****

Copyright

© 2021 PG Online Limited

The contents of this unit are protected by copyright.

This Sample Paper, Sample Paper Mark Scheme and other associated files distributed with it are supplied to you by PG Online Limited under licence and may be used and copied by you only in accordance with the terms of the licence agreement between you and PG Online Limited. Except as expressly permitted by the licence, no part of the materials distributed with this unit may be used, reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic or otherwise, without the prior written permission of PG Online Limited.

Licence agreement

This is a legal agreement between you, the teaching institution, and PG Online Limited. PG Online Limited grants to you a non-exclusive, non-transferable, revocable licence to use this Sample Paper, Sample Paper Mark Scheme and other associated files distributed with it in the course of teaching by your teachers and/or employees.

The materials distributed with this unit may be copied and used by your teachers and/or employees on a single site only in the course of their teaching. You warrant that you shall not, and shall procure that each of your teachers and/or employees shall not, share in any way any of the materials or part of the materials with any third party, including users on another site or individuals who are teachers and/or employees of a separate institution. You acknowledge and agree that the materials must remain with you, the teaching institution, and no part of the materials may be transferred to another institution. You also warrant that you shall not, and shall procure that each of your teachers and/or employees shall not, procure, authorise, encourage, facilitate or enable any third party to reproduce these materials in whole or in part without the prior permission of PG Online Limited.

In consideration of the licence granted to you, you shall indemnify PG Online Limited against all liabilities, costs, expenses, damages and losses (including but not limited to any direct, indirect or consequential losses, loss of profit, loss of reputation and all interest, penalties and legal costs and all other professional costs and expenses) suffered or incurred by PG Online Limited arising out of or in connection with the exercise by you of your rights granted under this licence.

Please write clearly, in BLOCK CAPITALS and black ink

Centre number Candidate number

Forename(s)

Surname

Date of Exam Time allowed: 2 hours

GCSE Computer Science

Paper 1: Computational thinking and programming skills

Total Marks

PAPER 1A

Instructions

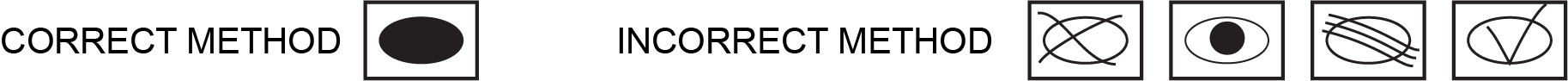
* Write in black ink or black ball-point pen. Use pencil only for drawing.
* Write your answer to each question in the space provided
* Answer all questions
* Do all rough work in this book
* Cross through any work you do not want marked
* **Questions that require a coded solution must be answered in Python 3**
* **You are not allowed to use a calculator**

Information

* The total mark for this paper is **90**
* The student version of this paper has **23** pages

**Advice**

* For multiple-choice questions, completely fill in the lozenge next to the answer you want to select.



* Icon

  Description automatically generatedTo change your answer, cross out your original answer like this: Icon

  Description automatically generated
* If you want to go back to an answer you previously crossed out, circle the answer you now want to select like this:

**Answer ALL questions.**

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **1** | **.** | **1** |

Define the term decomposition.

[2 marks]

Breaking down a problem into a number of sub-problems (1)  
so that each sub-problem accomplishes an identifiable task (1)  
which might itself be further subdivided (1)  
each sub-problem is easier to solve than the main problem (1).

Accept different wording with the same meaning.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **2** |  |  |

The pseudo-code in **Figure 1** assigns values to variables. The values represent the number of whole pounds that a person earns.

**Figure 1**

salary ← 16  
bonus ← 2  
salary ← salary + bonus

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **2** | **.** | **1** |

Shade **one** lozenge that shows the value stored in the variable salary once the program in **Figure 1** has finished running.

[1 mark]

|  |  |  |
| --- | --- | --- |
| A | 2 | Icon  Description automatically generated |
| B | 16 | Icon  Description automatically generated |
| C | 18 | A picture containing icon  Description automatically generated |
| D | 32 | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **2** | **.** | **2** |

Shade **one** lozenge that shows the data type for the variable salary in **Figure 1**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| A | Boolean | Icon  Description automatically generated |
| B | character | Icon  Description automatically generated |
| C | integer | A picture containing icon  Description automatically generated |
| D | real | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** |  |  |

The algorithm in **Figure 2** is used to calculate the area of a circle according to the formula:

*area = πr2*

The radius entered by the user must be between 1 and 100.

* Line numbers are included but are not part of the algorithm.

**Figure 2**

1 CONSTANT PI ← 3.141

2 SUBROUTINE CIRCLE\_AREA(r)

3 area ← PI \* r \* r

4 RETURN area

5 ENDSUBROUTINE

6 radius ← STRING\_TO\_REAL(USERINPUT)

7 IF radius ≥ 1 OR radius ≤ 100 THEN

8 result ← REAL\_TO\_STRING(CIRCLE\_AREA(radius))

9 OUTPUT result

9 ENDIF

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **1** |

Shade **one** lozenge which explains the meaning of the relational operator ≥ in line 7 in **Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| A | Greater than | Icon  Description automatically generated |
| B | Greater than or equal to | A picture containing icon  Description automatically generated |
| C | Less than | Icon  Description automatically generated |
| D | Less than or equal to | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **2** |

Shade **one** lozenge which shows the line number where the subroutine is called in **Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| A | Line 2 | Icon  Description automatically generated |
| B | Line 4 | Icon  Description automatically generated |
| C | Line 5 | Icon  Description automatically generated |
| D | Line 8 | A picture containing icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **3** |

Shade **one** lozenge which shows the data type of the variable radius in   
**Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| A | Character | Icon  Description automatically generated |
| B | Integer | Icon  Description automatically generated |
| C | Real | A picture containing icon  Description automatically generated |
| D | String | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **4** |

Shade **one** lozenge which shows the value stored in the variable result when CIRCLE\_AREA returns the value 12.56 in **Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| A | 12.56 | Icon  Description automatically generated |
| B | 12 | Icon  Description automatically generated |
| C | 13 | Icon  Description automatically generated |
| D | '12.56' | A picture containing icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **5** |

State the reason that PI (π) is stored as a constant rather than a variable in **Figure 2**.

[1 mark]

The value stored in PI cannot be changed**.**

Because the value of pi / π is a constant / is always the same.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **6** |

The variable area is a local variable.  
  
Explain why the use of a local variable is more appropriate than a global one in this situation.

[2 marks]

Once the subroutine has finished the area variable will no longer be needed / will be destroyed (1).   
This reduces the amount of memory required (1).  
Only code inside the subroutine can access the area variable (1) which prevents other code outside the subroutine causing a bug (1) and leads to more reusable code (1).

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **4** |  |  |

A programmer has written a Python program that asks the user to input a number. The program then calculates the times table for the number input for 10 rows. For example, if the user enters 5, the program will output:

1 x 5 = 5  
2 x 5 = 10  
3 x 5 = 15  
4 x 5 = 20  
5 x 5 = 25  
6 x 5 = 30  
7 x 5 = 35  
8 x 5 = 40  
9 x 5 = 45  
10 x 5 = 50

Complete the program below by filling in the gaps using the items in **Figure 3**. You will not need to use all the items in **Figure 3**. Each item in **Figure 3** should only be used once.

[5 marks]

**Figure 3**

|  |  |  |  |
| --- | --- | --- | --- |
| 0,10 | 0,9 | 1,10 | 1,11 |
| print | output |  |  |
| while | for | if | elif |
| int | float | str |  |
| + | x | \* | = |

timesTable = int(input("Enter times table: "))

for i in range(1,11):

result = timesTable \* i

print(i, "x", timesTable, "=", result)

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** |  |  |

The algorithm in **Figure 4** is a searching algorithm.

* Array indexing starts at 0.
* Line numbers are included but are not part of the algorithm.

**Figure 4**

1. names ← ["Amy", "Ava", "Joe", "Tim"]
2. OUTPUT "Enter name to search for: "

3 item ← USERINPUT

4 found ← False

5 i ← 0

6 WHILE NOT found AND i < LEN(names)

7 IF names[i] = item THEN

8 found ← true

9 OUTPUT names[i]

10 ENDIF

11 i ← i + 1

12 ENDWHILE

13 IF NOT found THEN

14 OUTPUT "Name not found"

15 ENDIF

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **.** | **1** |

State the data structure that is used to store the names in the variable names in the algorithm shown in **Figure 4**.

[1 mark]

Array (accept list) (1).

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **.** | **2** |

Shade **one** lozenge to show which of the following contains the **false** statement about the algorithm in **Figure 4**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| A | The algorithm uses Boolean operators | Icon  Description automatically generated |
| B | The algorithm uses definite iteration | A picture containing icon  Description automatically generated |
| C | The algorithm uses selection | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **.** | **3** |

Complete the trace table for the algorithm shown in **Figure 4** when the user enters "Joe". Some values have already been entered.

[6 marks]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| names | | | | item | found | i | OUTPUT |
| [0] | [1] | [2] | [3] |
| "Amy" | "Ava" | "Joe" | "Tim" | "Joe" | False | 0 |  |
|  |  |  |  |  |  | 1 |  |
|  |  |  |  |  | True | 2 | "Joe" |
|  |  |  |  |  |  | 3 |  |

1 mark for each correct entry. Allow variable values to be copied down if they are unchanged.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **.** | **4** |

The algorithm is for a linear search.

State **one** advantage of using a linear search rather than a binary search.

[1 mark]

The list doesn’t need to be sorted for a linear search (1).  
Fast searching for small lists with a linear search (1).  
There is no need to re-sort the list after insertions or deletions when using a linear search (1).

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **6** |  |  |

Write a Python program that converts a temperature in Centigrade to the equivalent temperature in Fahrenheit.

The program should:

* Allow the user to enter the temperature in Centigrade (no validation is required)
* Validate that the temperature entered is between 0 and 100 Centigrade (no other validation is required and both 0 and 100 are to be treated as valid temperatures)
  + Convert the temperature to Fahrenheit by the following formula:  
    Fahrenheit = Centigrade x (9 ÷ 5) + 32
* If a valid temperature is entered, output the converted temperature in Fahrenheit
* Otherwise output "Temperature must be between 0 and 100"
  + The program does not need to ask the user to enter the temperature again if they enter an invalid number

You should use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

[7 marks]

**Program Design (2 marks)**

Meaningful variable names throughout (even if the logic is incorrect) (1).  
Suitable data types throughout (must be a float for any temperatures entered or answers) (1).

**Program Logic (5 marks)**

Get user input for Centigrade temperature (1).  
Check temperature range is between 0 and 100 (1).  
Correct calculation for Fahrenheit conversion (1).  
Output the correct temperature (1) (award marks with or without unit)  
Output “Temperature must be between 0 and 100” if invalid temperature entered (1).

**Maximum 6 marks** if any errors in code

Python example (fully correct)

tempC = float(input("Enter the temperature in Centigrade: "))

if tempC >= 0 and tempC <= 100:

tempF = tempC \* (9/5) + 32

print(tempF)

else:

print("Temperature must be between 0 and 100")

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **7** |  |  |

Write a Python program that allows the user to input a word. The program then outputs each character in the word with the corresponding position starting at 1. For example, if the user enters “Hello”, the program will output:

1 H  
2 e  
3 l  
4 l  
5 o

Your program should work as follows:

* Gets the user to enter a string and store it in a suitable variable
* Output each character along with its position in the string. The positions output should start at 1

You should use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

[8 marks]

Program Design (3 marks)

The idea of inputting a string and using a loop to output each character (even if the code doesn’t work) (1).  
Use of for or while loop construct (even if the logic is incorrect) (1).  
Meaningful variables names (even if the logic is incorrect) (1).

Program Logic (5 marks)

Get user input for word correctly (1).  
Store the result in a variable (1).  
Use correct condition in FOR or WHILE statement (including any associated flags or variables) (1).  
Calculate and output the correct position (this will be 1 more than the array reference of the character) (1).  
Output each character (1).

Maximum 7 marks if any errors in code

Python example 1 (fully correct)

word = input("Enter word: ")

for i in range(0, len(word)):

position = i + 1

print(position, word[i])

Python example 2 (fully correct)

position = 1

word = input("Enter word: ")

for letter in word:

print(position, letter)

position = position + 1

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** |  |  |

Four in a row is a board game which has a grid with a width of seven holes and height of six holes.

Two players take turns to place a piece in one of the seven columns. One player is yellow, whilst the other player is red.

The following subroutines control the dropping of pieces into a column.

|  |  |
| --- | --- |
| DROP(colour, column) | Drops one piece of colour 'R' or 'Y' in column number 0 to 6. |
| HEIGHT(column) | Returns the number of pieces currently in the column as an integer between 0 and 6. |
| FULL(column) | returns True if the column is full, False if the column is empty or partially full |

The game has started and three moves have been made.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 5 |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |
| 1 |  |  |  | 'R' |  |  |  |
| 0 |  |  | 'R' | 'Y' |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** | **.** | **1** |

State a suitable data structure that could be used to store the grid.

[1 mark]

2-D array.

Accept 2-D list.  
Do not accept array in this case, as this implies a 1-D array.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** | **.** | **2** |

State the value that will be returned with the following statements.

[2 marks]

HEIGHT(3) 2

FULL(3) False

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** | **.** | **3** |

Draw the grid after the following sequence has run.

DROP('Y', 2)  
DROP('R', 3)  
DROP('Y', 2)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 5 |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |
| 2 |  |  | 'Y' | 'R' |  |  |  |
| 1 |  |  | 'Y' | 'R' |  |  |  |
| 0 |  |  | 'R' | 'Y' |  |  |  |

[3 marks]

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** | **.** | **4** |

Develop an algorithm using pseudo-code that will check if a column is full before dropping a piece into it.

Your algorithm will get the input from the user for both the colour and the column they want to drop their piece into. The piece will either be dropped in the column, or if there isn’t space, the algorithm will tell the user to try a different column.

[5 marks]

* Get user input twice (1).
* Store user input in two variables (1).
* Use of IF statement that checks the subroutine FULL(column) (1).
* Correct output if full (1).
* Use of subroutine DROP(colour, column) if not full (1).

Example 1

colour ← USERINPUT  
column ← USERINPUT  
IF FULL(column) THEN  
 OUTPUT "Try a different column"  
ELSE  
 DROP(colour, column)  
ENDIF

Example 2

colour ← USERINPUT  
column ← USERINPUT  
IF NOT FULL(column) THEN  
 DROP(colour, column)  
ELSE  
 OUTPUT "Try a different column"  
ENDIF

Allow minor spelling errors such as colum or colunm.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** | **.** | **5** |

The game is a draw if every column is full.

Develop an algorithm which will determine if the game is a draw by checking each column, using iteration, to see if it is full. If all columns are full, "Draw" will be displayed.

[7 marks]

* Mark A: A counter to store how many columns are full, initialised to zero (1)
* Mark B: A for or while loop (1)
* Mark C: Correct condition in while or for loop and any necessary flags (1)
* Mark D: Selection to determine if current column is full (1)
* Mark E: Update full column counter or appropriate flag (1)
* Mark F: Selection to determine if all columns are full (1)
* Mark G: Output "Draw" (1)

**Example 1**

fullColumnCount ← 0

FOR column ← 0 TO 6  
 IF FULL(column) THEN  
 fullColumnCount ← fullColumnCount + 1  
 ENDIF  
ENDFOR  
IF fullColumnCount = 7 THEN  
 OUTPUT "Draw"  
ENDIF

Accept answers which use WHILE loop

Accept answers that make use of constants for the number of columns.

Do not accept marks B, C or E if iteration is not used.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **9** | **.** | **1** |

A farmer has a computer-controlled tractor. The tractor needs to plough a field that has been divided into squares. The following instructions are available:

* Forward(n) moves the tractor n squares forward
* TurnLeft(deg) turns the robot deg degrees left (anti-clockwise)

Draw the path of the tractor through the grid below if the following program is executed (the tractor starts in the square marked by the facing in the direction of the arrow).

Forward(4)

TurnLeft(90)

Forward(1)

TurnLeft(90)

Forward(4)

[3 marks]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Horizontal line at the bottom (1)  
Vertical line at right (1)  
Horizontal line above the bottom line (1)

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **9** | **.** | **2** |

The grid below represents a field that needs to be ploughed. The tractor must visit each square in turn.

Develop an algorithm using pseudo-code which will plough the whole field as indicated by the arrow on the grid. The algorithm must use iteration. Note that the tractor can use the TurnLeft subroutine to make a turn.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

[3 marks]

FOR i ← 1 TO 3

Forward(4)

TurnLeft(90)

Forward(1)

TurnLeft(90)

Forward(4)

TurnLeft(270)

Forward(1)

TurnLeft(270)

ENDFOR

FOR loop (for count-controlled iteration) used (1).  
Correct condition in the FOR loop (1).  
First five instructions used for bottom two rows (as in the code for 09.1) (1)  
Last three instructions with TurnLeft(270) used to make the tractor effectively turn right by 90 degrees. (1)

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **0** |  |  |

Below is an array of numbers.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 6 | 4 | 2 | 3 | 7 | 8 | 1 |

Describe the stages required to sort this array using a merge sort algorithm.

[5 marks]

* Split the array into two equal length arrays (1)
* Continue to split each of these arrays… (1)
* …until there are 8 arrays with length 1 (1)
* Now merge each pair of arrays together in order (1)
* Until all sub-arrays have been merged together into one array (1)

Or a diagram to show this process:

A picture containing arrow

Description automatically generated

Diagram shows:

* First array split into two arrays length 4 (1).
* Further splits of each array into arrays of length 2 (1).
* Further splits of each array into arrays of length 1 (1).
* Merge single arrays into pairs (1).
* Merge pairs into two arrays length 4, and two arrays into one array (1).

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **1** |  |  |

The following Python program asks the user to input a sentence that must be at least three characters long and then works out the number of words it contains. The first part of the program validates the sentence entered and is shown in **Figure 5**.

**Figure 5**

validSentence = False

sentence = ""

while not validSentence:

sentence = input("Enter sentence: ")

if not len(sentence) >= 3:

print("Invalid choice")

else:

print("Valid choice")

validSentence = True

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **1** | **.** | **1** |

Shade **one** lozenge which shows invalid data that could be input to the program in **Figure 5**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| A | The dog. | Icon  Description automatically generated |
| B | The dog | Icon  Description automatically generated |
| C | I. | A picture containing icon  Description automatically generated |
| D | Me. | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **1** | **.** | **2** |

Complete the following test plan for the code shown in Figure 5.

[2 marks]

|  |  |  |
| --- | --- | --- |
| **Test type** | **Test data** | **Expected result** |
| Normal data | Apple | Valid choice message displayed |
| Invalid data | An empty string or one character | Invalid choice Message displayed |
| Boundary data | Any words that are two or three characters long | If three characters given as the test data  Valid choice message displayed  If two characters given as the test data  Invalid choice message displayed |

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **1** | **.** | **3** |

The remainder of the program, in **Figure 6**, determines the number of words that have been entered by the user and stored in the variable named sentence.

**Figure 6**

spaceCount = 0

for letter in sentence:

if letter == " ":

spaceCount = 1

totalWords = spaceCount

print("Number of words:", spaceCount)

The program works by counting the number of spaces between each word in the sentence entered.

The program currently always reports that 1 word has been entered no matter how many characters or spaces were entered.

The program contains three lines of code that each contain one error.

Using Python only, rewrite the code in the space below to correct the three errors.

[3 marks]

spaceCount = 0

for letter in sentence:

if letter == " ":

spaceCount = spaceCount + 1

totalWords = spaceCount + 1

print("Number of words:", totalWords)

spaceCount = spaceCount + 1 (1)  
totalWords = spaceCount + 1 (1) (the number of words is one more than the number of spaces)  
print(“Number of words:”, totalWords) (1)

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** |  |  |

Write a Python program that asks the user to enter the number of whole seconds they took in a race. The program then outputs the number of minutes and seconds the race took them. The format of the time must be mm:ss (two digits for the minutes and two digits for the seconds).

Example 1, if the user enters 127 seconds, the program will output 02:07  
Example 2, if the user enters 180 seconds, the program will output 03:00  
Example 3, if the user enters 117 seconds, the program will output 01:57

The program should:

* Ask the user to enter the number of seconds (there is no need to validate this number)
* Calculate the number of minutes that were required by performing an integer division on the seconds by 60
* Calculate the number of seconds that were required by using modulus
* If the minutes are less than 10, concatenate a leading zero
* If the seconds are less than 10, concatenate a leading zero
* Output the minutes and seconds with a colon between them

You should use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

[8 marks]

**Program Design (1 mark)**

Meaningful variable names throughout **and** suitable data types throughout (1).

**Program Logic (7 marks)**

Get user input for the number of seconds (1).  
Convert to integer and store in variable (1).  
Calculate the number of minutes using integer division (1).  
Calculate the number of seconds using modulus (1).  
IF statement with condition to correctly concatenate leading zero for minutes (1).  
IF statement with condition to correctly concatenate leading zero for seconds (1).  
Output minutes and seconds with a colon between them (1).

**Maximum 7 marks** if any errors in code

Python example 1 (fully correct)

time = int(input("Race time in seconds: "))

minutes = time // 60

seconds = time % 60

if minutes < 10:

minutes = "0" + str(minutes)

if seconds < 10:

seconds = "0" + str(seconds)

print(minutes, ":", seconds)

Python example 2 (fully correct)

time = int(input("Race time in seconds: "))

minutes = time // 60

seconds = time % 60

if len(str(minutes)) == 1:

minutes = "0" + str(minutes)

if len(str(seconds)) == 1:

seconds = "0" + str(seconds)

print(minutes, ":", seconds)

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **3** |  |  |

A Python program has been written that creates 5 random integers between 1 and 100 and stores them in a list. The program is shown in **Figure 7**. The function generateRandom returns a random number between 1 and 100.

**Figure 7**

import random

randomNumbers = [0]\*5

for i in range(0,5):

randomNumbers[i] = generateRandom()

print(randomNumbers)

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **3** | **.** | **1** |

State the purpose of the instruction:

import random

[1 mark]

This imports the random library so that random integers can be generated (1).

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **3** | **.** | **2** |

State the contents of randomNumbers after the instruction.

randomNumbers = [0]\*5

[1 mark]

[0, 0, 0, 0, 0]

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **3** | **.** | **3** |

The subroutine generateRandom has not yet been written.

Write Python code for the subroutine generateRandom. The subroutine will be a function that has no arguments and returns a random integer between 1 and 100.

[5 marks]

**Program Design (1 marks)**

Correctly structure the answer as a subroutine even if the syntax is not correct (1).

**Program Logic (5 marks)**

def generateRandom (1)   
(accept a space inside the brackets)  
Generate a random integer between 1 and 100 (1)  
Use of the random library to generate the random number (1)  
Return the random number (1).

**Maximum 5 marks** if any errors in code

Python example 1 (fully correct)

def generateRandom():

randNum = random.randint(1,100)

return randNum

Python example 2 (fully correct)

def generateRandom():

return random.randint(1,100)

Acknowledgements

Artwork



*Change of Heart*

© Karen Stamper (30 cm × 30 cm)

Paper collage and acrylic on board

www.karenstampercollage.com