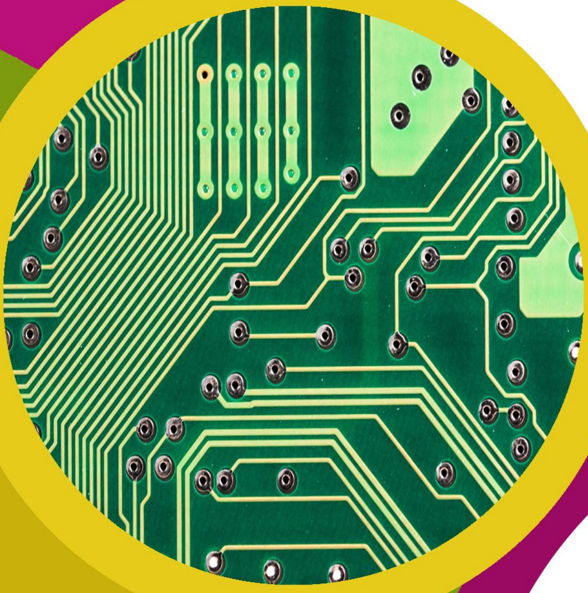
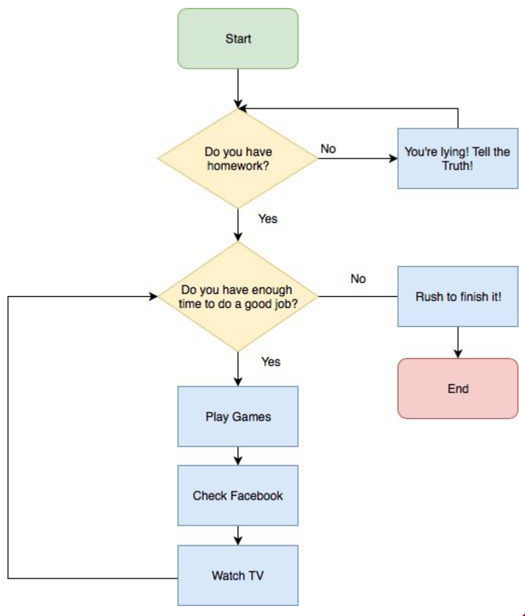
technocamps

Algorithms Session Plan





UNDEB EWROPEAIDD EUROPEAN UNION
Llywodraeth Cymru Welsh Government
Cronfa Gymdeithasol Ewrop European Social Fund
Prifysgol Abertawe Swansea University
Cardiff University Prifysgol Caerdydd
Prifysgol Bangor University
Cardiff Metropolitan University | Prifysgol Metropolitan Caerdydd
i.t.wales
Prifysgol Aberystwyth University
Prifysgol Glyndŵr Wrecsam | Wrexham Glyndŵr University
University of South Wales Prifysgol De Cymru

Workshop Schedule

Introduction - 10 minutes

Decomposition & Abstraction - 1 hour

Flowcharts, Selection - 1 hour

Iteration, Subroutines - 1 hour 30 minutes

Sort and Search Algorithms - 40 minutes

Conclusion - 10 minutes

Post-Day Questionnaires - 10 minutes

Note: These are estimated times, these will vary between classes, schools etc. so times will need to be adjusted accordingly.

Total: 4 hours 40 minutes

Learning Outcomes

Attendee Prerequisites

1. Basic experience of Python programming.

Learning Outcomes

1. Improved knowledge of Decomposition and Abstraction.
2. Greater experience of designing, writing and using Algorithms.
3. Improved confidence in implementing Algorithms using Python.
4. Greater experience in applying algorithms to other STEM subjects.

Preparation

1. Ensure all computers have Python 3 installed and ready to be used.
2. Print out Algorithm workbooks, one for each student attending workshop.
3. Each student will need a pencil for drawing of flowcharts.

Session Plan Key

In this session plan we use the following colours to differentiate the types of activities:

- **Yellow - Explain.** Teachers should explain the slide/example to the class.
- **Green - Discuss.** Teachers should start an open discussion with the class to get them to feedback some answers/ideas.
- **Purple - Activity.** Students are expected to complete an activity whether it be in their workbooks or on the computer, followed by a discussion of their solutions.
- **Green - Introduction/Conclusion.** The introduction/conclusion is also colour coded green. Teachers should hand out materials in the introduction and conclude the day and collect materials at the end.

Introduction

Begin with introductions, and a brief explanation of the Technocamps programme, before handing out pre-day questionnaires to be filled out by the students and teacher.

Activity: What is an Algorithm?

What is an Algorithm? - Students to write their definition of an Algorithm in their workbooks.
Ask for answers from students to see what they think:

- What is an Algorithm?
- What is important to remember when writing an algorithm?
- Where do we use algorithms in everyday life?

What is an Algorithm?

Explain: Algorithms

An Algorithm is a set of simple instructions that are done in a specific order to solve a problem.
When writing an algorithm is it important to remember to keep the instructions simple, in the correct order, unambiguous and relevant to solving the problem at hand.

Activity: Making a Cup of Tea

Students should write down a list of instructions in their workbooks for making tea (the drink). Once they have had 5 minutes to do this ask the class to give you their instructions. Use the a mug, teabags, sugar, kettle, milk to follow their instructions as literally as you can. Try to demonstrate how important it is to give clear, properly ordered, unambiguous instructions and what can go wrong if you don't. Explain that computers are not particularly clever and will literally do what you tell them.

Activity: Define Algorithms

Students should fill out the definition of an algorithm in the space provided in their workbooks.

Activity: Guess Who

Guess Who game - slide has pictures of numerous celebrities; each student selects a celebrity and in pairs they take turns asking their partner a question about their chosen celebrity. First person to guess correctly wins.

Decomposition

Discuss: Guess Who

- How many questions were needed?
- Which questions were useful or not useful?
- Did you change your questioning? How?

Activity: What is Decomposition?

Students should fill out the definition of decomposition in the space provided in their workbooks.

Explain: Decomposition

Decomposition is the process of breaking down a complex problem or system into smaller parts that are more manageable and easier to understand.

Activity: Decomposition of a Game

Students should think about the following ideas and then write down in their booklets:

- Who are the characters?
- What is the world they live in?
- What can the characters do?
- Is it a single-player game or multiplayer game?

Activity: LEGO Building

1. Student A and Student B are given a matching set of LEGO bricks.
2. Tell Student A to build the most interesting thing they can with their bricks, without letting Student B see what they are building.
3. Make sure Student B cannot see what is being built.
4. Student A should give instructions to Student B explaining how to build what they have created.

Student B is not allowed to ask any questions.

Discuss: LEGO Building

- Did the creation look the same?
- Whose fault is it?
- What would have made the task easier?
- Who or what may experience the same problems as student B?
  Answer: a computer.
- What makes a good algorithm?

Outcome:
This activity is a great way of showing how important it is to give clear, simple instructions that provide enough detail, especially to a computer who cannot ask questions. It highlights the importance of decomposition when faced with a task which may seem complex at first, but can be achieved by breaking the problem down into smaller steps.

Activity: Get Artistic

Students are shown a detailed picture which they must try to draw to the best of their ability. They should only be given 1 minute to work on their drawings.

Discuss: Get Artistic Reflection

Has anyone drawn a perfect replica? How did students decide what to include?What were their reasons for including the things they did and their reasons for leaving out other details?

Explain: Abstraction

Abstraction is the process of removing unnecessary detail and simplifying. Abstraction is used to remove unnecessary detail from a real world situation and to model the simplified result in an algorithm or program.
Real world examples of abstraction include:

- **Driving a car**

Someone who builds a car needs to understand how each and every part works. On the other hand, the driver only needs to know hoe to operate the steering wheel, gearstick, brakes, clutch and accelerator. They do not need to know the inner workings of the engine or how things function behind the dashboard and under the bonnet.

- **Programming**

How does the print/square root function work in Python? We don't need to know, but maybe we'll find out one way of doing it later on!

- **Teaching**

Teachers using Abstraction:
Show the slide with the animal cell (at GCSE level) and ask the students if they can identify the parts of the cell. Then point out that this is a simplified view of a cell to make it easier for GCSE students to understand. Then show the A-level view of an animal cell to show that they'll enjoy learning about if they continue to study it!

Do the same with the GCSE picture of an atom, and then the A-level view of an atom. This highlights the ways teachers and educators abstract detail in order to teach at appropriate levels of difficult.

Activity: What is Abstraction?

Students should write down the definition of Abstraction in their workbook.

Explain: Flowchart Conventions

Start with explaining the different types of shapes used in flowcharts. Show the different examples on the slides showing the flowchart templates and then an example of an actual flowchart which follows the template.

Explain: Sequence

Explain how a sequence is an action, or event, that leads to the next ordered action in a predetermined order. Talk about the making toast flowchart.

Activity: Making a Cup of Coffee

Students have a blank flow chart and all the instructions they need in their workbook. They should fill in the flowcharts with the instructions in the correct order.

Explain: Selection

Explain selection and show the getting ready flowchart. Explain the selection or choice required for this flowchart and how the outcome of the selection or choice changes the results or the next action in the flowchart.

Explain: Selection Flowchart

Show the flowchart that differentiates between three celebrities (The Queen, Stephen Hawking and Donald Trump). Point out that the different conditions or yes/no questions can differentiate between the three celebrities. If someone was attempting to guess the correct celebrity, they could ask the two questions in order to do so.

Activity: Vertebrates Flowchart

Students are given the 5 classes of vertebrates (Mammals, Birds, Reptiles, Amphibians and Fish) and are asked to make their own flowchart which differentiates between them with conditions i.e.

- "Do their offspring drink Milk?"
- "Do they lay eggs?"
- "Do they have feathers?"
- "Do they have limbs?"
- "Do they live solely on land?" etc.

Time will likely be needed to research this - Explain there may be animals who are exceptions to the general 'rules' of classification. e.g. a Duck-billed Platypus is a mammal but also lays eggs. We can ignore these exceptions.

Discuss: Vertebrates Flowchart

- What questions did the students end up using?
- Was it hard to find true/false questions?

Explain: Iteration

Explain the concept of iteration and illustrate this concept using the simple traffic light system. Show them the flowchart example of the simple traffic light system.

Discuss: Simple Traffic Lights

- What instructions would you use for this process?
- What needs to be iterated?
- How could you show that in a flowchart?
- Does this process ever end?

Activity: Login System Flowchart

Students are required to create a flowchart for a program which asks a user for a password. If the password is incorrect it says **“Password incorrect. Please Try Again.”** and returns to the original message. Otherwise it will say **“Password accepted. Welcome!”** and end the program. Their workbook contains a list of all the instructions and flowchart shapes they will need.

Subroutines, Pizza & Login

Activity: Login System in Python

Students should take the flowchart they created in the previous activity and use it to write a python program that follows the same brief. Allow someone to enter a password, if the password is incorrect print "Password incorrect. Please Try Again." and gives the user another chance to enter the password. Otherwise it will say "Password accepted. Welcome!" And end the program.

Solution at end of session plan.

Explain: Subroutines

Explain subroutines, return to flowchart example with cubeVolume flowchart.
Students are shown the Python cubeVolume program to see how subroutines work when implemented. (Program can be found in the resources link.)

Activity: Pizza Flowchart

Students are to create a flowchart on cooking pizza or chips in the oven. Students are to use all the symbols introduced so far.

Extension Task: Students are to add a subroutine symbol to their flowchart. An example of a subroutine they could add to the cooking pizza or chips problem is a timer subroutine or a preheating subroutine.

Compound Interest

Discuss: Flowchart Shape Recap

Discuss the flowchart symbols with the students ensuring they remember each shape and usage.

Discuss: Simple vs. Compound Interest

Talk about how a bank will pay you interest on most money you have deposited with them. They use your money for various things and the interest is essential a payment to you for the use of your money. Explain what **simple interest** is when compared to **compound interest.** Ask the students if they would prefer 10% interest every year for 10 years or 100% interest after 10 years. Start talking through how to calculate how much 10% every year for 10 years.

Explain: GCSE Compound Interest

Compound Interest: This is the process of calculating how much a given sum has increased over a period of time due to a particular interest rate being applied at certain intervals.
Go through the slides step by step explaining the process of answering the compound interest question, noting the use of abstraction to focus on the important detail and decomposition to break down the problem into individual steps. Once the final amount is reached, ask did Carys have enough money to buy the bike? (No.)
Show a flowchart which would work for any question of this format.

Activity: Compound Interest

Get students to calculate how much money they would have if they got 10% interest each year for 10 years on an initial balance of £1000. They have a table in their workbooks that they can use to calculate each years balance.

Activity: Compound Interest in Python

Students are required to implement the solution in Python so that given any inputs doe the starting amount, number of years, cost of the bike and interest rates, the program will output whether or not the bike can be afforded. They'll need to think about:

- The inputs needed and what type they are
- Which steps are repeated and how many times
- Where selection is needed

Solution at end of session plan.

Discuss: Compound Interest Solution

Ask the students if there is an easier way to implement this solution, assuming they used a loop and iteration to solve the problem. Is there an easier way to do it? (Yes there is)

Explain: Compound Interest Equation

Run through the slides which explain how to arrive at the final equation for working out Compound Interest:

$$T = A(1 + R)^n$$

Where T is the **total** amount, A is the initial **amount**, R is the decimal format of the interest **rate** and n is the number of years.

Explain: Bubble Sort

Bubble sort is an algorithm that repeatedly steps through a list to be sorted. The algorithm compares each pair of adjacent items and swaps them if they are in the wrong order. The passes through the list are repeated until no swaps are needed and the list is sorted. Big numbers **bubble up** through the list of other numbers to the end of the list like bubbles in a glass of water.

Explain: Bubble Sort (continued)

Lets us say that our unsorted list is [6 5 3 1 8 7 2 4] and we want to sort them into the smallest to the largest, we look at the first element in our list which is '6', we then check the next adjacent element which is '5' and we sort and order these two elements. Our list now becomes [5 **6** 3 1 8 7 2 4]. Next we compare '6' and '3' then sort. The list becomes [5 3 **6** 1 8 7 2 4]. This process is created giving the following list at each stage:

[5 3 1 **6** 8 7 2 4]
[5 3 1 6 **8** 7 2 4]

Noted there that nothing has changed because we are comparing '6' and '8', but we now know that '6' is less than '8' and therefore stays where it is for now.

[5 3 1 6 7 **8** 2 4]
[5 3 1 6 7 2 **8** 4]
[5 3 1 6 7 2 4 **8**]

As you can see here we are at the end of the list and have moved and sorted the '8' successfully. We then go back to the first of element of the list which is '5' and iterate the process until the list is fully sorted.

Activity: Bubble Sort

Students are asked to write down a brief description of Bubble Sort and the steps for completing the Bubble sort process in their own words. They are then provided with 8 cards numbered 1 to 8 and asked to sort the numbers using the algorithm.

Solution: 1,2,3,4,5,6,7,8

Explain: Merge Sort

Merge sort is a “divide and conquer” algorithm where: An unsorted list (let’s say **[6 5 3 1 8 7 2 4]** is our unsorted list) is divided into n individual sublists, (in this case 8 individual sublists) each containing 1 element. The list now becomes:

[6], [5], [3], [1], [8], [7], [2], [4]

The sublists are repeatedly merged to produce new sorted sublists until there is only 1 sorted sublist remains. Our sublists now become:

[5 6], [1 3], [7 8], [2 4]

This is because as we merge the individual sublists into pairs we also put them in order (smallest to largest).
This process is repeated and out sublists becomes:

[1 3 5 6], [2 4 7 8]

Again the sorting occurs during the merge process. As you can see the pair sublists **[5 6]** and **[1 3]** are sorted and merged into **[1 3 5 6]**. **1** is moved to then **3**, then **5** and lastly **6**.
We then finally merge the sublists into one sorted list with every element in the correct place:

[1 2 3 4 5 6 7 8]

Activity: Merge Sort

Students are asked to write down the steps for completing the Merge sort process in their own words. They are then provided with 8 cards numbered 1 to 8 and asked to sort the numbers using the algorithm.

Search Algorithms

Explain: Linear Search

A linear search is a simple search process where a list is searched form the first element to the last until the required value is found.
Show the step-by-step slides on how a linear search is performed.
Example: Looking for a students book in an unordered pile.

Activity: Linear Search

Students are asked to write down the steps for completing the Linear search process in their own words.
They are then asked to perform a linear search to find the number '42' from a list. They are then expected to either write down the steps or create a flowchart to show how the search was performed.

Explain: Binary Search

A binary search algorithm is a "divide and conquer" algorithm where:

- The middle value in a **sorted** list is inspected to see if it matches the search value.
- If the middle value is greater than the search value, the upper half of the list is discarded. If it is less than the search value, the lower half is discarded.
- This process is repeated with the list halving in size each time until the search value is found.

Show the step-by-step slides on how a linear search is performed.

Activity: Binary Search

Students are asked to write down the steps for completing binary search process in their own words.
They are then asked to perform a binary search to find the number '35' from a list. They are then expected to either write down the steps or create a flowchart to show how the search was performed.

Explain: Square Root

Remind the students what a square root is ie what number multiplied by itself makes this number. The square root of 25 is 5 because 5 * 5 = 25.

If someone gives us a number, how do we calculate its square root?
As a person you might try a bunch of numbers The Newton-Raphson method is a way of finding successively better approximations to the square root.

The equation is in the slides and must be applied iteratively to improved the estimation of the square root. We must provide an initial "guess" that will be refined (in the slides this is 3) and the number we wish to find the square root of. We take the value produced and feed it back into the equation to improve the estimate.

When do we stop?
When the value stops changing it has converged upon a solution. Be aware that some square roots will not converge with this method so you may have to set a maximum number of iterations.

Extension Activity: Square Root

Students should try to draw the flowchart for the Newton-Raphson method. They should make sure their algorithm will stop either when it has found the solution or when after a finite number of iterations.
Students who finish the flowchart use it to attempt to write a Python program that works out square roots.

Discuss: Computational Complexity

Discuss computational complexity of sorting and search algorithms.

What is computational complexity?
An estimate of the resources required by an algorithm. We are normally interested in how the amount of time required for an algorithm to finish (time complexity) changes as the input size changes. However, sometimes we are interested in the amount space required by the algorithm (space complexity).

Bubble sort and Merge Sort
The average time complexity of Bubble Sort is **n^2** whereas for Merge sort it is **n*log(n)**. They don't need to understand why this is the case, just that different algorithms have different time complexities and that they are important in determining which algorithm to use when.

If we needed to sort 1000 items then (on average) the amount of time taken would be:
Bubble sort: 1000^2 = 1,000,000
Merge Sort: 1000 * log2(1000) = 9966 (100 times faster than bubble sort)

Activity: Python Login Solution

```
def main():
        secretPassword = 'p4ssw0rd'
        passwordCorrect = False

        while not passwordCorrect:
                enteredPassword = input('Please enter your password: ')
                if enteredPassword == secretPassword:
                        print("Password accepted. Welcome!")
                        passwordCorrect = True
                else:
                        print("Password incorrect. Please try again.")
main()
```

Activity: Python Compound Interest Solution

```
def calculateNewTotalAmount(money, interestRate):
        interestGained = money * interestRate;

        return money+ interestGained

def main():
        money = float(input('How much money do you have?: £'))
        numberOfYears = int(input('Please enter the number of years: '))
        interestRate = float(input('Please enter the interest rate: '))
        costOfBike = float(input('How much does the Bike cost?: '))

        count = 0

        while count != numberOfYears:
                money = calculateNewTotalAmount(money, interestRate)
                count = count + 1

        if money >= costOfBike:
                print('You can buy the bike!')
        else:
                print('You can\'t buy the bike.')
main()
```

technocamps

@Technocamps

Find us on Facebook