Python Intermediate

Introduction to OOP

{cdenation}®

Learning Objectives

- To learn about OOP
- Understand fundamental principles of OOP
- To code a simple class and understand instantiation

{codenation}

Coding is all about data.

The data is information.

The properties are features of the data.

The methods are things we can do to the data.

```
1 print("Hello World!")
```

The string itself is the data.

A property of it is its length.

A method is .upper() – which transforms the characters.

```
1 print("Hello World!")
```

My string is an object.

Objects are collections of data and methods.

They are built from a Class.

What does this mean?

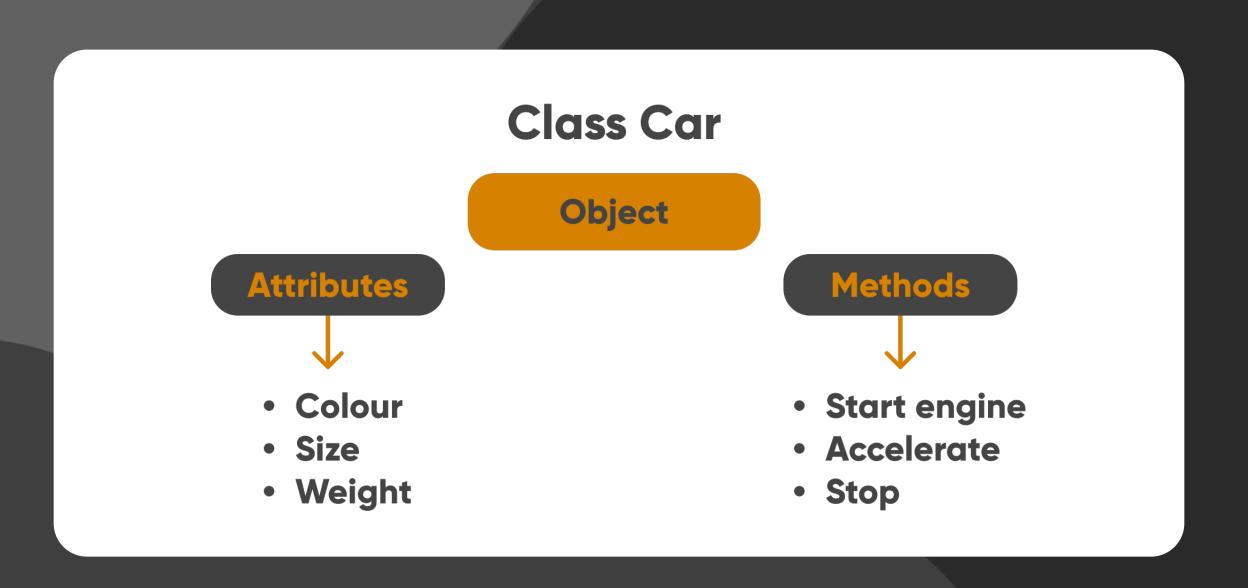
Think about how many strings you've used since day 1.

What are the similarities between them? What do they all share?

- All strings have a length.
- All strings can use string methods .upper(), .lower()
 etc.
- All strings can be sliced.
- All strings can be concatenated.
- All strings are indexed.

All strings inherit these things when they are created from their Class.

Every string is an object, instantiated from the class String.



Classes are like blueprints.
They outline a structure for an object.

They make it easy for us to make new objects to work with – especially lots of similar objects.

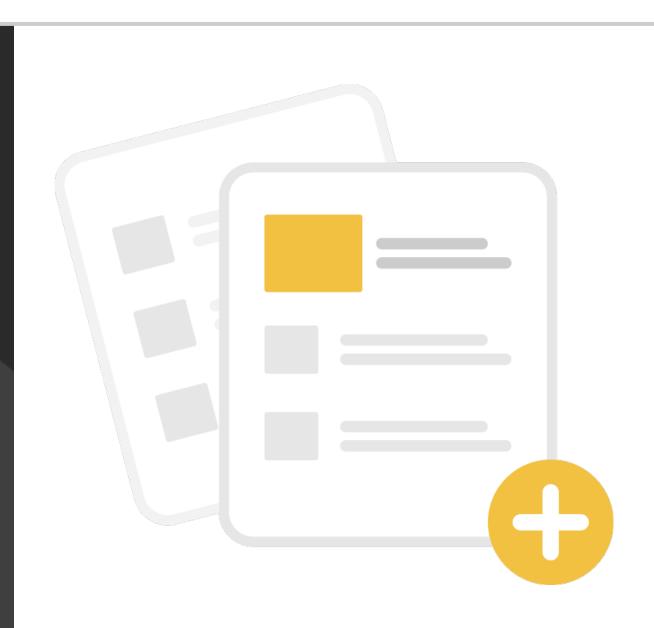
From my car Class, I can build lots of cars.
They can all start, accelerate, and stop.
They all have a colour, a size, and a weight.

Everything in Python is an object. string objects are built from the String class, integer objects are built from the Integer class.

We've been using classes and objects without realising!

We can also write our own.

Create a new Python file and name it:



Let's break this down.

```
1 class Person():
2  def __init__(self, person_name):
3  self.name = person_name
```

They keyword class is used to define a new class.

```
1 class Person():
2  def __init__(self, person_name):
3  self.name = person_name
```

We use PascalCase to name our classes rather than snake_case.

This is to distinguish classes from regular functions.

```
1 class Person():
2  def __init__(self, person_name):
3  self.name = person_name
```

This is called a constructor or an initialiser.

It tells Python how to create an object of this class.

```
1 class Person():
2  def __init__(self, person_name):
3  self.name = person_name
```

self is one of the constructor's parameters.

It's a default that we create with all our classes – it allows us to go onto make objects from this class.

```
1 class Person():
2  def __init__(self person_name):
3    self.name = person_name
```

If we give the constructor a second parameter of person_name, we can set the attributes ourselves when we create our object.

```
1 class Person():
2  def __init__(self, person_name):
3  self.name = person_name
```

Now we set the attribute to person_name and will be able to fill this in (or name the person) ourselves.

```
1 class Person():
2  def __init__(self, person_name):
3  self.name = person_name
```

Why stop with name?

A class can have as many attributes as you need.

```
1 class Person():
2  def __init__(self, person_name, person_age, person_height):
3    self.name = person_name
4    self.age = person_age
5    self.height = person_height
```

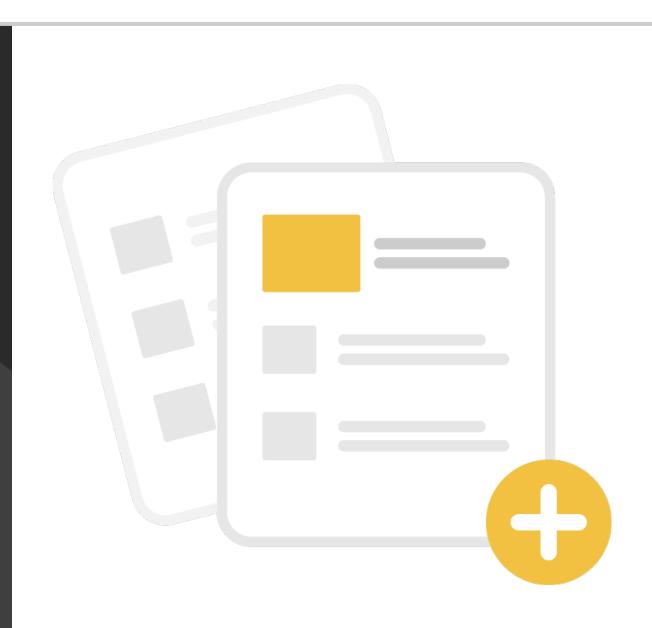
This is just a template to build a person from.

Let's use this template.

```
1 class Person():
2  def __init__(self, person_name, person_age, person_height):
3   self.name = person_name
4   self.age = person_age
5   self.height = person_height
```

Create another new Python file and name it:

main.py



Import the Person class from the person.py file.

This is exactly like importing a library like random, sys or time.

```
1 from person import Person
2
3 liam = Person("Liam", 30, "Tall")
```

Use the class Person to create an object called liam.

Pass the name, age and height as arguments.

```
1 from person import Person
2
3 liam = Person("Liam", 30, "Tall")
```

liam is the object we create.

It was constructed from the Person class with a name, age and height.

We defined the specifics when we asked for the liam object to be built.

```
1 from person import Person
2
3 liam = Person("Liam", 30, "Tall")
```

Reminder – file structure

We have two files working together.

person.py draws up the blueprint for Person, main.py does the building.

main.py

```
1 from person import Person
2
3 liam = Person("Liam", 30, "Tall")
```

```
1 class Person():
2  def __init__(self, person_name):
3  self.name = person_name
```

Reminder – file structure

OOP follows the SOLID principles.

S stands for single responsibility.

main.py

```
1 from person import Person
2
3 liam = Person("Liam", 30, "Tall")
```

```
1 class Person():
2  def __init__(self, person_name):
3  self.name = person_name
```

Reminder – file structure

person.py is responsible for the blueprint and the blueprint only.

If something is wrong with the blueprint, you know exactly where to go!

main.py

```
1 from person import Person
2
3 liam = Person("Liam", 30, "Tall")
```

```
1 class Person():
2  def __init__(self, person_name):
3  self.name = person_name
```

Print your object.

What do you see?

```
1 from person import Person
2
3 liam = Person("Liam", 30, "Tall")
4
5 print(liam)
6
7 # Output: person.Person
object at
0x7f817acf0fd0>
```

Dot Notation

Dot notation

We can access information about our object using dot notation!

object.property object.method()

At the minute we only have properties – how would I use the liam object to see Liam's height?

```
1 from person import Person
2
3 liam = Person("Liam", 30, "Tall")
5 # Object.property
6
7 print(liam.height)
8
9 # Output: "Tall"
```

Try it yourself!

Make a new you object!

Print out your name, age and height in a string from this object.

```
1 dave = Person("Dave", 50, "Tall")
2
3 print(f"My name is {dave.name}, I am {dave.age} and I am {dave.tall}")
```

Learning Objectives

- To learn about OOP
- Understand fundamental principles of OOP
- To code a simple class and understand instantiation

{codenation}

Activity 1

Using your person as an example, create a new class to build superheroes from.

Each superhero will have:

- A superhero name
- A secret identity
 - A superpower
- An arch enemy.

Create 4 superheroes from your class and use their properties to write a string about each.