Scientific Programming Using Object-Oriented Languages

Module 10: Other languages

Aims of Module 10:

After this module you should be able to

- explain the difference between a statically and dynamically typed programming language;
- list some similarities and differences between Java, C++ and Python;
- create and run a simple program including at least one custom class in C++ or Python.

A simple C++ program

```
#include <iostream>

int main() {
    std::cout << "Hello World!" << std::endl;
}
```

- To compile and run this program, log on to the Aristotle Linux system, or install the Gnu Compiler Collection (gcc) on a Linux or Mac OSX laptop.
- Type these commands to compile and then run the code:

```
g++ -o helloworld
./helloworld
```

A TwoVector class in C++ (part 1 of 3)

```
#include <iostream>
#include <cmath>

class TwoVector {
    private:
        double x_;
        double y_;
    public:
        TwoVector(double x, double y);
        TwoVector operator+(TwoVector other);
        double magnitude();
};
```

- This is the “header”, which *declares* the structure of the class so it can be used by a separate program.

A TwoVector class in C++ (part 2 of 3)

```
TwoVector::TwoVector(double x, double y): x_(x), y_(y) {}

TwoVector TwoVector::operator+(TwoVector other) {
    double x = x_ + other.x_;
    double y = y_ + other.y_;
    TwoVector result(x,y);
    return result;
}

double TwoVector::magnitude() {
    return std::sqrt(x_*x_ + y_*y_);
}
```

- This code *defines* the behaviour of the class.

A TwoVector class in C++ (part 3 of 3)

```
int main() {
    TwoVector v1(1,0);
    TwoVector v2(0,1);
    TwoVector z = v1 + v2;
    std::cout << z.magnitude() << std::endl;
}
```

- This is a main program that uses the `TwoVector` class.
- To compile this program, you need to *include* the header declaring the class.
- Before running it you need to link it to

Features of C++ (1 of 2)

- C++ is a statically typed language, like Java:
  - variables and functions have types
  - types must be specified at compile time
  - the compiler checks for type errors
- C++ is a compiled language
  - several stages to create executable program from source code: preprocess, compile, assemble, link
- C++ syntax:
  - similar to Java, but not identical
- Classes
  - constructor similar to Java, but different syntax to call it
  - special names like `operator+` allow overloading of operators like +

Features of C++ (2 of 2)

- Memory management in C++
  - variables can be created in two ways: on the stack or the heap
  - different syntax for both
  - programmer has to deal with pointers to memory addresses
  - programmer has to explicitly delete objects that are no longer needed
  - this is complex and can lead to errors
  - compare to Java, which does “garbage collection” to automatically find and delete unused objects
- C++ gives the programmer a great deal of control, for better and for worse!

A simple Python program

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

- You can also run this on Aristotle, or a Mac OSX laptop.
- There are also several ways of running it on Desktop@UCL or another Windows system.
- To run from a text console:

```
python helloworld.py
```

A TwoVector class in Python

```
from math import sqrt

class TwoVector:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return TwoVector(x,y)
    def magnitude(self):
        return sqrt(self.x*self.x + self.y*self.y)

v1 = TwoVector(1,0)
v2 = TwoVector(0,1)
z = v1 + v2
print(z.magnitude())
```

Features of Python

- Python is a dynamically typed language:
  - variables and functions don't have types
  - values still have types
  - functions can deal with any type that satisfies certain conditions
  - type errors are only found at run time
- Python is an interpreted language
  - but it does have an intermediate byte code form like Java
- Python syntax:
  - indentation counts, not braces or semicolons
- Classes
  - `__init__` is equivalent of constructor
  - special names like `__add__` allow overloading of operators like +
  - self is equivalent of this, but needed as explicit argument to functions