Student responses with examiner commentary

V1.0 29/10/2014

Strictly confidential

Student responses with examiner commentary

A-level Computer Science 7517
Paper 1 7517/D

For teaching from September 2015

For assessment from summer 2017

Specimen assessment paper 1 7517/D

Introduction

These resources should be used in conjunction with the Specimen Assessment material (7517/D from the AQA website. This document illustrates how examiners intend to apply the mark scheme in live papers. While every attempt has been made to show a range of student responses examiners have used responses, and subsequent comments, which will provide teachers with the best opportunity to understand the application of the mark scheme. Examples given in this commentary use Python 3.

<table>
<tr><th>Question Number</th><th>Student answer</th><th>Marks awarded</th><th>Marks available</th><th>Commentary</th></tr>
<tr><td>01.1</td><td>Steve</td><td>1</td><td>1</td><td>The student should have written B (the letter corresponding to the correct answer) as this is what the question asked for. As this answer is unambiguous and clearly correct the mark has been awarded – however, if a student doesn’t answer in the way asked in the question they have a higher chance of writing something that is not worth a mark.</td></tr>
<tr><td>01.2</td><td>Nathan was killed by a blow on the head not by poison. This means that as Ian did not kill him with poison Peter was not in the kitchen and Martin was not in the dining room and Suzanne was in the dining room and therefore Steve killed Nathan.</td><td>2</td><td>2</td><td>There are no references to the exact rules but the chain of logic is clearly fully there so both marks are awarded.</td></tr>
<tr><td>02.1</td><td><table><tr><th>Original state</th><th>Input</th><th>New state</th></tr><tr><td>S3</td><td>1</td><td>S2</td></tr><tr><td>S3</td><td>0</td><td>S4</td></tr></table></td><td>1</td><td>1</td><td>The order of the two rows is not important. Both rows are correct.</td></tr>
<tr><td>02.2</td><td>((0|1)+00(0|1)+) | ((0|1)+11(0|1)+)</td><td>1</td><td>3</td><td>The student has got muddled between the + and * characters – if the four +s were replaced with *s then the answer would be correct and worth 3 marks (though it wouldn’t be exactly the same as either of the two answers shown on the mark scheme).

The student has been awarded a mark for there being a 00 or 11 in the string.</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>02.3</td><td><table><tr><th>Rule number
(given in Figure 2)</th><th>Could be defined using a regular expression</th></tr><tr><td>1</td><td>Y</td></tr><tr><td>2</td><td>Y</td></tr><tr><td>3</td><td>Y</td></tr><tr><td>4</td><td>N</td></tr><tr><td>5</td><td>N</td></tr><tr><td>6</td><td>Y</td></tr></table></td><td>1</td><td>1</td><td>Correct answer.</td></tr>
<tr><td>02.4</td><td>The rule will recurse forever by keep on adding more characters. To fix this the rule needs to change to:
<word> ::= <char>|<char><word></td><td>2</td><td>2</td><td>Correct answer.

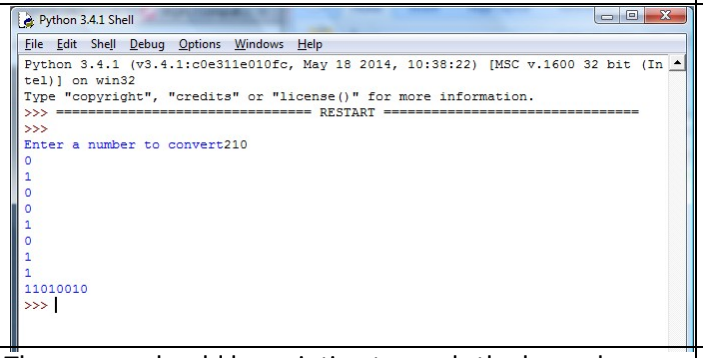
The order of the two parts on the right-hand side of the BNF rule does not matter.</td></tr>
<tr><td>03.1</td><td>1, 2, 3 make a cycle.</td><td>1</td><td>1</td><td>Equivalent to the answer shown in the mark scheme. The use of an example is not necessary but shows that the idea of the graph not being a tree because it contains a cycle is clearly there.</td></tr>
<tr><td>03.2</td><td><table><tr><th>Vertex (in Figure 3)</th><th>Adjacent vertices</th></tr><tr><td>1</td><td>2, 3</td></tr><tr><td>2</td><td>1, 3, 4</td></tr><tr><td>3</td><td>1, 2, 5</td></tr><tr><td>4</td><td>2</td></tr><tr><td>5</td><td>3</td></tr></table></td><td>2</td><td>2</td><td>Correct answer.</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>03.3</td><td>1) Adjacency lists are the better choice when there are lots of connections between the vertices
2) When someone wants to be able to see which vertices are connected to another vertex easily.</td><td>0</td><td>2</td><td>Answer 1 is wrong – these are the circumstances when an adjacency matrix would be the better choice.
Answer 2 is also not worth a mark. What is easier for people is irrelevant – adjacency lists and adjacency matrices are used to represent a graph in a computer's memory.</td></tr>
<tr><td>03.4</td><td>
<table>
<tr><th></th><th></th><th></th><th></th><th colspan="5">Cat</th></tr>
<tr><th>NoOfCats</th><th>A</th><th>B</th><th>C</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>5</td><td>2</td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>1</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>1</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td></td><td></td><td>2</td><td></td><td></td><td></td></tr>
<tr><td></td><td>3</td><td>1</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>1</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>1</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>3</td><td></td><td></td><td></td><td>3</td><td></td><td></td></tr>
<tr><td></td><td>4</td><td>1</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>4</td><td></td><td></td><td></td><td></td><td>1</td><td></td></tr>
<tr><td></td><td>5</td><td>1</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>5</td><td></td><td></td><td></td><td></td><td></td><td>1</td></tr>
</table>
</td><td>6</td><td>6</td><td>Correct answer. The value of 2 being assigned to A is on a different row to that shown on the mark scheme, but is fine.</td></tr>
</table>

03.5	It splits the cats up into different journeys so that they don't get stressed. The cats given a value of 1 will travel together, the cats given a 2 will travel together and the cats given a 3 will travel together, etc...	1	1	Correct answer – goes into more detail than needed for the mark to be awarded (the first sentence on its own would have been sufficient).
03.6	A problem that can't be solved except when the problem size is small.	0	2	The problem can always be solved – but when the problem size is large it takes an unreasonable amount of time to do so.
03.7	To solve a different problem that is similar to this one and use the answer to this different problem.	1	2	This is equivalent to the idea of solving a simple version of the problem (which is worth one mark).
04.1	False	1	1	Correct answer.
04.2	THEN Failed = True	1	1	Correct answer. The use of = instead of ← is fine.
04.3	L = M – 1	2	2	Correct answer. The use of = instead of ← is fine.
04.4	N^2	0	1	Incorrect – the student has probably got muddled between polynomial and exponential time complexity as both have one term to the power of another.
04.5	Log N	1	1	Correct answer. The missing O and brackets don't matter.
04.6	1	1	1	Correct answer. The missing O and brackets don't matter.
04.7	N	1	1	Correct answer. The missing O and brackets don't matter.

04.8	If there are n items in the list it will take no more than n comparisons to find an item in the list.	1	2	What has been written is correct but not enough has been written to get two marks.
05.1	3*4	1	1	Correct answer.
05.2	12 * 8 + 4	0	1	Wrong answer.
05.3	Never needs to use brackets	1	1	Correct answer.

06.1	Answer for task 1:
	`Dec = int(input("Enter a number to convert"))`
	`if Dec < 0:`
	`  print("Can't be less than 0")`
	`else:`
	`  Rem = 0`
	`  while Dec > 0:`
	`  Rem = Dec % 2`
	`  Dec = Dec // 2`
	`  print (Rem)`
	
	Answer for task 2:
	
	`Dec = int(input("Enter a number to convert"))`
	`if Dec < 0:`
	`  print("Can't be less than 0")`
	`else:`
	`  Rem = 0`
	`  Str = ""`
	`  while Dec > 0:`
	`  Rem = Dec % 2`
	`  Dec = Dec // 2`
	`  print (Rem)`
	`  Str = str(Rem) + Str`
		12	12	Answer written using Python 3 programming language.
				
				Correct answer.
				
				There was no need to show the answer for Task 1 as well as the answer for Task 2 as the answer for Task 2 includes the answer for Task 1.
				
				The student has added an additional validation rule which was not asked for – this is ignored (no marks awarded, no marks taken away) as it does not impact on the desired functionality of the program.

	`print (Str)`			
06.2		2	2	Correct test data used and correct result shown. Screen capture matches what would be obtained by running the student's code. The student has left `print (Rem)` in from Task 1 (see 06.1) so that the test shows both the unreversed and reversed output but the desired functionality is there.
07.1	The arrows should be pointing towards the base class not towards the subclasses. There is no class called Monster, there is a class called Enemy.	2	2	Both answers correct.
07.2	`self.Flask = Item()`	1	1	A correct example given.
07.3	`CavernState`	1	1	Correct answer. Strictly-speaking this is a Python list – but it is being used as an array.
07.4	`class Enemy(Item):`	0	1	This includes code other than the identifier – so the mark is not given as it is not clear they have clearly recognised what the identifier is. `Enemy` would be a correct answer.
07.5	`choice`	1	1	Correct answer. Technically, there should be a capital c on the identifier as Python is case-sensitive. However, case of identifiers is

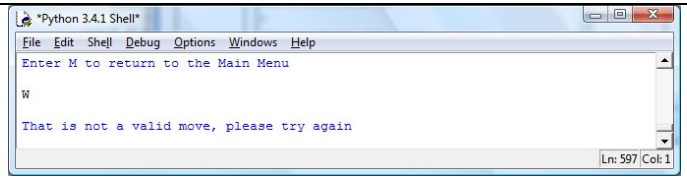
				0ignored in Section C (one reason for this is because case could be changed by auto-correct options in the word processor).
07.6	`class Game:`	0	1	This includes code other than the identifier – so the mark is not given as it is not clear they have clearly recognised what the identifier is. `Game` would be a correct answer.
07.7	So that (0,0) is not allowed as this is where the player is and it would make the game too easy if the flask could start there and too hard if the monster or trap could start there.	1	1	Mark awarded – goes into more detail than needed. Having a trap at (0,0) would not make the game harder as a trap is only triggered when the player moves into a cell containing a trap – but it would give the unexpected result of a trap not being triggered when the player starts in the same cell as the trap but then being triggered if the player moves back to that cell later in the game. This minor inaccuracy in the student's answer is ignored as it does not change the meaning of the part of their answer that was being looked for in this question.
07.8	Named constants make program code easier to understand and make it easier to change the program in the future.	2	2	Two correct answers - first point is same as mark scheme, second point clearly equivalent to "easier to update".
07.9	Create a new Trap in the Game class - `self.Trap3 = Trap()` Change the If statement `if not self.Monster.GetAwake() and not`	2	2	Both marks awarded. There was no need to include actually modifications made to the code – descriptions of the changes needed would have been sufficient.

	FlaskFound and not Eaten and (self.Player.CheckIfSameCell(self.Trap1.GetPosition()) and not self.Trap1.GetTriggered() or self.Player.CheckIfSameCell(self.Trap2.GetPosition()) and not self.Trap2.GetTriggered()): to if not self.Monster.GetAwake() and not FlaskFound and not Eaten and (self.Player.CheckIfSameCell(self.Trap1.GetPosition()) and not self.Trap1.GetTriggered() or self.Player.CheckIfSameCell(self.Trap2.GetPosition()) and not self.Trap2.GetTriggered()or self.Player.CheckIfSameCell(self.Trap3.GetPosition()) and not self.Trap3.GetTriggered()):			

08.1	def CheckValidMove(self, Direction):
	  ValidMove = True
	  if not(Direction in ['N', 'S', 'W', 'E', 'M']) or self.Player.GetPosition().NoOfCellsEast == 0 and Direction == 'W':
	  ValidMove = False
	  return ValidMove
		3	3	Answer written using Python 3 programming language.
				
				Correct answer – rather than using a separate If statement the student has combined the new conditions with the conditions in the existing If statement. All new functionality is correct and none of the existing functionality has been changed so full marks have been given.

08.2	while not ValidMove:	0	2	Answer written using Python 3 programming language.

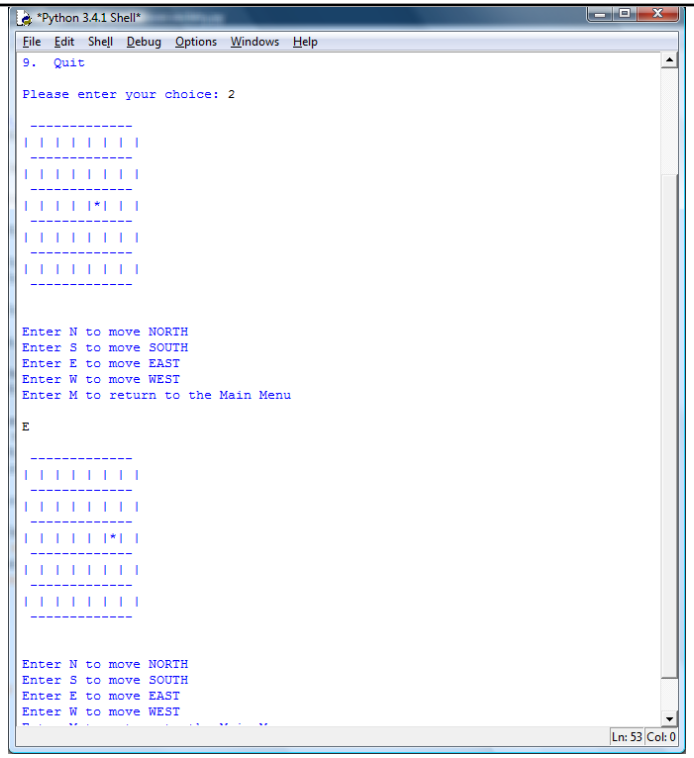
	`self.DisplayMoveOptions()`
	`MoveDirection = self.GetMove()`
	`ValidMove = self.CheckValidMove(MoveDirection)`
	`if ValidMove = False:`
	`print ("That is not a valid move, please try again")`
				The if statement should use == not =. This means that the condition is not correct and also that the error message will not be displayed when it should be.

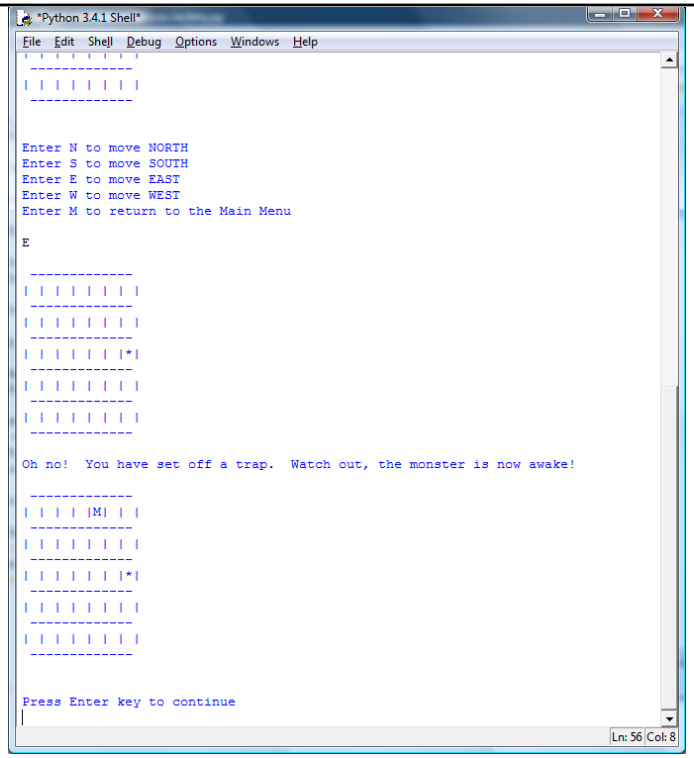
08.3		0	1	While this screen capture shows the correct test data being entered and it also shows the correct result being obtained no marks have been awarded because this screen capture was not obtained by running the code that the student has shown in parts 8.1/8.2. The student has either corrected their program code but forgotten to show the latest version of their code in their answer to 8.2 or has faked this screen capture (possibly by getting the error message to always be displayed when W is entered). Even if this screen capture could have been obtained from the code provided by the student it would not be given the mark as it is not possible to see if this is the correct test as the screen capture does not show the position of the player prior to the W being entered.

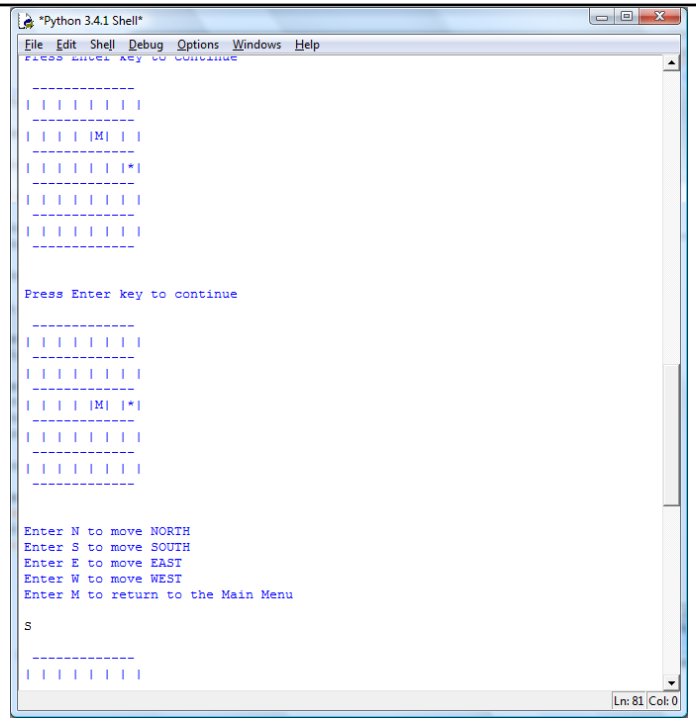
09.1	`class SleepyEnemy(Enemy):`
	`def __init__(self):`
	`Enemy.__init__(self)`
	
	`def ChangeSleepStatus(self):`
	`self.MovesTillSleep = 4`
		8	8	Answer written using Python 3 programming language.
				
				Fully-working solution has been developed.
				
				`self.MovesTillSleep` is only created in

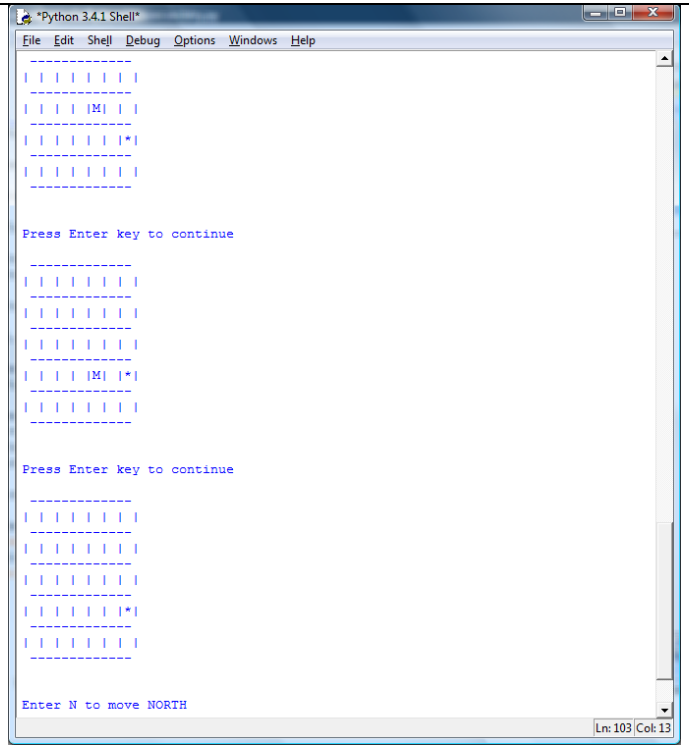
	`Enemy.ChangeSleepStatus(self)`
	
	`def MakeMove(self, PlayerPosition):`
	`Enemy.MakeMove(self, PlayerPosition)`
	`self.MovesTillSleep = self.MovesTillSleep - 1`
	`if self.MovesTillSleep < 1:`
	`self.ChangeSleepStatus()`
				`ChangeSleepStatus` (and this statement earns two marks, the first for creating the property and the second for assigning it the value 4) but this will not affect the functionality as long as `ChangeSleepStatus` is called before `MakeMove`.

09.2		2	2	Correct test data and results shown. Screen capture has been obtained by running student's code. There are some minor gaps missing between the screen captures but there is enough evidence to show all the test data and all the intermediary results of the test.


Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
9. Quit
Please enter your choice: 2
Enter N to move NORTH
Enter S to move SOUTH
Enter E to move EAST
Enter W to move WEST
Enter M to return to the Main Menu
E
Enter N to move NORTH
Enter S to move SOUTH
Enter E to move EAST
Enter W to move WEST
Ln: 53 Col: 0


Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Enter N to move NORTH
Enter S to move SOUTH
Enter E to move EAST
Enter W to move WEST
Enter M to return to the Main Menu
E
Oh no! You have set off a trap. Watch out, the monster is now awake!
Press Enter key to continue
Ln: 56 Col: 8


Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Press Enter key to continue
Press Enter key to continue
Enter N to move NORTH
Enter S to move SOUTH
Enter E to move EAST
Enter W to move WEST
Enter M to return to the Main Menu
S
Ln: 81 Col: 0

10.1	def DisplayMoveOptions(self):
	  print()
	  print("Enter N to move NORTH")
	  print("Enter S to move SOUTH")
	  print("Enter E to move EAST")
	  print("Enter W to move WEST")
	  print("Enter A to shoot")
	  print("Enter M to return to the Main Menu")
	  print()
		1	1	Answer written using Python 3 programming language.
				
				Mark awarded.

10.2	`def CheckValidMove(self, Direction):`
	`  ValidMove = True`
	`  if not(Direction in ['N', 'S', 'W', 'E', 'M']) or self.Player.GetPosition().NoOfCellsEast == 0 and Direction == 'W':`
	`  ValidMove = False`
	`  if Direction == 'A':`
	`  ValidMove = self.Player.GetHasArrow()`
	`  return ValidMove`
		2	2	Answer written using Python 3 programming language.
				
				Code has all the required functionality so gets both marks.

10.3	`class Character(Item):`
	`  def __init__(self):`
	`  Item.__init__(self)`
	`  self.HasArrow = True`
	
	`  def MakeMove(self, Direction):`
	`  if Direction == 'N':`
	`  self.NoOfCellsSouth = self.NoOfCellsSouth - 1`
	`  elif Direction == 'S':`
	`  self.NoOfCellsSouth = self.NoOfCellsSouth + 1`
	`  elif Direction == 'W':`
	`  self.NoOfCellsEast = self.NoOfCellsEast - 1`
	`  elif Direction == 'E':`
	`  self.NoOfCellsEast = self.NoOfCellsEast + 1`
		8	8	Answer written using Python 3 programming language.
				
				A fully-working solution that maps onto all points on the mark scheme.

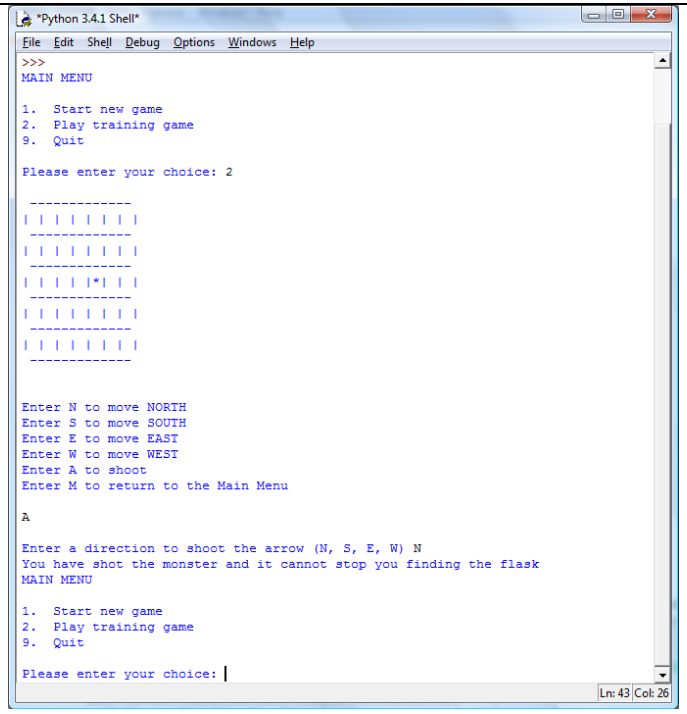
	`  def GetHasArrow(self):`
	`    return self.HasArrow`
	
	`  def GetArrowDirection(self):`
	`    choice = ""`
	`    self.HasArrow = False`
	`    while choice not in ['N', 'S', 'W', 'E']:`
	`      choice = input("Enter a direction to shoot the arrow (N, S, E, W) ")`
	`      if choice not in ['N', 'S', 'W', 'E']:`
	`        print('That is not a valid direction - choose again')`
	`  return choice`			

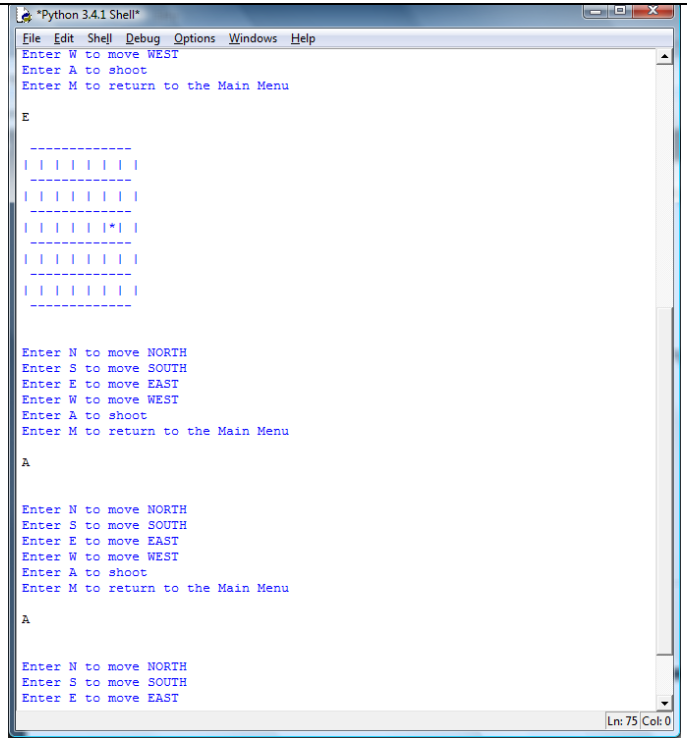
10.4	`def Play(self): Eaten = False FlaskFound = False`
	`  MoveDirection = ' '`
	`  Position = CellReference()`
	
	`self.Cavern.Display(self.Monster.GetAwake())`
	`  while not Eaten and not FlaskFound and (MoveDirection != 'M'):`
	`    ValidMove = False`
	`    while not ValidMove:`
	`      self.DisplayMoveOptions()`
	`      MoveDirection = self.GetMove()`
	`      ValidMove = self.CheckValidMove(MoveDirection)`
		5	6	Answer written using Python 3 programming language.
				
				Very close to a fully-working solution. It would have been a bit neater to use a compound condition rather than nested-ifs, but student is not penalised for this.
				
				However, the order of the if statements means that there is only a call to `GetArrowDirection` if the monster is directly to the north of the player – if the monster is anywhere else then the player will not be given the option to choose which direction to shoot in. This means that the 2nd mark point on the mark scheme has not been awarded.

```
    if MoveDirection != 'M':
      if MoveDirection == 'A':
        if
self.Monster.GetPosition().NoOfCellsEa
st ==
self.Player.GetPosition().NoOfCellsEas
t:
          if
self.Monster.GetPosition().NoOfCellsSo
uth <=
self.MonsterPlayer.GetPosition().NoOfC
ellsSouth:
            if
self.Player.GetArrowDirection() ==
'N':
              FlaskFound = True
              print("You have shot the
monster and it cannot stop you finding
the flask")
      else:

self.Cavern.PlaceItem(self.Player.GetP
osition(), ' ')

self.Player.MakeMove(MoveDirection)

self.Cavern.PlaceItem(self.Player.GetP
osition(), '*')

self.Cavern.Display(self.Monster.GetAw
ake())
        FlaskFound =
self.Player.CheckIfSameCell(self.Flask
```

```
.GetPosition())
        if FlaskFound:
          self.DisplayWonGameMessage()
        Eaten =
self.Monster.CheckIfSameCell(self.Play
er.GetPosition())
        if  not
self.Monster.GetAwake() and \
        not FlaskFound and not Eaten
and \

(self.Player.CheckIfSameCell(self.Trap
1.GetPosition()) and \
        not self.Trap1.GetTriggered()
or \

self.Player.CheckIfSameCell(self.Trap2
.GetPosition()) and \
        not
self.Trap2.GetTriggered()):

self.Monster.ChangeSleepStatus()
          self.DisplayTrapMessage()

self.Cavern.Display(self.Monster.GetAw
ake())
      if (self.Monster.GetAwake()) and
not Eaten and not FlaskFound:
        Count = 0
        while Count != 2 and not
Eaten:

self.Cavern.PlaceItem(self.Monster.Get
```

	Position(), ' ') Position = self.Monster.GetPosition() self.Monster.MakeMove(self.Player.GetP osition()) self.Cavern.PlaceItem(self.Monster.Get Position(), 'M') if self.Monster.CheckIfSameCell(self.Flas k.GetPosition()): self.Flask.SetPosition(Position) self.Cavern.PlaceItem(Position, 'F') Eaten = self.Monster.CheckIfSameCell(self.Play er.GetPosition()) print() print('Press Enter key to continue') input() self.Cavern.Display(self.Monster.GetAw ake()) Count = Count + 1 if Eaten: self.DisplayLostGameMessage()				

10.5		1	1	Screen capture genuinely from student's code and has correct test data and result.

10.6		0	1	Mark not awarded – user was not given the choice of direction to shoot the arrow in and when they choose A for a 2nd time the illegal move error message was not displayed.

Version 1.0
First published (07/10/2014)
Last updated (29/10/2014)