0                 Reception



1     ITClass      Hallway      DTClass



2                 Cafeteria
```

Rooms

I want to be able to have multiple **rooms** that I can visit within my game.

I will therefore need to create a **Room object** that defines each room in my game - i.e. the information and methods belonging to each room.

If I want to be able to create **Room objects**. What will I need to create first?

Rooms

I want to be able to have multiple **rooms** that I can visit within my game.

I will therefore need to create a **Room object** that defines each room in my game - i.e. the information and methods belonging to each room.

If I want to be able to create **Room objects**. What will I need to create first?

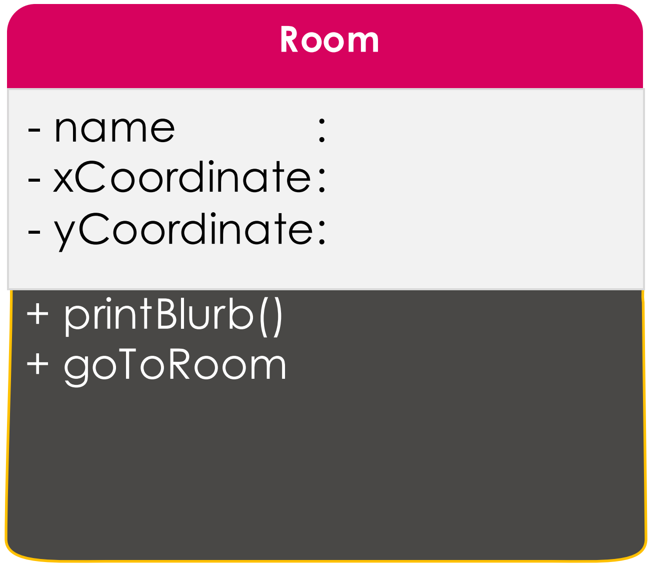
- A **Room Class**!

Your task:

- Design a Class to model a Room!

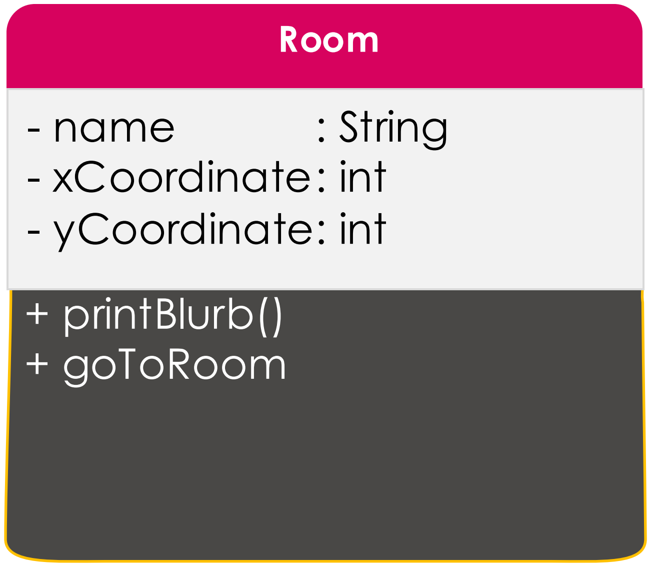
Room Class

Maybe something like this:



Room Class

Maybe something like this:



Got a different idea?

- There are a few variations here!

Room Objects

Okay so we now have a Room class that can be used to create Room objects for us!

But now we're going to need something that can hold onto all our rooms for us.

What could we make to hold our Room objects (**Hint:** remember we're using object-oriented programming)?

Room Objects

Okay so we now have a Room class that can be used to create Room objects for us!

But now we're going to need something that can hold onto all our rooms for us.

What could we make to hold our Room objects (**Hint:** remember we're using object-oriented programming)?

- A **Map** class!

Adding Rooms

Now we want to add our Room objects, as defined by those we made in our mini maps.

Ideally, when adding stuff into our game it should be stored separately to our code; this means it can be easily modified without having to modify the game code.

We will use a csv file to accomplish this (a csv file is just a text file with **comma separated values**). It should look something like this:

technocamps

Adding Rooms

We can make our **csv** files using Notepad or a similar **basic** text editor.

Remember, your commas are separating the values, so we cannot use commas in our blurbs.

It should look something like this:

```
RoomName, xCoordinate, yCoordinate, Blurb

Reception,1,0,This seems to be the schools' reception. Broken glass and paper covers the floor.

Hallway,1,1,You enter the main hallway. It is pitch black. There are no windows at all. Classro
```

Importing Rooms

Now that we have our rooms described, we need to import these definitions into our program and start creating Room objects out of them!

We will create a new method in the **Map** class to handle this.

Type Casting

We want to be able to use the data from the room file for specific purposes, but currently everything we read from the file is a String.

What data do we need to change?

Type Casting

We want to be able to use the data from the room file for specific purposes, but currently everything we read from the file is a String.

What data do we need to change?

Our coordinates need to be of type int!

We will need a new method in our Room class to type cast our imported data.

Get A New Room

We already have a method in our Room class that works out the coordinates for each new Room object.

Now we need a method for our Map class that searches through the list of rooms to find a matching object!

Making our Game

Okay, so we have some essential classes set up!

We will need more of these as our game grows in complexity – remember this is OOP and everything in the game should be expressed as some sort of object.

Making our Game

Okay, so we have some essential classes set up!

We will need more of these as our game grows in complexity – remember this is OOP and everything in the game should be expressed as some sort of object.

However, for now, we can begin tying this all together by creating a class that will run the program.

Making our Game

Okay, so we have some essential classes set up!

We will need more of these as our game grows in complexity – remember this is OOP and everything in the game should be expressed as some sort of object.

However, for now, we can begin tying this all together by creating a class that will run the program.

This is often called Main, but we will call our main class **Game**, which will create a **Game object** from which the game is run!

Starting the Game

As we are performing object orientation, we will need to initialise an object of the Game class before running our program.

Again, with OOP this should be defined as a method.

The only thing outside our Game Class should be the creation of the Game Class object.

Moving Rooms

We now have a Game class and a partial method that runs the main game loop!

However, we're going to need to interface with the get Room methods we already created.

The Game object will need to use these methods to check if it is possible for the user to move rooms in the specified direction, and either:

1. Move rooms (i.e. change Room object)
2. Tell the user they cannot move

technocamps

Llywodraeth Cymru
Welsh Government

Prifysgol Abertawe
Swansea University

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

PRIFYSGOL BANGOR UNIVERSITY

Cardiff Metropolitan University | Prifysgol Metropolitan Caerdydd

University of South Wales
Prifysgol De Cymru

Cyngor Cyllido Addysg Uwch Cymru
Higher Education Funding Council for Wales
hefcw

Prifysgol Cymru Y Drindod Dewi Sant
University of Wales Trinity Saint David

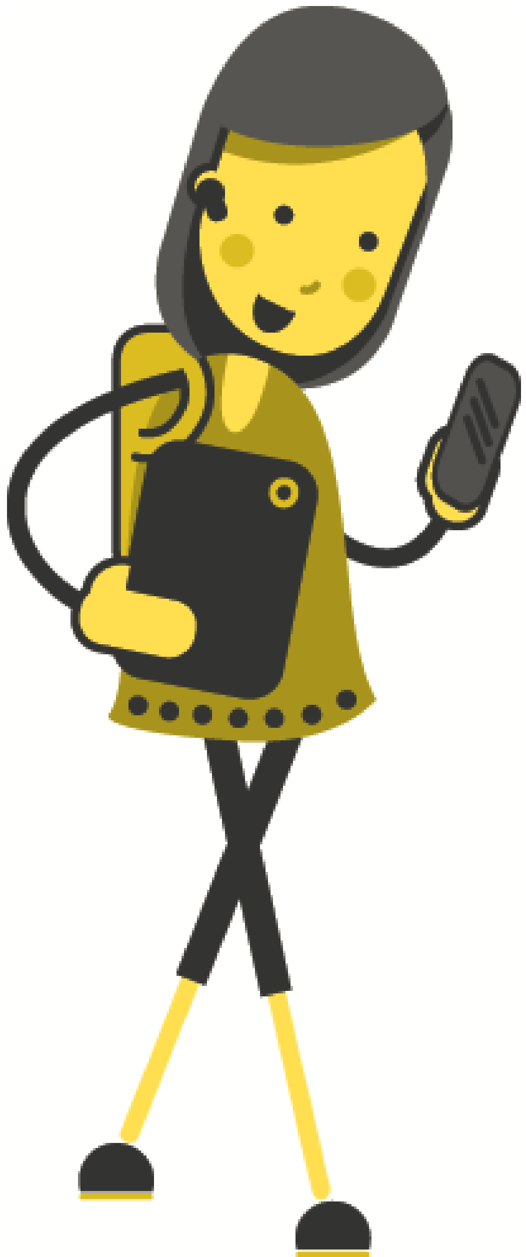
PRIFYSGOL ABERYSTWYTH UNIVERSITY

PRIFYSGOL Glyndŵr Wrecsam | Wrexham Glyndŵr UNIVERSITY

institute of CODING in wales
technocamps

technocamps

Text Adventure with OOP



OOP Recap

Object oriented programming is, as the name suggests, all about **objects**.

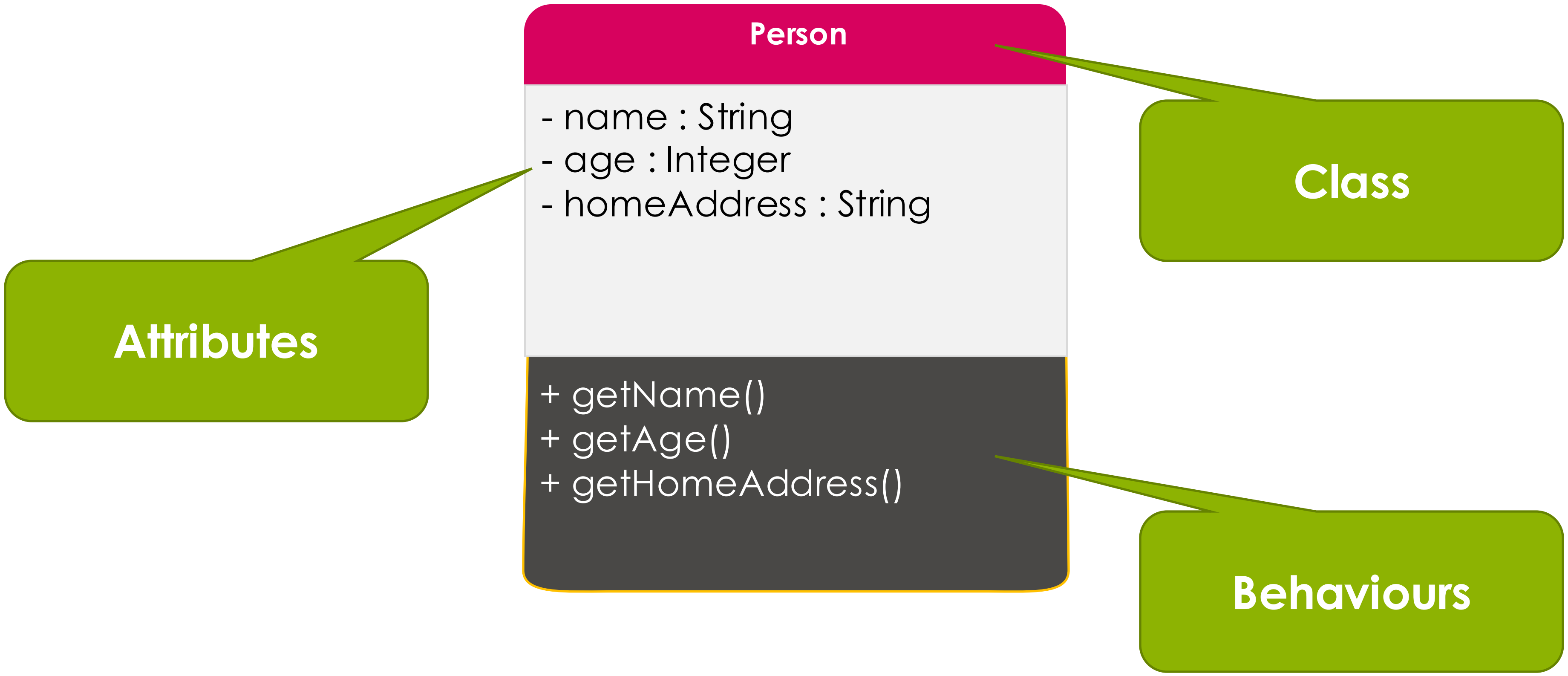
Classes are **blueprints** that can be used to create objects. A class defines:

- The **data (attributes)** an object will store.
- The **behaviour** an object will be capable of.
- How other objects can interact with it.

To create an Object, we must specify the blueprint (Class) for it!

technocamps

OOP Recap



Inheritence

We regularly encounter Objects which are **specific types** of some abstract concept.

In day-to-day life, we often use the abstract concept to refer to things:

- *"Where are the **fruits**?"*
- *"Heavy **vehicles** are harder to drive"*
- *"Turn on the **heater**"*
- *"Our shelter houses wounded **animals**"*
- *"I will visit the **doctor**"*

Inheritence

Each of these concepts on its own is too abstract to think about...

- But it is useful to group things with common **attributes** and **behaviours** together!

We can think of specific types of each of them

For example, **Animal**:

- Cat
- Dog
- Bird

Can you think of other examples?

The '*is - a*' Relationship

When we have a specific type of a more general concept, we use the ***is-a*** relationship between the two:

- "A dog ***is-a*** Animal"
- "A car ***is-a*** Vehicle"
- …

It's good to get in the habit of using the ***is-a*** relation!

What About Classes?

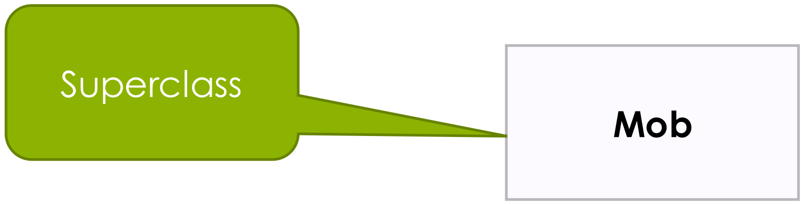
Inheritance is a core component between classes!

- The more general class is the **superclass**
- Specific types of this general class are **subclasses**

What About Classes?

Inheritance is a core component between classes!

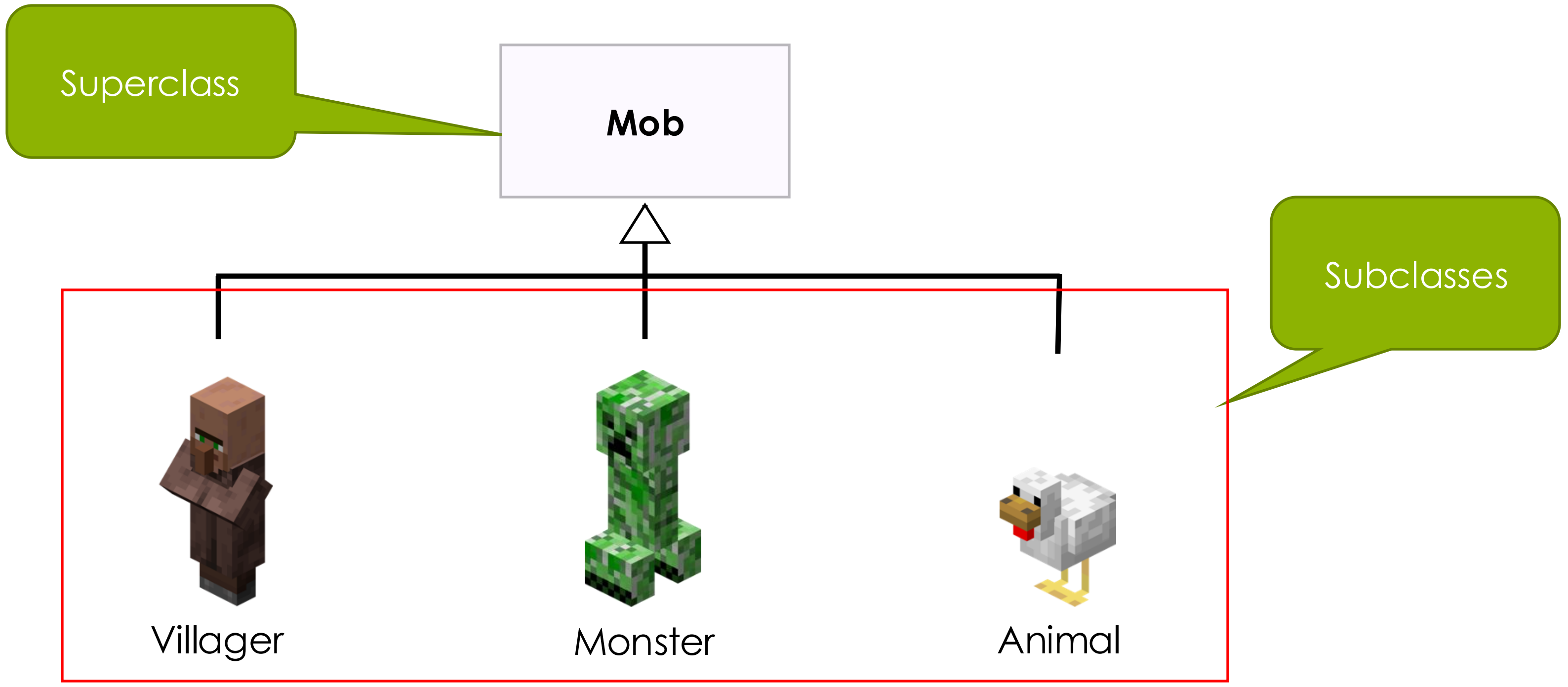
- The more general class is the **superclass**
- Specific types of this general class are **subclasses**



What About Classes?

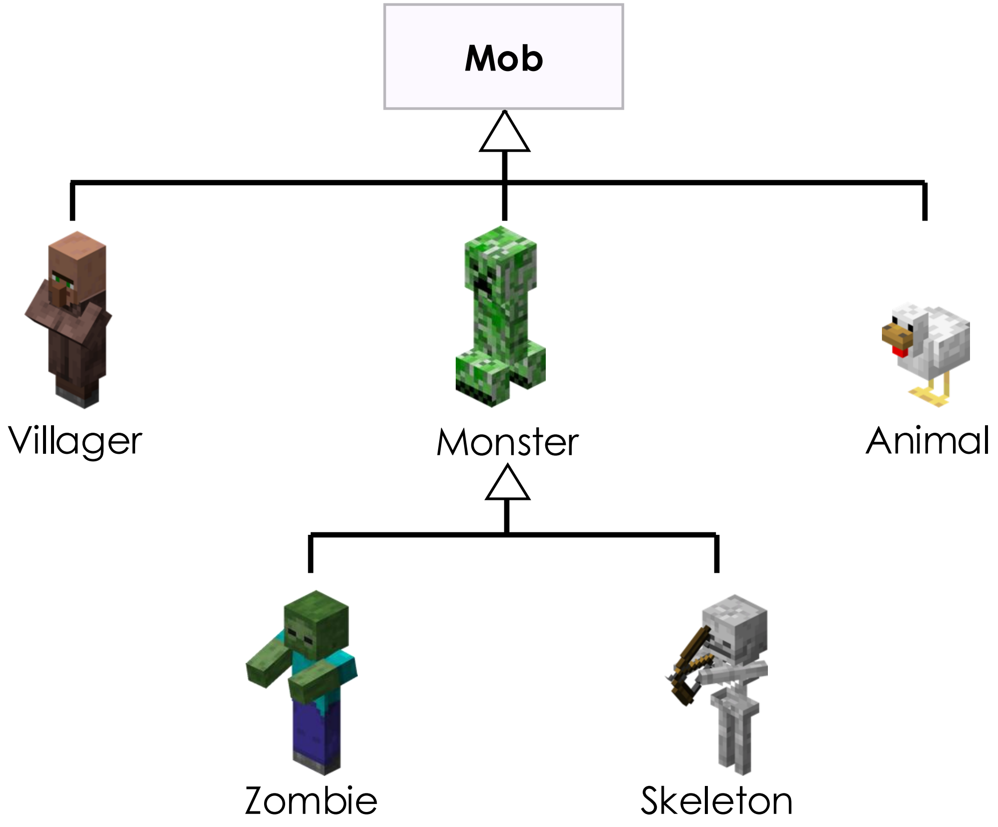
Inheritance is a core component between classes!

- The more general class is the **superclass**
- Specific types of this general class are **subclasses**



Fun Fact!

Inheritance is usually multi-level! Let's revisit the previous hierarchy:

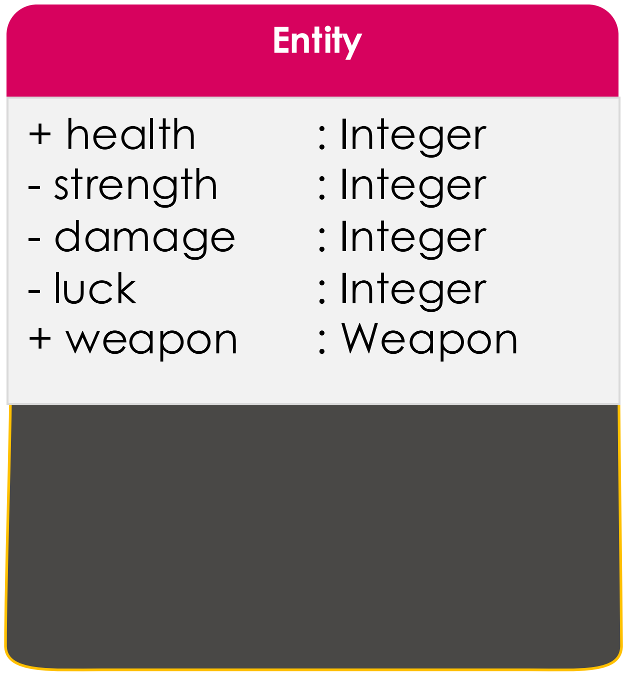


Unified Modelling Language

We need a standardised way to talk about Classes

The Unified Modelling Language (**UML**) can do that!

- A very big, complicated language…
- But don't worry we'll only need to use a fraction of it!



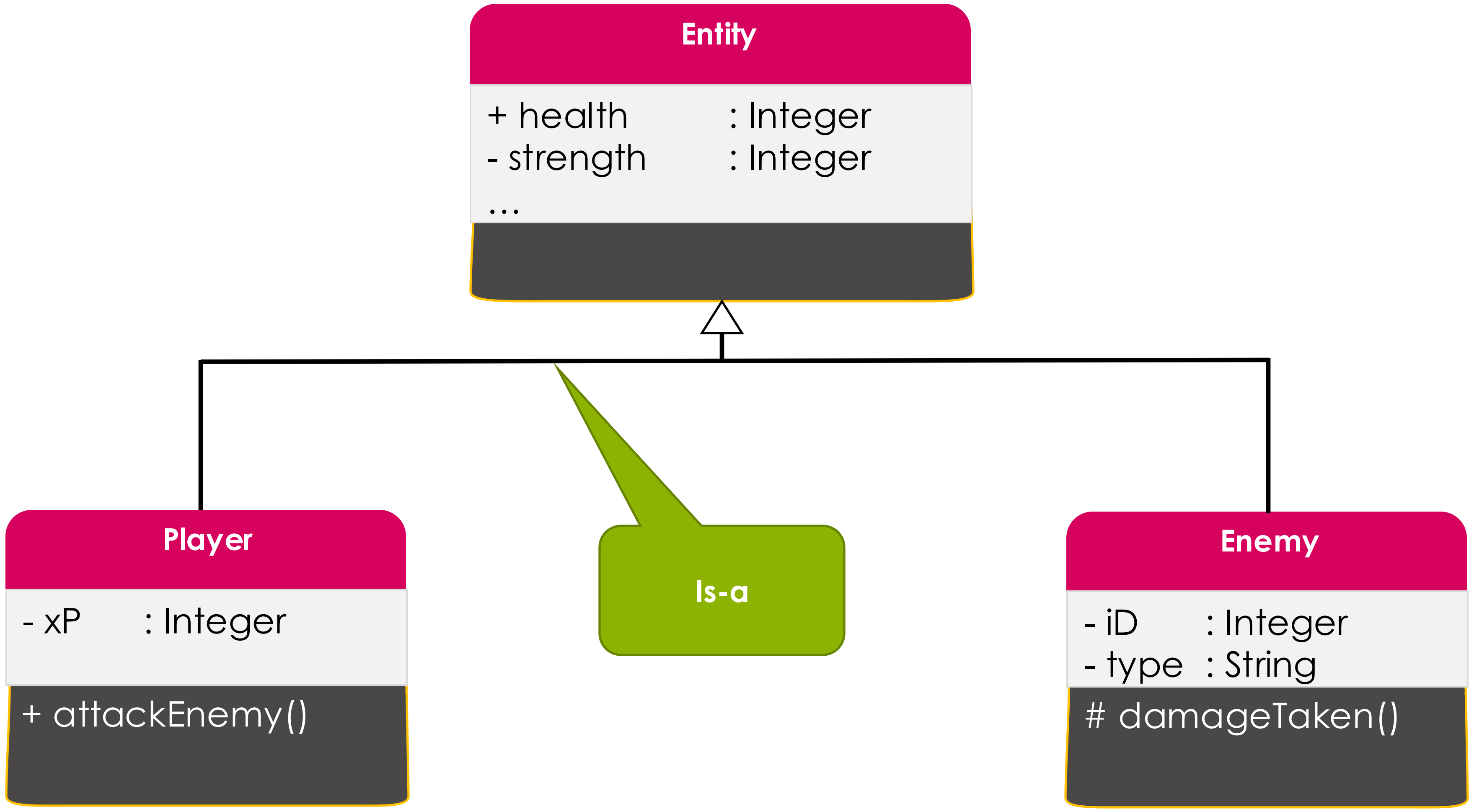
Entity

+ health	: Integer
- strength	: Integer
- damage	: Integer
- luck	: Integer
+ weapon	: Weapon

__**Class Diagrams**__:

- Show Classes pictorially
- … We've used them already!
- There's just a bit more to them…

Visualising Relationships *ft. is-a*



The super() Method

When programming inheritance we generally want our subclasses to inherit all attributes (and possibly methods) from the superclass.

i.e. All **Entities** should have the attribute **Health**

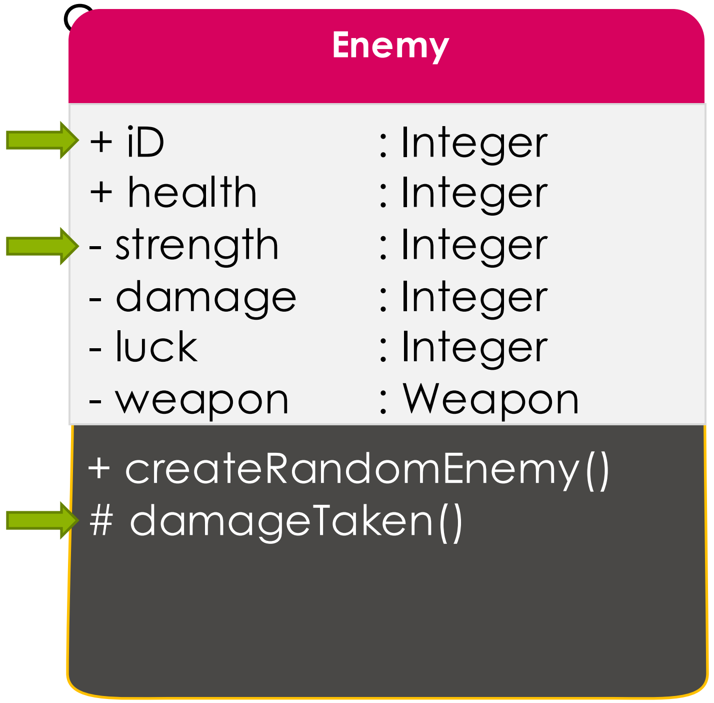
- this may well have different default values for different subclasses, but all entities have the attribute
- after all – that's why we made it an attribute of entities!

To inherit both attributes and methods in Python we can use the super() method.

What’s up with the +/- symbols?

In a Class we must select which information is visible to the world

- And we may choose to limit, or restrict it completely!



```
Enemy
+ iD          : Integer
+ health      : Integer
- strength    : Integer
- damage      : Integer
- luck        : Integer
- weapon      : Weapon
+ createRandomEnemy()
# damageTaken()
```

Symbol	Meaning	Who can see it?
+	public	Everyone!
#	protected	The class & its subclasses
-	private	The class itself only

How do we do this in Python?

Python is good at emulating both scripting and object-oriented languages!

Unfortunately, in Python these are only conventions and they have no impact on the code itself (i.e. all variables are public and there is no such thing as private or protected)

Symbol	Meaning	Who can see it?	Naming Python Variables & Methods
+	public	Everyone!	variableName
#	protected	The class & its subclasses	_variableName
-	private	The class itself only	__variableName

Create Enemies

There's a few ways we could create enemies in our game:

- We could load them in from a file like we did with our rooms (this would be a good method for bosses with specific stats)
- We could generate them randomly

For now, we will generate them randomly, as we've already looked at reading from a file.

Printing Enemies

Now we have enemies in each room, but the game never tells us that – so the user would have no idea.

We should print out the enemies in each room as we enter!

Attack Enemies

Finally, we'll begin adding combat to our game!

We'll add a method to the **Player** to attack an **Enemy**, and a method to the **Enemy** to calculate the damage done by the **Player**.

We will then need to add this option to our user decisions.

technocamps

Continue your game!

We have now looked at many methods for creating a game:

Continue to develop your game with these concepts!

- Finish the combat system to allow enemies to attack.
- Add an Items class with a Weapons subclass (this could change the damage dealt by entities)
- Read your items from files - a different file for each Item subclass would allow you to import unique attributes (i.e. value, weight, damage…)
- And whatever else you can imagine!