COMP642

Object Oriented Programming

Lectorial 1

Learning Objectives

- Describe OOP concepts
- Differentiate between an object and a class
- Understand the four characteristics of the object-oriented paradigm
- Understand the OOP approach to problem-solving

What is OOP?

- Refers to a programming methodology that organises a program into a collection of interacting **objects**.
- Objects are organised into **classes**, which allow individual objects to be grouped together.
- Historically, a program has been viewed as a logical procedure that takes input data, processes it, and produces output data.
- Everything in OOP is grouped as "objects".
- OOP is implemented by sending **messages** to objects.

What is an Object? (1)

- An **object** can be considered a "thing" that can perform a set of activities.
- An object has characteristics or <u>**attributes**</u> (data).
- An object has actions or <u>**behaviours**</u> (methods).
- Example: Car

Attributes?

Behaviours?

What is an Object? (1)

- An **object** can be considered a "thing" that can perform a set of activities.
- An object has characteristics or **attributes** (data).
- An object has actions or **behaviours** (methods).
- Example: Car

Car has attributes such as model, brand name, registration number, colour, year.

Car has behaviours such as going forward, backward, stop, and accelerate.

What is an Object? (2)

Make: Toyota
Colour: Red
Year: 2010

Forward()
Backward()
Accelerate()
Stop()

Make: VW
Colour: Blue
Year: 2013

Forward()
Backward()
Accelerate()
Stop()

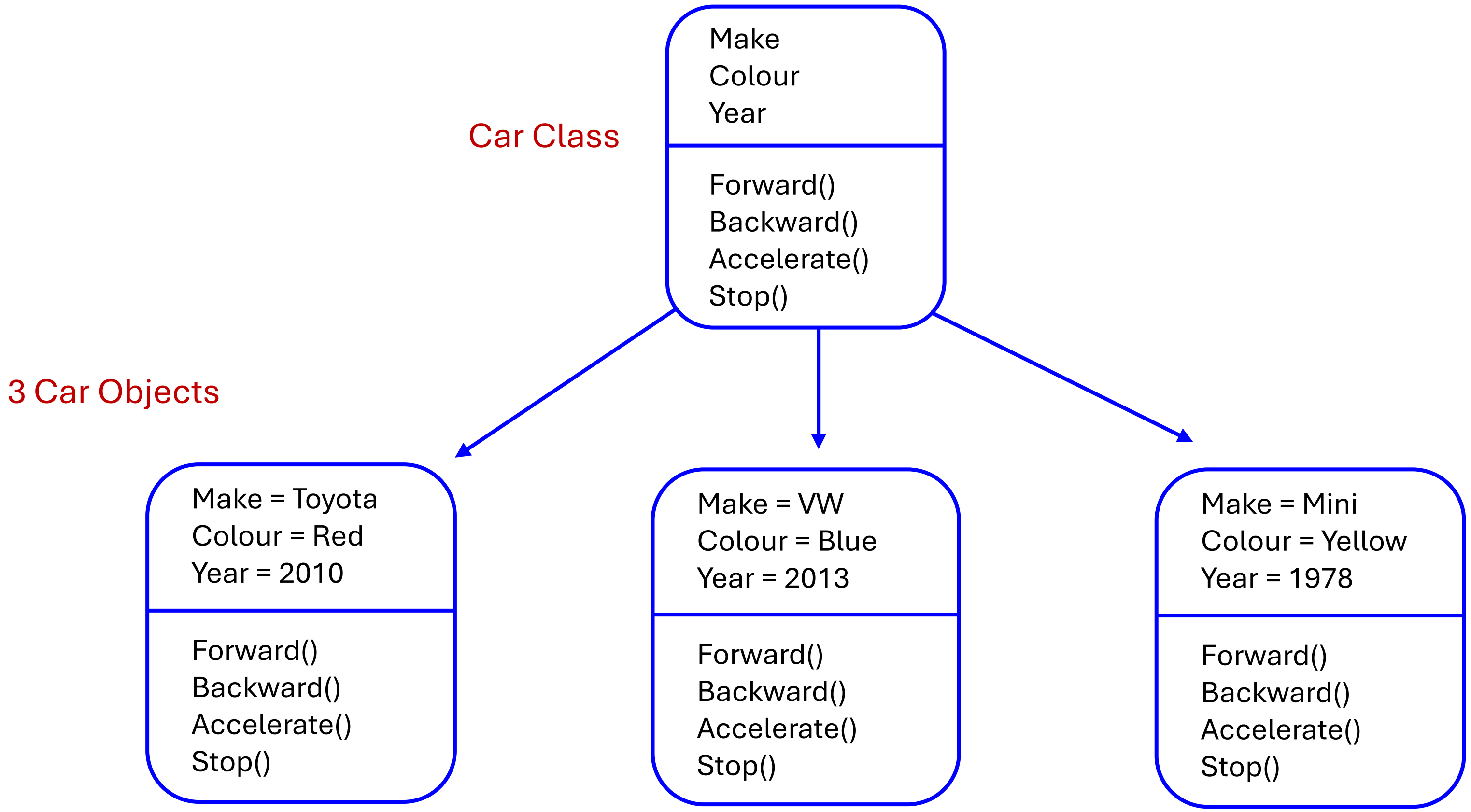
Make: Mini Cooper
Colour: Yellow
Year: 1978

Forward()
Backward()
Accelerate()
Stop()

Data (Attributes)

Methods (Behaviours)

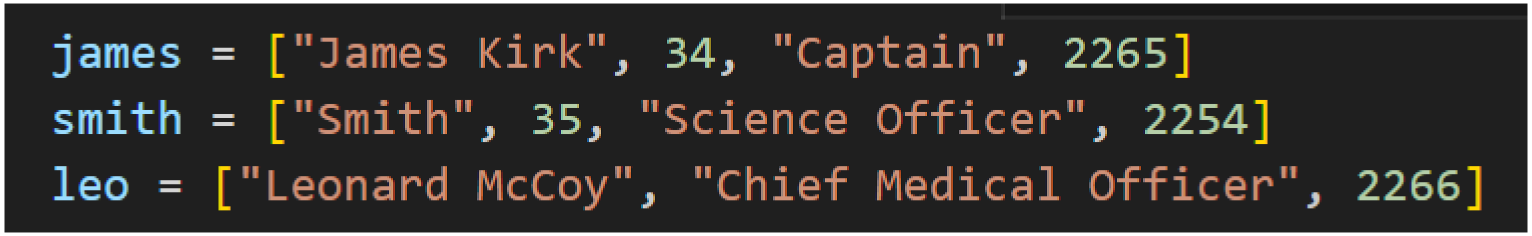
What is a Class?



- Every object is a member of a **class**.
- A class is a template or blueprint from which **objects** are created.

Example - Track Employees

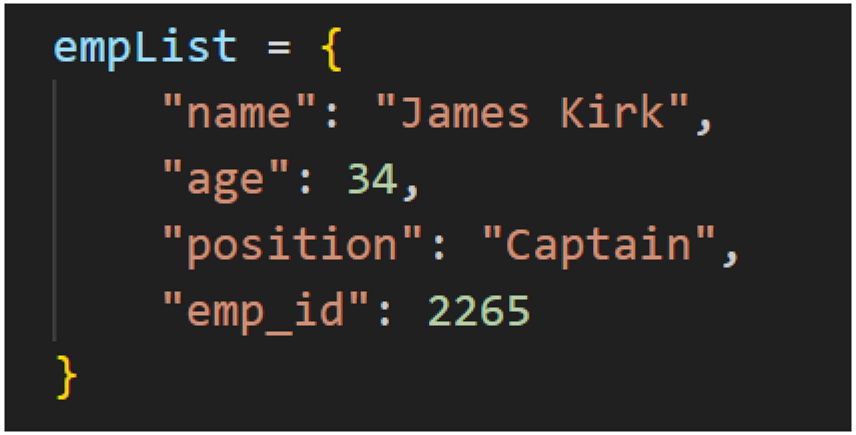
Using a List:



```
james = ["James Kirk", 34, "Captain", 2265]
smith = ["Smith", 35, "Science Officer", 2254]
leo = ["Leonard McCoy", "Chief Medical Officer", 2266]
```

What is the problem with this?

Using a Dictionary:



```
empList = {
    "name": "James Kirk",
    "age": 34,
    "position": "Captain",
    "emp_id": 2265
}
```

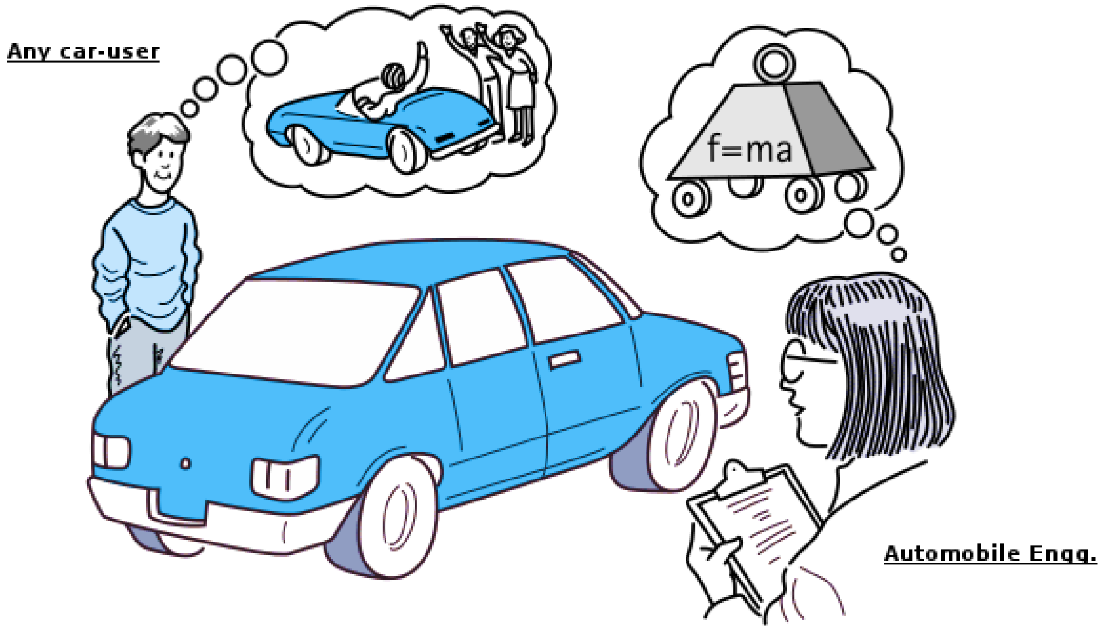
What is the problem with this?

Characteristics of OOP

Abstraction	Eliminate the irrelevant
Encapsulation	Hiding the unnecessary
Inheritance	Modelling the similarity
Polymorphism	Same function different behavior

Abstraction

- **Abstraction** focuses on the essential and eliminates the irrelevant.

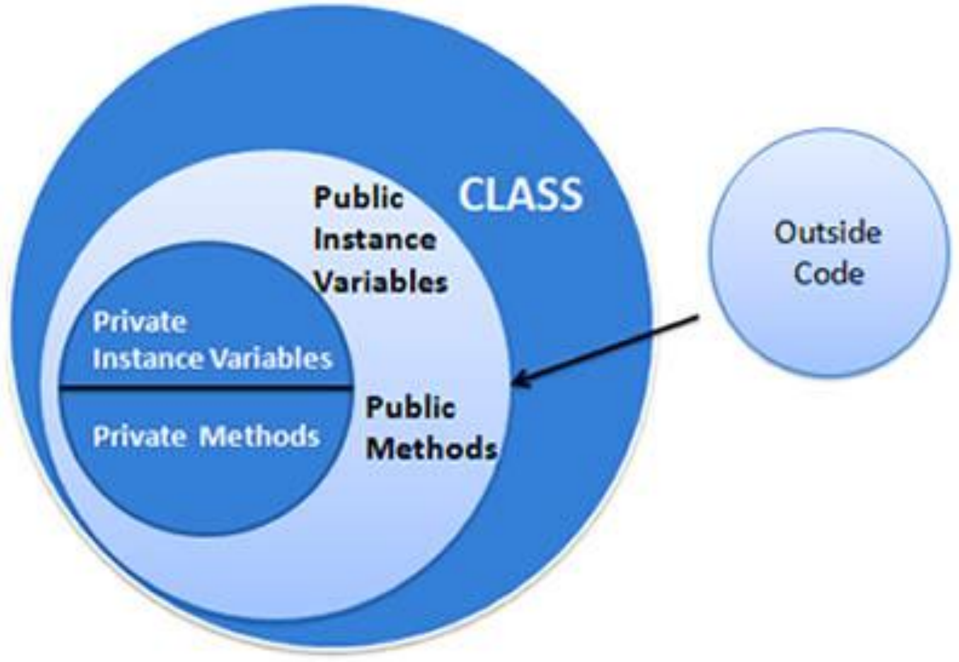


An abstraction includes the essential details relative to the perspective of the viewer.

https://logicmojo.com/assets/dist/new_pages/images/abstraction-example.png

Encapsulation

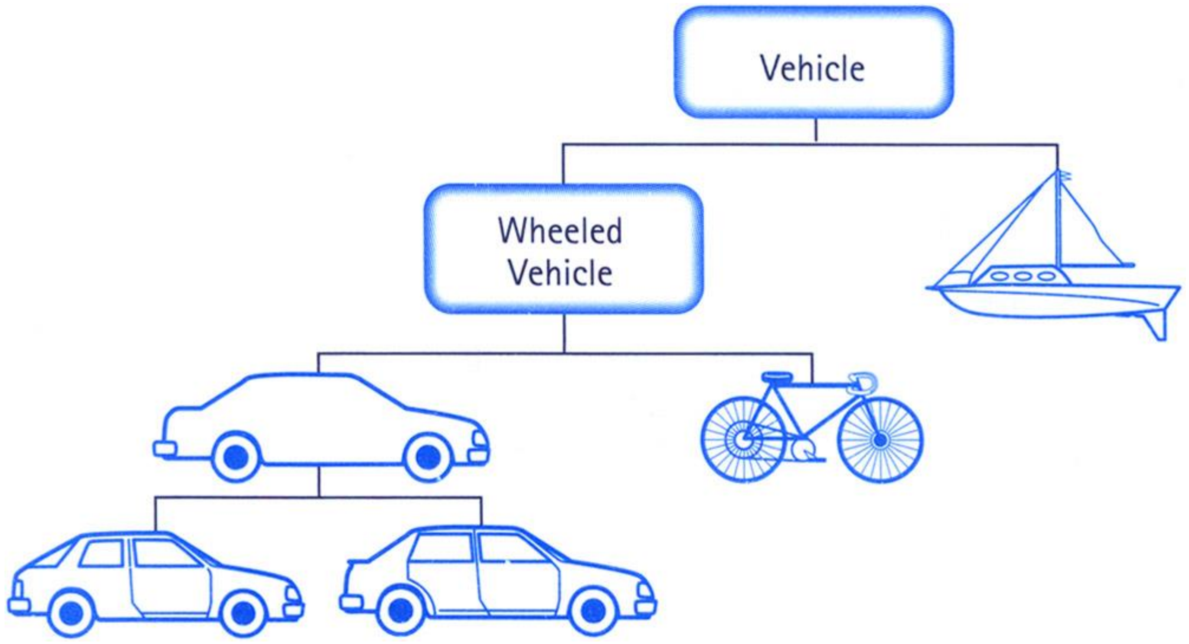
- Things that we do not want other people/classes to know are to be hidden.
- The only thing that an object knows about another object is the object's interface.



http://www.differencebetween.info/sites/default/files/images/4/image.jpg

Inheritance

- **Inheritance** allows a class to have the same behavior as another class and extend or tailor that behavior to provide special action for specific needs.
- It models an "is a" relationship between objects.



https://asiablog.acumatica.com/wp-content/uploads/2017/10/inheritance.jpg

Polymorphism

- **Polymorphism** allows two or more objects to respond to the same message.
- It allows a sending object to communicate with different objects in a consistent manner without worrying about how many different implementations of a message.

"Now Speak!"
"Quack"
"Woof"
"Meow"

http://technofriends.files.wordpress.com/2008/02/polymorphism.gif

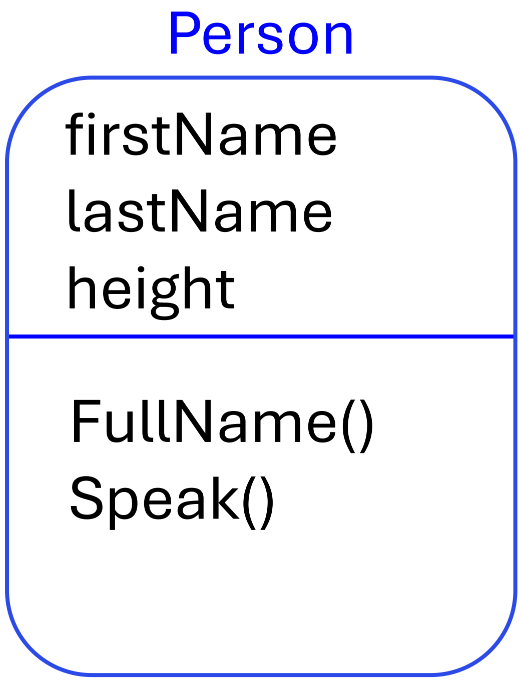
Object-Oriented Approach to Problem-Solving

1. Problem identification
2. Object identification - identify objects needed to solve the problem
3. Identify the attributes (data) and the behaviours (methods) and define the relevant classes
4. Identify relationships between classes
5. Work on the objects - message sequences to solve the problem

OOP in Python

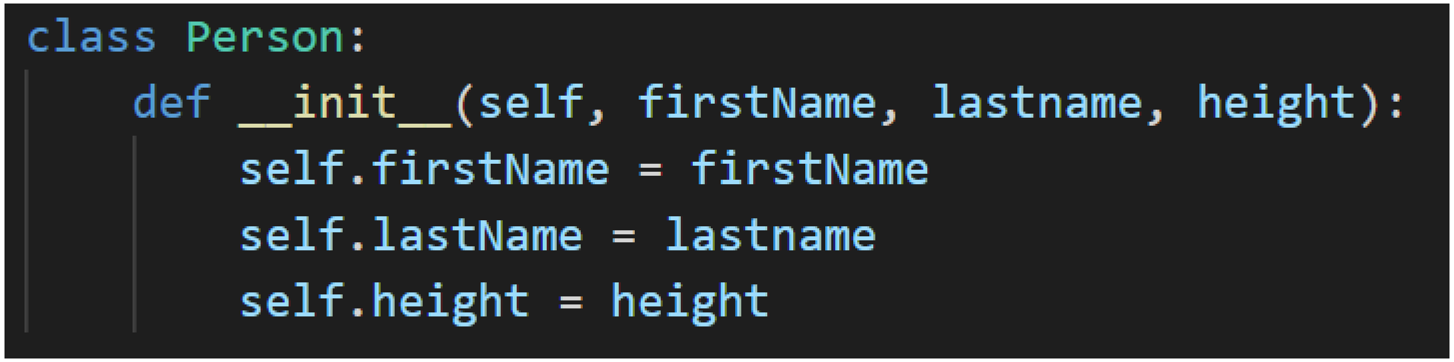
- Python supports OOP
- Provides a clear program structure and clean code
- Facilitates easy maintenance and modification of existing code
- Since class is shareable, the code can be reused
- Does not allow strong data encapsulation (hence not a 100% pure OO language)

Example



Example - Declaring Class

- Classes are defined usually (but not always) in a separate file.
- The class **Person** below has 3 attributes/properties.



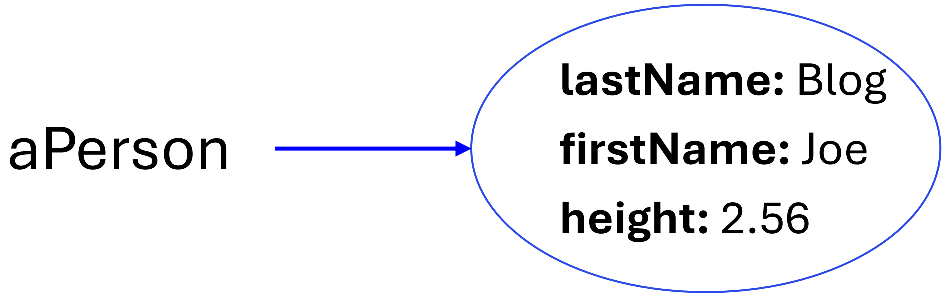
```
class Person:
    def __init__(self, firstName, lastname, height):
        self.firstName = firstName
        self.lastName = lastname
        self.height = height
```

Example - Making Objects

Declare a variable to refer to your object

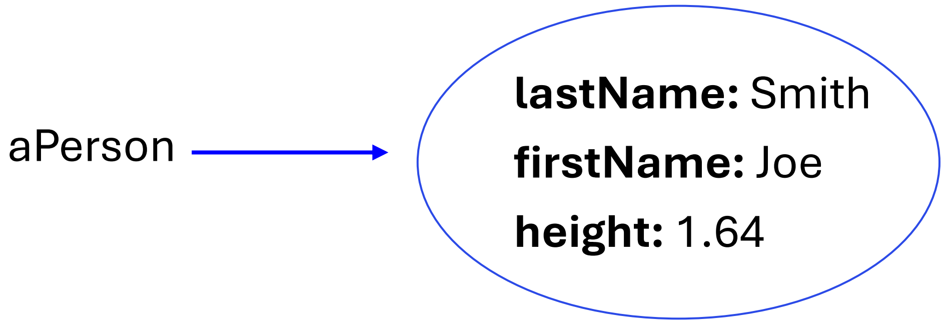
```
aPerson = Person("Joe", "Blog", 2.56)
```

Create the new object



Example - Using Public properties

```python
aPerson.lastName = "Smith"
aPerson.height = 1.64
```



- The properties can then be given a value.
- The values of these properties can be used like any other variable.

```python
print("{} {} {}".format(aPerson.firstName,aPerson.lastName, aPerson.height))
```

```
Joe Smith 1.64
```

Example - Methods (1)

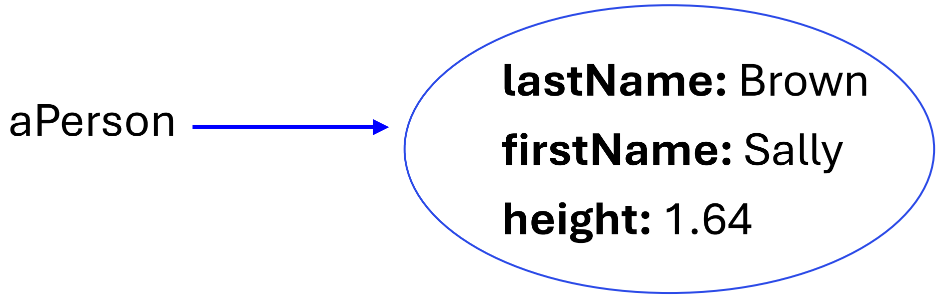
- Methods are procedures or functions in the class definition.

```python
def fullName(self):
    return self.firstName + " " + self.lastName
```

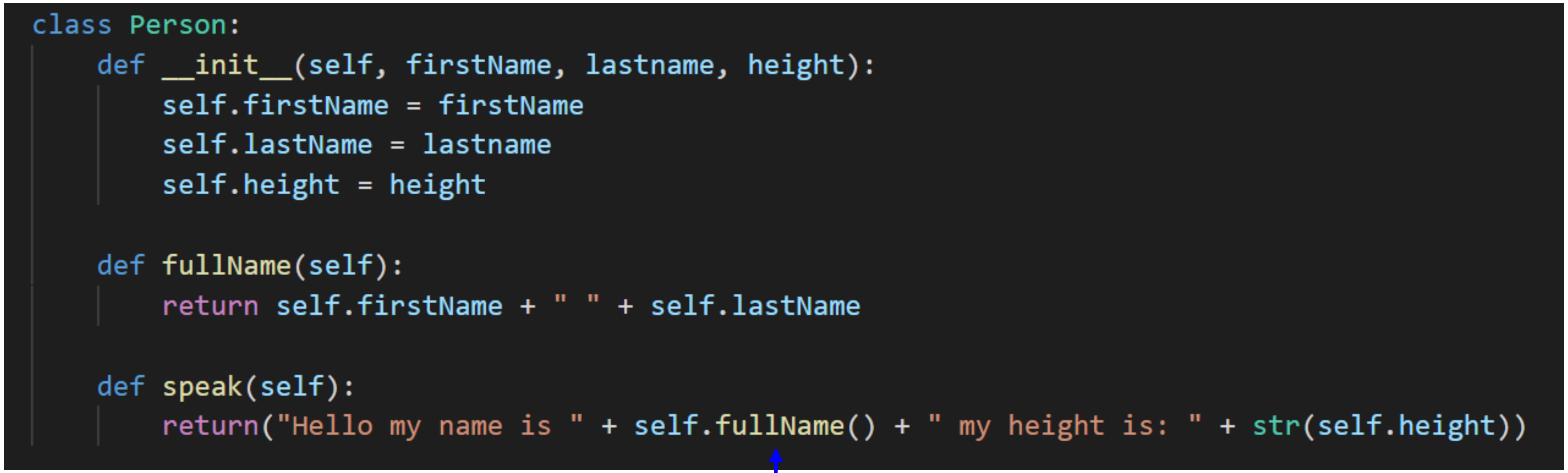
- You can call the method from elsewhere (e.g.; from the form).

```python
print(aPerson.fullName())
```

- When the aPerson object is asked to return fullName it uses its own values of firstName and lastName.



Example - Methods (2)



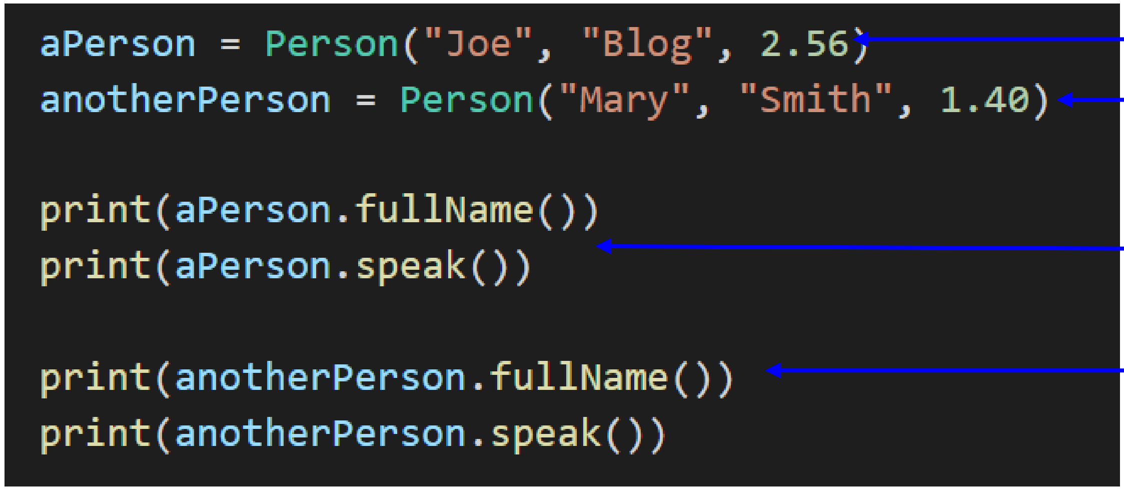
```python
class Person:
    def __init__(self, firstName, lastname, height):
        self.firstName = firstName
        self.lastName = lastname
        self.height = height

    def fullName(self):
        return self.firstName + " " + self.lastName

    def speak(self):
        return("Hello my name is " + self.fullName() + " my height is: " + str(self.height))
```

Can call the method of the same class

Example - Methods (3)



```python
aPerson = Person("Joe", "Blog", 2.56)
anotherPerson = Person("Mary", "Smith", 1.40)

print(aPerson.fullName())
print(aPerson.speak())

print(anotherPerson.fullName())
print(anotherPerson.speak())
```

Each object has its own values for its properties.

Each object can access methods of the class.

Example

We want to build a simple application that will manage a savings account.

The transactions supported are deposit, withdraw, calculate interest, and check balance.

Problem??

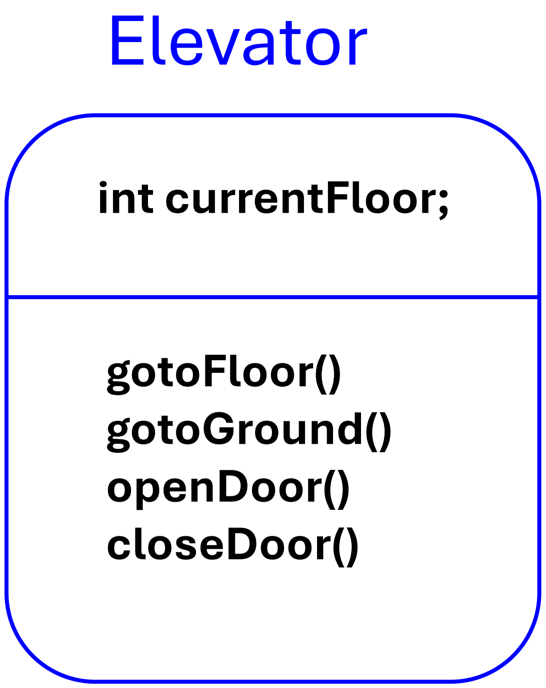
Object?? Class??

Data?? Methods??

How do we use the class to assemble the application?

Test Your Knowledge

Create a class called **Elevator** based on the diagram below. Write a simple application to simulate the elevator's operation. You may assume that the building has 6 floors.


Elevator
int currentFloor;
gotoFloor()
gotoGround()
openDoor()
closeDoor()