Achievement Standard: 91075\_1.45*version 5*

**Construct a plan for a basic computer program for a specified task**

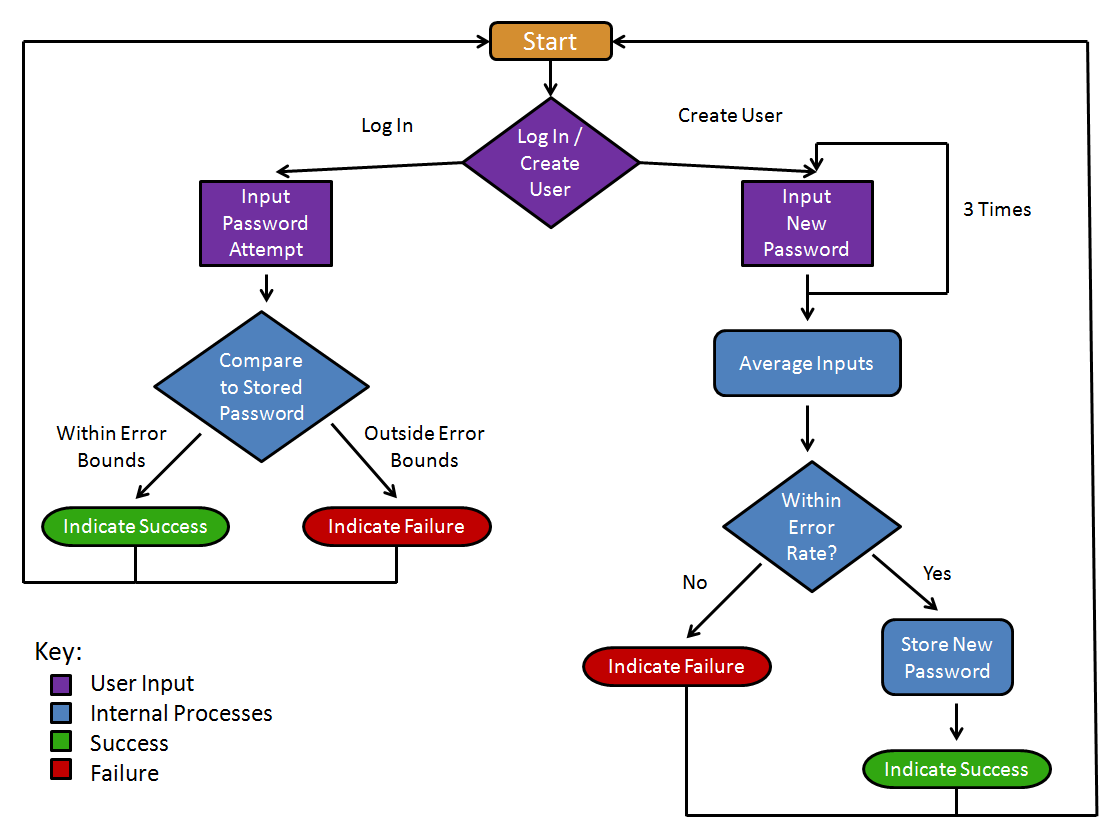
Achievement Standard: 91076\_1.46*version 5*

**Construct a basic computer program for a specified task**

**Python**

**STUDENT EVIDENCE**

**DOCUMENT**



**STUDENT NAME:** Matthew King

**Submission Date:** 26/ 06 / 2018

**Submission Number:** 1

Task 1. Plan The Programme

Program Purpose

Give Feedback on how much time a user spends playing video games each week

Target Language

The programming language used is PYTHON

Program Flow – Pseudo code

1. Ask the user their name “What is your name?”
2. If the name is an actual name (not a number) then continue otherwise loop
3. Ask the user how much time they spent playing games on Monday
4. If the time is reasonable (not more than 24 hours or less than 0 and is an actual number without decimals) continue, if not loop
5. Loop 3-4 for each day of the week
6. Calculate and tell the user what their average time spent gaming per day is
7. Tell the user what the most amount of time they spent gaming this week was
8. Tell the user what the least amount of time they spent gaming this week was
9. Tell the user if they are spending more or less than the recommended amount of time spent gaming each week (and how much they should cut back by if appropriate)

Program Flow – Flowchart

START

Ask the user “what is your name?”

If the name is not a name

“the minimum hours you spent gaming was (minimum hours)”

“The maximum hours you spent gaming was (maximum hours)”

Calculate the; minimum, maximum and average amount of time the user spent gaming using the time list

Add 1 to the loop number

Add the number to a list of total numbers.

If loop number is 7, exit the loop, otherwise loop again

Ask the user how much time they spent gaming on (loop number) day of the week

If the number is > 24 or < 0 or is not a number

If the user spent > 2 hours gaming on average per day.

END

“You spent (average hours – 2) hours less than you should playing games on average each day. Well done you!

“You spent (average hours – 2) hours more than you should playing games on average each day.You should slow down”

**VARIABLES MADE**

Must list every variable you have created

*Delete all yellow boxes once you have completed each table.*

Must test every input variable and with at least two data inputs

|  |  |  |  |
| --- | --- | --- | --- |
| **Input / Output / Both / Process** | **Name** | **Data-type** | **Description** |
| Input | name | String | *The user’s name* |
| Input | timeSpentDaily | integer | *How much time the user spent each day* |
| Output | averageTime | Float | *The average amount of time the user spent gaming each day* |
| Output | minTime | Integer | *The minimum amount of time the user spent gaming each day* |
| Output | maxTime | Integer | *The maximum amount of time the user spent gaming each day* |

Task 2. Test the Programme

**VALID TESTING**

**Purpose:** <Fill me in>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Which input is being tested** | **Data entered to test the input** | **Reason for test** | **Expected outcome** | **Actual outcome** | **Pass / Fail** |
| 1 | Name | 567 | *To check if name will not crash the program when number is entered* | *Please Don’t use numbers* | *Please Don’t use numbers* | *Pass* |
| 2 | matthew | *To check if the program will automatically give users a capital name* | *“Hello Matthew”* | *“Matthew is a nice name”* | *Pass* |
| 3 | timeSpentDaily | Twenty four | *to check that the code uses numbers only* | *Please enter a valid number* | *Please enter a valid number* | *pass* |
| 4 | 2.5 | *To check that the code wont crash if a decimal is entered* | *Please enter a valid number* | *Please enter a valid number* | *pass* |
| 5 |  |  |  |  |  |  |
| 6 |  |  |  |  |  |
|  |  |  |  |  |  |  |

**BOUNDARY TESTING**

**Purpose:** to check the code wont except any values that are not possible e.g. 36 hours in one day

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Which Boundary is being tested** | **Data entered to test the input** | **Reason for test** | **Expected outcome** | **Actual outcome** | **Pass / Fail** |
| 1 | The number of hours a user spends gaming in one day. Can’t be > 24 or < 0 | 25 | *To check the user cant put in more than 24 hours* | *LIAR* | *LIAR* | *Pass* |
| 2 | -1 | *To check the user can’t put in less than 0 hours* | *LIAR* | *LIAR* | *Pass* |
| 3 | The number of hours a user spends gaming in one day. Can’t be > 24 or < 0 | 24 | *To check the user can input 24 hours* | *Hmm a bit more than expected* | *Hmm a bit more than expected* | *Pass* |
| 4 | 0 | *To check the user can input 0 hours* | *Good Job* | *Good Job* | *Pass* |
| 5 |  |  |  |  |  |  |
| 6 |  |  |  |  |  |
|  |  |  |  |  |  |  |

**INVALID TESTING**

**Purpose:**To check the user cannot put in invalid data types that could break the code i.e. strings instead of integers

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Which input is being tested** | **Data entered to test the input** | **Reason for test** | **Expected outcome** | **Actual outcome** | **Pass / Fail** |
| 1 | Name | 56 | *To check that name won’t accept numbers* | *Please don’t use numbers* | *Please don’t use numbers* | *Pass* |
| 2 | Fifty6 | *To check that name wont, accept strings and integers combined* | *Please don’t use numbers* | *Please don’t use numbers* | *Pass* |
| 3 | TimeSpentDaily | Nine | *To check that the code will accept integers only* | *Please enter a valid number* | *Please enter a valid number* | *Pass* |
| 4 | 5.6 | *To check that the code will accept integers only* | *To check that the code will accept integers only* | *To check that the code will accept integers only* | *pass* |
| 5 |  |  |  |  |  |  |
| 6 |  |  |  |  |  |
|  |  |  |  |  |  |  |

Task 3. Write the Programme

1. Create a **folder** in your drive space and name it: ‘LastNameFirstName\_Assessment 1.45+1.46’.
2. Rename this Evidence document to: LastNameFirstName\_1.45 Evidence.
3. Save it in the above **folder**.
4. Type your name into the space shown on the Evidence document – in black.
5. Open IDLE and save a new file LastNameFirstName\_Assessment\_1.46.
6. Save it in the above **folder**.
7. Name, Date, Purpose and Version your EVERY program you code.
8. It is strongly suggested that a new version is saved each day as you develop your program.
9. Test as you go.
10. Record each test date.
11. If testing does not go as expected, then record that fact in the test table as you go.

*NOTE :*

*You may have used Pseudocode instead of a flowchart, so substitute Pseudocode for Flowchart*

AS 1.45: PLANNING Checklist – can you tick all these boxes?

**For Achieved I have:**

**A**

* Described the program I am creating in a short introduction
* Listed the variables I need in a clearly laid out table
  + At least two of these are different types (e. g. string, integer, Boolean)
* Drawn a flowchart that shows the steps my program will carry out to solve the problem
  + My flowchart has a START and END
  + My flowchart shows where my variables are created/modified
  + My flowcharts show where any calculations are done
  + My flowchart shows at least 1 selection/conditional structure (there is a fork with at least 2 branches in it somewhere!)
  + My flowchart shows at least 1 loop (part of it repeats and this is shown clearly with the arrows)
* Written a plan for testing my program on expected input
  + My testing plan has **at least 2** expected values that would be entered **for each place** the user can enter input

**For Merit I have also:**

* Completed my plan independently - I have taken ownership of the project and completed it to a high standard by the due date, and not depended on other people’s help for most of it.

**M**

* Named my variables well - logically, and according to Python naming conventions
* Made sure my flowchart is detailed
  + My flowchart shows how my program will deal with boundary values
  + My flowchart is clearly laid out
  + My flowchart shows I have chosen good conditions and structures and if I follow it I can see that there would be no unexpected behaviour.
  + There is no unnecessary repetition in my flowchart
* Included Boundary test cases in my testing plan
  + My testing plan has also covered any boundary values that apply to user inputs

**For Excellence I have also:**

* Shown any CONSTANTS or derived values I will use in my variable list

**E**

* Made sure that my flowchart shows an effective and logical solution to the program
* Made sure my flowchart shows how my program will be flexible and robust (handling invalid input, showing how constants and derived values are used etc.)
* Included invalid test cases in my testing plan for any inputs that have invalid values - at least 2 values/test cases for each of these inputs.
* My testing plan shows that my testing could/would be done in an organised and comprehensive way.

AS 1.46: IMPLEMENTING Checklist – can you tick all these boxes?

**A**

**For Achieved I have:**

* Used at least 2 variables
  + At least two of these are different types (e.g. string, integer, Boolean)
* Written a program in Python that does the required task.
  + My program works from start to end without crashing on expected input
  + My program includes comments
  + My program has at least 1 selection/conditional structure (there is an if/else somewhere)
  + My program has at least 1 loop (part of it repeats)
* I have tested my program on expected input
  + I have documented at least 2 expected values being entered for each place the user can enter input (in testing table or screenshots)

**For Merit I have also:**

**M**

* Constructed my program independently - I have taken ownership of the project and completed it to a high standard by the due date, and not depended on other people’s help for most of it.
* Named my variables well - logically, and according to Python naming conventions
* Made sure my program is well-structured
  + My program deals appropriately with boundary values
  + My code is clearly laid out with comments that accurately describe how my code functions
  + I have chosen good conditions and each structures has a purpose
  + There is no unnecessary repetition in my code
* Included Boundary test cases in my testing
  + My testing documentation also shows my program being tested on any boundary values that apply to user inputs

**For Excellence I have also:**

**E**

* Made sure that my program is an effective and logical solution to the program (could this code be any simpler and still work?)
* Made sure my program is flexible and robust
  + It handles invalid input
  + I have used CONSTANTS or derived values where relevant
  + It is easy to change things in my program without having to change a lot of code
* I have included invalid test cases in my testing documentation for any inputs that have invalid values - at least 2 values/test cases for each of these inputs.
* My testing documentation shows that my testing has been done in an organised and comprehensive way.