



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

**Introduction to Computer Programming
(ACCA4020) - Assignment**

Alex Walton

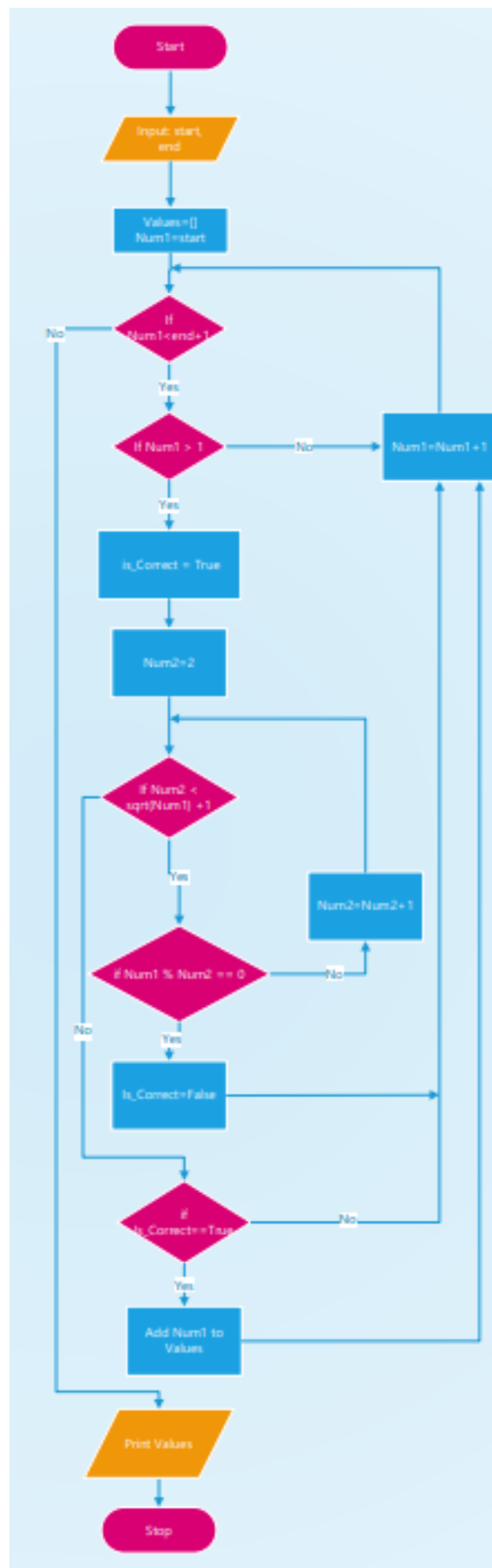
(Ba) Games Design w/ Foundation Year
University Of Wales Trinity Saint David

29/04/2025

Table of Contents

Table of Contents.....	
Flowchart.....	3
1. Question 1.....	4 - 5
1.a Describe the purpose of the flowchart. What is it designed to address.....	4
1.b Identify the number of loops used in the flowchart and the role of each loop.....	5
2. Question 2.....	6 - 7
2.a Explain how inner loop termination effects outer loops.....	6
2.b Identify the Boolean variable in the flowchart and the role of Boolean.....	7
3. Question 3.....	8 - 9
3.a Explain the purpose of a conditional statement in programming.....	8
3.b Explain each of the above three conditional statements.....	9
4. Question 4.....	10
4.a Write a program that implements logic outlined in the flowchart.....	10
4.b Test the program and display results.....	10
5. Question 5.....	11
6. Question 6.....	12
7. Question 7.....	13
8. References.....	14

Flowchart



1. Question 1

1.a Describe the purpose of the flowchart. What problem or task is the flowchart designed to address?

According to a guide provided online by Team Asana (1), a flowchart can be described as “a diagram that illustrates the steps, sequences, and decisions of a process or workflow”. What this means is that the purpose of a flowchart tool is to abstract a complex set of information into smaller more understandable pieces enabling its creator(s) to efficiently and easily provide explanations to non-professionals or otherwise bring others up to speed in a fraction of the time it would normally take.

The flowchart tool is an adaptable application designed to counter a variety of everyday problems. The Team Asana guide names various instances where the tool could be used, three of these instances are and are not limited to:

“Track progress”: With a flowchart, resources are saved as various symbols are utilised to document where a workflow process is currently stationed allowing the viewer to quickly decide whether overall progress is ahead or behind of schedule therefore improving efficiency as resources and time can be allocated where necessary.

“Identify bottlenecks or issues”: An efficient error checking method due to visualisation, flowcharts can be used to rapidly determine where errors or otherwise unexpected results may possibly occur enabling the industry professional to implement the correct solutions before the errors could happen saving time, money and resources

“Standardise processes”: A single flowchart can be created to provide an optimal solution to a task or problem allowing others to make use of it leading to the standardisation of processes in various applications further increasing workflow efficiency and reduction in resource usage. Further on in the guide, it states that there are various flowchart types, including but not limited to:

“Decision flowchart”: A decision flowchart “outlines the different paths and outcome based on specific conditions or criteria” meaning that this flowchart type will highlight various scenarios and therefore human decisions can be made where necessary.

“Process flowchart”: “Provides a step by step representation of a process” effectively breaking down and explaining what would happen in a given scenario.

“Program flowchart”: “Represents the sequence of steps and decisions in a computer program or algorithm”. A hybrid of the two above mentioned flowchart types, a program flowchart makes use of both process sequences and decision making to perform a set task within given parameters.

1.b Identify the number of loops used in the flowchart and explain the role of each loop.

Using the flowchart attached in pg. 3. A total of 6 loops can be identified within the program which will execute under different conditions but will often execute the same process (as 4 of these loops will lead to executing the process $\text{Num1} = \text{Num1} + 1$) to begin executing the whole program again until termination conditions are met. Below explains the role of each loop:

Loop 1: The first loop also serving as the outer loop, will check if Num1 is smaller than the user inputted end value + 1, if Num1 is determined to be larger than the provided value the program will execute a full termination condition. However if Num1 is smaller then the program will progress to the following loops below.

Loop 2: Loop 2 will execute if the decision making determines that the variable Num1 is not greater than 1, in this event it will switch to $\text{Num1} = \text{Num1} + 1$ and start the program over again repeating until Num1 becomes greater than 1.

Loop 3: In the 4th decision making process of the flowchart, if the program determines that the remainder of the division of Num1 and Num2 does not equal to 0 then a loop will execute causing Num2 to increase by 1 each loop iteration until the remainder of the division becomes 0.

Loop 4: Loop 4 shares the same decision making branch as loop 2. While the previous loop executes under a no condition loop 3 will execute when the remainder is 0 leading to $\text{Num1} = \text{Num1} + 1$ thus creating a new iteration of the whole program.

Loop 5: Loops 3 and 4 are nested inside of loop 5, as highlighted where the 3rd decision making branch of the program, if the variable Num2 is not smaller than the square root of $\text{Num1} + 1$ then a Boolean will execute to check if the statement is true and if it isn't will return to $\text{Num1} = \text{Num1} + 1$ and start a new program iteration.

Loop 6: The yes value of the Boolean executed by loop 5, this loop will rewrite the value of the variable Values with Num1 before returning to $\text{Num1} = \text{Num1} + 1$ and creating a new program iteration.

The above 6 loops will continue producing program iterations until the termination condition in loop 1 (Num1 is smaller than the inputted end value + 1) is met, thus exiting the loops and the program completely.

2. Question 2

2.a Explain how the termination conditions of the inner loop affect the outer loop's processing. What happens when the inner loop exits early?

The inner and outer loop will often execute together but in different stages as the inner loop will execute specific sets of instructions in several times until a set condition is met leading to the outer loop executing and restarting the inner loop across numerous iterations until the outer loop termination conditions are met.

In the context of the attached flowchart, the inner loops affect the outer loop by slowly increasing the value of Num1 by executing various instructions, for example checking if the division remainder is 0, as previously mentioned in 1.b. These iterations will each reach their own termination conditions where they will 'break' allowing the outer loop to begin executing 'if Num1 < end + 1' instruction set and will exit the program or restart the inner loops dependant on the Boolean of this code.

Break, is a command in which "as soon as Python encounters this statement, it immediately exits the loop, regardless of whether the loop has fully completed its iterations or not." as stated in the basic use section of a blog provided by ioflood.com (3).

With the definition of the break condition in mind, should the inner loops of the flowchart exit early the Num1 = Num1 + 1 loop can execute early saving both time and resources as the program doesn't need to fully check an iteration and a new one can begin straight away to check for any changes in variables leading to different termination states across the iterations.

2.b Identify the Boolean variable used in the flowchart and what is the role of this Boolean variable?

Referring back to the attached flowchart, the Boolean variable of the program is identified by the name "is_Correct". But what the definition of a Boolean. In programming, a Boolean is described by BBC Bitesize (2) as "a form of algebra where all values are either true or false". This simple logic tool is widely used in basic comparisons leading to various termination conditions as a result. The role of the Boolean variable in the attached flowchart is to determine if various processed calculations inside of the inner loops match the programmed value. In the event that the Boolean is false then the inner loops will exit and initiate outer loop processing. The Boolean variable is used throughout the inner loops of the attached flowchart and is the variable that will determine if the loop should exit early meaning that Boolean is a variable that will save resources, a concept briefly mentioned at the end of 2.a.

3. Question 3

3.a Explain The Purpose of a Conditional Statement In Programming

According to CLRN (4), a conditional statement is a “fundamental concept that allows developers to make decisions within a program”. Just like the role of Boolean discussed in 2.b, conditional statements are also true / false instruction sets executing different code sets under varying criteria. CLRN breaks down the conditional statement into three main components: “Condition, Boolean, Action”. Unlike Boolean which is only a variable the conditional statement will launch full instructions sets after condition evaluation and can branch into more instruction sets than the limiting two sets provided by Boolean.

The guide explains that the conditional statement can be expressed as:

If-else: “Single condition”. The more Boolean format, the if-else code is utilised to split into two different instruction sets dependant on a single defined or inputted condition.

If-elif-else: “Multiple condition”. More advanced than if-else, this statement is utilised to branch into three or more instruction sets as require based on two or more conditions being checked by this statement.

3.b Explain each of the above three conditional statements and discuss how these conditional statements contribute to the final results of the program.

- If $\text{Num1} > 1$: This single condition conditional statement serves as the first loop of the nested program loops, as previously mentioned it will check if Num1 is greater than the value of 1. It contributes to the final program results as it will save time and resources by switching to the outer loop early on if the condition is not met however if it is met then the program will proceed as instructed.
- if $\text{Num1} \% \text{Num2} == 0$: This statement, is nested inside of a inner loop and checks of there is a division remainder that matches the value of 0, if not the program will increase Num2 by 1 each time until the 0 remainder is achieved in which the is_Correct Boolean executes leading to outer loop processing.
- if $\text{Is_Correct} == \text{True}$: In the event that Num2 is not smaller than the square root of $\text{Num1} + 1$ then $\text{is_Correct} == \text{True}$ is used to determine whether this is true or not, if the condition is true then the Boolean will rewrite the values variable as Num1 before entering the outer loop and restarting the whole program.

4. Question 4

4.a Write a Python program named, `generate_special_numbers` that implements the logic described in the flowchart.

```
Start = int(input("Enter a value: \n"))
End = int(input("Enter a value: \n"))
Values = []
Num1 = Start

while Num1 <= End:
    if Num1 > 1:
        is_Correct = True
        Num2 = 2
        while Num2 < Num1:
            if Num1 % Num2 == 0:
                is_Correct = False
                break
            Num2 = Num2 + 1
        if is_Correct == True:
            Values.append(Num1)
        Num1 = Num1 + 1

print(Values)
```

As requested by 4.a, the provided program attached on the left has been produced in line with required criteria. The start and end values are designed to be user inputted, asking the user to do so with "Enter a value:" which provides the operating parameters for the program to cycle through until all values are identified in the given range. The outer loop `Num1 <= End` will continually run iterations of the inner loops to identify prime numbers through division and Boolean and will exit inner loop functions early if conditions have been met allowing the program to save resources. All values listed throughout the iterations will print at the end of the full program as shown in 4.b

4.b Test the `generate_special_numbers` program with the following inputs and display the results.

1. Start = 3, end = 16

The above program has printed a series of prime numbers successfully identified within the defined range and clearly highlights this within the square brackets as shown in img. 1.

```
Enter a value:
3
Enter a value:
16
[3, 5, 7, 11, 13]
```

img.1

2. Start = 25, end = 42

Using a different range the program has changed its operations and again provided a series of prime values within the defined inputted range highlighting that the program can perform the required task with a variety of ranges.

```
Enter a value:
25
Enter a value:
42
[29, 31, 37, 41]
```

img. 2

5. Question 5

Test cases

5a. Test case 1: start = 12 and end = 22

```
Enter a value:
12
Enter a value:
22
[13, 17, 19]
```

Just like the first two test cases in 4.b the program has rapidly identified prime numbers within the inputted range and saving resources because it has identified every prime number and exited the program early before checking any extra numbers.

5b. Test case 2 : start = 30 and end = 8

```
Enter a value:
30
Enter a value:
8
[]
```

The program isn't instructed to check start and end values in a reverse order therefore the program isn't able to identify any prime numbers leading no values being printed.

5c. Test case 3 : start = - 11 and end = 9

```
Enter a value:
-11
Enter a value:
9
[2, 3, 5, 7]
```

In maths logic, a negative number that is either multiplied or divided by another negative value will result in a positive value. Due to the range provided and the calculations conducted inside the program no negative numbers can be produced therefore the program is only able to print prime numbers that are positive in the given range.

5d. Test case 4 : start = 24 and end = 27

```
Enter a value:
24
Enter a value:
27
[]
```

In this test case, the program cannot detect a prime number within the provided range and therefore the program will exit at the earliest opportunity and therefore will print a value of nothing.

5e. Test case 5 : start = 13 and end = 13

```
Enter a value:
13
Enter a value:
13
[13]
```

As the provided start and end values are the same the program will cycle only once to check this number and as 13 is identified as a prime number the program will proceed to print 13 as the only value since 13 only number it can check.

6. Question 6

Identify any optimization techniques used in the given program, generate_special_numbers. Explain how these optimizations affect the program's performance, particularly in terms of time complexity. Include an example to illustrate the impact of the optimization.

Various optimisation tools could be used in order to improve the performance of the program, especially in the way it is coded. The first optimisation is method is to utilise a reduced number of variables leading to enhanced efficiency at runtime as unnecessary variables are removed saving on runtime and resource usage. A reduction in the number of loops used can also enhance efficiency as the program can check values more directly and execute the same task in a shorter time frame while using less loops. To implement these changes, it is first important to identify where they can be substituted.

The while loop can be substituted for a range loop replacing a constant runtime iteration of increments of 1 for a direct start to end range check skipping iterations. The optimisation techniques can also be used to assume that the number is prime until it is proven otherwise through a fast division calculation rather than cycling through various calculations and loops.

By consolidating multiple loops until a few and with the reduction of variables used the program is now able to perform the same task as instructed while taking up half the space and saving both time and resources.

Below is an optimised version of the assignment program that is based on the original coding produced with guidance from generative AI.

```
start = int(input("Enter a start value: \n"))
end = int(input("Enter an end value: \n"))
primes = []

for num in range(start, end + 1):
    if num > 1:
        is_prime = True
        for divisor in range(2, int(num ** 0.5) + 1):
            if num % divisor == 0:
                is_prime = False
                break
        if is_prime:
            primes.append(num)
print(primes)
```

7. Question 7

Reflect on your learning journey in this module. What specific programming concepts did you find most challenging, and how did you overcome these challenges?

Throughout the module I've found optimisation to be one of the hardest concepts to learn and understand because especially in python, it can be quite difficult to identify what sections of code can be altered to provide the same results while also saving on resources and without causing other sections of a program to have errors.

To overcome this, I have chosen to utilise generative AI to assist in the syntax checking and optimisation of the program in small increments. With some of the results being shown in Q6.

Furthermore, I have found difficulties in understanding how to create different types of loops, especially making loops that will return to the beginning without a clear indication of such. To understand this concept I have researched into how to initiate the described problem and have discovered that utilising the while loop will enable me to create loop iterations until the while loop termination has been achieved.

I have also had troubles on how to properly program syntax logic as key concepts I have learned such as conditional statements and use of Boolean lead to several errors both in syntax and at runtime in which case I have also utilised generative AI to syntax check each error and provide an explanation how to fix them.

Finally, I have had a logic error in which the program will operate as instructed however it would not print any prime numbers leading to confusion on how to ensure that the program will utilise the value variable. I later discovered that while the syntax was correct the way I implemented the coding did not allow for values to be rewritten therefore with further assistance from generative AI a small change was introduced enabling values to correctly store prime numbers.

8. References

1. Flowchart 101: Symbols, types and how to create them. (2024, May 17).
<https://asana.com/resources/what-is-a-flowchart>
2. <https://www.bbc.co.uk/bitesize/guides/zqp9kqt/revision/1>
3. Gabriel Ramuglia. (2024, June 6). Python break: Guide to Controlling Loop Execution.
<https://ioflood.com/blog/python-break/>
4. What is a conditional statement in coding? (2024, November 8). CLRN.
<https://www.clrn.org/what-is-a-conditional-statement-in-coding/>