GCSE Computer Science

Deep Dive into Paper 2 – Questions 5 and 6

Pearson Edexcel

Introduction

- Welcome

Poll

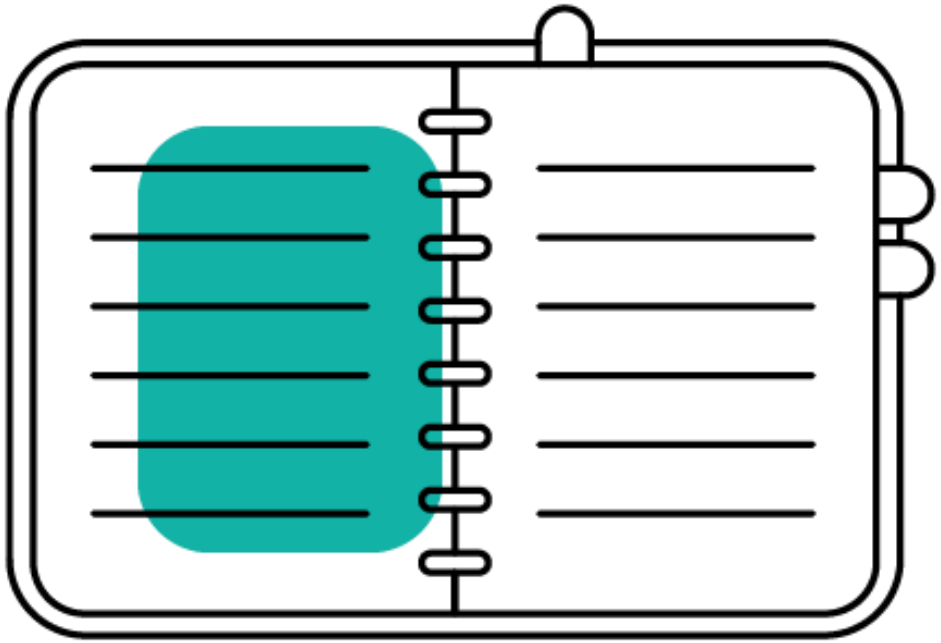
Who is new to the qualification?

Poll

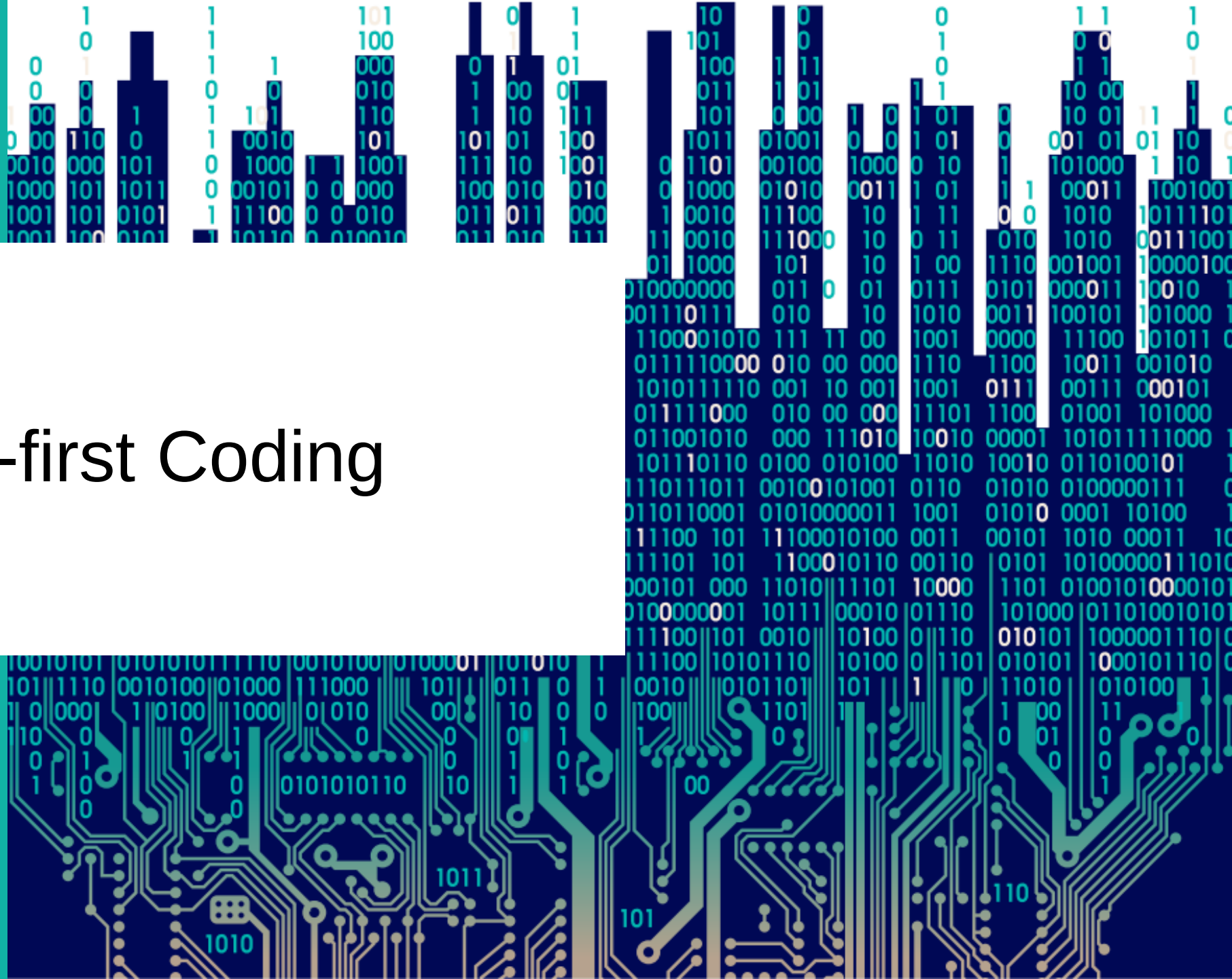
Did your students sit the summer 2024 examination?

Agenda

- Welcome and introduction
- Deep dive into Specimen 4 Q05
- Deep dive into Specimen 4 Q06
- Using the PLS in the exam
- Gender Gap in Computer Science
- Support and resources



Comment-first Coding



Pearson
Edexcel

Accessing materials

- Open Specimen 4, Paper 2, Question paper
- Open Q05.py in a Python IDE

Q05.py

```
# ------------------------------------------------------------------
# Constants
# ------------------------------------------------------------------
TABLE_WIDTH = 66            # Divider lines

# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
headers = ["Name", "ID", "Q 1", "Q 2", "Q 3",
           "Q 4", "Yr Total"]
names = ["Adams", "Baker", "Collins", "Dalton", "East", "Ford",
         "Green", "Hill"]
empID = [434, 161, 427, 285, 460, 889, 275, 789]
salesQuarter1 = [942.45, 1566.99, 924.59, 197.71, 764.20,
                 279.43, 867.03, 880.43]
salesQuarter2 = [865.78, 337.10, 1597.64, 171.13, 552.89,
                 495.23, 637.09, 469.96]
salesQuarter3 = [973.25, 466.54, 288.54, 979.36, 2780.42,
                 898.52, 522.45, 979.11]
salesQuarter4 = [535.84, 919.83, 661.63, 852.07, 315.02,
                 120.78, 2748.53, 755.71]
salesTotalPerName = []

# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------

# Calculate total sales for each employee and add it
#   to the salesTotalPerName array

# Display the headers with a pipe symbol separating columns

# Display a divider for the width of the table

# Display a row to show data for each employee
```

Suggested time: 30 minutes

5 A program is needed to display quarterly and annual sales data.

Here is the output produced by a fully functional program.

```
Name     |  ID  |   Q 1    |   Q 2    |   Q 3    |   Q 4    | Yr Total
-------------------------------------------------------------------------
Adams    |  434 |   942.45 |   865.78 |   973.25 |   535.84 |   3317.32
Baker    |  161 |  1566.99 |   337.10 |   466.54 |   919.83 |   3290.46
Collins  |  427 |   924.59 | 1597.64  |   288.54 |   661.63 |   3472.40
Dalton   |  285 |   197.71 |   171.13 |   979.36 |   852.07 |   2200.27
East     |  460 |   764.20 |    552.8 | 2780.42  |   315.02 |   4412.53
Ford     |  889 |   279.43 |   495.23 |   898.52 |   120.78 |   1793.96
Green    |  275 |   867.03 |   637.09 |   522.45 | 2748.53  |   4775.10
Hill     |  789 |   880.43 |   469.96 |   979.11 |   755.71 |   3085.21
-------------------------------------------------------------------------
                                                 Year Total 26347.25
                                                 Mean Sales  3293.41
```

The pipe symbol (|) separates columns.

The program must use string formatting functions to control the output.

The table shows the format for the column headers.

Field	Align	Width	Precision	Type
Name	Left	9		String
ID	Centre	5		String
Q 1 to Q 4	Centre	9		String
Yr Total	Centre	10		String

Comment-first coding: Analysis

1	Read the question	Pick out the most important pieces of information to help understand the problem
2	Read the requirements for amending the code (after ‘Open file Q05.py’)	Make note of any requirements, variables, or information that needs to be kept track of
3	Read the last few instructions	Make note of any instructions needed in the response
4	Read the provided student code (after ‘The program must …’)	Make note of the types of coding needed to solve the problem

1. Read the question

Need data in rows and columns.
Exact output given, so can be checked to see if program works.

Are these totals in the data already or do they need to be calculated?

This says ‘string formatting functions’. May need to look that up in the PLS.

Found <string>.format () on page 12 of the PLS.
The question says ‘column headers’ which must be the ID, Q!, Q2, etc.

Suggested time: 30 minutes

5 A program is needed to display quarterly and annual sales data.

Here is the output produced by a fully functional program.

```
Name     |  ID  |   Q 1    |   Q 2    |   Q 3    |   Q 4    | Yr Total
---------------------------------------------------------------------
Adams    |  434 |   942.45 |   865.78 |   973.25 |   535.84 |   3317.32
Baker    |  161 |  1566.99 |   337.10 |   466.54 |   919.83 |   3290.46
Collins  |  427 |   924.59 |  1597.64 |   288.54 |   661.63 |   3472.40
Dalton   |  285 |   197.71 |   171.13 |   979.36 |   852.07 |   2200.27
East     |  460 |   764.20 |    552.8 |  2780.42 |   315.02 |   4412.53
Ford     |  889 |   279.43 |   495.23 |   898.52 |   120.78 |   1793.96
Green    |  275 |   867.03 |   637.09 |   522.45 |  2748.53 |   4775.10
Hill     |  789 |   880.43 |   469.96 |   979.11 |   755.71 |   3085.21
---------------------------------------------------------------------
                                                Year Total  26347.25
                                                Mean Sales   3293.41
```

The pipe symbol (|) separates columns.

The program must use string formatting functions to control the output.

The table shows the format for the column headers.

Field	Align	Width	Precision	Type
Name	Left	9		String
ID	Centre	5		String
Q 1 to Q 4	Centre	9		String
Yr Total	Centre	10		String

1. Read the question

The table shows the format for each row of employee data.

Field	Align	Width	Precision	Type
Name	Left	9		String
ID	Centre	5		Integer
Q 1 to Q 4	Right	9	2	Real
Yr Total	Right	10	2	Real

The data is stored in one-dimensional arrays.

More string formatting, but this time it's for each row of the data, which is the numbers.

Connection to the code file. The data is stored in arrays.
That's plural, so there must be multiple arrays.
Need to check that in the code.

2. Read the requirements

Open file **Q05.py**

Amend the code to:

- calculate the total sales for each employee and add it to an array
- display the column headers
- display the data for each employee
- calculate and display the total sales for the year
- calculate and display the mean sales for the year.

Need to add multiple numbers.
Check if needs a loop.

Display the headers.
Display each employee.
Will need a loop because there are multiple employees.

Need to sum up a total of all the employee data

Need to find a mean (total sales divided by number of employees)

3. Read the last instructions.

The program must work if the number of items in the arrays is changed.

Use the constants and variables provided.

Use string formatting functions to align the columns.

Do **not** add any additional functionality.

Save your amended code file as **Q05FINISHED.py**

(Total for Question 5 = 15 marks)

Remember to use length of array as upper bound on loop

Be sure to check for, and use, the constants and variables already defined in the code file.

Make sure to use the correct alignment for the columns. Use the output shown in the question paper.

Standard instructions.
Just focus on what has been asked.
Don't get distracted.

4. Read the Python code file

Remember to use these constants

Multiple arrays, so could use one loop, with the same index for each

Program logic is already provided in the comments. Don't need to decompose the whole solution, but do need to decompose and code the smaller pieces.

Q05.py

```
1  # ------------------------------------------------------------------
2  # Constants
3  # ------------------------------------------------------------------
4  TABLE_WIDTH = 66            # Divider lines
5
6  # ------------------------------------------------------------------
7  # Global variables
8  # ------------------------------------------------------------------
9  headers = ["Name", "ID", "Q 1", "Q 2", "Q 3",
10            "Q 4", "Yr Total"]
11 names = ["Adams", "Baker", "Collins", "Dalton", "East", "Ford",
12          "Green", "Hill"]
13 empID = [434, 161, 427, 285, 460, 889, 275, 789]
14 salesQuarter1 = [942.45, 1566.99, 924.59, 197.71, 764.20,
15                  279.43, 867.03, 880.43]
16 salesQuarter2 = [865.78, 337.10, 1597.64, 171.13, 552.89,
17                  495.23, 637.09, 469.96]
18 salesQuarter3 = [973.25, 466.54, 288.54, 979.36, 2780.42,
19                 898.52, 522.45, 979.11]
20 salesQuarter4 = [535.84, 919.83, 661.63, 852.07, 315.02,
21                 120.78, 2748.53, 755.71]
22 salesTotalPerName = []
23
24 # ------------------------------------------------------------------
25 # Main program
26 # ------------------------------------------------------------------
27
28 # Calculate total sales for each employee and add it
29 #   to the salesTotalPerName array
30
31 # Display the headers with a pipe symbol separating columns
32
33 # Display a divider for the width of the table
34
35 # Display a row to show data for each employee
36
37 # Display a divider for the width of the table
38
39 # Calculate total of all sales for all employees for the year
40
41 # Display a label and the total sales for the year
42
43 # Calculate mean of the total sales for the year
44
45 # Display a label and the mean of the total sales for the year
46
47
```

Comment-first coding: Develop code

A	Top-down	Start with the first comment, add the code below it, test it, get it working, then move to the next
B	Easiest-first	Read the comments and add code for any that are easy to code, test, and get working.
C	Subprograms-first (similar to bottom-up)	Code and test the subprograms first

Suggestions:

- Comment out a section, if it's a struggle to get working and try to move to another section.
- Remove all syntax and runtime errors.

Develop the code for each comment

‘for each employee’ means a loop.
Remember to use length of array as upper bound on loop
Need a common index for each array.
Add the total for each employee to the row in the list.
Run, Test, Fix.

The comment says to display the headers with the pipe symbol.
That means <string>.format(), but the <string> needs to be defined to match the first formatting table on the question paper.
Run, Test, Fix.

Display a dividing line.
Use multiplication of a string, which is found in the PLS on page 12.
Run, Test, Fix.

```
9   headers = ["Name", "ID", "Q 1", "Q 2", "Q 3",
10            "Q 4", "Yr Total"]
11  names = ["Adams", "Baker", "Collins", "Dalton", "East", "Ford",
12           "Green", "Hill"]
13  empID = [434, 161, 427, 285, 460, 889, 275, 789]
14  salesQuarter1 = [942.45, 1566.99, 924.59, 197.71, 764.20,
15                   279.43, 867.03, 880.43]
16  salesQuarter2 = [865.78, 337.10, 1597.64, 171.13, 552.89,
17                   495.23, 637.09, 469.96]
18  salesQuarter3 = [973.25, 466.54, 288.54, 979.36, 2780.42,
19                  898.52, 522.45, 979.11]
20  salesQuarter4 = [535.84, 919.83, 661.63, 852.07, 315.02,
21                  120.78, 2748.53, 755.71]
22  salesTotalPerName = []
23  salesYearTotal = 0.0
24  salesMean = 0.0
25  headerLayout = "{:<9s}|{:^5s}|{:^9s}|{:^9s}|{:^9s}|{:^9s}|{:^10s}"
26  rowLayout =
    "{:<9s}|{:^5d}|{:>9.2f}|{:>9.2f}|{:>9.2f}|{:>9.2f}|{:>10.2f}"
27  yearTotalLayout = "{:>55s}{:>11.2f}"
28  meanSalesLayout = "{:>55s}{:>11.2f}"
29
30  # ---------------------------------------------------------------
31  # Main program
32  # ---------------------------------------------------------------
33
34  # Calculate total sales for each employee and add it
35  #   to the salesTotalPerName array
36  for index in range (len (salesQuarter1)):
37      total = salesQuarter1[index] + salesQuarter2[index] + \
38              salesQuarter3[index] + salesQuarter4[index]
39      salesTotalPerName.append (total)
40
41  # Display the headers with a pipe symbol separating columns
42  print (headerLayout.format (headers[0], headers[1], headers[2],
43                              headers[3], headers[4], headers[5],
44                              headers[6]))
45  # Display a divider for the width of the table
46  print ("-"*TABLE_WIDTH)
```

Develop the code for each comment

The comment says to run the total of all sales, so one more loop is required.
Run, Test, Fix.

One more format string needed for the total for the whole year.
Run, Test, Fix.

Calculation of the mean needs to be done. There is no formula provided.
Run, Test, Fix.

One last format string needed to output the mean sales.
Run, Test, Fix.

```
 9  headers = ["Name", "ID", "Q 1", "Q 2", "Q 3",
10             "Q 4", "Yr Total"]
11  names = ["Adams", "Baker", "Collins", "Dalton", "East", "Ford",
12           "Green", "Hill"]
13  empID = [434, 161, 427, 285, 460, 889, 275, 789]
14  salesQuarter1 = [942.45, 1566.99, 924.59, 197.71, 764.20,
15                   279.43, 867.03, 880.43]
16  salesQuarter2 = [865.78, 337.10, 1597.64, 171.13, 552.89,
17                   495.23, 637.09, 469.96]
18  salesQuarter3 = [973.25, 466.54, 288.54, 979.36, 2780.42,
19                  898.52, 522.45, 979.11]
20  salesQuarter4 = [535.84, 919.83, 661.63, 852.07, 315.02,
21                  120.78, 2748.53, 755.71]
22  salesTotalPerName = []
23  salesYearTotal = 0.0
24  salesMean = 0.0
25  headerLayout = "{:<9s}|{:^5s}|{:^9s}|{:^9s}|{:^9s}|{:^9s}|{:^10s}"
26  rowLayout =
    "{:<9s}|{:^5d}|{:>9.2f}|{:>9.2f}|{:>9.2f}|{:>9.2f}|{:>10.2f}"
27  yearTotalLayout = "{:>55s}{:>11.2f}"
28  meanSalesLayout = "{:>55s}{:>11.2f}"
```

```
57  # Calculate total of all sales for all employees for the year
58  for index in range (len (salesTotalPerName)):
59      salesYearTotal = salesYearTotal + salesTotalPerName[index]
60
61  # Display a label and the total sales for the year
62  print (yearTotalLayout.format ("Year Total", salesYearTotal))
63
64  # Calculate mean of the total sales for the year
65  salesMean = salesYearTotal / len (names)
66
67  # Display a label and the mean of the total sales for the year
68  print (meanSalesLayout.format ("Mean Sales", salesMean))
69
```

Mark scheme

Answer
Award marks as shown. **Calculating sales for each employee** • Loop used to process all items in attempt to calculate sales for each employee (1) • Length of array with 8 elements used to bound loop (1) • Total for employee calculated by adding item at same index in all four salesQuater arrays (1) • Total added to salesTotalPerName array (1)

Levels-based mark scheme to a maximum of 3, from:

- Design (3)
- Functionality (3)

[6.3.3] Length of arrays used for bounds on all loops and
<string>.format() (f-strings) attempted for all outputs, even if layout is inaccurate
[6.3.2] Constant used for table width.
[6.4.1] Currencies must be right aligned.

[6.1.5] Translates.
[6.1.3] Translates and executes without runtime errors. Some calculations are accurate.
[6.1.6] Table output is fit for purpose.
Alignment as given in exemplar, even if field widths are inaccurate

Comment-first coding last check

Double check that all of the last instructions (Step 3) have been addressed:

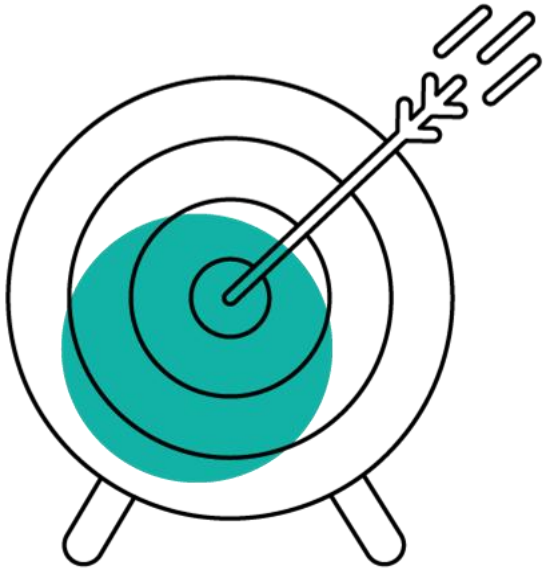
- Must work with any length arrays
- Use the constants and variables in the given code
- Use string formatting

Double check that the output matches the output given in the question paper:

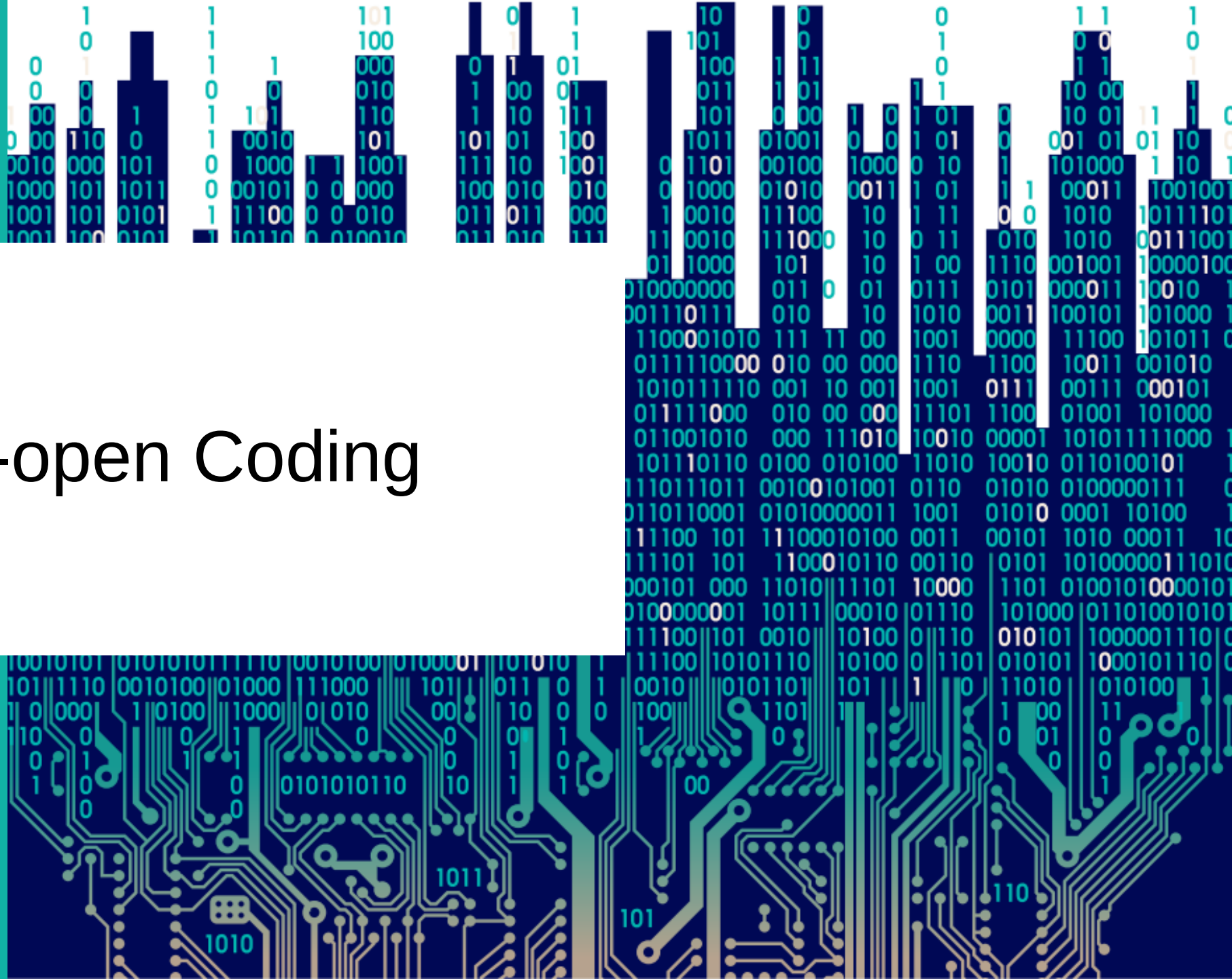
- Columns are aligned as given in the tables of the question paper
- Dividers are present
- Totals are calculated and displayed accurately

Double check that good practice is followed:

- Variables are named consistently and meaningfully at the top of the file
- White space is used to help show the logic of the code
- Comments added where they might be needed to explain what logic



Extended-open Coding



Pearson
Edexcel

Accessing materials

- Open Specimen 4, Paper 2, Question paper
- Open Q06.py in a Python IDE

Q06.py

```
# -----------------------------------------
# Constants
# -----------------------------------------
X = "X"
O = "O"
B = "_"

# -----------------------------------------
# Global variables
# -----------------------------------------
Grid_01 = [[X, B, B],
           [O, O, O],
           [X, B, X]]

Grid_02 = [[X, O, X],
           [O, B, X],
           [O, B, X]]

Grid_03 = [[B, B, X],
           [B, B, B],
           [O, B, B]]

Grid_04 = [[B, B, X],
           [O, X, B],
           [X, B, O]]

Grid_05 = [[O, B, B],
           [X, O, B],
           [X, B, O]]

# -----------------------------------------
# Subprograms
# -----------------------------------------
# ==========> Write your code here


# -----------------------------------------
# Main program
# -----------------------------------------
# ==========> Write your code here

```

Suggested time: 30 minutes

6 A program is being developed to create a noughts and crosses game. The game grid has nine cells. The user symbols are × and O. A blank cell has an underscore (_) symbol in it.

The game functions are being developed by different team members. One part of the development is to find if there is a winner.

A winning symbol is one that appears in three horizontal, vertical or diagonal cells in a line.

The table shows example grids and winners.

× wins	O wins	× wins	No winner
_ O O × × × _ _ _	O × _ O _ × O _ _	× _ O O × _ _ _ ×	_ × O × _ _ O _ _

Open file **Q06.py**

Write a program to meet these requirements:

- process each of the game grids given in the code
- display each game grid by row, no grid lines required
- determine if there is a winner for each game
- display the winning symbol and an appropriate message when there is a winner
- display an appropriate message when there is no winner
- include at least one user-devised subprogram.

Use comments, white space and layout to make the program easier to read and understand.

Do **not** add any additional functionality.

Save your amended code as **Q06FINISHED.py**

(Total for Question 6 = 15 marks)

Extended-open coding: Analysis

1	Read the question	Pick out the most important pieces of information to help understand the problem
2	Read the requirements for amending the code (after ‘Open file **Q06.py**’)	Make note of any requirements, variables, or information that need to be kept track of
3	Read the last few instructions	Make note of any instructions that are needed in the response

1. Read the question

Suggested time: 30 minutes

6 A program is being developed to create a noughts and crosses game. The game grid has nine cells. The user symbols are X and O. A blank cell has an underscore (_) symbol in it.

The game functions are being developed by different team members. One part of the development is to find if there is a winner.

A winning symbol is one that appears in three horizontal, vertical or diagonal cells in a line.

The table shows example grids and winners.

X wins	O wins	X wins	No winner
_ O O X X X _ _ _	O X _ O _ X O _ _	X _ O O X _ _ _ X	_ X O X _ _ O _ _

Familiar game.
X and O are usual symbols.
_ for blank is unexpected.

Finding the winner seems to be the focus.

The rule for finding a winner matches the familiar rules of the game.

Test data is provided.
Is it sufficient?

2. Read the requirements

Open file **Q06.py**

Write a program to meet these requirements:

- process each of the game grids given in the code
- display each game grid by row, no grid lines required
- determine if there is a winner for each game
- display the winning symbol and an appropriate message when there is a winner
- display an appropriate message when there is no winner
- include at least one user-devised subprogram.

Only need to process the grids given.
No user input required.

Display each grid but can ignore the grid lines.

Find the winner for each grid.
Display the winning symbol, if there is a winner.
Display a message, if there is no winner.

A non-functional requirement that means a subprogram.

3. Read the last instructions

Use comments, white space and layout to make the program easier to read and understand.

Do **not** add any additional functionality.

Save your amended code as **Q06FINISHED.py**

(Total for Question 6 = 15 marks)

Be sure to check for and use the constants and variables already defined in the code file.

Make sure to use the correct alignment for the columns. Use the output shown in the question paper.

Standard instructions.
Just focus on what has been asked.
Don't get distracted.

4. Read the Python code file

Remember to use these constants.

Test data is hard coded here.

Subprograms should be separated from the main program code.

Q06.py

```
# ------------------------------------------------------------------
# Constants
# ------------------------------------------------------------------
X = "X"
O = "O"
B = "_"

# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
Grid_01 = [[X, B, B],
           [O, O, O],
           [X, B, X]]

Grid_02 = [[X, O, X],
           [O, B, X],
           [O, B, X]]

Grid_03 = [[B, B, X],
           [B, B, B],
           [O, B, B]]

Grid_04 = [[B, B, X],
           [O, X, B],
           [X, B, O]]

Grid_05 = [[O, B, B],
           [X, O, B],
           [X, B, O]]

# ------------------------------------------------------------------
# Subprograms
# ------------------------------------------------------------------
# ==========> Write your code here


# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# ==========> Write your code here

```

Extended-open coding: Design & Develop

A	Design a part
B	Code the part
C	Test the part
D	Refine/fix the part
E	Repeat any of A, B, C, D until the part is working

Suggestions:

- Get the solution to work.
- Perfecting the output can come last.
- Remove all syntax and runtime errors.

A. Design the big picture

Add design comments into the code file.

Consider decomposition (subprograms, blocks).

Subprograms should be separated from the main program code.

Put in enough to test the subprograms.

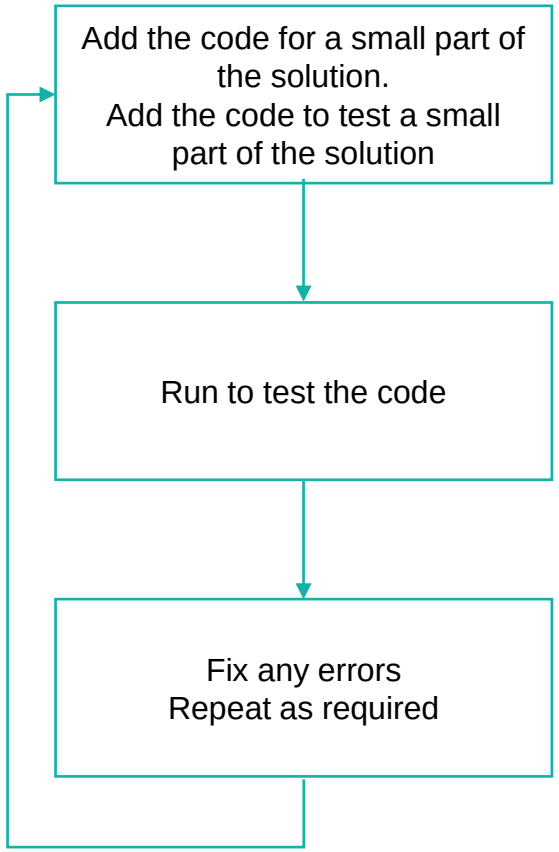
```
31  # ---------------------------------------------------------------
32  # Subprograms
33  # ---------------------------------------------------------------
34  # ==========> Write your code here
35
36  # Display a grid.
37  #   Pass the grid as an input parameter.
38  #   No return value.
39
40  # Check rows.
41  #   Pass the grid as an input parameter.
42  #   Return value is the symbol of the winner, if found
43  #   Return value is the blank symbol, if no winner found
44
45  # Check columns.
46  #   Pass the grid as an input parameter.
47  #   Return value is the symbol of the winner, if found
48  #   Return value is the blank symbol, if no winner found
49
50  # Check diagonals.
51  #   Pass the grid as an input parameter.
52  #   Return value is the symbol of the winner, if found
53  #   Return value is the blank symbol, if no winner found
54
55  # ---------------------------------------------------------------
56  # Main program
57  # ---------------------------------------------------------------
58  # ==========> Write your code here
59
60  # Display Grid_05
61  # Check rows Grid_01
62  # Check columns Grid_02
63  # Check diagonals Grid_04
64
```

B. Code a small part of a solution

1. Work on displaying the grid

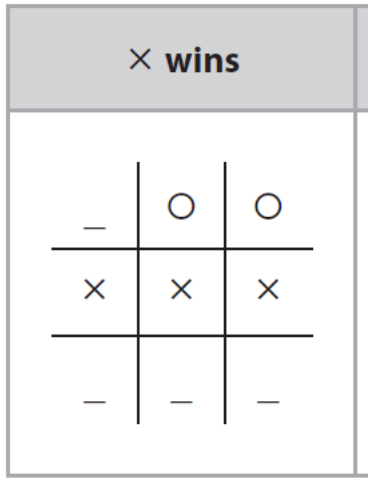
```
31  # ---------------------------------------------------------
32  # Subprograms
33  # ---------------------------------------------------------
34  # ==========> Write your code here
35
36  # Display a grid.
37  #   Pass the grid as an input parameter.
38  #   No return value.
39  def displayGrid (pGrid):
40      for row in pGrid:
41          print (row)
```

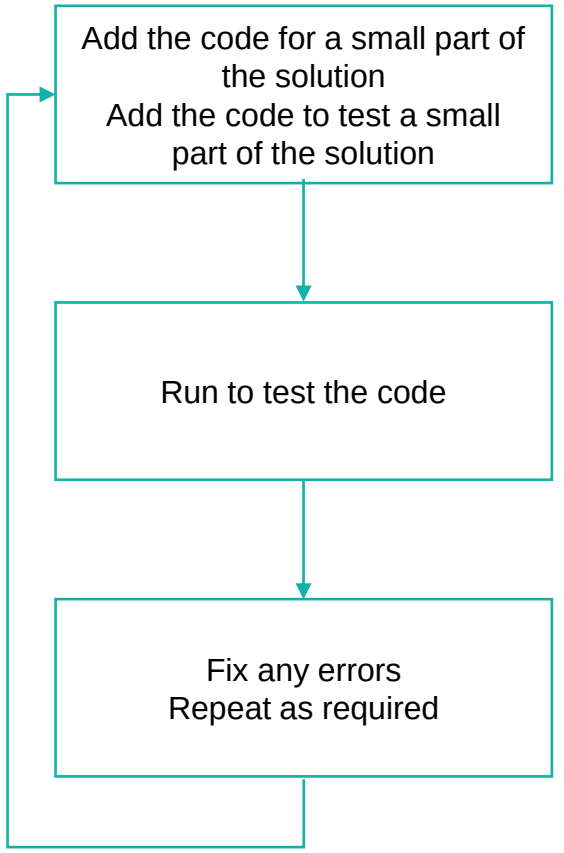
```
74  # ---------------------------------------------------------
75  # Main program
76  # ---------------------------------------------------------
77  # ==========> Write your code here
78
79  # Display Grid_05
80  displayGrid (Grid_05)
81
```



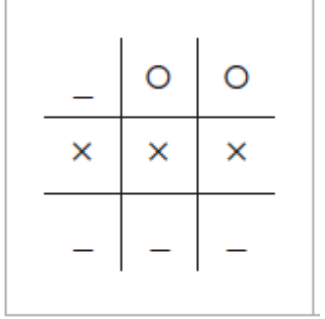
B. Code a small part of a solution

1. Work on displaying the grid
2. Work on checking the rows for a winner





A. Design a small part of a solution
B. Code a small part of a solution




```
43  # Check rows.
44  #   Pass the grid as an input parameter.
45  #   Return value is the symbol of the winner, if found
46  #   Return value is the blank symbol, if no winner found
47  def checkRows (pGrid):
48      winner = B                  # No winner to start
49      foundWinner = False
50      row = 0
51
52      # Go down each of the three rows
53      while ((row < 3) and (not foundWinner)):
54          # Check if match all three and not blank
55          if ((pGrid[row][0] == pGrid[row][1]) and
56              (pGrid[row][0] == pGrid[row][2]) and
57              (pGrid[row][0] != B)):
58              foundWinner = True
59              winner = pGrid[row][0]
60          row = row + 1
61
62      return (winner)
```

Blank is the default value to return (no winner). We stop if found.

Look at all three rows in turn. If found, then can exit the loop.

Compare 1st cell to 2nd cell. Compare 2nd cell to 3rd cell. Make sure the cell is not blank.

If found, stop the loop and set the winner to the symbol used in the row.

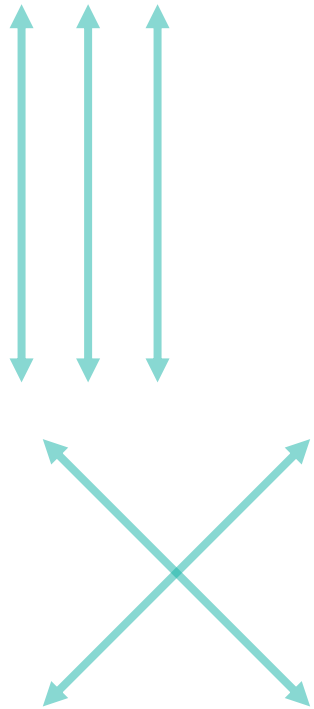
Point to the next row.

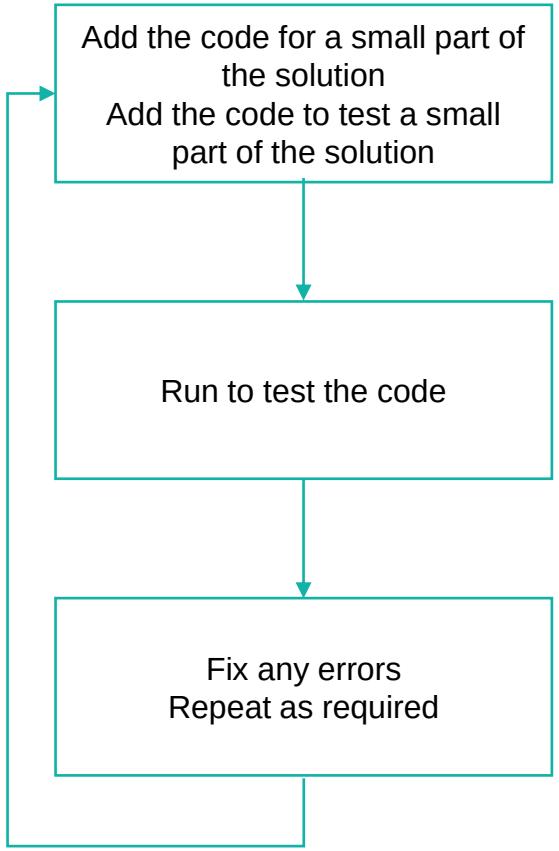
Return the winner to the calling part of the program.

B. Code a small part of a solution

1. Work on displaying the grid
2. Work on checking the rows for a winner
3. Work on checking the columns for a winner
4. Work on checking the diagonals for a winner

○ wins	× wins
○ × _ ○ _ × ○ _ _	× _ ○ ○ × _ _ _ ×





A. Design a small part of a solution
B. Code a small part of a solution

```
73  def checkDiagonals (pGrid):
74      winner = B                  # No winner to start
75
76      # Check left to right
77      if ((pGrid[0][0] == pGrid[1][1]) and
78              (pGrid[0][0] == pGrid[2][2]) and
79              (pGrid[0][0] != B)):
80          winner = pGrid[0][0]
81
82      # Check right to left
83      elif ((pGrid[0][2] == pGrid[1][1]) and
84              (pGrid[0][2] == pGrid[2][0]) and
85              (pGrid[0][2] != B)):
86          winner = pGrid[0][2]
87
88      return (winner)
```

Blank is the default value to return (no winner)
We stop if found

Look at all three cells going left to right

Look at all three cells going right to left

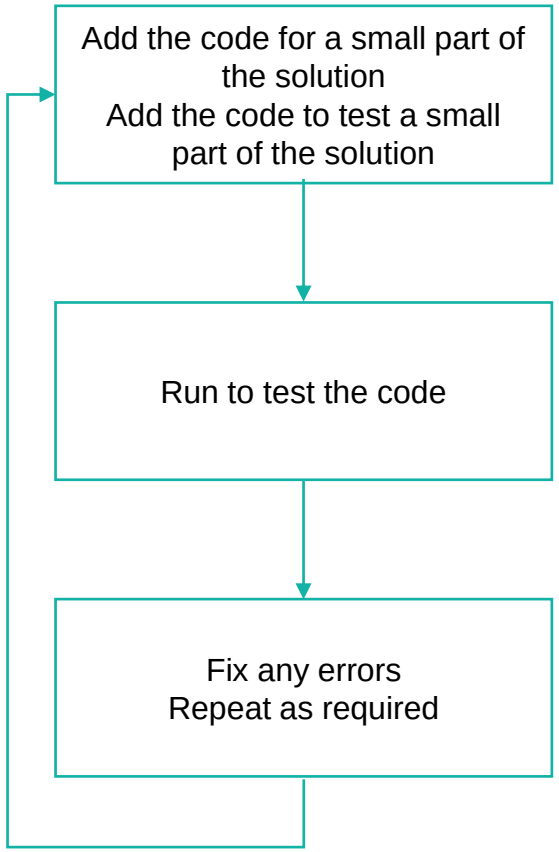
Return the winner to the calling part of the program.

× wins

×	_	○
○	×	_
_	_	×

B. Code a small part of a solution

1. Work on displaying the grid
2. Work on checking the rows for a winner
3. Work on checking the columns for a winner
4. Work on checking the diagonals for a winner
5. Display the winner symbol and a message
6. Process all the grids



A. Design a small part of a solution
B. Code a small part of a solution

```
90  def displayWinner (pWinner):
91      if (pWinner == B):
92          print ("No winner")
93      elif (pWinner == X):
94          print ("Winner: ", X)
95      elif (pWinner == O):
96          print ("Winner: ", O)
97      else:
98          print ("Unknown error")
99
100 # Main processing loop for each grid
101 def processGrid (pGrid):
102     theWinner = ""
103
104     displayGrid (pGrid)
105     theWinner = checkRows (pGrid)
106     if (theWinner == B):
107         theWinner = checkColumns (pGrid)
108         if (theWinner == B):
109             theWinner = checkDiagonals (pGrid)
110     displayWinner (theWinner)
111
112 # ---------------------------------------------------------------
113 # Main program
114 # ---------------------------------------------------------------
115 # ==========> Write your code here
116 processGrid (Grid_01)       # Every grid
117 processGrid (Grid_02)
118 processGrid (Grid_03)
119 processGrid (Grid_04)
120 processGrid (Grid_05)
```

Display a message based on the winning symbol given as a parameter.

Control the sequence:
Display the grid
Check the rows
Check the columns
Check the diagonals
Display the winner

Control the sequence of processing each grid.

Mark scheme

Award marks as shown.

- Looping for displaying the grid (1)

  - `for row in pGrid:`

  For loop only

  Allow any row-oriented output:

```
['X', '_', '_']
['O', 'O', 'O']
['X', '_', 'X']
```

```
X  _  _
O  O  O
X  _  X
```

Levels-based mark scheme to a maximum of 3, from:

- Design (3)

  [6.1.2] While loop with Booleans, rather than for loop with break/return
  [6.6.2] Use of parameters in subprogram and arguments on the call
  [6.1.1] Fully decomposed with a variety of subprograms to ensure code is not duplicated and the number of global variables is minimised.

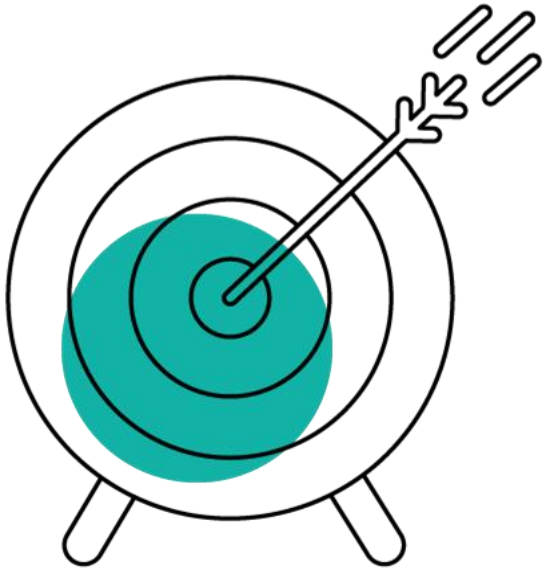
Extended-open coding last check

Double check that the output meets requirements:

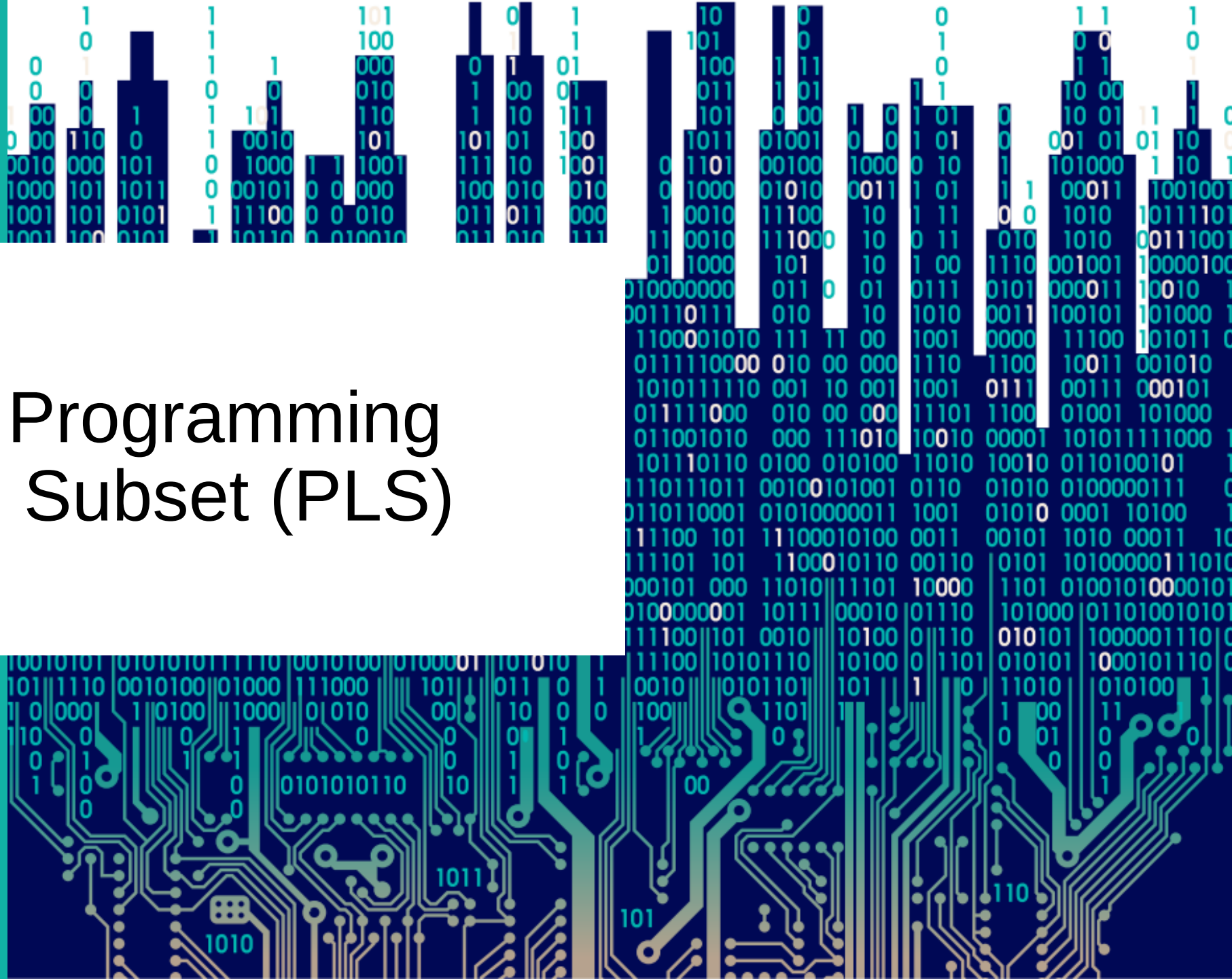
- The grids are displayed
- Winners are displayed
- At least one user-devised subprogram

Double check that good practice is followed:

- Variables are named consistently and meaningfully at the top of the file
- White space is used to help show the logic of the code
- Comments added where they might be needed to explain logic



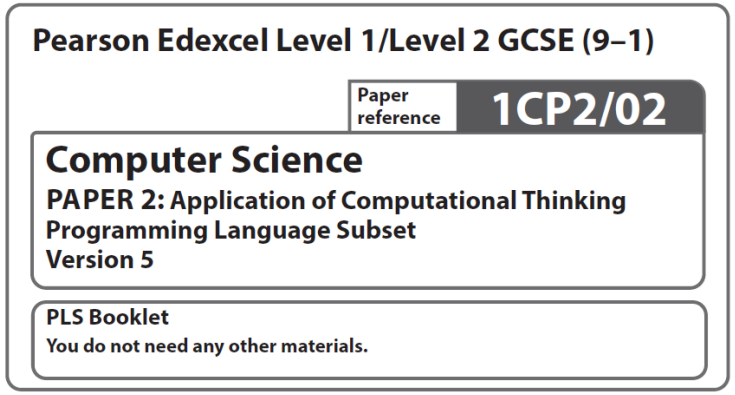
Using the Programming Language Subset (PLS)



Pearson
Edexcel

PLS: What is it?

- The PLS specifies which parts of Python 3 are used to set questions.
- Students familiar with everything in this document will be able to access all parts of the Paper 2 assessment.
- Teachers may go beyond the scope of the PLS into techniques and approaches that they may consider to be more efficient or engaging.
- Pearson will **not** go beyond the scope of the PLS when setting assessment tasks.
- Any student successfully using more esoteric or complex constructs or approaches not included in this document will still be awarded marks in Paper 2 if the solution is valid.



Q01

Suggested time: 10 minutes

1 A program is being developed to show the pattern of traffic light colours.

The pattern is Red, Red and Amber, Green, Amber.

The program displays the full pattern three times.

Open file **Q01.py**

Amend the lines at the bottom of the code to give the:

- name of a library used in this program
- name of an array used in this program
- line number of a variable initialisation
- line number of a repetition
- name of a data type conversion function used in this program
- name of a built-in subprogram used on line 21
- name of an arithmetic operator used in this program.

Do **not** add any additional functionality.

Save your amended code file as **Q01FINISHED.py**

(Total for Question 1 = 7 marks)

Repetition, the while loop is explained on page 6.

Repetition

```
while <condition>:
    <command>
```

Pre-
<co

All operators are found on page 6.

Arithmetic operator	Meaning
/	division
*	multiplication
**	exponentiation
+	addition
–	subtraction
//	integer division
%	modulus

Q02

Open file **Q02.py**

Amend the code to:

- fix the syntax error on original line
  ```
  numYears =
  ```
- fix the syntax error on original line 11
  ```
  numYears = float (input ("How many years do you want? ")
  ```
- fix the syntax error on original line 20
  ```
  else
  ```
- fix the TypeError on original line 11
  ```
  numYears = float (input ("How many years do you want? ")
  ```
- fix the NameError on original line 13
  ```
  for count in rang (numYears):
  ```
- fix the NameError on original line 14
  ```
  theyear = int (input ("Enter a year: "))
  ```
- fix the NameError on original line 23
  ```
  print (Goodbye)
  ```

Looking up float or input, both on p. 9, will show how the brackets are used.

```
float (<item>)
```

Q03

```
52          # =====> Choose the correct line to validate the input
53          #if (not choiceStr.isdigit()):
54          #if (choiceStr.isalnum()):
55          #if (choiceStr.isalpha()):
56          #if ("{:<8.2f}".format(choiceStr):
57              error = True
58          else:
59              choiceInt = int (choiceStr)
```

Looking up the string-handling subprograms can help determine which is required to make the logic work.

Q04

```
18  # ---------------------------------------------------------------
19  # Main program
20  # ---------------------------------------------------------------
21
22  # =====> Complete the line to open the file for reading
23  theFile = open
24
25  # =====> Complete the line to read every line from the file
26  for
27
28      # =====> Complete the line to remove the line feed
29      record =
30
31      # =====> Complete the line to separate the record into fields
32      fields =
33
```

On p. 8, in the PLS, all the file-handling subprograms are set out.

The split subprogram is found on p. 11.

Gender Gap in Computer Science

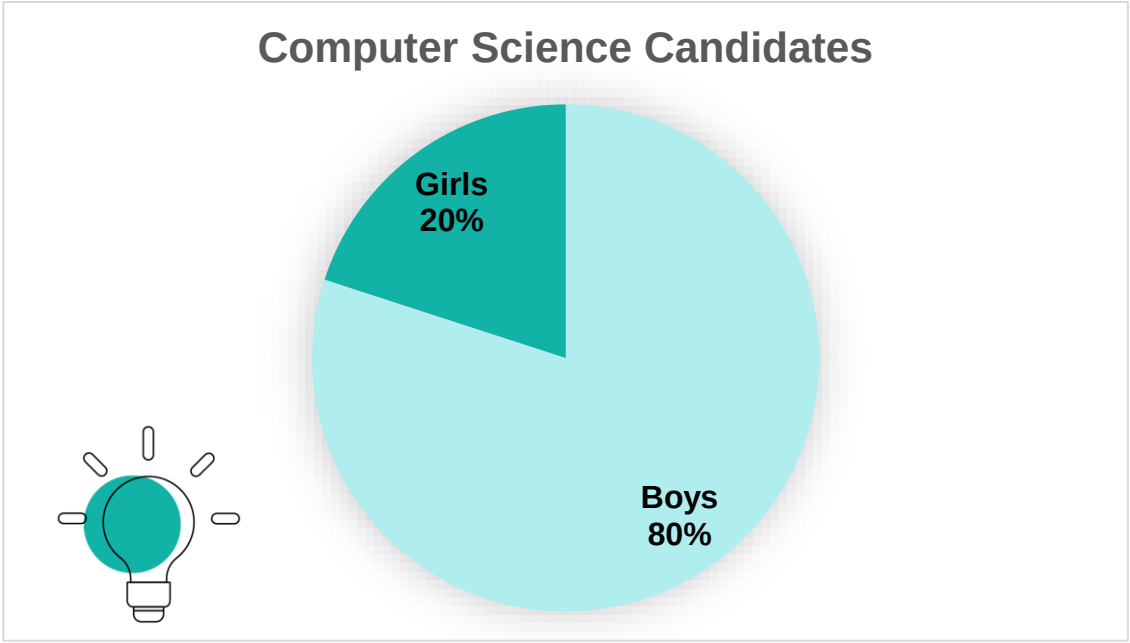
Pearson Edexcel

Gender Gap in Computer Science

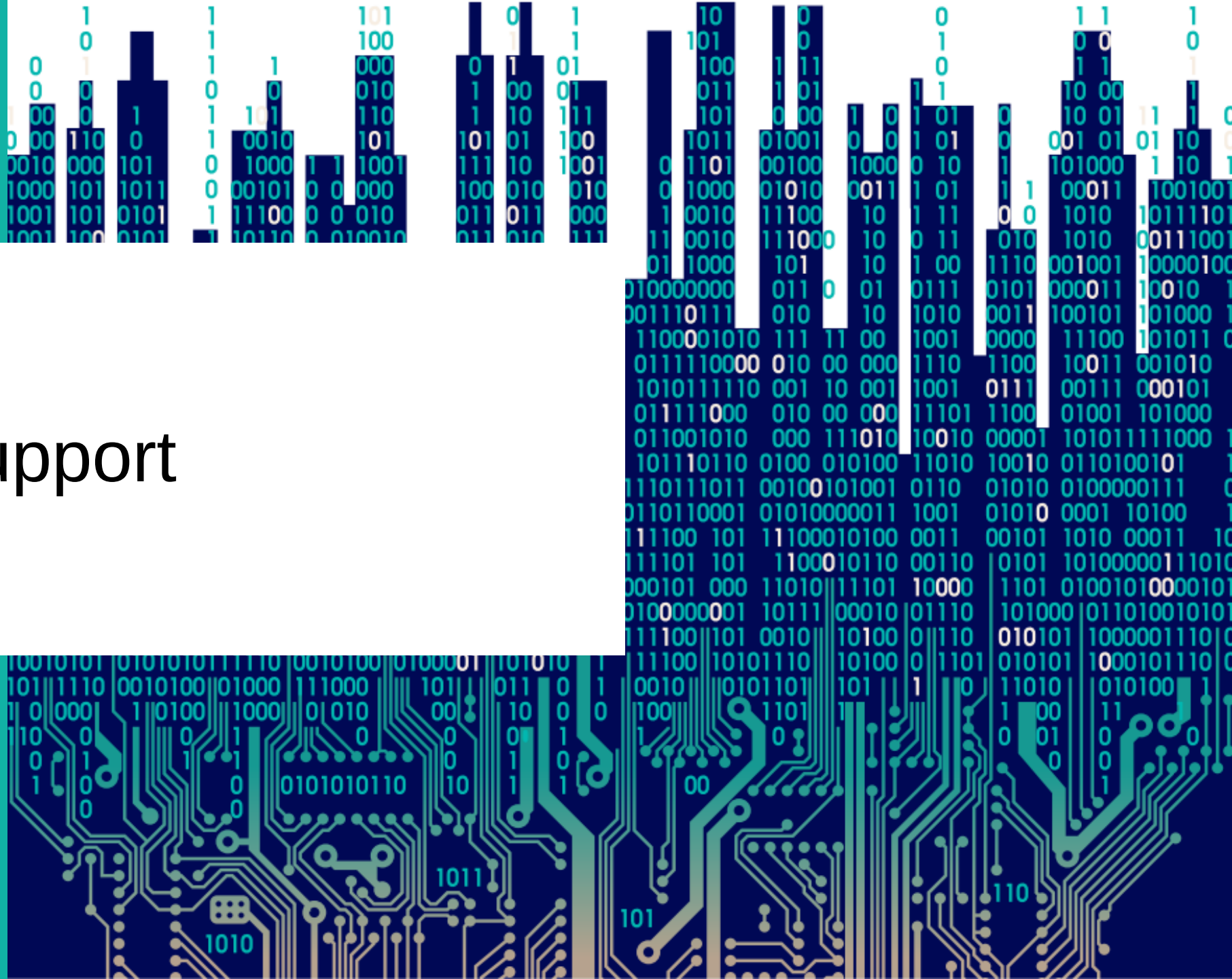
Pearson are taking up the challenge to improve uptake of girls in tech.

Visit the webpage to find:

- Women in computing talking about their own inspirational career journeys
- Organisations addressing careers and removing barriers to engaging girls.



Further support



Pearson
Edexcel

Specimen materials

Paper 1

- Question paper
- Mark scheme

Paper 2

- Question paper
- Programming language subset
- Python code files for the student
- Mark scheme
- Python code files for the code shown in the mark scheme
- Tips for how to write your own questions based on the patterns in the specimen

Assessment – teacher training

An additional package for each Specimen set.

Each Paper 2 question has:

- Pointers to more information about the specific items assessed
- Two or more student responses, in the form of Python code files
- A detailed mark explanation, including justification for the marks awarded and further information that you and students might find helpful.

Subject Advisor Support

Our subject advisors are experts in their fields and are here to support you throughout the year.

Computer Science

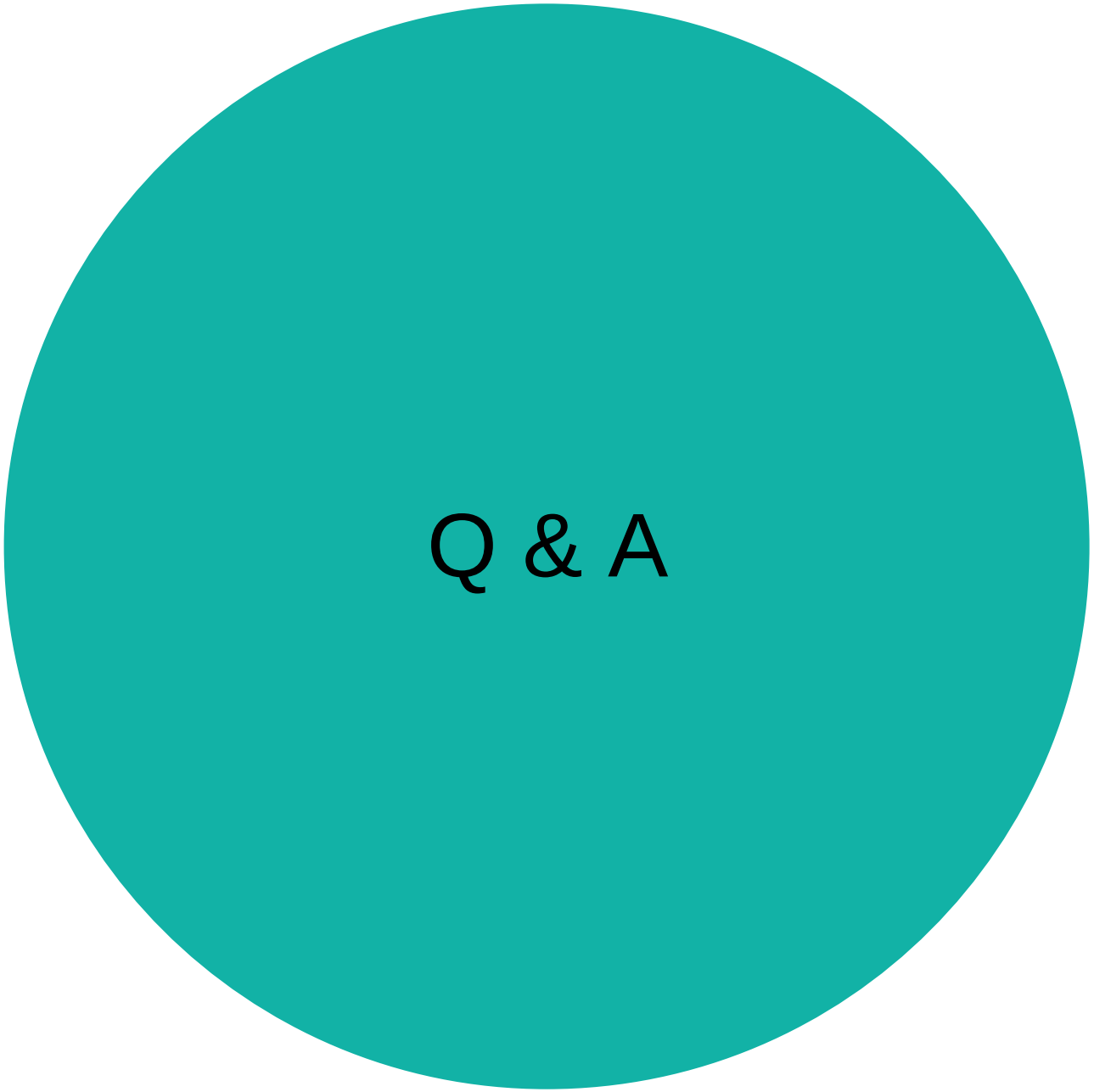
Email: TeachingComputerScience@pearson.com

Phone: +44 (0) 344 463 2535
(Mon–Fri, 9.00–17.00)

Book an appointment with your Subject Advisor

Sign up to receive regular updates from your Subject Advisor on qualification news and support for your subject.

Tim Brady
Computer Science and ICT


Q & A

Pearson

Find out more

For more professional development courses please see Pearson's Professional Development Academy

Your Feedback Matters

Following this event, you will receive an invitation to share your thoughts about the session. Your feedback is invaluable to us, as it helps us tailor our professional development materials to better meet your needs. Please don’t hesitate to let us know what you’d like to see more of and what areas you think could be improved.

Pearson