**Pearson Higher Nationals in**

**Computing**

Unit 01: Programming

Assignment 01

**SH. Mohamed Sihan**

**E204387**

**ESOFT**

Acknowledgement

Table of Contents

Acknowledgement i

Table of Contents ii

**List of figures**

**List of tables**

Introduction

1. Activity - 01
   1. Algorithm

An algorithm is a process for executing a calculation or problem-solving. Algorithms function as a precise set of instructions that carry out predetermined operations in a hardware- or software-based procedure(Gillis, 2023)

Here showing two types of algorithms

1. Pseudocode

The description of an algorithm or computer program's intended behavior that is both comprehensive and accessible is called pseudocode. To make it easy to understand by programmers and other developers, it is written in a formal but readable language with natural grammar and structure (Sheldon, 2023)

1. Flow charts
   * 1. Characteristics of a good algorithm

According to Bhatt (2019),

1. **Input** - The data that needs to be changed for the computation to get the desired result is the input. A well-defined algorithm should contain zero or more inputs. To ensure input precision, you must be aware of the kind, quantity, and format of the data.
2. **Output** - The data that comes out of the computation is the output (the desired outcome). An algorithm should produce one or more clearly specified outputs that correspond to the intended result. Knowing the type of data, quantity, and format of the output or even whether any at all is also necessary for output precision.
3. **Finiteness** - At some point, the algorithm has to stop. Stopping could result in the desired result OR a response indicating that is no way to solve the problem. Algorithms have a limited number of steps and must end. An algorithm must always end after a predetermined number of steps and not be limitless.
4. **Independent** - Step-by-step instructions that don't depend on programming code should be part of an algorithm. It ought to be designed so that any programming language might use it.

According to RishabhPrabhu (2023),

1. **Feasibility** - for the algorithm to be implemented using the resources at hand, it needs to be straightforward, universal, and workable. It can't have any futuristic technology in it at all.
2. **Unambiguous** - There should be no doubt in the algorithm. Its steps must all lead to the same meaning and be obvious.
   * 1. Algorithm for Fibonacci series

* **Pseudocode for Fibonacci series**

Begin

Initialize the list 0,1 in Fibonacci\_Numbers

Read the input integer in n to enter the Fibonacci number.

Set Fn to 2

If n less than or equal to 0:

Then display The Fibonacci series of (n) : [ ]

Else if the n equal to 1 :

Then show the factorial value of (n) : [0]

Else :

While Fn less than to n

next\_number = Fibonacci\_Numbers [-1] + Fibonacci\_Numbers [-2]

Get the next\_number into list of Fibonacci\_Number

Increment Fn by 1

Display the list of Fibonacci\_Number

End.

In this pseudocode...

First, initialize [0, 1] as Fibonacci\_Number.

Get the input into n.

After set a variable Fn to 2,

Next, the conditional statement (if n is less than or equal to 0) checks whether the input n is less than or equal to 0. If the statement is true, show a box bracket without characters like []. If the statement is false, check that input n is equal to 1. If it is, the output is [0]. If it is not 1, the loop condition is checked.

In the loop (while) condition…

Checks if Fn is less than input n. If Fn is low, the loop starts. First in the loop (next\_number = Fibonacci\_Numbers [-1] + Fibonacci\_Numbers [-2]) will get the last digit of Fibonacci\_Numbers (list) and the Fibonacci\_Numbers digit before the last digit. Then, it adds the two digits and sets the output to next\_number. After that, it adds the next\_number to Fibonacci\_Numbers. Next, Fn is incremented by one. Then, Fn checks again if the input is less than n.

When the (Fn is less than input n) condition is false, the Fibonacci\_Numbers will display with the included other digits. If pass the input as 2, based on the coding, the system will display [0, 1] because the while loop will check the input (Fn less than n) and the inputted 2 is not less than Fn (set Fn to 2) because Fn is also 2, then 2 is not less than 2, so the system will display the Fibonacci\_Number and the Fibonacci\_Number is [0, 1]. (Initialize the list 0,1 in Fibonacci\_Numbers.).

* **Flow chart for Fibonacci series**
  + 1. Algorithm for Factorial value
* **Pseudocode for Factorial value**

Begin

Read the input integer in n to enter the value.

Create variable i and fac to 2 and 1.

If the n less than or equal to 0

Then show the factorial value of (n): 0.

Else

While i less than or equal to n:

fac multiply by i

i increment by 1

Display the factorial value of n : fac.

End

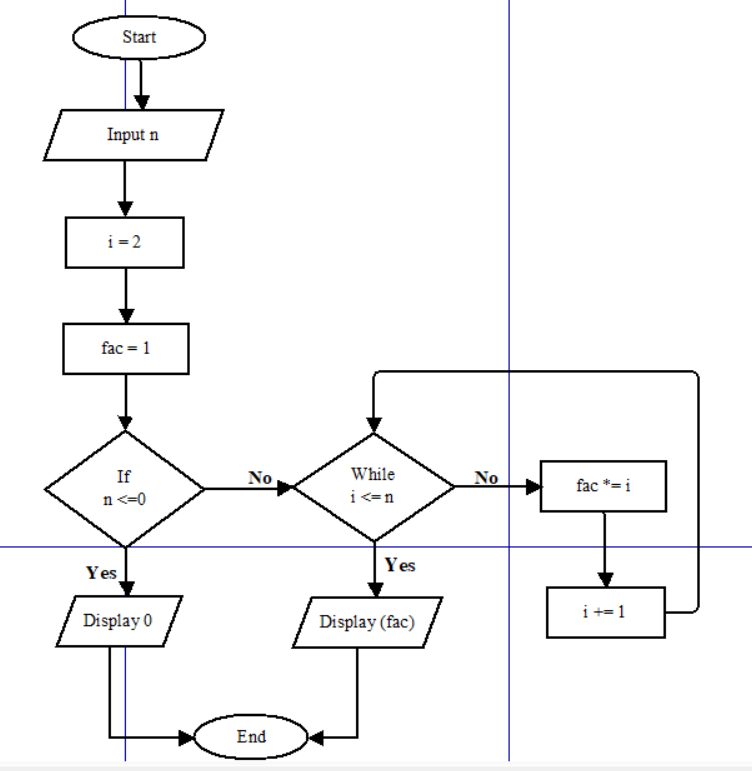
In this pseudocode...

First, get the input into n.

Next, create the variables for the calculation: I as two and fac as one.

After, check the input n is less than or equal to 0 using the if condition. If it is true, display 0; if it is false, the while loop condition checks if i is less than or equal to input n, and if i is less than or equal to n, the loop will start. Next, (fac \*= i) multiplies the current value of fac by i. This accumulates the factorial value. After increasing i by one, after again checking, the increased i is less than or equal to n.

When i is greater than n, the loop will stop and display the factorial value of the input number as fac (Display the factorial value of n as fac). If pass 1 as input, based on the code, it will display 1 because the while condition checks the input 1 (i less than or equal to n). the i is already 2 (according to Create variable i and fac to 2 and 1), so the input 1 is not less than or equal to 2(i), then display the fac. The fac is 1.

* **Flow chart for Factorial value**
  1. A diagram of software development

     Description automatically generatedSteps involved in the process of writing and executing a program.
  2. Potential challenges faced.
* Faced challenge – No ideas about the Fibonacci series and factorial value.

While the learner starts to code the Fibonacci series and factorial value, firstly, the learner doesn’t have any ideas about the Fibonacci series and factorial value. So, the learner learns it by searching for Fibonacci series and factorial value on YouTube and Google. After gaining knowledge about the Fibonacci series and factorial value, began writing the code.

* Faced challenge – How to write the code.

According to the programming assignment brief, write the code using the Python programming language. But the learner does not know how to write the code after writing an algorithm for the Fibonacci series and factorial value. After writing Python code for the algorithm in the diary, Later, the learner wrote the Python code in Thonny software.

**Fibonacci Series**

* Faced challenge – Initialize variables.

The learner decides to initialize Fibonacci\_Numbers = [0,1], and the [0, 1] list is number 2 of the Fibonacci series and must have a minimum of two digits in the list to calculate the Fibonacci series. The step of next\_number = Fibonacci\_Numbers[-1] + Fibonacci\_Numbers[-2] has two digits. After creates Fn = 2 because the [0,1] list is number 2 of the Fibonacci series.

After creates this n = int(input("Enter the number of Fibonacci: ")). The step is to get the input into n. It is important to calculate which number sequence it wants.

* Faced challenge – How to calculate the Fibonacci series.

Break down the problem into small steps and understand the logic of the algorithm to compute the Fibonacci sequence. (next\_number = Fibonacci\_Numbers [-1] + Fibonacci\_Numbers [-2]) The step of calculation was very difficult for the learner. Finally found out how to take the last two digits from the list for the Fibonacci calculation.

* Faced challenge – While Fn < n or while Fn <= n

If the learner use (while loop) Fn <= n and passes the input (n) as 2 already, the Fn is 2, so Fn is not less than 2, but Fn 2 is equal to 2, then based on the statement, this is displayed as [0, 1, 1]. but this is not Fibonacci sequence of 2. If the learner use (While loop) Fn < n this, Fn is 2 and n is 2 then 2 is not less than 2 so the loop does not start and display the {print("The Fibonacci series of " + str(n) + " : " + str(Fibonacci\_Numbers))} Fibonacci sequence. The Fibonacci sequence is [0, 1]. and this sequence is correct.

* Faced challenge – if pass the input as 0 or 1 or less than 0.

The Fibonacci series only gives the sequence to non-negative numbers, and if anyone passes the input as 0 or less than 0, the system wants to display something like ([ ]) this. So, the learner decided to use the if condition to solve the issue. First, the system checks if the input n is less than or equal to 0 (if n <= 0). If it is less than or equal to 0 {print ("The Fibonacci series of " + str(n)+ " : " + str([])} the system will display [ ].

* Faced challenge – if pass the input as 1.

Based on the Fibonacci series, the Fibonacci series of 1 is [ 0 ]. So, after checking the input is less than or equal to 0, and if it is not, after the system will check if the input is equal to 1, like elif n = = 1. If it is equal, the system displays [ 0 ] { print ("The Fibonacci series of " + str(n) + " : " + str([0])}.

**Factorial value**

* Faced challenge – Initialize variables.

The learner created n = int(input("Enter the value : ")) this statement to get the input number into n and it helps to calculate the factorial value of which number of values want.

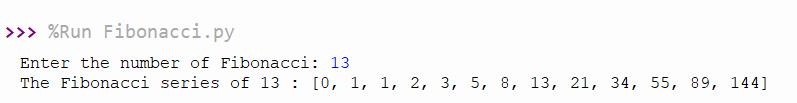
After created i is equal to 2 (i = 2) for checking the loop condition. Next, create fac = 1 to get the calculated values into the fac.

* Faced challenge – calculate the factorial value.

Using the fac \*= i step to calculate the factorial value here the system checks if i is less than or equal to input n if it is the system multiple the fac with i into fac (fac = fac \* i). after if i is not less than or equal to n the system displays the multiplied value of fac { print ("The factorial value of " + str(n) + " : " + str ( fac ) )}.

* Faced challenge – if pass the input as 0 or less than 0.

Based on the programming assignment brief, the factorial of a non-negative integer, is multiplication of all integers smaller than or equal to n. So, if anyone passes the input in negative numbers the system wants to show 0. Then the learner decided to use if condition to solve the issue. So, the system will check if the passed input is less than or equal to 0 (if n <= 0). If it is display 0 { print ("The factorial value of " + str(n) + " : " + str (0)) }.

* 1. Dry run table for created algorithms.
* **Fibonacci series**

Input value = 13 n = 13

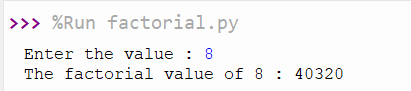
|  |  |  |  |
| --- | --- | --- | --- |
| **Loop** | **Fn** | **next\_number** | **Fibonacci\_Numbers** |
| Initial | 2 | - | [0, 1] |
| 1 | 3 | 1 | [0, 1, 1] |
| 2 | 4 | 2 | [0, 1, 1, 2] |
| 3 | 5 | 3 | [0, 1, 1, 2, 3] |
| 4 | 6 | 5 | [0, 1, 1, 2, 3, 5] |
| 5 | 7 | 8 | [0, 1, 1, 2, 3, 5, 8] |
| 6 | 8 | 13 | [0, 1, 1, 2, 3, 5, 8, 13] |
| 7 | 9 | 21 | [0, 1, 1, 2, 3, 5, 8, 13, 21] |
| 8 | 10 | 34 | [0, 1, 1, 2, 3, 5, 8, 13, 21, 34] |
| 9 | 11 | 55 | [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55] |
| 10 | 12 | 89 | [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] |
| 11 | 13 | 144 | [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,144] |

1. initialize the Fibonacci\_Numbers as [0, 1]
2. get the input into n n = 13
3. set Fn = 2
4. Check 13 < = 0, 13 is not less than or equal to 0, so it will go to the next step.
5. Here check the 13 is equal to 1. 13 is not equal to 1, it will go to the next step.
6. Here the while loop condition checks the 2 (Fn) is less than 13 (n). Yes 2 is less than 13 so the loop will start.
   * 1. Take the last two digits from the Fibonacci\_Numbers (0, 1) and add the total into next\_number (1).
     2. Add the total value (1) into Fibonacci\_Numbers list [0, 1, 1].
     3. Add 1 to Fn (2). So Fn is 3.
   1. Check the 3 (Fn) is less than 13 (n). Yes so,
      1. Take the last two digits from the Fibonacci\_Numbers (1, 1) and add the total into next\_number (2).
      2. Add the total value (2) into Fibonacci\_Numbers list [0, 1, 1, 2].
      3. Add 1 to Fn (3). So Fn is 4.
   2. Check the 4 (Fn) is less than 13 (n). Yes so,
      1. Take the last two digits from the Fibonacci\_Numbers (1, 2) and add the total into next\_number (3).
      2. Add the total value (3) into Fibonacci\_Numbers list [0, 1, 1, 2, 3].
      3. Add 1 to Fn (4). So Fn is 5.

The next four to eleven loops continue in the same pattern.

1. Check the 11 (Fn) is less than 13 (n). Yes so,
2. Take the last two digits from the Fibonacci\_Numbers (34, 55) and add the total into next\_number (89).
3. Add the total value (89) into Fibonacci\_Numbers list [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89].
4. Add 1 to Fn (11). So Fn is 12.
5. Check the 12 (Fn) is less than 13 (n). Yes so,
6. Take the last two digits from the Fibonacci\_Numbers (55, 89) and add the total into next\_number (144).
7. Add the total value (144) into Fibonacci\_Numbers list [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144].
8. Add 1 to Fn (12). So Fn is 13.
9. Here the while loop condition checks the 13 (Fn) is less than 13 (n). No 13 is not less than 13 so the loop will end.
10. Display the Fibonacci\_Numbers. The Fibonacci\_Numbers is [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144].

* **Factorial value**



Input value = 08 n = 08

|  |  |  |
| --- | --- | --- |
| **Loop** | **i** | **fac** |
|  | 2 | 1 |
| 1 | 3 | 2 |
| 2 | 4 | 6 |
| 3 | 5 | 24 |
| 4 | 6 | 120 |
| 5 | 7 | 720 |
| 6 | 8 | 5040 |
| 7 | 9 | 40320 |

* 1. Big-O notation
     1. Definition
     2. Big-O notation role in evaluating efficiencies of algorithms.
  2. Python program code for created both algorithms.
  3. Critically evaluate the created algorithm efficiencies using Big-O notation

1. Activity – 02
   1. Programming paradigm.
   2. Main characteristics of programming paradigm
   3. Main characteristics of object oriented.
   4. Main characteristics of event driven paradigm
   5. Relationship between the programming paradigms
   6. Code as example for the above three programming paradigms
   7. Critically evaluate the code samples that you have given above in relation to their structure and the unique characteristics.
2. Activity – 3
   1. Pseudo code for the salary component.
   2. Implement the above three components.
   3. Design a suitable database structure for keeping the data of the above system.
   4. Analyze the features of an Integrated Development Environment (IDE).
   5. How those features help in application development
   6. Evaluate the use of the visual studio IDE for your application development contrasted with not using an IDE (evaluate with using Grifindo)
3. Activity – 4
   1. GUI system for the above scenario
   2. Database structure
   3. Examine debugging process.
   4. The features available in visual studio IDE for debugging
   5. Use of Visual studio IDE for debugging features in developed system
   6. Evaluate how you used the debugging process to develop a more secure, robust application with examples.