**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**

No table of figures entries found.

**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 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.).

* A diagram of a flowchart

  Description automatically generated**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

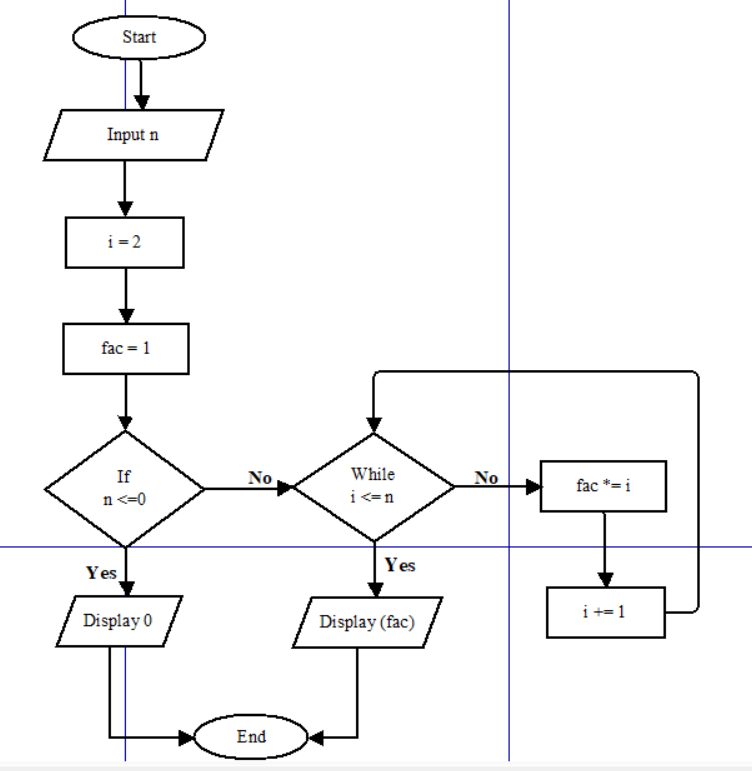
In this pseudocode...

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.

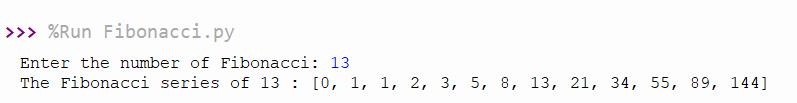
After created i is equal to 2 (i = 2) for checking the loop condition. Next, create fac = 1 to get the calculated values into the fac.

* 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 multiply 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

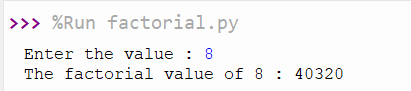
|  |  |  |  |
| --- | --- | --- | --- |
| **Loop** | **Fn** | **next\_number** | **Fibonacci\_Numbers** |
| Initial | 2 | - | [0, 1] |
| 1 | 3 | 1 | [0, 1, 1] |
| 2 | 4 | 2 | [0, 1, 1, 2] |
| 3 | 5 | 3 | [0, 1, 1, 2, 3] |
| 4 | 6 | 5 | [0, 1, 1, 2, 3, 5] |
| 5 | 7 | 8 | [0, 1, 1, 2, 3, 5, 8] |
| 6 | 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** | **n** | **i** | **fac** |
| Initial | 8 | 2 | 1 |
| 1 | 8 | 3 | 2 |
| 2 | 8 | 4 | 6 |
| 3 | 8 | 5 | 24 |
| 4 | 8 | 6 | 120 |
| 5 | 8 | 7 | 720 |
| 6 | 8 | 8 | 5040 |
| 7 | 8 | 9 | 40320 |

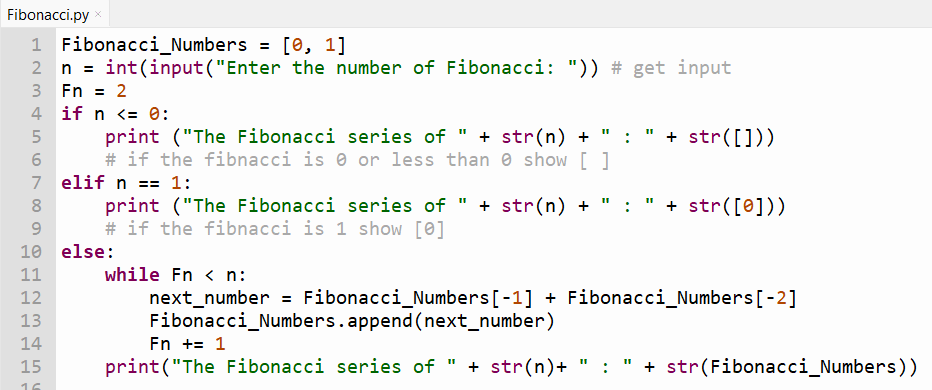
1. Get the input into n. the input is 8 so n = 8
2. Initialize i equal 2.
3. Set fac to 1.
4. The if condition checks if the 8 (n) is less than or equal to 0. 8 is not less than or equal to 0. So, it will go to the next step.
5. Into else, the while loop condition checks if the 1 (i) is less than or equal to 8 (n). Yes, it is. Then the loop will start.
6. * 1. 1 (fac) multiply with 2 (i) and replace the multiplied value (2) in fac. (fac = fac \* i)
     2. Increase i by 1. So, i is 3
7. Check the increased i (2) is less than or equal to 8. Yes, it is. So,
8. 2 (fac) multiply with 3 (i) and replace the multiplied value (6) in fac. (fac = fac \* i)
9. Increase i by 1. So, i is 4
10. Check the increased i (4) is less than or equal to 8. Yes, it is. So,
11. 6 (fac) multiply with 4 (i) and replace the multiplied value (24) in fac. (fac = fac \* i)
12. Increase i by 1. So, i is 5

The next four and five loops continue in the same pattern.

1. Check the increased i (7) is less than or equal to 8. Yes, it is. So,
2. 720 (fac) multiply with 7 (i) and replace the multiplied value (5040) in fac. (fac = fac \* i)
3. Increase i by 1. So, i is 8
4. Check the increased i (8) is less than or equal to 8. Yes, it is equal to 8. So,
5. 5040 (fac) multiply with 8 (i) and replace the multiplied value (40320) in fac. (fac = fac \* i)
6. Increase i by 1. So, i is 9
7. Here checks the increased i (9) is less than or equal to input 8 (n). No, 9 is not less than or equal to 8. So, the loop will end. And it will go to the next step.
8. Display the fac value. The value is 40320.
   1. Big-O notation

Big O Notation is a vital tool in computer science that is used to determine an algorithm's time efficiency. Programmers can categorize algorithms using Big O Notation based on how their space requirements or run times change with an algorithm's input size (Simplilearn, 2023)

* + 1. Big-O notation role in evaluating efficiencies of algorithms.

* 1. Python program code for created both algorithms.
* **Fibonacci series.**

In the Fibonacci calculation…

Get the input into n and after check if the input is less than or equal to 0 (if n < = 0). If it is display [ ]. If it is not, the elif condition will check if the input is equal to I (elif n = = 1). if it is, the system display [0], if it is not the else condition check the while loop condition.

The loop condition checks if the Fn is less than input (while Fn < n), if the input is 2 the system will display [0, 1] because the Fn is 2 and n (input) is also 2 and 2 is not less than 2 so the system will display Fibonacci\_Numbers list. The list is [0, 1] (Fibonacci\_Numbers = [0, 1]).

The loop will begin if the Fn is less than the input. and until the Fn is equal to the input, the loop will keep going. In the loop, first step is “next\_number = Fibonacci\_Numbers[-1] + Fibonacci\_Numbers[-2]” and here the system takes the total of last two digits from the Fibonacci\_Numbers list into next\_number (variable).

After the following step “Fibonacci\_Numbers.append(next\_number)” the system stores the total value (next\_number) into Fibonacci\_Numbers list. Next, increase the Fn by 1. Again, the loop checks the increased Fn and continues the loop until the Fn is equal to the input.

**Comparison between the Fibonacci series, pseudocode, and the python code.**

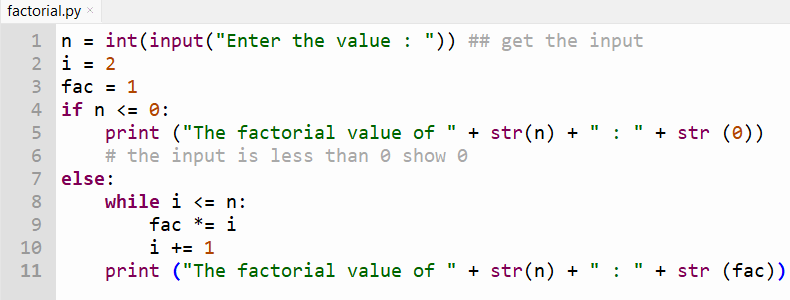
The Python code is built based on the pseudocode. According to the pseudocode, want to “initialize the list 0,1 in Fibonacci\_Numbers”. So, the learner initialized that in python language. “Fibonacci\_Numbers = [0, 1]”. After want to get the input into n (Read the input integer in n to enter the Fibonacci number) so the learner wrote “n = int(input("Enter the number of Fibonacci: "))” the step to get the input. In the pseudocode. Set Fn to 2. So, the learner wrote Fn = 2 in the python programming language.

Based on the pseudocode, the system wants to check “If n less than or equal to 0”. And if the n is less than or equal to 0 “Then display The Fibonacci series of (n) : [ ]” So, the learner wrote if condition to solve the issue “if n <= 0” the step will check the n is less than or equal to 0. And if it is yes, then the system will display the box bracket like this [ ] using the following python step “print ("The Fibonacci series of " + str(n) + " : " + str([ ]) )”. After if the n is not less than or equal to 0, it will go to the next step (else if).

In the pseudocode, the else if condition check like “Else if the n equal to 1” and if the n is equal to 1, “Then show the factorial value of (n) : [0]”. So, the learner wrote “elif n = = 1” the step to check the n is equal to 1 using the python language. If n is equal to 1, the system will display [0], using the following the python code “print ("The Fibonacci series of " + str(n) + " : " + str([0]))”. After if the n is not equal to 1, it will go to the next step (else).

The next step in pseudocode, (“Else”) check “While Fn less than n”. so, the learner wrote the “while Fn < n” step to check the Fn less than n, if it is less than, “next\_number = Fibonacci\_Numbers [-1] + Fibonacci\_Numbers [-2]” then want to find the total of last two digits from Fibonacci\_Numbers list into next\_number. So, the learner wrote the step “next\_number = Fibonacci\_Numbers[-1] + Fibonacci\_Numbers[-2]” to find the total into next\_number. After want to “Get the next\_number into list of Fibonacci\_Number”. So, the learner wrote “Fibonacci\_Numbers.append(next\_number)” the step to store the next\_number into Fibonacci\_Numbers list.

Next, Increment Fn by 1. So, the learner wrote the following step “Fn += 1”Again, check the increased Fn is less than n and continue the loop until the Fn is equal to the input. If Fn is not less than n “Display the list of Fibonacci\_Number”. So, the learner wrote the following step to display the output is “print("The Fibonacci series of " + str(n)+ " : " + str(Fibonacci\_Numbers))”. After display the system will “End”.

* **Factorial value.**

In the factorial value calculation…

Get the input into n. After, check the input is less than or equal to 0 (if n < = 0). If the input is less than or equal to 0, then want to show 0. If the input is not less than or equal to 0, then the while loop condition will check i is less than or equal to n. if the i is less than or equal to n, then (fac \*= i) Replace the multiplied value in fac after the fac multiply with I (fac = fac \* i). Again, the loop checks the increased i and continue the loop until the i is greater than input. If the i greater than n the loop will end and display the last replaced fac as factorial value.

**Comparison between the factorial value, pseudocode, and the python code.**

The python code is built based on the pseudocode. According to pseudocode, “Read the input integer in n to enter the value” the for get the input so, the learner wrote the following python statement to get the input “n = int(input("Enter the value : "))”.

Next step in the pseudocode, want to “Create variable i and fac to 2 and 1”. Then the learner wrote the following statement to generate the variables “i = 2”, “fac = 1”.

Next steps in the pseudocode, want to check “If the n less than or equal to 0”. And if the is less than or equal to 0, “Then show the factorial value of (n): 0”. So, the learner wrote if condition to solve the issue. “If n <= 0” the step is checking the n is less than or equal to 0. And if the n is less than or equal to 0, the system will display 0 using the following python code. The code is “print ("The factorial value of " + str(n) + " : " + str (0))”. If n is not less than or equal to 0, it will go to the next step (else).

Into the else statement, want to check the while loop condition i is less than or equal to n, based on the following pseudocode statement. The pseudocode statement is “While i less than or equal to n”. So, the learner wrote following python code to check the while loop condition i is less than or equal to n. the python code is “while i < = n”.

After, according to the factorial value pseudocode, if the i is less than or equal to n, want to start the loop. Into the loop, wan to “fac multiply by i”. So, the learner wrote python. And the code is “fac \*= i”. after accounting to the pseudocode, want to “i increment by 1” so the learner wrote python code to increase the i by 1 is “i += 1”. Again, check the increased i is less than or equal to n and continue the loop until the i is greater than the input. If the i is not less than or equal to n, according to the pseudocode, the system wants to “Display the factorial value of n : fac”. So, the learner wrote the following python code to display the output, and the python code is “print ("The factorial value of " + str(n) + " : " + str (fac))”.

* 1. 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. Pseudocode for the salary component.

* **Pseudocode for the Find group find button.**

Begin

If employeenamebox is index value not equal to -1:

Then display the message “Please Confirm the Salary issue date, Date Range Begin Date and End Date” and alert icon.

Call SalaryDetailsCalculation.find\_work\_date\_range(daterangebox)

Call SalaryDetailsCalculation.find\_total\_salary(employeenamebox, totalsalarybox)

Call SalaryDetailsCalculation.find\_total\_leave(employeenamebox, totalleavebox, begindatebox, enddatebox)

Call SalaryDetailsCalculation.find\_total\_OThourse(employeenamebox, OThoursebox, begindatebox, enddatebox, salaryissuedatebox, daterangebox)

Else:

Display MessageBox with message "Please Select Employee Name."

End.

If the user presses the find group find button in salary form, then the system wants to show the message first. The message is “Please Confirm the Salary issue date, Date Range Begin Date and End Date” because the system wants to find the total leaves and worked OT between the given date range so must want to set the date range to calculate the total leaves and worked OT hours correctly. And the message showing the “confirm the salary issue date” because it wants to calculate the gross pay value calculation.

After, the system will find the work date range and call and calculate the total salary, total leave and worked OT hours. And the called functions are want to create in SalaryDetailsCalculation Class.

* **Pseudocode for the Find group find\_work\_date\_range.**

Begin

Create function find\_work\_date\_range and take parameter (\_daterangebox):

Get DataTable dt from the database using SQL query to find Work\_Date\_Range from workdays table.

If dt has rows

Set \_daterangebox.Text to the string representation of Work\_Date\_Range from the first row of dt.

End

Want to create “find\_work\_date\_range” function in SalaryDetailsCalculation Class and get \_daterangebox (text box) as parameter. After, want to get Work\_Date\_Range from workdays table using SQL query into dt. After if the dt show any outputs, then want to set the output into \_daterangebox (text box).

* **Pseudocode for the Find group find\_total\_salary.**

Begin

Create function find\_total\_salary and take parameters (\_employeenamebox, \_ totalsalarybox):

Try

Get the ID from the selected value of \_employeenamebox.

Create SQL query \_totalSalary to calculate the total salary to an employee as "TotalSalary.”

Get DataTable dt from the database using \_totalSalary

If dt has rows

Set \_totalsalarybox.Text to the string representation of “TotalSalary” from the first row of dt.

Else:

Set \_totalsalarybox.Text to "No data found".

Catch Exception ex:

Display the message "Error: " and ex.Message

End.

Create the function find\_total\_salary and get the required tools as parameters. And want to do the calculation into try, catch error handling method. Check the calculation in the try area and have any errors the catch area wants to get the error and show the error as a message.

Next, get the \_employeenamebox is selected value into ID. After want to create the SQL query to find the total salary based on the programming assignment brief (allowance + basic salary). After want to check the query into dt and the dt show any output as “TotalSalary” then set the output into \_totalsalarybox(text box) or if it is not showing any outputs then show “No data found” into \_totalsalarybox.

* **Pseudocode for the Find group find\_total\_leave.**

Begin

Create function find\_total\_leave and take parameters (\_employeenamebox, \_totalleavebox, \_begindatebox, \_enddatebox):

Try:

Create SQL query totalleave to calculate Total Leaves of employee between the date range as “Total\_Leaves”

Get DataTable dt from the database using totalleave

If dt has rows:

Try to parse Total\_Leaves from dt as an integer.

If parsing is successful:

If parsed value (LValue) is greater than 0:

Set \_totalleavebox.Text to the string representation of LValue

If parsing fails or LValue is not greater than 0:

Set \_totalleavebox.Text to "0.”

Catch Exception ex:

Display the message "Error: " and ex.Message

End.

Here, want to create find\_total\_leave function and get the required tools as parameters. And want to do the calculation into try, catch error handling method. Check the calculation in the try area and have any errors the catch area wants to get the error and show the error as a message.

Next, create the SQL query to find a specific employee total leaves between the given date range as “Total\_Leaves” and want to give the name as totalleave to the query. After, “Get DataTable dt from the database using totalleave” if dt show any outputs then, “Try to parse Total\_Leaves from dt as an integer.”

* **Pseudocode for the Find group find\_total\_OThourse.**

Begin

Create function find\_total\_OThourse and take parameters (\_employeenamebox, \_OThoursebox, \_begindatebox, \_enddatebox, \_salaryissuedatebox, \_daterangebox):

Try:

Create OTQuery to calculate OT hours as "Worked\_OT\_Hours."

Get DataTable dt from the database using OTQuery

If dt has rows:

Try to parse Worked\_OT\_Hours from dt as an integer.

If parsing is successful:

If parsed value (OTValue) is greater than 0:

Set \_OThoursebox.Text to the string representation of OTValue

Else:

Set \_OThoursebox.Text to "0.”

If parsing fails or OTValue is not greater than 0:

Set \_OThoursebox.Text to "0.”

Catch Exception ex:

Display the message "Error: " and ex.Message

End.

* **Pseudocode for the payment group payments button.**

Begin

If totalsalarybox.Text is not an empty :

Call SalaryDetailsCalculation.find\_Nopayvalue (totalsalarybox, employeenamebox, begindatebox, enddatebox, nopaylevebox, nopaybox, daterangebox)

Call SalaryDetailsCalculation.find\_Basepayvalue(totalsalarybox, OThoursebox, basepaybox, employeenamebox)

Call SalaryDetailsCalculation.find\_Grosspayvalue(grosspaybox, basepaybox, nopaybox, salaryissuedatebox)

Else:

Display the message "Please find data in the Find Group".

End.

* **Pseudocode for the No pay value**

Begin

Create function find\_Nopayvalue and take the parameters (\_totalsalarybox, \_employeenamebox, \_begindatebox, \_enddatebox, \_nopaylevebox, \_nopaybox, \_daterangebox):

Try

Get the ID from the selected value of \_ employeenamebox as a string:

If \_totalsalarybox.Text can be parsed to a decimal as TotalSalary:

Create a NopayleveQuery to calculate the no-pay leaves as "No\_of\_Leaves" based on the condonation.

Create a noPayQuery to calculate the no-pay value as "NoPayValue" based on the condonation of the no-pay value.

Get DataTable dt from the database using NopayleveQuery;

If dt has rows:

Try to parse the value in dt.Rows[0]["NopayLeaves"] as an integer and store it in nopayLeaves

If the parsing is successful:

Set \_nopaylevebox.Text to NopayLeaves.

Get DataTable dt1 from the database using noPayQuery

If dt1 has rows:

Set \_nopaybox.Text to NoPayValue from dt1.

End if.

Else:

Set \_nopaybox.Text to "0.”

Set \_nopaylevebox.Text to "0.”

End if.

Catch Exception ex:

Display MessageBox with message "Error: " and ex.Message

End.

* **Pseudocode for the base pay value**

Begin

Create function find\_Basepayvalue and take parameters (\_totalsalarybox, \_OThoursebox, \_basepaybox, \_employeenamebox)

Try

If \_totalsalarybox.Text can be parsed to a decimal as TotalSalary and \_OThoursebox.Text can be parsed to a decimal as Worked\_OT\_Hours:

Create a basePayQuery to calculate the base pay as “BasePayValue” based on the condition of the base pay value.

Get DataTable dt from the database using basePayQuery

If dt has rows:

Set the “BasePayValue” from dt to base paybox.text.

Else:

Set 0 into \_basepaybox.Text

End if.

Catch Exception ex:

Display the message "Error: " and ex.Message

End

* **Pseudocode for the gross pay value**

Begin

Create function find\_ Grosspayvalue and take parameters (\_grosspaybox, \_basepaybox, \_nopaybox, \_salaryissuedatebox):

Try

If \_basepaybox.Text can be parsed to a decimal as basepayvalue and \_nopaybox.Text can be parsed to a decimal as nopayvalue:

Create a grossPayQuery to calculate the gross pay value as "GrossPayValue" based on the condonation of the base pay value.

If dt has rows:

Set the “GrossPayValue” from dt to \_grosspaybox.Text

Else:

Set 0 to \_grosspaybox.Text

End if.

Catch Exception ex:

Display the message "Error: " and ex.Message

End.

* 1. Implement the above three components.
  2. Design a suitable database structure for keeping the data of the above system.
  3. Analyze the features of an Integrated Development Environment (IDE).
  4. How those features help in application development
  5. Evaluate the use of the visual studio IDE for your application development contrasted with not using an IDE (evaluate with using Grifindo)

1. 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.