Pearson
Edexcel

Mark Scheme (Results)

June 2022

Pearson Edexcel GCSE In
Computer Science (1CP2/02)
Paper 2: Application of Computational
Thinking

Edexcel and BTEC Qualifications

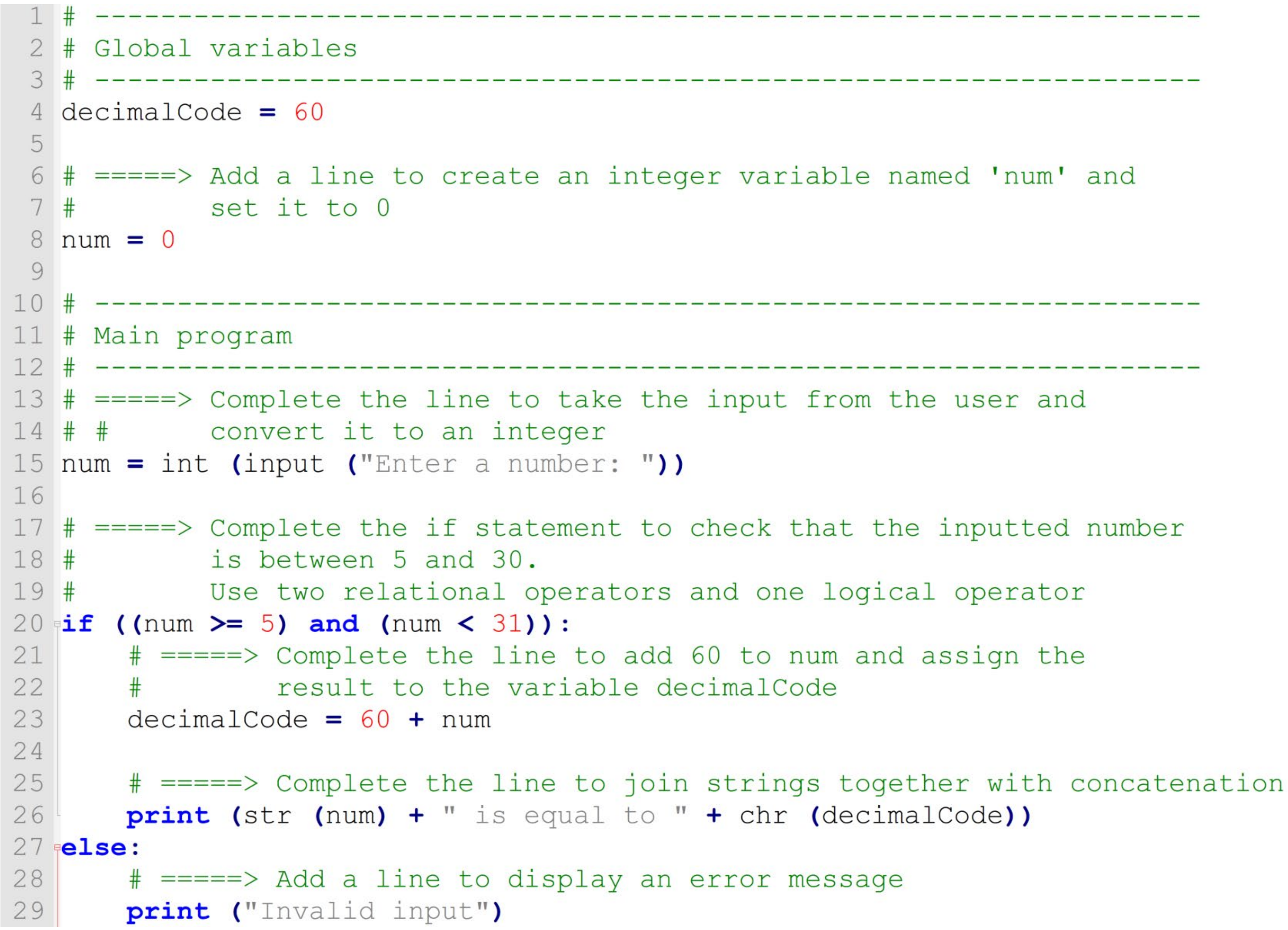
Pearson: helping people progress, everywhere

June 2022
Publications Code 1CP2_02_rms_20220825

General Marking Guidance

- All candidates must receive the same treatment. Examiners must mark the first candidate in exactly the same way as they mark the last.
- Mark schemes should be applied positively. Candidates must be rewarded for what they have shown they can do rather than penalised for omissions.
- Examiners should mark according to the mark scheme not according to their perception of where the grade boundaries may lie.
- There is no ceiling on achievement. All marks on the mark scheme should be used appropriately.
- All the marks on the mark scheme are designed to be awarded. Examiners should always award full marks if deserved, i.e. if the answer matches the mark scheme. Examiners should also be prepared to award zero marks if the candidate’s response is not worthy of credit according to the mark scheme.
- Where some judgement is required, mark schemes will provide the principles by which marks will be awarded and exemplification may be limited.
- When examiners are in doubt regarding the application of the mark scheme to a candidate’s response, the team leader must be consulted.
- Crossed out work should be marked UNLESS the candidate has replaced it with an alternative response.

Question number	MP	Appx. Line	Answer	Additional guidance	Mark
1			Award marks as shown.		
	1.1	8	New line added to create integer and setting it to 0 (1)	• num = 0 • num = int () num = 0	
	1.2	15	Use of input (<prompt>) to display a prompt (1)	• Allow input("")	
	1.3	15	Conversion of string input to integer using int () (1)		
	1.4	20	Use of correct variable and relational operator for lower bound (1)	• num > 4 • num >= 5 • 5 <= num • Allow num > 5 • Allow =<	
	1.5	20	Use of correct variable and relational operator for upper bound (1)	• num < 31 • num <= 30 • 30 >= num • Allow num < 30 • Allow =>	
	1.6	20	Use of AND to join the range checks (1)	• Allow the use of OR to correctly exclude invalid inputs. Review mp1.4 and 1.5 to check relational operators matches logic.	
	1.7	23	Use of addition to convert to decimal code (1)	• 60 + num • decimalCode + num • Allow + as part of compound operator,	
	1.8	23	Use of assignment to set value of decimalCode (=) (1)	• Allow = as part of compound operator, independent of mp1.7	
	1.9	26	String concatenation used to join parts of string output (1)		
	1.10	29	Invalid input message displayed is fit for purpose (1)		**(10)**



```python
# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
decimalCode = 60

# =====> Add a line to create an integer variable named 'num' and
#        set it to 0
num = 0

# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# =====> Complete the line to take the input from the user and
# #      convert it to an integer
num = int (input ("Enter a number: "))

# =====> Complete the if statement to check that the inputted number
#        is between 5 and 30.
#        Use two relational operators and one logical operator
if ((num >= 5) and (num < 31)):
    # =====> Complete the line to add 60 to num and assign the
    #        result to the variable decimalCode
    decimalCode = 60 + num

    # =====> Complete the line to join strings together with concatenation
    print (str (num) + " is equal to " + chr (decimalCode))
else:
    # =====> Add a line to display an error message
    print ("Invalid input")
```

Question number	MP	Appx. Line	Answer	Additional guidance	Mark
2			Award marks as shown.		**(10)**
	2.1	19	Any comment with the word “string” in it near the turtle.mode () call (1)		
	2.2	23	Name error – correct spelling of constant HEIGHT (1)		
	2.3	28	Attribute error – Requires a capital letter <turtle>.Turtle () (1)		
	2.4	36	Type error – Remove argument to <turtle>.pendown () (1)		
	2.5	42	Logic error – Move vertical grid line back to origin (1) `theTurtle.setpos (0, 200)`		
	2.6	48	Logic error – Correct length of vertical grid line (1) `theTurtle.forward (400)`		
	2.7	56	Logic error – Correct heading for starting point of square (1) `theTurtle.setheading (90)`		
	2.8	68	Control pen size with a constant (1) `theTurtle.pensize (BIG)`	• Do not allow mark for first added line that uses the default ‘turtle’ rather than ‘theTurtle’, even if the rest of the line is correct. Allow follow through.	
	2.9	71	Set the pen colour to “gold” (1) `theTurtle.pencolor ("gold")`		
	2.10	78	Hide the turtle (1) `theTurtle.hideturtle ()`		

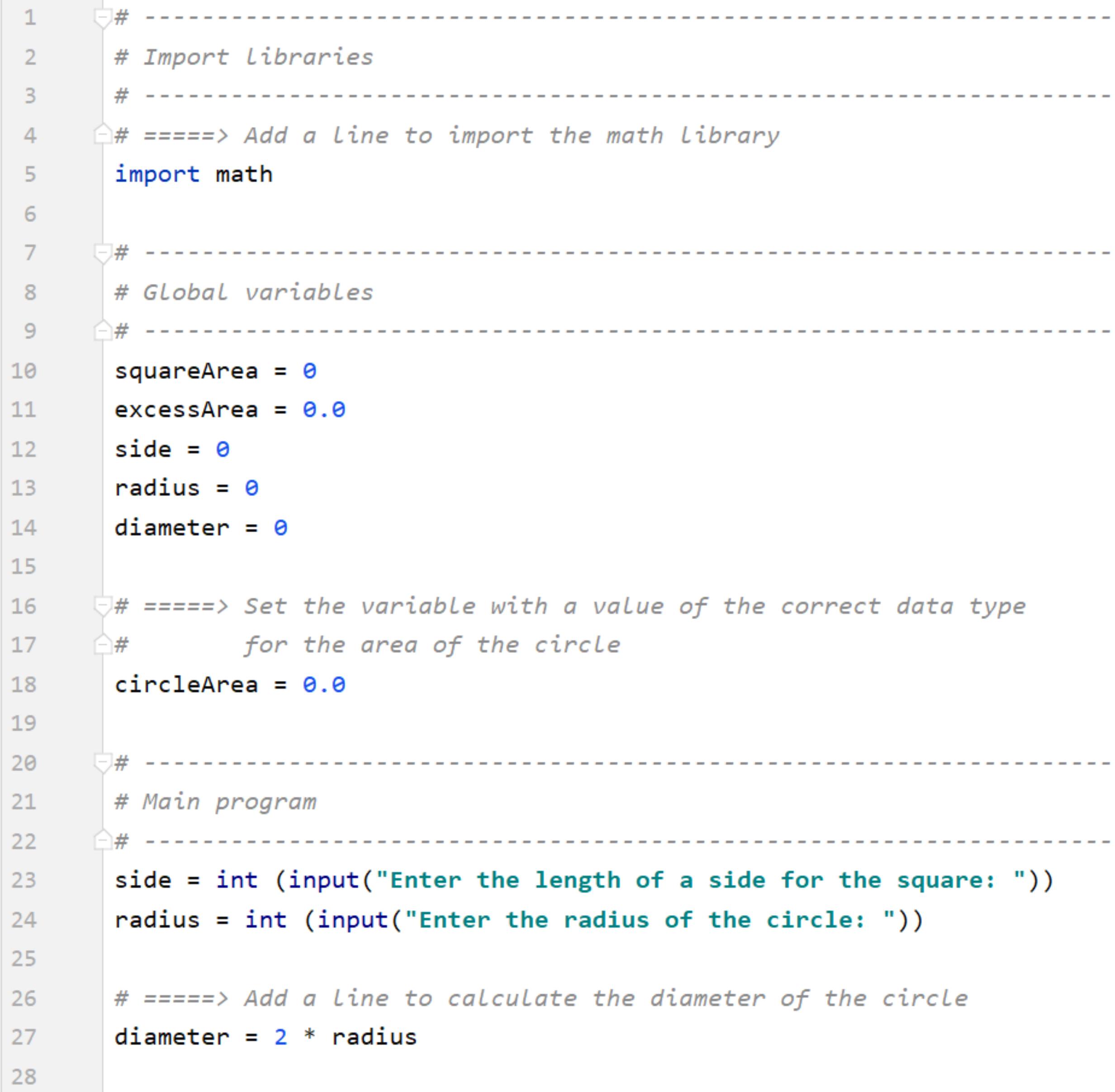
```python
# Import libraries
# -------------------------------------------------------------------
import turtle

# -------------------------------------------------------------------
# Constants
# -------------------------------------------------------------------
WIDTH = 800
HEIGHT = 600
BIG = 8

# -------------------------------------------------------------------
# Main program
# -------------------------------------------------------------------
# Setup the turtle environment
# =====> Add a comment to identify the data type of the argument
#        to the turtle.mode () subprogram
turtle.mode ("standard")            # A string
screen = turtle.Screen ()

# =====> Fix the NameError
screen.setup (WIDTH, HEIGHT)
turtle.screensize (WIDTH, HEIGHT)

# Prepare the turtle
# =====> Fix the AttributeError
theTurtle = turtle.Turtle ()                # Create a turtle
theTurtle.penup ()

# Draw grid lines
theTurtle.setpos (-200, 0)
theTurtle.setheading (0)

# =====> Fix the TypeError
theTurtle.pendown ()
theTurtle.forward (400)
theTurtle.penup ()

```

```python
# ====> Fix the logic error that causes the vertical axis to be
#       too far right
theTurtle.setpos (0, 200)
theTurtle.setheading (270)
theTurtle.pendown ()

# =====> Fix the logic error that causes the vertical axis
#        to be drawn too short
theTurtle.forward (400)
theTurtle.penup ()

# Draw a square
theTurtle.setpos (-200, -200)               # Lower left

# =====> Fix the logic error that makes the outside square
#        tilt left of the vertical axis
theTurtle.setheading (90)                   # Point north
theTurtle.pendown ()
for count in range (4):
    theTurtle.forward (400)                 # Side
    theTurtle.right (90)                    # Turn
theTurtle.penup ()

# Draw a circle
theTurtle.setpos (100, 0)                   # Right side of circle
theTurtle.setheading (90)                   # Point north

# ====> Add a line to set the size of the pen to the constant BIG
theTurtle.pensize (BIG)

# =====> Add a line to set the colour of the pen to gold
theTurtle.pencolor ("gold")

theTurtle.pendown ()
theTurtle.circle (100)                      # Radius of 100
theTurtle.penup ()

# =====> Add a line to hide the turtle
theTurtle.hideturtle ()

print ("Be sure to close the turtle window.")
turtle.done ()
```

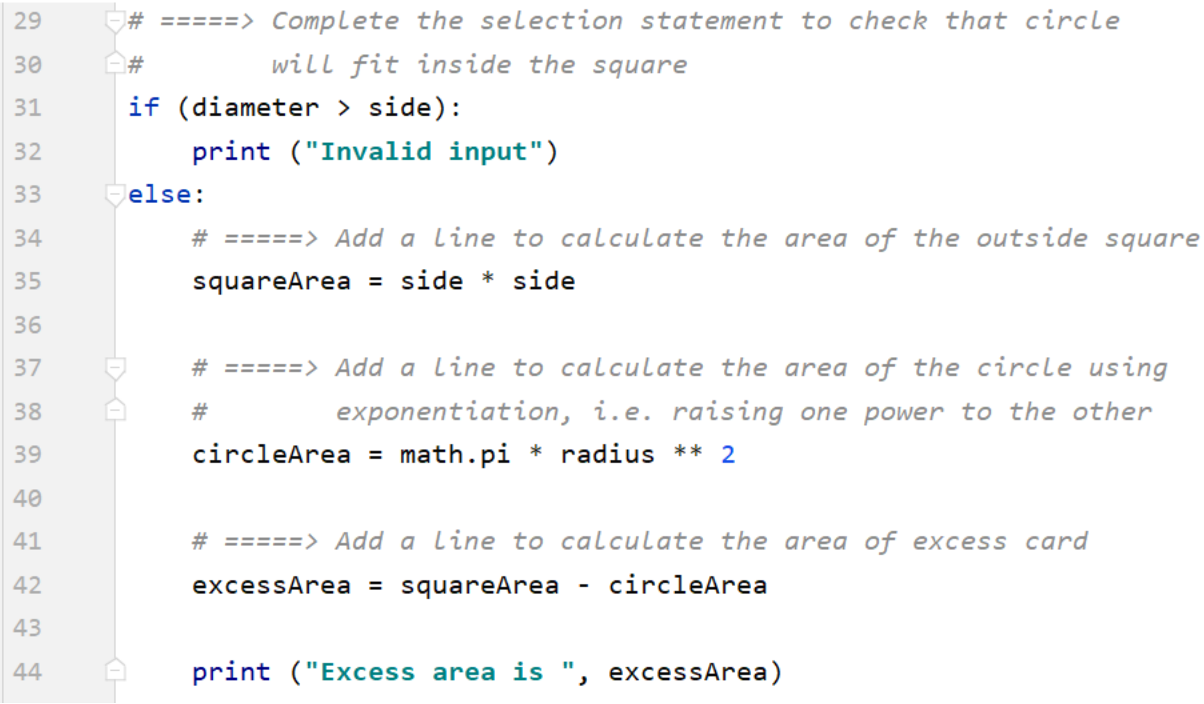
Question number	MP	Appx. Line	Answer	Additional guidance	Mark
3			Award marks as shown.		
	3.1	5	Import the math library (1)	• import math • from math import pi • from math import pow	
	3.2	18	Initialise ‘circleArea’ to a real number (1)		
	3.3	27	Correct translation of diameter calculation and assignment to ‘diameter’ (1)		
	3.4	31	Use of relational operator and two correct variables to construct a test for invalid input (1)	• diameter > side • diameter >= side • Allow comparisons between areas of the circle and areas of the square (circleArea > squareArea)	
	3.5	35	Calculation of the area of the square (1)	• side * side • side ** 2 • math.pow (side, 2)	
	3.6	39	Correct translation of exponentiation (**2) for circle area, even if remainder of formula is incorrect (1)	• radius ** 2 • math.pow (radius, 2)	
	3.7	42	Subtraction used to calculate the positive difference between the area of the square and the area of the circle (1)	• excessArea = squareArea - circleArea	
			Levels-based mark scheme to a maximum of 3, from:		
	3.8 3.9 3.10		Functionality (3) Execute with test data given in question paper.	Considerations for levels-based mark scheme: • [6.1.2] Translates without error, even if reduced functionality • [6.1.6] Functions correctly to produce the required output • [6.6.1] Use of constant (math.pi) in preference to estimated value (3.14…)	**(10)**

Functionality (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	**Functionality (when the code is run)** • The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements. • Program outputs are of limited accuracy and/or provide limited information. • Program responds predictably to some of the anticipated input. • Solution is not robust and may crash on anticipated or provided input.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that meets most of the stated requirements. • Program outputs are mostly accurate and informative. • Program responds predictably to most of the anticipated input. • Solution may not be robust within the constraints of the problem.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that fully meets the given requirements. • Program outputs are accurate, informative, and suitable for the user. • Program responds predictably to anticipated input. • Solution is robust within the constraints of the problem.	3



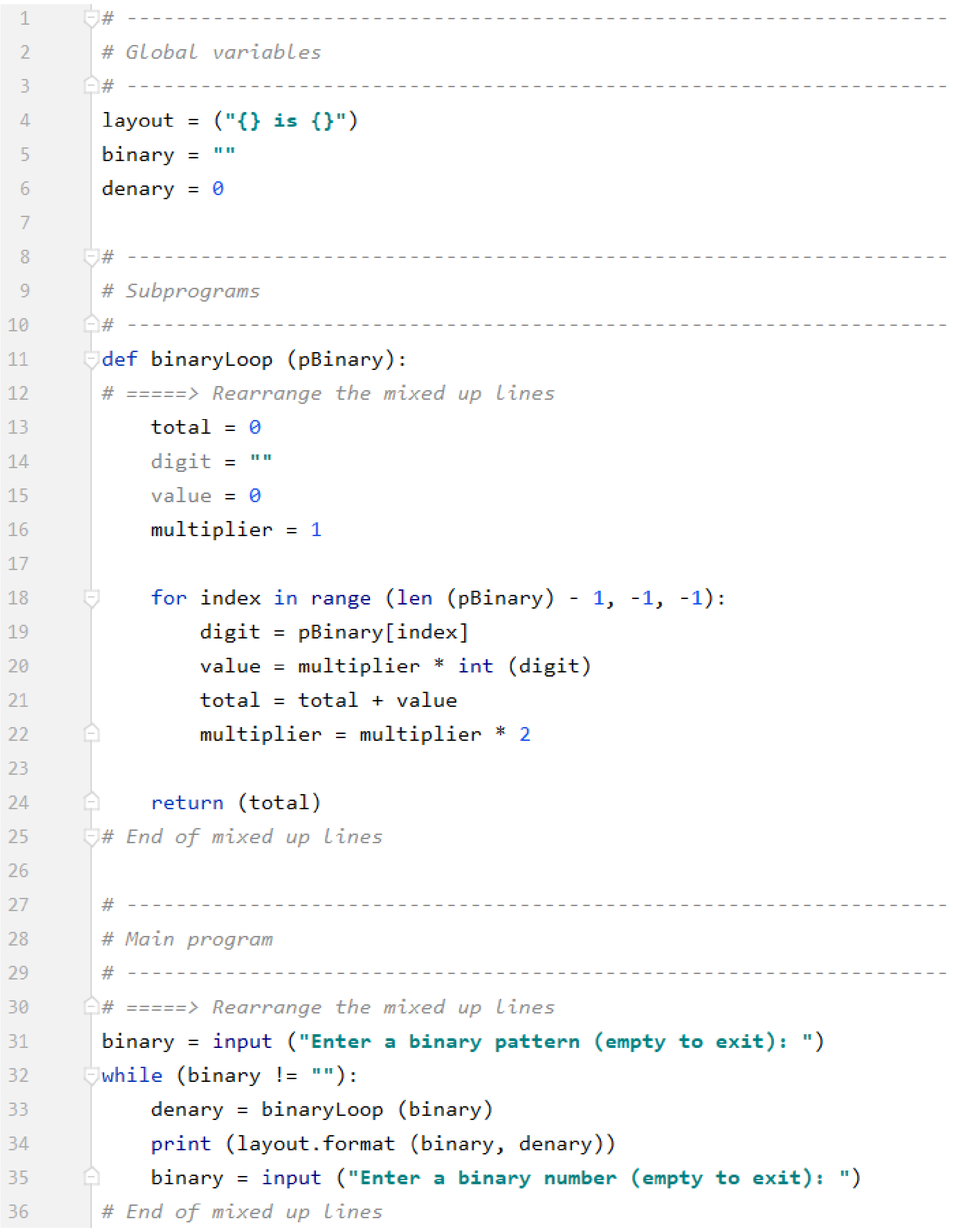
```python
# -------------------------------------------------------------------
# Import libraries
# -------------------------------------------------------------------
# =====> Add a line to import the math library
import math

# -------------------------------------------------------------------
# Global variables
# -------------------------------------------------------------------
squareArea = 0
excessArea = 0.0
side = 0
radius = 0
diameter = 0

# =====> Set the variable with a value of the correct data type
#        for the area of the circle
circleArea = 0.0

# -------------------------------------------------------------------
# Main program
# -------------------------------------------------------------------
side = int (input("Enter the length of a side for the square: "))
radius = int (input("Enter the radius of the circle: "))

# =====> Add a line to calculate the diameter of the circle
diameter = 2 * radius

```



```
# =====> Complete the selection statement to check that circle
#        will fit inside the square
if (diameter > side):
    print ("Invalid input")
else:
    # =====> Add a line to calculate the area of the outside square
    squareArea = side * side

    # =====> Add a line to calculate the area of the circle using
    #        exponentiation, i.e. raising one power to the other
    circleArea = math.pi * radius ** 2

    # =====> Add a line to calculate the area of excess card
    excessArea = squareArea - circleArea

    print ("Excess area is ", excessArea)
```

Question number	MP	Appx. Line	Answer	Additional guidance	Mark
4			Award marks as shown.	• Award sequence only. • Ignore intervening lines. • Ignore changes made to provided lines. • Do not award same sequence in supplied file, if no lines are moved in that subsection	
			Subprogram	• If no lines are moved, not changing the sequence, then award no marks in this subsection	
	4.1 4.2	13	Initialisation of variables before calculations inside the subprogram, all together (One mark for any two, up to a maximum of 2) multiplier, total, digit, value (2)		
	4.3	18	Iteration (for loop) placed at highest level inside subprogram, after initialisation of local variables (1)		
			Order of calculations maintained:	• Order of lines must be digit, followed by value, followed by total. • Multiplier must be after value.	
	4.4	19	digit must be the first of the sequence (1)		
	4.5	20	value (1)		
	4.6	21	total (1)		
	4.7	22	multiplier (1)		
	4.8	24	Return statement is last line in subprogram (1)		
			Main Program	• If no lines are moved, not changing the sequence, then award no marks in this subsection	
	4.9	31	User input accepted as first operation in main program (1)		
	4.10	32	Repetition (while loop) after any input (1)		**(15)**

	4.11	33	Call to ‘binaryLoop’ subprogram inside repetition (while loop) (1)		
	4.12	34	Output of result follows call to subprogram (1)		
	4.13	35	User input accepted as last operation in main program, inside loop (1)		
			Additional		
	4.14		Functions for any sequence of 1s and 0s **and** exits on empty string (1)	• Execute with test data given in question paper	
	4.15		Adherence to accurate indentation (1)		



```python
# ---------------------------------------------------------------------
# Global variables
# ---------------------------------------------------------------------
layout = ("{} is {}")
binary = ""
denary = 0

# ---------------------------------------------------------------------
# Subprograms
# ---------------------------------------------------------------------
def binaryLoop (pBinary):
# =====> Rearrange the mixed up lines
    total = 0
    digit = ""
    value = 0
    multiplier = 1

    for index in range (len (pBinary) - 1, -1, -1):
        digit = pBinary[index]
        value = multiplier * int (digit)
        total = total + value
        multiplier = multiplier * 2

    return (total)
# End of mixed up lines

# ---------------------------------------------------------------------
# Main program
# ---------------------------------------------------------------------
# =====> Rearrange the mixed up lines
binary = input ("Enter a binary pattern (empty to exit): ")
while (binary != ""):
    denary = binaryLoop (binary)
    print (layout.format (binary, denary))
    binary = input ("Enter a binary number (empty to exit): ")
# End of mixed up lines
```

Question number	MP	Appx. Line	Answer	Additional guidance	Mark
5			Award marks as shown.		**(15)**
			Preparation		
	5.1		File opened for writing only (1)		
	5.2		Constant used as file name to open (1)		
			Processing all items in array		
	5.3		A loop to process every item in the given data structure (1)	• for or while	
			Controlling width of data output to file		
	5.4		Mechanism for controlling seven items (1)	• if or for	
	5.5		Constant used to compare against count for column control (1)		
			Formatting of output line		
	5.6		Line feed added to each line of output (1)		
	5.7		Comma added to each output item, except the last on each line (1)		
			Exiting		
	5.8		File closed before exiting program (1)	• Award if using ‘with open’	
			Additional		
	5.9		Use of techniques to ensure code is readable (1)		
			Levels-based mark scheme to a maximum of 6, from:		
	5.10 5.11 5.12		Solution design (3)	Considerations for levels-based mark scheme: • [6.1.2] Translates without error, even if reduced functionality • [6.1.6] Format of file matches requirement of seven columns per line (addition of line feed). • [6.3.3] Output file contains string values on separate lines	
	5.13 5.14 5.15		Functionality (3)		

				• [6.2.2] Use of 'for' loop in preference to a 'while' loop for iteration across an entire data structure or across the seven weights • [6.3.3] Conversion of integers in data structures to strings for output file • [6.3.3] Use of string concatenation to join items for output line	

Solution design (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	• There has been little attempt to decompose the problem. • Some of the component parts of the problem can be seen in the solution, although this will not be complete. • Some parts of the logic are clear and appropriate to the problem. • The use of variables and data structures, appropriate to the problem, is limited. • The choice of programming constructs, appropriate to the problem, is limited.	• There has been some attempt to decompose the problem. • Most of the component parts of the problem can be seen in the solution. • Most parts of the logic are clear and appropriate to the problem. • The use of variables and data structures is mostly appropriate. • The choice of programming constructs is mostly appropriate to the problem.	• The problem has been decomposed clearly into component parts. • The component parts of the problem can be seen clearly in the solution. • The logic is clear and appropriate to the problem. • The choice of variables and data structures is appropriate to the problem. • The choice of programming constructs is accurate and appropriate to the problem.	**3**

Functionality (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	**Functionality (when the code is run)** • The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements. • Program outputs are of limited accuracy and/or provide limited information. • Program responds predictably to some of the anticipated input. • Solution is not robust and may crash on anticipated or provided input.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that meets most of the stated requirements. • Program outputs are mostly accurate and informative. • Program responds predictably to most of the anticipated input. • Solution may not be robust within the constraints of the problem.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that fully meets the given requirements. • Program outputs are accurate, informative, and suitable for the user. • Program responds predictably to anticipated input. • Solution is robust within the constraints of the problem.	3

```python
# ------------------------------------------------------------------
# Constants
# ------------------------------------------------------------------
OUTPUT_FILE = "QO5_OUTPUT.TXT"
MAX_PER_LINE = 7

# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
weightsUsed = [3.79, 4.16, 1.52, 3.66, 2.58, 4.98, 4.37, 2.95, 2.58,
               4.37, 4.59, 2.61, 6.13, 4.49, 1.66, 2.65, 4.64, 4.72,
               3.59, 4.56, 4.23, 2.15, 4.03, 2.47, 4.61, 4.55, 6.31,
               5.81, 2.63, 3.61, 3.49, 4.49, 3.02, 3.86, 6.26, 3.11,
               1.79, 2.62, 2.23, 2.34, 5.66, 4.58, 3.52, 1.53, 2.07,
               3.89, 3.48, 5.52, 6.38, 3.77, 1.74, 1.78, 3.87, 3.45,
               3.79, 3.36, 1.87, 2.12, 2.09, 2.84, 2.29, 4.46, 3.63]

# =====> Write your code here
count = 1
outLine = ""
```

```python

# -------------------------------------------------------------------
# Main program
# -------------------------------------------------------------------
# =====> Open the output file
theFile = open (OUTPUT_FILE, "w")

# =====> Process each item in the data structure
for num in weightsUsed:
    outLine = outLine + str (num)
    if (count == MAX_PER_LINE):
        outLine = outLine + "\n"
        theFile.write (outLine)
        outLine = ""
        count = 1
    else:
        outLine = outLine + ","
        count = count + 1

# =====> Close the output file
theFile.close ()
```

Question number	MP	Appx. Line	Answer	Additional guidance	Mark
6			Award marks as shown.		**(15)**
			Input		
	6.1		Convert input string to uppercase to match data given in data structure (1)	• <string>.upper()	
			Linear search and terminating conditions		
	6.2		Linear search uses length of list for upper boundary of loop (1)		
	6.3		Mechanisms to identify when item is found in the list (1)	• Boolean variables • Appropriate ordering of if/elif/else	
	6.4		Mechanism to identify when item location is passed over in search (1)	• Boolean variables • Appropriate ordering of if/elif/else	
			Identifying suggested word		
	6.5		A method for tracking the suggested word is used (1)	• index, whole record	
	6.6		Use of two-dimensional indexing (1)		
			Levels-based mark scheme to a maximum of 9, from:		
	6.7 6.8 6.9		Solution design (3)	Considerations for levels-based mark scheme: • [6.1.1] Use decomposition to solve problem and create solution • [6.2.2] Use of ‘while’ loop to traverse data structure, rather than a ‘for’ loop • [6.3.1] Choice of variable data types to hold suggested word is appropriate, i.e. a list rather than a string and an integer • [6.1.2] Write in a high-level	
	6.10 6.11 6.12		Good programming practices (3)		
	6.13 6.14 6.15		Functionality (3)		

				language • [6.1.4] Program code is laid out in clear sections; white space is used to show different parts of the solution/functionality • [6.1.4] Variable names are meaningful; comments are provided and are helpful in explaining logic • [6.4.1] Printed outputs are fit for purpose • [6.1.6] Functions correctly for all anticipated inputs in the constraints of the problem definition • [6.1.6] Functions correctly if a word greater than “ZA” is entered (special case)	

Test Data:

Input	Expected output
no AA ZA	NO is worth 2 points. AA is worth 2 points. ZA is worth 11 points.
MU	MU is worth 4 points.
nz / NZ	NZ is not in the list. Use OD worth 3 points.
zh / ZH	ZH is not in the list. Use ZA worth 11 points.

Solution design (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	• There has been little attempt to decompose the problem. • Some of the component parts of the problem can be seen in the solution, although this will not be complete. • Some parts of the logic are clear and appropriate to the problem. • The use of variables and data structures, appropriate to the problem, is limited. • The choice of programming constructs, appropriate to the problem, is limited.	• There has been some attempt to decompose the problem. • Most of the component parts of the problem can be seen in the solution. • Most parts of the logic are clear and appropriate to the problem. • The use of variables and data structures is mostly appropriate. • The choice of programming constructs is mostly appropriate to the problem.	• The problem has been decomposed clearly into component parts. • The component parts of the problem can be seen clearly in the solution. • The logic is clear and appropriate to the problem. • The choice of variables and data structures is appropriate to the problem. • The choice of programming constructs is accurate and appropriate to the problem.	**3**

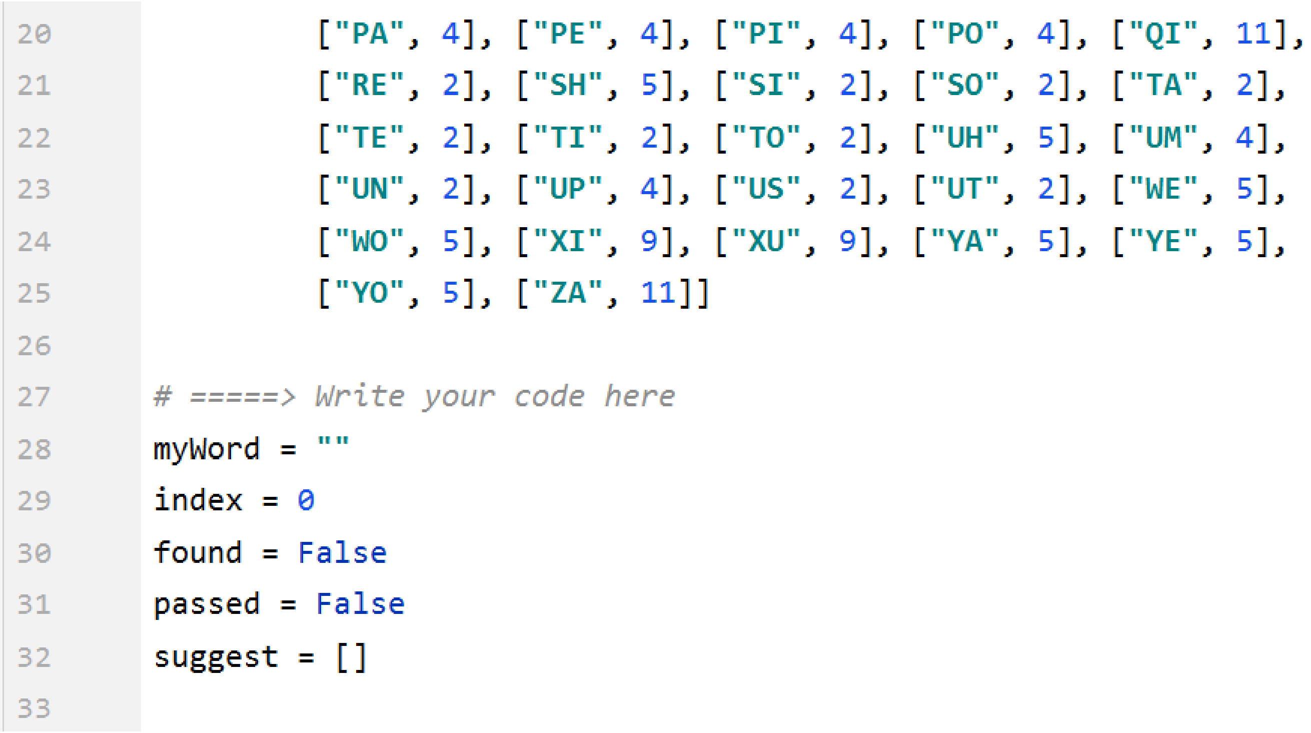
Good programming practices (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	• There has been little attempt to lay out the code into identifiable sections to aid readability. • Some use of meaningful variable names. • Limited or excessive commenting. • Parts of the code are clear, with limited use of appropriate spacing and indentation.	• There has been some attempt to lay out the code to aid readability, although sections may still be mixed. • Uses mostly meaningful variable names. • Some use of appropriate commenting, although may be excessive. • Code is mostly clear, with some use of appropriate white space to aid readability.	• Layout of code is effective in separating sections, e.g. putting all variables together, putting all subprograms together as appropriate. • Meaningful variable names and subprogram interfaces are used where appropriate. • Effective commenting is used to explain logic of code blocks. • Code is clear, with good use of white space to aid readability.	**3**

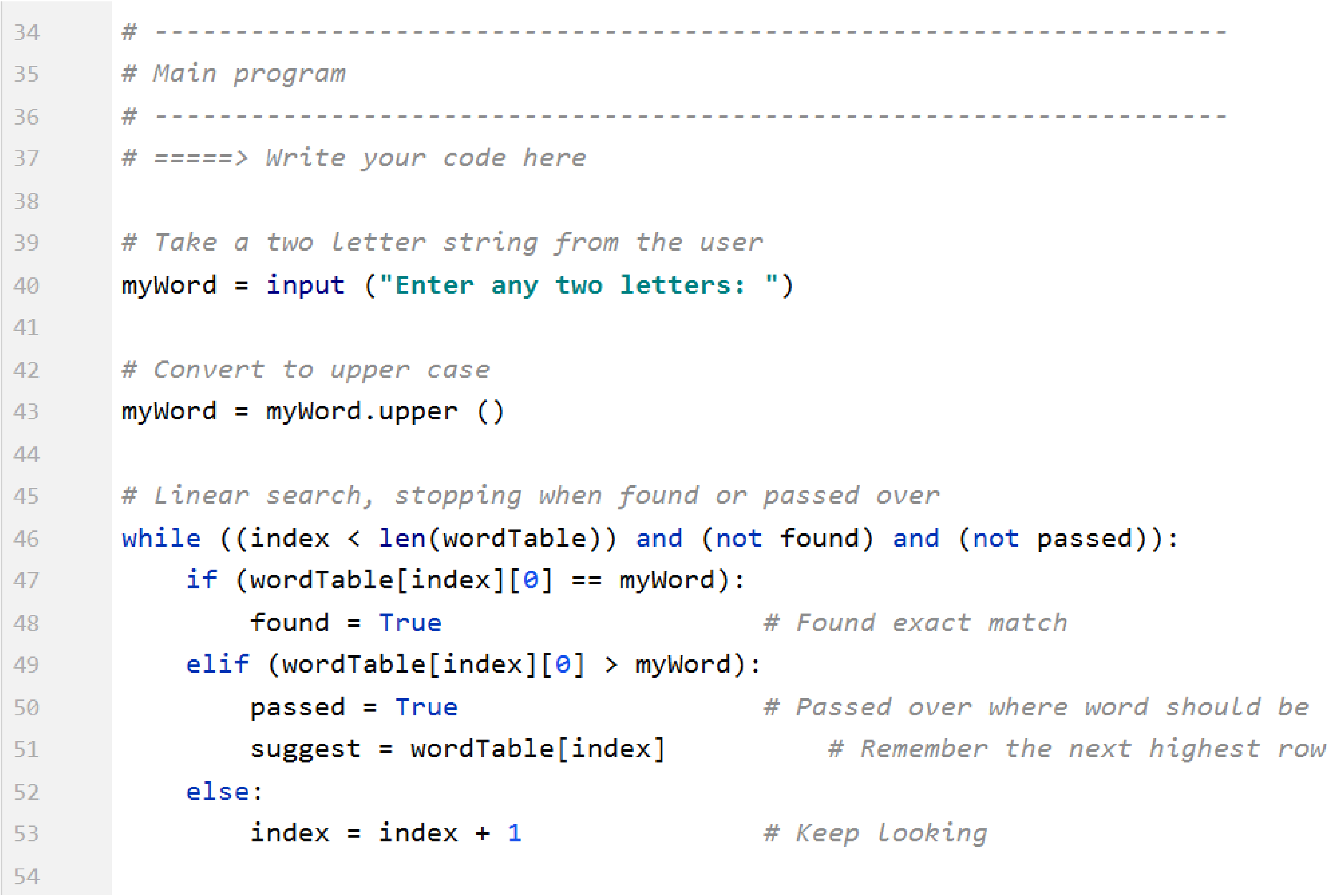
Functionality (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	**Functionality (when the code is run)** • The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements. • Program outputs are of limited accuracy and/or provide limited information. • Program responds predictably to some of the anticipated input. • Solution is not robust and may crash on anticipated or provided input.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that meets most of the stated requirements. • Program outputs are mostly accurate and informative. • Program responds predictably to most of the anticipated input. • Solution may not be robust within the constraints of the problem.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that fully meets the given requirements. • Program outputs are accurate, informative, and suitable for the user. • Program responds predictably to anticipated input. • Solution is robust within the constraints of the problem.	**3**

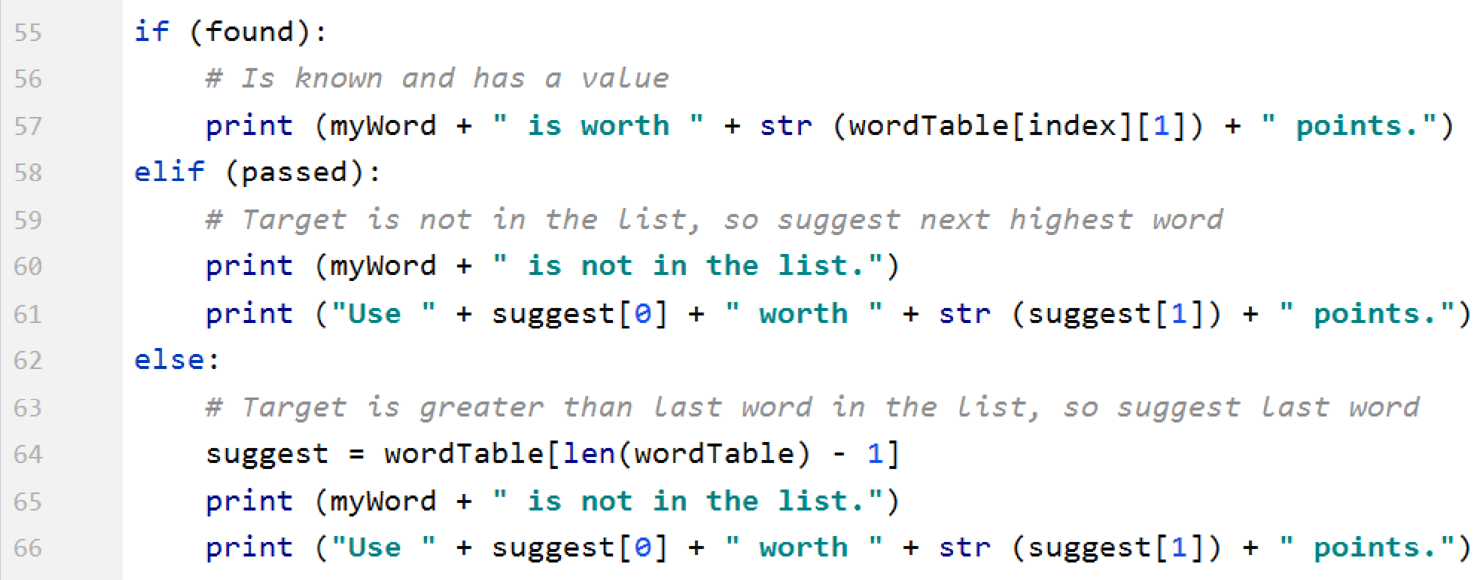
```
# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
wordTable = [["AA", 2], ["AB", 4], ["AD", 3], ["AE", 2], ["AG", 3],
         ["AH", 5], ["AI", 2], ["AL", 2], ["AM", 4], ["AN", 2],
         ["AR", 2], ["AS", 2], ["AT", 2], ["AW", 5], ["AX", 9],
         ["AY", 5], ["BA", 4], ["BE", 4], ["BI", 4], ["BO", 4],
         ["BY", 7], ["DA", 3], ["DE", 3], ["DO", 3], ["ED", 3],
         ["EF", 5], ["EH", 5], ["EL", 2], ["EM", 4], ["EN", 2],
         ["ER", 2], ["ES", 2], ["ET", 2], ["EW", 5], ["EX", 9],
         ["FA", 5], ["FE", 5], ["GI", 3], ["GO", 3], ["HA", 5],
         ["HE", 5], ["HI", 5], ["HM", 7], ["HO", 5], ["ID", 3],
         ["IF", 5], ["IN", 2], ["IS", 2], ["IT", 2], ["JO", 9],
         ["KA", 6], ["KI", 6], ["LA", 2], ["LI", 2], ["LO", 2],
         ["MA", 4], ["ME", 4], ["MI", 4], ["MM", 6], ["MO", 4],
         ["MU", 4], ["MY", 7], ["NA", 2], ["NE", 2], ["NO", 2],
         ["NU", 2], ["OD", 3], ["OE", 2], ["OF", 5], ["OH", 5],
         ["OI", 2], ["OK", 6], ["OM", 4], ["ON", 2], ["OP", 4],
         ["OR", 2], ["OS", 2], ["OW", 5], ["OX", 9], ["OY", 5],
```



```
         ["PA", 4], ["PE", 4], ["PI", 4], ["PO", 4], ["QI", 11],
         ["RE", 2], ["SH", 5], ["SI", 2], ["SO", 2], ["TA", 2],
         ["TE", 2], ["TI", 2], ["TO", 2], ["UH", 5], ["UM", 4],
         ["UN", 2], ["UP", 4], ["US", 2], ["UT", 2], ["WE", 5],
         ["WO", 5], ["XI", 9], ["XU", 9], ["YA", 5], ["YE", 5],
         ["YO", 5], ["ZA", 11]]

# =====> Write your code here
myWord = ""
index = 0
found = False
passed = False
suggest = []

```



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

# Take a two letter string from the user
myWord = input ("Enter any two letters: ")

# Convert to upper case
myWord = myWord.upper ()

# Linear search, stopping when found or passed over
while ((index < len(wordTable)) and (not found) and (not passed)):
    if (wordTable[index][0] == myWord):
        found = True                    # Found exact match
    elif (wordTable[index][0] > myWord):
        passed = True                   # Passed over where word should be
        suggest = wordTable[index]          # Remember the next highest row
    else:
        index = index + 1               # Keep looking

```



```python
if (found):
    # Is known and has a value
    print (myWord + " is worth " + str (wordTable[index][1]) + " points.")
elif (passed):
    # Target is not in the list, so suggest next highest word
    print (myWord + " is not in the list.")
    print ("Use " + suggest[0] + " worth " + str (suggest[1]) + " points.")
else:
    # Target is greater than last word in the list, so suggest last word
    suggest = wordTable[len(wordTable) - 1]
    print (myWord + " is not in the list.")
    print ("Use " + suggest[0] + " worth " + str (suggest[1]) + " points.")
```