COMPUTER SCIENCE

Teacher name: Mike Collins

Exam board: OCR

Email address: mike.collins@stockton.ac.uk

Are you thinking about studying Computer Science at A Level? The activities in this four-week study programme will give you head start and an insight into what it is like to study Computer Science at Bede. Practical problem solving and programming is at the heart of what we do and through these activities you will develop both of these skills. Some of these activities will build upon GCSE Computing but you do not need to have studied the GCSE programme in order to do them.

Week 1: Learn to programme or develop your existing skills

Programming is at the core of what we do and although we programme in a different language, Python is an excellent start. Its easy to read and understand and its very accessible i.e. you can develop online or from your home computer.

Go to: https://www.learnpython.org/

1. Have a look at “Learn the Basics” and work through each section up to and including “Basic String Operations”. Each section builds upon the last, and it is useful to make your own programming notes as you go along. This way you can refer back to them when you need to.

2. Once you have complete the various exercises, see if you can apply your skills to solving this problem.

   You need to write a basic program where the user can enter a sum in format:

   123 + 945

   When the user presses enter, the program should calculate and output the answer. The user must be able to type the sum all in one go, on the same line. Your job is to use string operations to split the sum up into its component parts i.e. first number, second number and operators, then calculate the answer.

   First try and get it to work with addition and fixed length numbers, then see if you can make it work with any operator and variable length numbers. Remember to save your code, you will need it later on in this course.

Week 2: Getting under the hood of how the CPU works with LMC

In week one, you started to develop your programming skills using the high-level programming language, Python. This week we are going to look at a low-level programming language called Assembly and use this to try and help us understand how a CPU works. To do this we are going to use a simulator called LMC or Little Man Computer. There are literally hundreds of tutorials on this but I have picked two YouTube videos to get you started:

Tutorial 1: https://youtu.be/kCyyZI1GgsQ
Tutorial 2: https://youtu.be/sEFnRDgkaWA
Simulator: http://peterhigginson.co.uk/LMC/

1. Work through both tutorials – the first is theoretical, the second practical. Again, it is worthwhile making notes on how the process works and what each instruction does.

2. You might have to do a little more research – these two tutorials might not give you all the knowledge you need (everyone learns differently).

3. Once you have completed these tutorials, see if you can write a programme that will multiply together two numbers that the user has inputted. Hint: there is no multiplication function in LMC but multiplication is really just a succession of additions.

Week 3: Understanding the Fetch Execute Cycle

Alan Turing is often referred to as the father of modern computing, however in his day there wasn't even an electronic calculator. What Turing conceived, be it theoretical, was a machine which humans could command. He called this a Universal Machine. Some years later, another computer scientist call John von Neumann built upon Turing's idea to develop the first programmable computer. His concept – **the Fetch Execute Cycle** – is still used today in all computers from Desktops to Smart watches.

1. Watch the following video on YouTube: https://youtu.be/OTDTdTYId2g, again you might want to do some additional research as there are lots of tutorials that cover the Fetch Execute Cycle.

2. See if you can construct a flow-diagram to represent the steps involved in fetching an instruction and executing it.

3. Stretch & Challenge: programming languages allow developers to create decisions and loops – can you work out what, in the FEC, is changed to make these decisions and loops work?

Week 4: Back to Python

For the last task, we are going to jump back to the Python tutorials that you looked at in week 1. Go to: https://www.learnpython.org/

1. Continue on from where you left off and complete:
   (a) Conditions
   (b) Loops
   (c) Functions
   (d) Directories

Make sure you do all the exercises at the end of each section.

2. Look back at the code that you wrote in week 1 and see if you can create a set of functions, which you can pass two numbers, to carry out the four arithmetic operations i.e. addition, subtraction, division and multiplication.

Stretch and Challenge Tasks

Below are a number of Code Challenges that allow you to practise and consolidate what you have learned:

1. Speed Tracker
Create a program that takes a time for a car going past a speed camera, the time going past the next one and the distance between them to calculate the average speed for the car in mph. The cameras are one mile apart.

2. Unit Converter (temp, currency, volume)
Converts various units between one another. The user enters the type of unit being entered, the type of unit they want to convert to and then the value. The program will then make the conversion.

3. Credit Card Validator
Takes in a credit card number from a common credit card vendor (Visa, MasterCard, American Express, Discoverer) and validates it to make sure that it is a valid number (look into how credit cards use a checksum).

4. Name that Number
Telephone Keypads often have letters associated with each number. This means that 0141 117 2556 could be stored as 0141-CAT-DOOR. Create a program that can convert a phone number with “letters” into one that only contains digits.

Reading list

https://craigndave.org/ocr-alevel-h046-h446/
https://cyber-school.joincyberdiscovery.com/
http://www.cs4fn.org/