Teaching London Computing

A Level Computer Science

Topic 6: Introducing OOP

William Marsh
School of Electronic Engineering and Computer Science
Queen Mary University of London

Aims

- What and why OOP?
  - The problem of software development
- OOP concepts
  - Data hiding
  - Class and instances
- Using classes in Python
  - Using classes – example of files
  - ‘Methods’ versus functions
- Create a new class in Python

What is OOP and Why?

What is OOP?

- Object-oriented programming IS
  - An idea for organising programs
- Object-oriented programming IS NOT
  - A completely different type of programming
    - Builds on if, while, **functions** etc
  - Necessary: remember it's all assembly code eventually
- At first, OOP is more complex

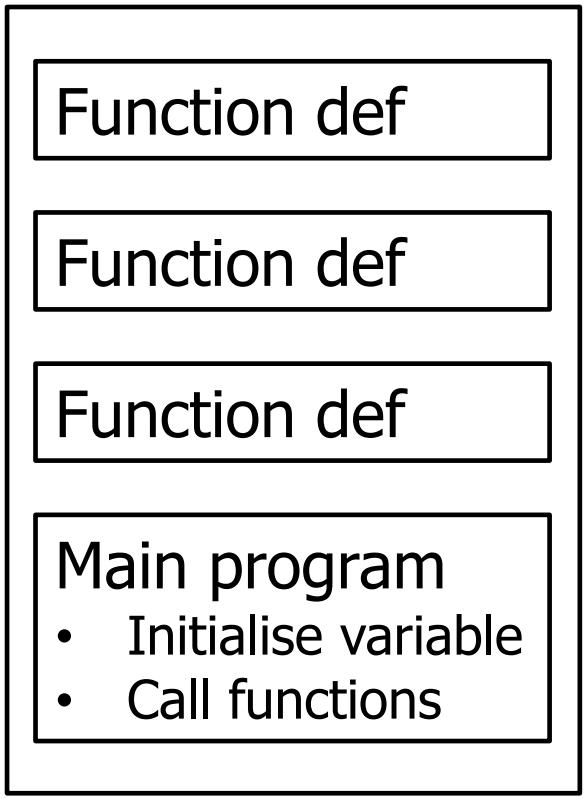
Why Organise S/W?

- Hard to organise large problem
- Work must be shared across a team
  - Imagine building a house with no plan?

- Advantages claimed for OOP organisation
  - Better reuse of code in libraries
  - Software easier to change
- OOP very popular for Graphical User Interface (GUI) libraries

Software Organisation So Far

- Break overall program into functions
- **Discussion**: is it obvious what functions to choose?
- Aside: more complex organisation possible



Exercise 1.1 (and 1.2)

- Use google to find an example of a failed software project in the UK
  - How late?
  - How much money wasted?

OO Concepts

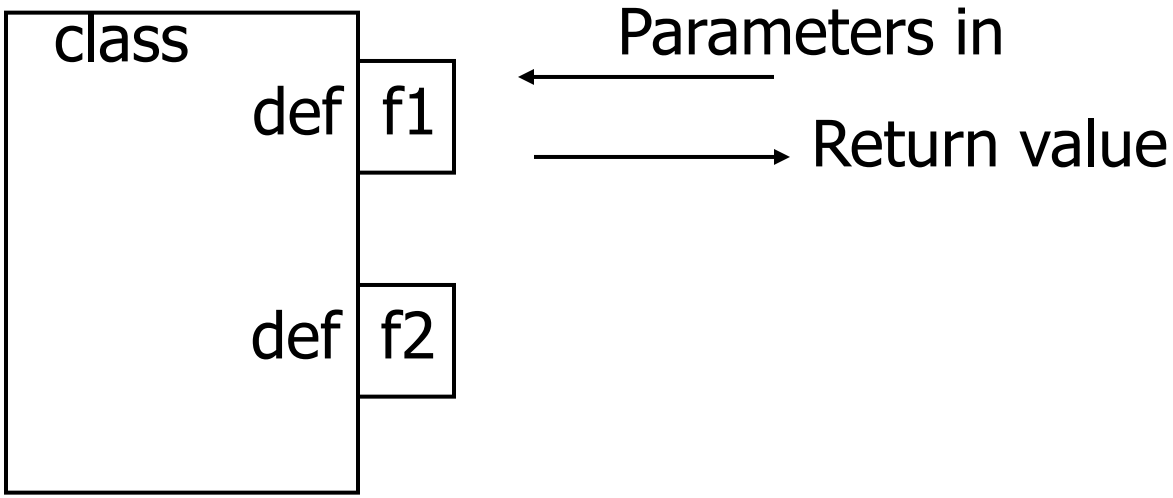
Data Hiding – Abstraction

- Different ways to represent complex data
- Example: shopping list
  - List of pairs: (item *string*, amount *integer*)
  - Dictionary: map from item to amount required

- Data hiding principle: **the user should not know the representation**
  - It may change
  - Instead, provide functions (or ‘operations’)

What's a CLASS – I

- A box with buttons (functions or operations or methods)



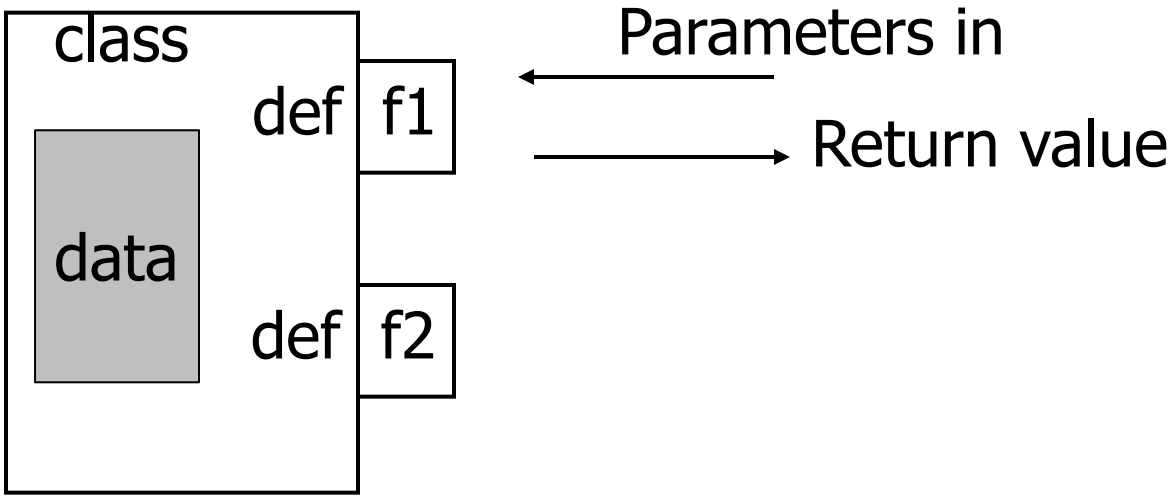
- A class is just a template

Words

- **Method**: this word is used in OOP theory
- **Function**: Python has these, as do other programming languages
- **Operation**: this word is used on OO analysis

What's a CLASS – II

- A box containing data (variables)

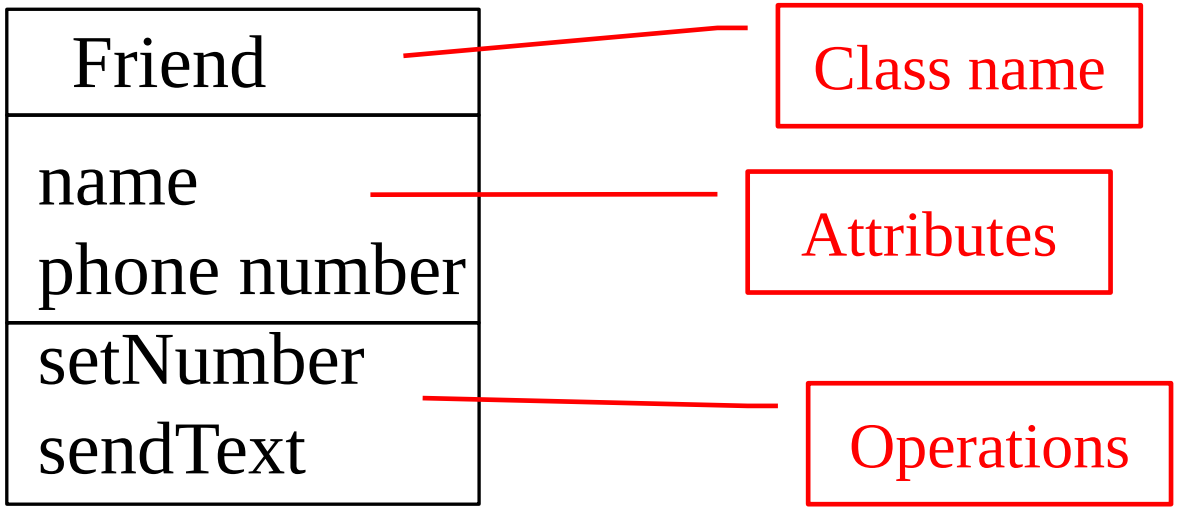


- A class is just a template

Picture of Classes

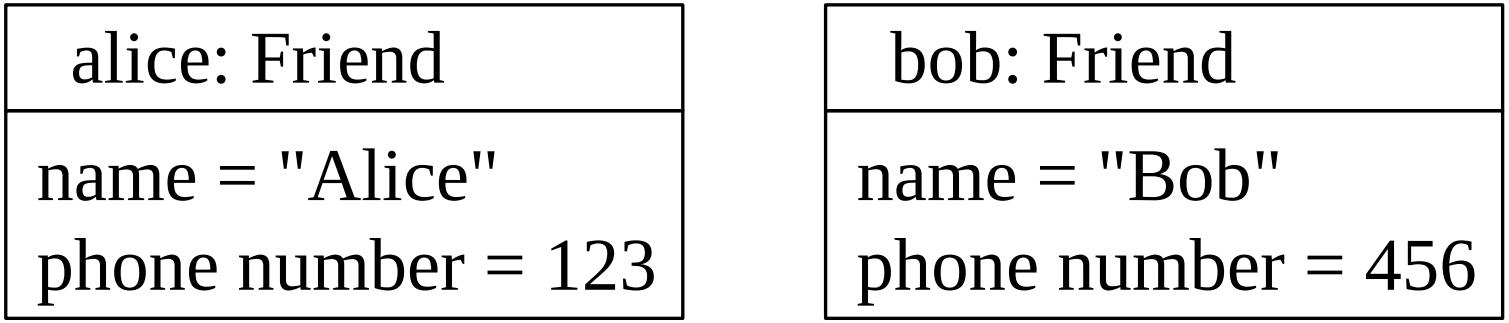
- A class has
  - A name
  - Attributes (i.e. variables)
  - Operations (i.e. functions)

Data hiding:
- Hide the attributes
- Use the operations



Object: An Instance of a Class

- A class is a template
- An object is a particular instance of a class
  - Different data (attribute values)
  - Same code



Exercise 2.1 – Shopping Functions

- Recall the shopping list representations:
  - List of pairs: (item *string*, amount *integer*)
  - Dictionary: map from item to amount required

- Suggest the **functions** that would be useful
  - What do you do with a shopping list?

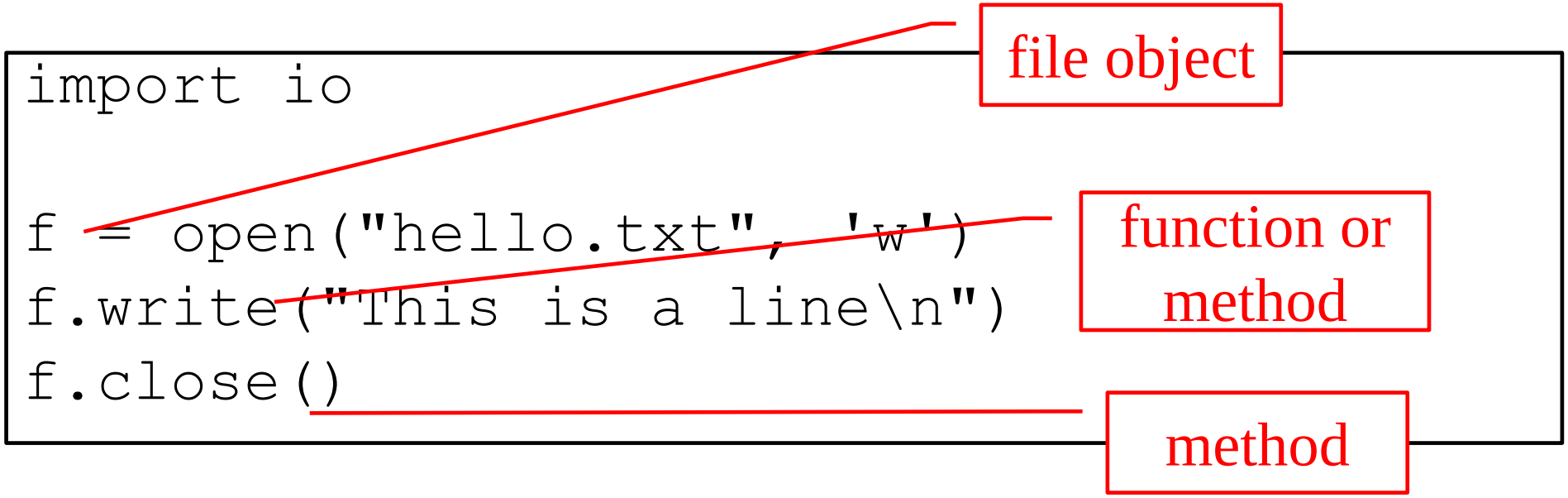
Using Objects in Python

You already do this

Example: Files

```
import io

f = open("hello.txt", 'w')
f.write("This is a line\n")
f.close()
```



- There is not a ‘file’ class; the object is of one of several classes

Example: Two Files

f and g are different objects

```
import io

f = open("hello.txt", 'w')
g = open("bye.txt", 'w')
f.write("Hello to you\n")
g.write("Good bye. Go away.\n")
f.write("You are welcome\n")
g.close()
f.close()
```

What Data is in the File Object?

- We are not told: details probably depend on the OS

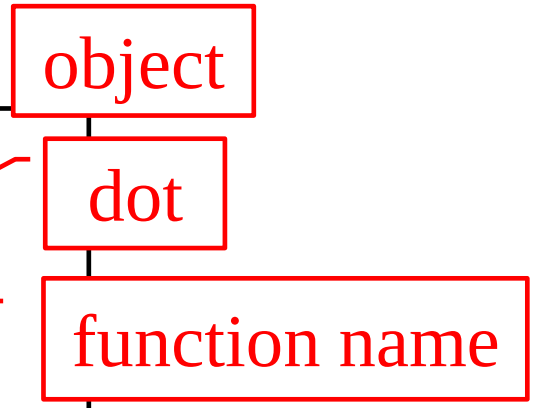
- File name
- Location of file on disk
- Buffer of text

- Each file object must have different data

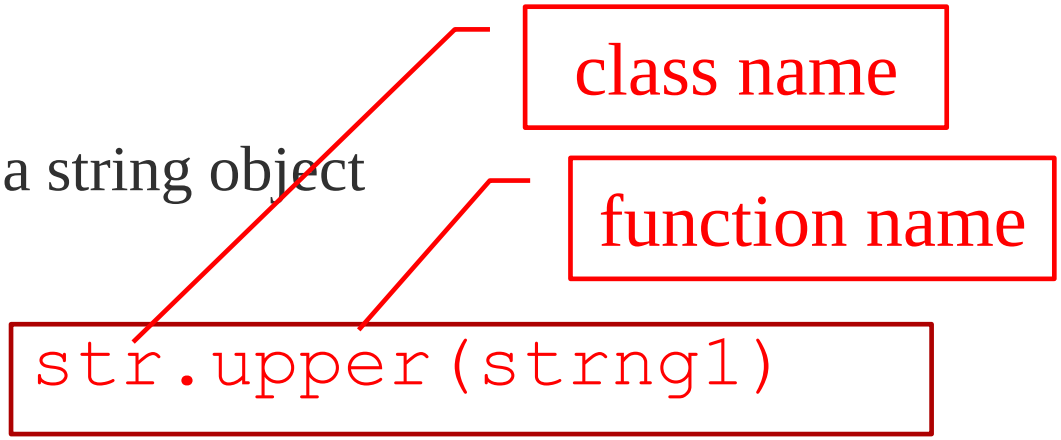
Function and Method Syntax

```
strng1 = "hello william"
n = len(string1)

strng2 = strng1.upper()
```



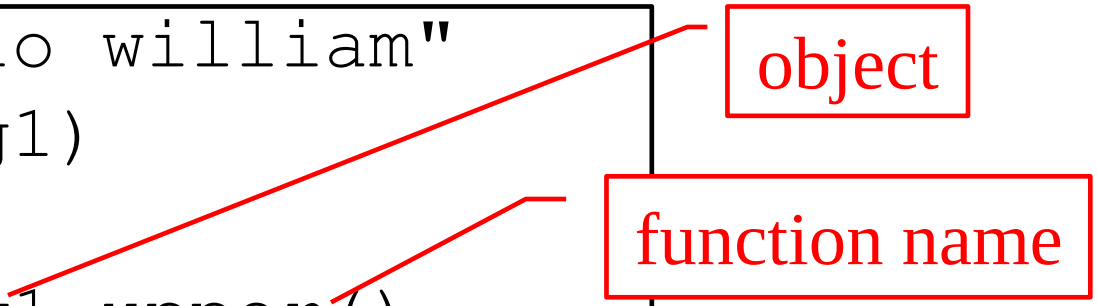
- 'str' is a class
  - `str(99)` – returns a string object
  - Equivalent syntax



Functions and Methods

```
strng1 = "hello william"
n = len(string1)

strng2 = strng1.upper()
```



- The `strng1` object has a class
- Take the `upper()` function from this class
- Call it with the object as the first parameter
- … add any further parameters

Lists are Objects

```
>>> lst = [1,2,3,4]
>>> lst.append(99)
>>> lst
[1, 2, 3, 4, 99]
```

- The list is changed
  - Append 99 to the list lst
- Nothing is returned

```
>>> lst = [1,2,3,4]
>>> type(lst)
<class 'list'>
>>> list.append(lst, 99)
>>> lst
[1, 2, 3, 4, 99]
```

Exercise 3.1 and 3.2

- Look at String and List method in the Python documentation
- Try some out.

Define New Classes in Python

This bit is new

Declaring A Class

- A person class with two functions

```
class Person:
    def setAge(self, a):
        self.age = a

    def getAge(self):
        return self.age
```

- `setAge()` function sets an attribute `age`
  - Remember: in Python variables are initialised, not declared

Using The Person Class

- Create instance of the Person class
  - i.e. people!

```
p1 = Person()
p2 = Person()

p1.setAge(21)
print(p1.getAge())
p2.setAge(101)
print(p2.getAge())
```

Use class name to construct new objects

What is ‘self’?

- The name self is used by convention
  - Not a key word
  - **Always** use it

- Explanation (*not essential*)
  - In the ‘dot’ syntax, object is first parameter
  - … so function called with method syntax needs at least one parameter

Exercise 4.1 – Person Class

- Enter the Person class
  - The class declaration and the ‘using code’ go in the same file
- Add another attribute:
  - What else can you know about a person?

Problem – Initialising Attributes

- What happens if we get the age before it is set?

```
p1 = Person()
p2 = Person()

print(p1.getAge())
p1.setAge(21)
```

- Need to initialise the attributes

Constructor

- Constructor is a special function
- Called using class name

```
class Person:
    def __init__(self, n):
        self.name = n
        self.age = 0

    def setAge(self, a):
        self.age = a

    def getAge(self):
        return self.age
```

Using a Constructor

- Constructor called using class name

```
p1 = Person("Alice")
p2 = Person("Bob")

print(p1.getAge())
p1.setAge(21)
```

- If you do not define a `__init__` the default constructor creates an empty object

Exercise 4.2 – Add a Person Constructor

- Add a constructor to the person class
- Initialise all the attributes
  - **Either** to default values
  - **Or** to values given as parameters

- Write code to use the class

Working With Many Source Files

- Module – file containing Python definitions
  - Contains function and **class definitions**

- **Guideline**
  - Write each class in a separate file
  - Filename same as class name
  - Import:

```
from Person import Person
```

Summary

- Object-oriented programming is a way to organise more complex programs
  - Learn the syntax and behaviour
  - Learn how to use OO to organise a program

- A class is a template for an object. An object has
  - Attributes: *what is unique about this object?*
  - Operation: *what can you do to it?*
- Data and code are organised together
  - Supports data (information) hiding – abstraction