Introduction to Computer Programming

1.1 Course Introduction

What is computer programming?

Programming is a way to tell a computer to do a specific task e.g.

- adding two numbers
- plotting some data as a graph
- saving a file

Computers understand instructions that are written in a specific *syntax* form called a **programming language**.

Why study programming?

- A tool you can use for your studies (data, modeling, calculation, simulation...).
- A growing sector of the jobs market (software engineering, data science, AI, robotics...).
- Increasing use of coding within jobs and areas of academia not traditionally related to computing.
- Coding is fun! (personal projects, creative technology, competitions, charity sector...)

Why study Python?

- Free and open source
- "High level"
  - human-understandable
  - abstracted from machine-level computing processes (eg. memory management)
- Easy to learn
- Versatile - both a *scripting* and a *programming* langauge:
  - *scripting* : run in a host application for debugging and viewing output (e.g. a mathematical model)
  - *programming* : controls a computer or machine (e.g. a microcontroller on a robot)
- Increasingly used in industry and academia
- Large and diverse community of users and developers

Course Goals

- A good standalone programming toolkit.
- Skills to improve the quality of your work in other subjects.
- A fundamental base from which to start developing further as a programmer.

Course Entry Level

Beginner, no prior programming knowlegde.
First year (and upwards), engineering / engineering-related degree programmme

Course stucture: Weekly schedule

Group 1 : EENG

Group 2: everyone else - EMAT, Digital Health (MSc + CDT), Biorobotics (MSc) etc

Lecture/lab

Group 1	Group 2
Monday	Tuesday
10:00-12:00	13:00-15:00
MVB 2.11	MVB 2.11

Drop-in support sessions:

Bring your questions about the weekly exercises for 1-2-1 support from TAs

On-campus
Friday
15:00-17:00
MVB 1.15

Class Structure:

Lecturers: Hemma Philamore, Matthew Hennessy

Part 1 (1 hour)

- 15-30 mins theory/demos
- 30-45 mins practise exercises (essential + advanced questions)

Part 2 (1 hour)

- 15-30 mins theory/demos
- 30-45 mins practise exercises (essential + advanced questions)

Please sit in the same seat/area of the lab each week - TAs will monitor the same areas of the lab - look out for their lanyards!

Complete any unfinished 'essential' exercises for homework (not assessed).

Optional weekly drop-in sessions: Bring your questions about the weekly exercises for 1-2-1 support from TAs

How To Access the Course Material

Blackboard page for unit EMAT10007:

- Slides (pdf and Jupyter notebook)
- Weekly exercises (pdf)
- Example answers released the following week

Different ways of running Python / software we will use

We wil run Python code (scripts) in different environments:

- Jupyter notebook (lecture slides, demos)
- Spyder IDE (integrated development environment)

IDE (integrated development environment)

A single piece of software to:

- write and edit code
- run code and see the output
- debug code (graphical dispay showing the nature and cause of an error)

In [1]:

```
B = 7

print(B)
```

```
7
```

The focus of this course is to learn the Python **syntax** .

We will use a limited number of environments to run Python code.

Python may be be run in many other environments (computer command line, raspberry pi etc...)

Installing Anaconda on your personal computer

Anaconda installs a number of applications including:

- Jupyter notebook
- Spyder IDE

Instructions for installing Anaconda on your personal computer:

- Windows : https://docs.anaconda.com/anaconda/install/windows/ (https://docs.anaconda.com/anaconda/install/windows/)
- Mac : https://docs.anaconda.com/anaconda/install/mac-os/ (https://docs.anaconda.com/anaconda/install/mac-os/)
- Linux : https://docs.anaconda.com/anaconda/install/linux/ (https://docs.anaconda.com/anaconda/install/linux/)

Opening Anaconda

You may use the lab computers or personal laptop computers in the lecture:

Linux lab

- Click on Activities in the top left corner of the screen.
- Open Terminal from programs menu
- `/opt/anaconda/2020.07/bin/anaconda-navigator`
- Press enter

Personal computer

- Choose Anaconda from the programs menu

Opening Jupyter notebook

Click 'Launch' next to the Jupyter notebook application.

The application will open in a web browser.

Opening lecture notes in Jupyter notebook

1. Download lecture notes as a .ipynb file from blackboard.
2. Within the Jupyter notebook browser window you will see your computer file system.
3. Navigate to where you have saved the .ipynb file from blackboard.
4. Click on it to open.
5. You can now interactively edit and run the code you see on the lecture slides.

Alternativey you can open the slides as pdf (non-interactive) version.

Dependencies

If viewing slides using Jupyter notebook...

The folders `sample_data` , and `img` should be stored in the *same folder* as the lecture slides (.ipynb files) for the images and code examples to appear correctly in the lecture slides.

You can download `sample_data` , and `img` from the main blackboard page.

It is recommended that you store **all slides**, **`sample_data`** , **`img`** together in a single folder.

Opening Spyder

Click 'Launch' next to the Spyder application

How to save your work in Spyder

You will complete the weekly exercises in spyder.

These are effectively your notes.

Save them in an organised way (e.g using the week/class number).

File >> Save >> `Filename` (.py file extension automatically added)

For now, save your files under your username, you *should* be able to access this from elsewhere in the university.

How you will be assessed

Coursework 1 (50%): Syntax test. Set week 5, deadline week 7 (Feedback week 9)

Coursework 2 (50%): Modelling exercise. Set week 9, deadline week 11

Any Questions?

Let's give it a go!...