Puzzle answers

Puzzle set 1 – Input and output

	Answer	Additional notes
1	Bicycle Chain	The text is stored in the variable productName and is then displayed.
2	Labradoodle Two	The comma in the print( ) statement produces a space between the stored words.
3	Scott aged 23	The text "aged" is placed between the two stored words. Two commas create spaces between the words.
4	Lava JavaLava Java	The stored text is displayed twice. The concatenation means there is no space in the middle of the text.
5	video videovideo	The entered word "video" is stored and then displayed three times. The comma (,) creates a space between the first two words while the + concatenates (joins) the last two words.
6	Going GoingGone	A comma ensures the first two words are separated by a space. The last two words are concatenated so have no space.
7	This, is a plus + symbol	This puzzle highlights the importance of what is inside or outside of the inverted commas. Anything inside is text, anything outside is a statement. The comma (,) and + inside " " are displayed. The other two separate and concatenate the three parts of the output.
8	Be kind whenever possible.	With this puzzle it's best to work backwards from the print statement. textThree = Be kind + whenever (with a space in front of " whenever") textTwo = possible A full stop is concatenated on to the end of the displayed text.
9	energy(e) = mc squared	Using brackets in print statements can be confusing. A bracket inside " " is displayed, just like any other text character.
10	life live	Compare the print statement to the output and work backwards to work out what is stored in word1, word2 and word3. From the second half of the output we can see that the missing word in: Live the", word1, "you love." is "life". So word1 must be storing "life". This means that word2 = "the life you". As word3 follows this part of the print statement, word3 must be "live" to complete the sentence shown in the output window.

Puzzle set 2 – Simple calculations

	Answer	Additional notes
11	2.0	When your code does not state if the answer is an integer (int) or a number with decimal place (float) Python assumes that the result of a division will be a float and adds the decimal place. 12 divided by 6 is therefore 2.0 and not 2.
12	18	A simple calculation of 13 plus 7 minus 2.
13	16	Remember that * means multiply.
14	6	Remember the division takes place before the addition. The int( ) function ensures the answer is an integer.
15	5.0	Remember that the result within the brackets is calculated first before the answer is divided by 2. A division means that the answer is stored as a float with a decimal place.

	Answer	Additional notes
16	4.0	Both brackets are evaluated before the division of each result (20/5)=4.0 The division ensures the answer is a float.
17	19	Where variables are involved in a calculation, it is the stored numbers that are calculated. 13+6=19
18	80	The variable 'numberThree' stores the result of 5*16 (or 80). The result of the multiplication is then displayed.
19	5 10 25	As your programs get longer you may have multiple variables. In this puzzle you are required to track the value each variable stores. numberOne = 5 numberTwo = (5*2)=10 and numberThree = (5+10+10)=25. The values stored in the three variables are then displayed, separated by commas.

The next few puzzles are similar to number 19. Sometimes programmers will keep a note of the values stored in variables while they are testing their program. The example below is called a "trace" table.

A trace table is used to note the values stored in variables after each line of code has been executed (run).

<table>
<tr><th></th><th>Answer</th><th>Additional notes</th></tr>
<tr><td>20</td><td>5.0 60.0 10.0</td><td>
<table>
<tr><th>Trace table for puzzle 20</th><th>numberOne</th><th>numberTwo</th><th>numberThree</th></tr>
<tr><td>After: numberOne = 20 / 4</td><td>5.0</td><td></td><td></td></tr>
<tr><td>After: numberTwo = numberOne + 55</td><td>5.0</td><td>60.0</td><td></td></tr>
<tr><td>After: numberThree = numberTwo / 6</td><td>5.0</td><td>60.0</td><td>10.0</td></tr>
<tr><td>After: print (numberOne, numberTwo, numberThree)</td><td>5.0</td><td>60.0</td><td>10.0</td></tr>
</table>
The fact that numberOne stores a float ensures that every other number is a float.
A trace table also shows how variables keep storing values throughout the execution of a program.
</td></tr>
<tr><td>21</td><td>49</td><td>
<table>
<tr><th>Trace table for puzzle 21</th><th>numberOne</th><th>numberTwo</th><th>numberThree</th></tr>
<tr><td>After: numberOne = int(input("Please enter an integer."))</td><td>7</td><td></td><td></td></tr>
<tr><td>After: numberTwo = numberOne * numberOne</td><td>7</td><td>49</td><td></td></tr>
<tr><td>After: numberThree = numberTwo / numberOne</td><td>7</td><td>49</td><td>7.0</td></tr>
<tr><td>After: print (int(numberThree * numberThree))</td><td>7</td><td>49</td><td>7.0</td></tr>
</table>
In this example, the answer 49.0 (7.0*7.0) is never stored within a variable.
The int( ) ensures that the result is displayed as an integer.
</td></tr>
</table>

<table>
<tr><th></th><th>Answer</th><th>Additional notes</th></tr>
<tr><td>22</td><td>67.0</td><td>
<table>
<tr><th>Trace table for puzzle 22</th><th>numberOne</th><th>numberTwo</th><th>numberThree</th></tr>
<tr><td>After: numberOne = 12 + 8</td><td>20</td><td></td><td></td></tr>
<tr><td>After: numberThree = numberOne * 3</td><td>20</td><td></td><td>60</td></tr>
<tr><td>After: numberTwo = numberThree - 55</td><td>20</td><td>5</td><td>60</td></tr>
<tr><td>After: numberThree = numberTwo * (numberOne/2)</td><td>20</td><td>5</td><td>50.0</td></tr>
<tr><td>After: numberOne = numberThree + 17</td><td>67.0</td><td>5</td><td>50.0</td></tr>
<tr><td>After: print (numberOne)</td><td>67.0</td><td>5</td><td>50.0</td></tr>
</table>
Note that twice in this puzzle variables are assigned a new value.
</td></tr>
<tr><td>23</td><td>3 3.0 3</td><td>
<table>
<tr><th>Trace table for puzzle 23</th><th>numberOne</th><th>numberTwo</th><th>numberThree</th></tr>
<tr><td>After: numberOne = int(12 / 4)</td><td>3</td><td></td><td></td></tr>
<tr><td>After: numberTwo = 6 * 2</td><td>3</td><td>12</td><td></td></tr>
<tr><td>After: numberThree = 4 + 1</td><td>3</td><td>12</td><td>5</td></tr>
<tr><td>After: numberThree = numberTwo</td><td>3</td><td>12</td><td>12</td></tr>
<tr><td>After: numberTwo = float(numberOne)</td><td>3</td><td>3.0</td><td>12</td></tr>
<tr><td>After: numberThree = numberOne</td><td>3</td><td>3.0</td><td>3</td></tr>
<tr><td>After: print (numberOne, numberTwo, numberThree)</td><td>3</td><td>3.0</td><td>3</td></tr>
</table>
It is important in this puzzle to note when the int( ) and float( ) functions are used to change the data type.
</td></tr>
</table>

Puzzle set 3 – String functions

	Answer	Additional notes
24	A flee	Remember when you are programming similar problems that spaces are counted as a character.
25	A fleeting bea	When no number appears before the colon, the substring starts from the beginning of the string.
26	f hearts	When no number appears after the colon, the substring ends with the last character of the string.
27	arts	For a negative number, count backwards from the last character.
28	eting be	The output displays the string from after character 5 up to the character before –15 (backwards from the last characters).
29	ting	The variable lyric2 stores the result of the first substring [2:10] or "fleeting". Substring is then used again to display lyric2 from after character 4 to the end of the string.
30	GC	The code concatenates the first character of each stored string.
31	car wars return of the mini	Each character is converted to lower case and stored in the variable "tempFilm". The contents of the variable are then displayed.
32	WORLD WAR Z 31st oct	The string stored in filmRelease is converted to upper case and concatenated with the now lower case date.

	Answer	Additional notes
33	C = central P = processing U = unit	In this puzzle, the first letter of each word is extracted using substring and then changed to upper case. The resulting capital letters are then concatenated with an equal sign and the original words to build the answer shown.
34	10	The len( ) function calculates the number of characters (length) in a string. If you answered 9, remember that the space also counts as a character.
35	Your password Sphinx is 6 characters long	The stored string and its length are both built into the output.
36	4	The count function looks for the number of times one string appears inside another. There are four examples of "re" found in the string. There are only two kinds of people who are really fascinating: people who know absolutely everything, and people who know absolutely nothing. – Oscar Wilde
37	Whatever you are, be a good one. Abraham Lincoln	Replace swaps "bad" for "good". No other characters change.
38	I met an old gentleman once, almost a hundred years old, and he told me...	Remember that programs can update values in variables. Each line updates what is stored in the variable "quotation". First "lady" is swapped for "gentleman" then "she" is swapped for "he".
39	Programs & Coding 17	"Code" is replaced with "Programs" leaving "Programs & Coding" which has 17 characters (including the two spaces).
40	Number of characters in password = 12	After the first replace "02jjkk kkde" becomes "02jjddd dddde". After the second replace "02jjddd dddde" becomes "02jddd dddde" which is 12 characters long. The answer is then printed along with the message.
41	9	Three copies of the same sentence are stored with slight differences following replaces. The remaining code adds up the number of "w" and "b" characters stored in the three variables. sentenceOne – The to boys learned to new skills sentenceTwo – The two boys learned two new skills sentenceThree – The two girls learned two new skills
42	hen Mhe	This program begins by counting and storing how often four different letters appear in a string. After the first five lines: letter1position = 3 (there are 3 a characters) letter2position = 2 (there are 2 e characters) letter3position = 2 (there are 2 i characters) letter4position = 3 (there are 3 o characters) These values are then used to store a single letter from the string (e.g. letter1 = e, or statement[2:3]): After lines 6 to 9: letter1 = "e", letter2 = "hen", letter3 = "h", letter4 = " M" The password variable then stores a string that is created by concatenating the above variables in the order given. Note that letter4 stores a space character as well as M. This creates the space in the middle of the password.

Puzzle set 4 – Mathematical functions

	Answer	Additional notes
43	193.73	The value stored in the variable "height" is rounded to 2 decimal places and displayed.
44	10.0	As we started with a float, the answer will still be a float, even when rounded to 0 decimal places.
45	79	The number is rounded up to the nearest integer.
46	4	25 divided by 7 = 3, remainder 4.
47	1	The modulus of 12.5 divided by 3 is 0.5. The ceil( ) function rounds 0.5 up to the nearest integer.
48	The number of whole planks needed = 7	20/3 = 6.6666. The ceil( ) function rounds this up to 7.
49	0.88	value – int(value) = 57.884-57 The result of the above, 0.884, is rounded to 2 decimal places.
50	25	dog = cat rounded down to the nearest integer, 5 5 to the power of 2 = 25.
51	27	It is vital longer problems are broken down into stages. First work out the value stored by num3. int(num2) = 30, math.ceil(num1) = 13 so num3 = 30-13 = 17 num3%5 = 2 Next calculate the value stored in num4. so num4 = 3 to the power of 2 = 9 int(num2/10) = 3 Finally the result of 9 * 3 is printed (27)

Puzzle set 5 – "If", "else" and "elif" statements

		Answer	Additional notes
52	a	Low	The condition "if number <= 50" is true (23 is less than 50) so print("Low") is executed. The conditions in the other if statements are false so their print statements are not executed.
	b	Middle	Only the condition "if number > 50 and number < 100" is true (67 is more than 50 and less than 100) so "Middle" is displayed.
	c	High	Only the condition "if number >= 100" is true (100 is equal to 100) so "High" is displayed.
	d	High	Only the condition "if number >=100" is true (236 is more than 100) so "High" is displayed.
	e	Low	Only the condition "if number <= 50" is true (50 is equal to 50) so "Low" is displayed.
53	a	Liquid	Only the condition "elif temp > 43 and temp < 87" is true (60 is greater than 43 and 60 is less than 87) so "Liquid" is displayed.
	b	Solid	Only the condition "elif temp > =–273 and temp <= 42" is true (–50 is greater than –273 and -50 is less than 42) so "Solid" is displayed.
	c	Gas	None of the conditions are true so the else is executed. "Gas" is displayed.
	d	Gas	None of the conditions are true so the else is executed. "Gas" is displayed.
	e	c	43 should be either a solid or liquid as the temperature falls between the two ranges but neither of the conditions accounts for 43 as an input.

		Answer	Additional notes
	e	Change the first part of the second condition to read temp > 42 to include 43 in the condition.	Alternatively the same condition could be changed to temp>=43 as this would also include 43.
54	a	Valid age School age	The first condition checks if age is less than 0 or greater than 120. If neither of these is true the else is executed and "Valid age" is displayed. Of the four if statements only the condition age>=3 and age<=18 is true so "School Age" is also displayed.
	b	Age not valid	–2 is less than 0 so the very first condition is true meaning "Age not valid" is displayed.
	c	Valid age Working age	47 is not < 0 or > 120 so the first if condition is false. This means the else displays "Valid age". Of the other if statements only age > =16 is true. This means "Working age" is also displayed.
	d	16, 17 or 18	To display the correct output the following conditions are required to be true: age <0 or age > 120 is false age >=3 and age <=18 is true age >= 16 is true Any one of the three ages shown in the answer meet all three of these conditions.
55	The program has three outer if statements that evaluate the weight of the parcel. When one of these conditions is true the nested if statements then evaluate the value of the parcel. The combination of the two inputs results in one particular value being stored for postage. Each of the answers below notes which conditions were true.		
	a	2.75	if weight > = 0 and weight < 2: weight = 1.5 if value > = 50 and value < 150: value = 62
	b	Invalid Value 0.0	if value < = 0: value = 0 if weight >= 2 and weight < 10: weight = 2.2
	c	15.0	elif weight > = 10 and weight < 25: weight = 19 if value > = 150: value = 172
	d	25.0	weight (32.5) does not meet any of the weight conditions so else is true
	e	2.5	if weight > = 2 and weight < 10: weight = 2.5 if value > 0 and value < 50: value = 34
	f	12.3	elif weight > = 10 and weight < 25: weight = 10 if value >= 50 and value < 150: value = 50
56	a	bcbc	len(word[number:]) calculates the number of characters in "word" from position "number" to the end of the string. len("been"[3:]) = 1 so the if condition is false and the else is executed. This means that 'word' now stores word[0:2] + word[0:2] = bebe word.count("e") counts the number of e characters in "word". As this is 2 the else is executed and all the es in "word" are replaced with c.
	b	coffaacoffaa	len("coffee" [2:]) = 4 so the if condition is true and "word" now stores word + word = coffeecoffee word.count("e") = 4 so all the es in "word" are replaced with a.
	c	baanbaan	len("been"[1:]) = 3 so the if condition is true and "word" now stores word + word = word.count("e") = 4 so all the es in "word" are replaced with a.

Puzzle set 6 – Fixed loops part 1

	Answer	Additional notes
57	Graphics Graphics Graphics	range(3) in the for statement means that the print("Graphics") statement is repeated three times.
58	Processor Processor	The range(1,3) can be interpreted as: 1 – execute loop 2 – execute loop 3 – don't execute loop print("Processor") is only executed twice.
59	pug pug pug pug pug	The variable "type" is assigned the value "pug". This is then displayed 5 times in the loop.
60	2 5 8 11 14 17	This program displays the loop variable "numbers". The range starts counting at 2 going up in steps of 3. The next value 20 does not display as the range notes this as where the loop should not execute any more repetitions.
61	10 20 30 40 50	The range starts counting at 10 going up in steps of 10. The next value 60 does not display as the range stops at 55.
62	35 30 25 20 15 10	The range starts counting at 35 going down in steps of 5. The next value 5 does not display as the range stops counting down at 7.
63	4 7 10 13	During each repetition 3 is added onto the "number" variable. "number" is then displayed. As these two lines of code are inside a loop with range(4), this happens 4 times.
64	range(6)	This is a simple count from 0 so only requires one number in the range. There are 6 values displayed so the code repeated 6 times.
65	range(2,6)	The count starts at 2 and counts up by 1 each repetition (so a step is not required). The last value displayed is 5 meaning the loop stopped executing at 6.
66	range(7,4,-1)	This output requires a step of -1 to count down from 7. The final value displayed is 5 meaning the loop must have stopped executing at 4.
67	range(1,11,3)	This fixed loop counts from 1 in steps of 3. Any value from 11, 12 or 13 would stop the loop counting at 10.
68	range(45,8,-9)	The output shows the count beginning at 45 and counting in steps of -9. The value to stop the loop could be any number from 8 down to 0.

Puzzle set 7 – Trace tables

<table>
<tr><th></th><th>Answer</th><th>Additional notes</th></tr>
<tr><td>69</td><td>18 11</td><td>
<table>
<tr><th>Trace table</th><td>num1</td><td>num2</td></tr>
<tr><td>beginning of the program</td><td>3</td><td>2</td></tr>
<tr><td>loop = 0</td><td>5</td><td>5</td></tr>
<tr><td>loop = 1</td><td>10</td><td>8</td></tr>
<tr><td>loop = 2</td><td>18</td><td>11</td></tr>
<tr><td>final values displayed</td><td>18</td><td>11</td></tr>
</table>
As the loop is repeated (3 times), num1 is calculated as the current value in num1 + the value stored in num2. Then num2 is assigned a value of the current value of num2 + 3. This creates two running totals where both variables are incremented each repetition.
</td></tr>
<tr><td>70</td><td>20.0 10.0</td><td>
<table>
<tr><th>Trace table</th><td>num1</td><td>num2</td></tr>
<tr><td>beginning of the program</td><td>80</td><td>40</td></tr>
<tr><td>loop = 0</td><td>40</td><td>20.0</td></tr>
<tr><td>loop = 1</td><td>20.0</td><td>10.0</td></tr>
<tr><td>final values displayed</td><td>20.0</td><td>10.0</td></tr>
</table>
Note that the type of data stored changes when the first division takes place. Every value stored after that becomes a float.
</td></tr>
<tr><td>71</td><td>44 28 6</td><td>
<table>
<tr><th>Trace table</th><td>num1</td><td>num2</td><td>num3</td></tr>
<tr><td>beginning of the program</td><td>2</td><td>4</td><td>6</td></tr>
<tr><td>loop = 0</td><td>–4</td><td>10</td><td>6</td></tr>
<tr><td>loop = 1</td><td>–4</td><td>16</td><td>6</td></tr>
<tr><td>loop = 2</td><td>8</td><td>22</td><td>6</td></tr>
<tr><td>loop = 3</td><td>44</td><td>28</td><td>6</td></tr>
<tr><td>final values displayed</td><td>44</td><td>28</td><td>6</td></tr>
</table>
Note that the value of num3 is not changed within the loop so the value displayed is the same as the initial value
</td></tr>
<tr><td>72</td><td>156 77 79</td><td>
<table>
<tr><th>Trace table</th><td>num1</td><td>num2</td><td>num3</td></tr>
<tr><td>beginning of the program</td><td>1</td><td>3</td><td>5</td></tr>
<tr><td>loop = 0</td><td>6</td><td>2</td><td>4</td></tr>
<tr><td>loop = 1</td><td>16</td><td>7</td><td>9</td></tr>
<tr><td>loop = 2</td><td>36</td><td>17</td><td>19</td></tr>
<tr><td>loop = 3</td><td>76</td><td>37</td><td>39</td></tr>
<tr><td>loop = 4</td><td>156</td><td>77</td><td>79</td></tr>
</table>
Note that the order in which the variables were assigned new values was changed: num3 first, then num2 and num1.
</td></tr>
</table>

<table>
<tr>
<td>73</td>
<td>5 5 5</td>
<td>
<table>
<tr><th>Trace table</th><td>num1</td><td>num2</td><td>num3</td></tr>
<tr><td>beginning of the program</td><td>5</td><td>5</td><td>5</td></tr>
<tr><td>loop = 0</td><td>5</td><td>5</td><td>5</td></tr>
<tr><td>loop = 1</td><td>5</td><td>5</td><td>5</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>loop = 199</td><td>5</td><td>5</td><td>5</td></tr>
<tr><td>final values displayed</td><td>5</td><td>5</td><td>5</td></tr>
</table>
At the end of each repetition num1, num2 and num3 store the same values that they stored at the beginning of the program. This means that it doesn't matter how often this program loops, the final values will always be 5,5,5.
</td>
</tr>
<tr>
<td>74</td>
<td>44 41</td>
<td>
<table>
<tr><th>Trace table</th><td>num1</td><td>num2</td></tr>
<tr><td>beginning of the program</td><td>2</td><td>4</td></tr>
<tr><td>loop = 0</td><td>6</td><td>6</td></tr>
<tr><td>loop = 1</td><td>12</td><td>11</td></tr>
<tr><td>loop = 2</td><td>23</td><td>21</td></tr>
<tr><td>loop = 3</td><td>44</td><td>41</td></tr>
<tr><td>final values displayed</td><td>44</td><td>41</td></tr>
</table>
Note the pattern within the loop part of the trace table. num2 is at first equal to num1, then 1 less, then 2 less, then finally 3 less than num1. As the loop value increases (from 0 to 3) the value subtracted in the line num2 = num1 – loop changes.
</td>
</tr>
<tr>
<td>75</td>
<td>282 6 288</td>
<td>
<table>
<tr><th>Trace table</th><td>num1</td><td>num2</td><td>num3</td></tr>
<tr><td>beginning of the program</td><td>4</td><td>6</td><td>0</td></tr>
<tr><td>loop = 3</td><td>12.0</td><td>6</td><td>18.0</td></tr>
<tr><td>loop = 4</td><td>32.0</td><td>6</td><td>38.0</td></tr>
<tr><td>loop = 5</td><td>90.0</td><td>6</td><td>96.0</td></tr>
<tr><td>loop = 6</td><td>282.0</td><td>6</td><td>288.0</td></tr>
<tr><td>final values displayed</td><td>282</td><td>6</td><td>288</td></tr>
</table>
As you get better at following the logic of programs tracing code can become easier. num2 is never assigned a new value so you can predict that its final value will be 6. As num3 = the current value of num1 + 6 there is no need to trace it either. The final num3 value displayed will be the final num1 value + 6.
</td>
</tr>
</table>

<table>
<tr><th></th><th>Answer</th><th>Additional notes</th></tr>
<tr><td>76</td><td>5 0 5</td><td>
<table>
<tr><th>Trace table</th><th>start</th><th>mid</th><th>end</th></tr>
<tr><td>beginning of the program</td><td>1</td><td></td><td></td></tr>
<tr><td>end of loop = 1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>end of loop = 2</td><td>2</td><td>0</td><td>2</td></tr>
<tr><td>end of loop = 3</td><td>5</td><td>2</td><td>5</td></tr>
<tr><td>end of loop = 4</td><td>6</td><td>2</td><td>6</td></tr>
<tr><td>end of loop = 5</td><td>5</td><td>0</td><td>5</td></tr>
<tr><td>final values displayed</td><td>5</td><td>0</td><td>5</td></tr>
</table>
Remember that the modulus symbol % calculated the remainder when one number is divided by another.
The first repetition (where loop = 1) can be described as follows:
start = 1 * 10 = 10
mid = remainder of 10/1 = 0
end = 0 + 1 = 1
start = 1
</td></tr>
<tr><td>77</td><td>0a1a2ab3</td><td>
<table>
<tr><th>Trace table</th><th></th><th></th><th></th></tr>
<tr><td>beginning of the program</td><td colspan="3">text1 = a, text2 = b, text3 = null (empty) string</td></tr>
<tr><td>after that</td><td>phrase1</td><td>phrase2</td><td>text3</td></tr>
<tr><td>loop = 0</td><td>0a</td><td>b0</td><td>0a</td></tr>
<tr><td>loop = 1</td><td>1a</td><td>b1</td><td>0a1a</td></tr>
<tr><td>loop = 2</td><td>2a</td><td>b2</td><td>0a1a2a</td></tr>
<tr><td>loop = 3</td><td>3a</td><td>b3</td><td>0a1a2ab3</td></tr>
<tr><td>final values displayed</td><td></td><td></td><td>0a1a2ab3</td></tr>
</table>
While loop = 0, 1 or 2 phrase1 is concatenated onto the end of text3. In the final repetition loop is no longer <= 2 so phrase2 is concatenated onto the end of text3.
</td></tr>
<tr><td>78</td><td>number = 13</td><td>
<table>
<tr><th>Trace table</th><th></th><th></th></tr>
<tr><td>beginning of the program</td><td colspan="2">text1 = this, text2 = is,
text3 = hard, words = null string</td></tr>
<tr><td>after that</td><td>words</td><td>number</td></tr>
<tr><td>diamond = 2</td><td>thisis</td><td>2</td></tr>
<tr><td></td><td>isihard</td><td>5</td></tr>
<tr><td>diamond = 3</td><td>isihardthisis</td><td>8</td></tr>
<tr><td></td><td>ihahard</td><td>13</td></tr>
</table>
Lots to follow here:
The loop repeats twice (range(2,4) as len of "is" = 2, len of "hard" = 4
The 'words' variable is concatenated twice. After the first concatenation the s characters are counted and added onto 'number'. After the second concatenation (where the second, third and fourth characters of words are concatenated with text3) the h characters are counted and added onto number along with the current vlaue of the loop variable "diamond". This is then repeated.
8 s and h characters are counted across the two repetitions. A further 5 is added on for the two values of diamond (2 and 3) during the repetitions giving a final total of 13 when the value in "number" is displayed.
</td></tr>
</table>

Puzzle set 8 – Conditional loops

	Answer	Additional notes
79	B (22)	The loop keeps repeating while num is not equal (!=) to 22. So entering 22 would stop the loop and result in "program finished" being displayed.
80	D (34)	This puzzle demonstrates that conditions can be written in several ways. not(guess==34) is the same as guess!=34 so, as with the last puzzle, entering the number in the condition will stop the loop.
81	C (24)	The loop will continue while num is equal to one of the three values 34, 22 or 20. Any value other than these three numbers will stop the loop.
82	C (76)	Any temperature less than or equal to 45 will cause the loop to repeat again. 76 is the only temperature over 45 so will stop the loop.
83	A (17)	17 is the only one of the four values that is not less than 5 or greater than 17.
84	D (27)	This is a familiar bit of code in Python programming as it is often used for validating input. The while loop checks that the input is between two values. If it is not it will keep looping. The only valid input is 27.
85	B (hello)	The program continues looping while the length of the string input is less than 4 letters. As "hello" has 5 letters, inputting it stops any further repetition.
86	C (loss)	The program loops while the word inputted has 0 or 1 s characters in the string. "loss" has 2 s characters so stops the loop.
87	A (afar)	The while loop continues to repeat while the text inputted has: • less than or equal to 1 a character; or • the length of the text is less than 4 characters. When an "or" is used in this way both of the conditions have to be false before the loop stops. "afar" is the correct answer because it is 4 characters long and has 2 a characters.

Puzzle set 9 – Assigning values to lists

	index	numbers	Additional notes
88	0	0	All elements of the list are initially set to 0. Note the order in which the elements are then assigned new values (index 2, 4 and then 3).
	1	0	
	2	33	
	3	11	
	4	22	
89	0	5	All elements of the list are initially set to 9. In the following code only two of the elements are changed. number[2] = 9 has no effect as the element is already stores 9.
	1	9	
	2	9	
	3	9	
	4	0	
90	0	0	Elements of the list are initially set to [0,1,2,3,4]. Elements 1, 4 and 3 and then reassigned different values.
	1	4	
	2	0	
	3	2	
	4	1	

	index	numbers	Additional notes
91	0	5	All elements of the list are initially set to 5. Elements 2, 4 and 3 are then set to the answers of each of the three calculations. Finally index 2 is assigned its initial value of 5. Remember that index 4 should now store a float value because the result of a division was stored.
	1	5	
	2	5	
	3	12	
	4	10.0	
92	0	12	Elements of the list are initially set to [12,2,24,4,36]. Index 2 is assigned the value 3 and then index 4 is assigned the same value as numbers[4] = numbers[2]. Index 2 is then assigned a new value, 6. Finally index 1 is assigned 20 + 4 – 6.
	1	18	
	2	6	
	3	4	
	4	3	
93	0	9	The puzzle requires that you calculate the result of the maths on each line. Remember: math.ceil( ) rounds up to the nearest whole number, pow () works out the power of one number to another, int( ) rounds down to the nearest whole number.
	1	5	
	2	5	
	3	5	
	4	2	
94	0	20	In this puzzle you need to follow the logic of the if statements. As numbers[2] = 35 the first if condition is true and numbers[0] is assigned 10 + 10. As the remainder of 20/2 is 0 the condition in the second if statement is false. numbers[4] is therefore assigned 20+10
	1	10	
	2	25	
	3	5	
	4	30	
95	0	3	The loop assigns each element of the list the value 3. This occurs as the loop variable nums counts up from 0 to 4 with each repetition. So numbers[0] = 3, then numbers[1] = 3 and so on.
	1	3	
	2	3	
	3	3	
	4	3	
96	0	0	Within the loop each element of the list is assigned the value of 'counter'. The first time the loop executes counter = 0. At the end of each repetition 1 is added on to counter.
	1	1	
	2	2	
	3	3	
	4	4	
97	0	0	When the loop executes each element of the list is assigned the value of the loop variable. This stores 0, then 1, then 2 and so on in the list.
	1	1	
	2	2	
	3	3	
	4	4	
98	0	2	This puzzle requires a good understanding of the loop variable. During the first repetition loop = 0 so the line of code within the loop can be read as "numbers[1] = numbers[0] + 0 + 1" = 2 + 0 + 1 = 3 During the next repetition loop = 1 so "numbers[2] = numbers[1] + 1 + 1" = 5 This continues with the loop incrementing each time.
	1	3	
	2	5	
	3	8	
	4	12	

	index	wordLength	Additional notes
99	0	5	This puzzle simple calculates the number of characters in each element of the "words" list, using the len( ) function, and stores the result in another list called "wordLength".
	1	8	
	2	5	
	3	14	
	4	7	

Puzzle set 10 – More than one list

	Answers			Additional notes
100	**index**	**nameList1**	**nameList2**	Both lists are initially assigned 5 different names. The fixed loop then assigns each element in nameList1 the value that is stored in the same element of nameList2. The result of this is that the lists store the same vales as the whole of the nameList2 is copied into nameList1.
	0	Mary	Mary	
	1	Nida	Nida	
	2	Jill	Jill	
	3	Tracy	Tracy	
	4	Helen	Helen	
101	**index**	**townList1**	**townList2**	As with puzzle 100 one list of names is copied from one list into the other. The use of the lower( ) function changes the capital letter at the beginning of each town name to lower case as all the towns are copied from townList1 into townList2.
	0	Dover	dover	
	1	Maidstone	maidstone	
	2	Ayr	ayr	
	3	Shepway	shepway	
	4	Pembroke	pembroke	
102	**index**	**places**	**letterCount**	The puzzle begins by assigning the place names shown to the "places" list. The code then loops through each place name counting the number of times the "o" character appears in each name. This value is stored in the letterCount list.
	0	Glasgow	1	
	1	Swansea	0	
	2	Lisburn	0	
	3	Thurso	1	
	4	Bolton	2	
103	**index**	**elements**		Each time this code loops it counts the number of "i" characters in the element names. The first time the if statement condition is true (Titanium has >1 "i" characters) the loop variable = 1. elements[loop] would store elements[4-loop] or elements[3] or "Lead" This only happens once more during the loop for "Silicon" (when loop = 4). As [4-loop] = [0], this results "Copper" being copied into elements[4].
	0	Copper		
	1	Lead		
	2	Iron		
	3	Lead		
	4	Copper		

<table>
<tr><th></th><th colspan="3">Answers</th><th>Additional notes</th></tr>
<tr><td rowspan="6">104</td><td>index</td><td colspan="2">numbers</td><td rowspan="6">The program compares two elements of the list. If the value in the first element is greater than the second the code swaps the two values.
Swapping two values in programming requires an additional variable.
temporary value = first value
first value = second value
second value = temporary value:
Note: The complete algorithm is known as a Bubble Sort. You'll find the complete code in the Standard Algorithms chapter of this book.</td></tr>
<tr><td>0</td><td colspan="2">9</td></tr>
<tr><td>1</td><td colspan="2">35</td></tr>
<tr><td>2</td><td colspan="2">45</td></tr>
<tr><td>3</td><td colspan="2">67</td></tr>
<tr><td>4</td><td colspan="2">92</td></tr>
<tr><td rowspan="6">105</td><td>index</td><td>nameList1</td><td>nameList2</td><td rowspan="6">In this code the first character [0:1] of words, in the same index, in both lists are compared.
If the first character of the word in nameList1 comes alphabetically before the first character of the word in nameList2, the two names are swapped using a temporary value as explained in the previous puzzle. Every name is swapped except the two in element 3 of the lists (U is greater than T so the condition is false)</td></tr>
<tr><td>0</td><td>Mary</td><td>Bob</td></tr>
<tr><td>1</td><td>Nida</td><td>Derek</td></tr>
<tr><td>2</td><td>Jill</td><td>Fred</td></tr>
<tr><td>3</td><td>Usman</td><td>Tracy</td></tr>
<tr><td>4</td><td>Helen</td><td>Abubakar</td></tr>
<tr><td rowspan="6">106</td><td>index</td><td colspan="2">values</td><td rowspan="6">The puzzle extracts and stores the first decimal place of each number.
It does this by first subtracting the int( ) of the number from the original, leaving just the decimal places.
For the first element this would be: 45.78 – 45 = 0.78
The code then multiplies the result by 10: 0.78 * 10 = 7.8
Finally the int( ) function is used to remove the decimal place: int(7.8) = 7</td></tr>
<tr><td>0</td><td colspan="2">7</td></tr>
<tr><td>1</td><td colspan="2">3</td></tr>
<tr><td>2</td><td colspan="2">1</td></tr>
<tr><td>3</td><td colspan="2">2</td></tr>
<tr><td>4</td><td colspan="2">9</td></tr>
<tr><td rowspan="6">107</td><td>index</td><td>firstValue</td><td>secondValue</td><td rowspan="6">The first if statement performs one of two calculations based on whether firstValue[num] is rounded up (in which case it will be greater than the number it started as) or rounded down. The result of the calculation is stored in secondValue[num].
If the result of the calculation is greater than 50 then 99 is stored in firstValue[num], if it equal to or less than 50 then 0 is stored.</td></tr>
<tr><td>0</td><td>99</td><td>72</td></tr>
<tr><td>1</td><td>0</td><td>12.0</td></tr>
<tr><td>2</td><td>0</td><td>12.0</td></tr>
<tr><td>3</td><td>0</td><td>42.0</td></tr>
<tr><td>4</td><td>99</td><td>90</td></tr>
</table>

Puzzle set 11 – 2D lists

	Answer	Additional notes
108	Hsin stepped outside	Use the indexes to look up the position of each word in the 2D list. For example: words[1][0] = first index 1, second index 0 = Hsin.
109	The match is off today	As with the previous puzzle the indexes are used to look up the position of five words stored in the 2D list.
110	make	The code uses the lower( ) function to change the word in index 4,4 to lower case.
111	Set	The first character of index 0,3 ("SET") is concatenated with the rest of the word. Note that the rest of the word is changed to lower case.
112	Gillian eagerly stepped outside	While the program loops, the first index remains fixed so the words printed are indexes: [5][1], [5][2], [5][3], [5][4] Each of the words will display on a new line.
113	set the bar high	While the program loops, the second index remains fixed so the words printed (in lower case because the lower( ) function is used) are indexes: [0][3], [1][3], [2][3], [3][3] The end= ' ' has been used so each of the words will display on the same line.
114	Great lives significantly affect everyone else	Note that the answer should be written on a single line. The outer loop in this program has a step meaning that first equals only 0 and 2. The inner loop ranges from 0 to 2.
115	19 12 7	The code adds the three numbers in each sublist (row) before displaying the result. 5 + 8 + 6 = 19 3 + 4 + 5 = 12 6 + 1 + 0 = 7
116	1	The two indexes given contain the values 8 and 4. As 8 divided by 4 does indeed equal 2, index [2][1] is displayed.
117	586 345	There are two points to note in this puzzle: Firstly, the range can be calculated as the length of the main index of grid (3) – 1 = 2, so the outer loop ("first") repeats twice (0 then 1). The second point is that the print statement does not print the space " " included in earlier examples so the three numbers in each sub list are displayed with no spaces between them.
118	34	This puzzle changes the values of three elements before adding up each of the element in the list of lists.
119	056 148 635	The outer and inner loops have been swapped and the ranges in both loops result in a backwards count through the indexes. The code displays the third value, in each sublist, in reverse order: 056 [5,8,6], [3,4,5], [6,1,0] As the second index (the outer loop) now decrements (reduces) by 1, the second value in each sub list is displayed in reverse order: 148 [5,8,6], [3,4,5], [6,1,0] Finally, the first value in each sub list is displayed in reverse order: 635 [5,8,6], [3,4,5], [6,1,0]

Puzzle set 12 – Predefined functions (max, min, sum, split, index, append)

	Answer	Additional notes
120	The most hours studied in a day was 6	The max( ) function is used to find and display the largest value "6" in the list.
121	The least hours played in a day was 2	The min( ) function is used to find and display the smallest value "2" in the list.
122	The total weight is: 50	The sum( ) function is used to add up all the values in the list.
123	Minimum 1 + Maximum 99 = 100	The max( ) and min( ) of "numList" are appended to a new list: [1,99]. The sum( ) function then adds these two new elements together and appends the result. [1,99,100]. These three values are displayed in formatted output.
124	Maximum = 72 Minimum = 2	The program takes 5 elements of "numList" and appends them to the empty "limits" list. [3, 72, 7, 23, 2] The program then displays the maximum and minimum values in the new list.
125	Peas Carrots Milk Tea Bags Bread Marmalade	The program splits the string by the commas and stores each word in a new list. The program then prints each item from the list, displaying the contents of one element on each line.
126	The first ball = 2 The first ball = 0 The first ball = 1	The program splits the string using the – symbol. ['2,3,0,0,0,1', '0,0,0,2,3,0', '1,1,0,0,0,6'] This creates a list of three strings. The loop then uses substring to display the first character in each element: "oneOver[0:1]"
127	The longest book = 949 pages.	The program splits the string using the space characters. ['123', '949', '823', '546', '1002', '398'] This is a bit of a trick question. The number characters in the new list are stored as strings so the max( ) function is looking alphabetically for the maximum string, in the case "949" because 9 is last alphabetically.
128	The longest book = 1002 pages.	The following lines were added to puzzle 127 after the split( ) function. `eachBookLength[loop]=int(eachBookLength[loop])` This line changes the strings in the list to integers. As numbers are now stored in the list, the maximum value displayed would now be 1002.
129	The lowest temp = -2 The highest temp = 24	The code is similar to puzzle 128. A string is split and then converted to a list of integers. The min( ) and max( ) functions are then used to display the smallest and largest values in the list.
130	The letter is at position 4	The program finds and displays the index of the first "o" characters in the string.

	Answer	Additional notes
131	The word is at position 4	The program stores and then displays the index of "shout" in the list "vocals". 0 1 2 3 4 ["say","speak","dictate","verbalise","shout"]
132	The word is at position 2	The program splits the sentence using the space character and stores each word in a list. It then searches for the index of "at", which is stored in index 2.

Puzzle set 13 – Predefined functions (pop, insert, remove, extend and append)

	Answer	Additional notes
133	[2, 4, 8, 10, 7]	Element at index 2 is first removed (popped) from the list [2, 4, 8, 10]. The value 7 is then added (appended) to the end of the list.
134	[3, 5]	pop(3) removes index 3 from the list [1,3,5,11] The next pop(3) removes the new index 3 [1,3,5] Finally, the first example of the value 1 (found at index 0) is removed from the list [3, 5].
135	[2, 3, 5, 6, 1]	The value 2 is inserted into the start of the list at index 0. [2,3,4,5,6] pop(2) then removes index 2 from the list. [2,3,5,6] The value 1 is appended onto the end of the list. [2, 3, 5, 6, 1]
136	[33, 66, 55]	The first example of the value 44 is removed from the list. [33,66,44,55,44] This is repeated twice more inside the loop. [33,66,55,44] [33,66,55]
137	[4, 5, 6]	The program stores the value "4" of index 2 of "days". It then appends this value three times to the night list, adding one more each time.
138	[5, 1, 2, 3, 4]	The program loops four times, during which the code removes the first item of the list before appending the same item to the end of the list as shown below. [2,3,4,5,1] [3,4,5,1,2] [4,5,1,2,3] [5,1,2,3,4].
139	[7, 12, 8, 6, 7]	The program loops from 5 down to 1. Each time the loop variable is used to append a new item to the "nights" list. For example, when loop = 5: temp=days.pop(loop) is temp=days.pop(5) = [4,6,4,5,8,2,4] so therefore, nights.append(2 + 5) appends 7 in the "nights" list
140	[1, 1]	The program loops through the list checking to see if values are greater than 1. If they are it removes the value from the list.
141	[4, 5, 4]	The program works out the average of the values stored in the list "points" (sum(3+4+5+1+3+1+4)/7=21/7 so average = 3). The code then loops through the list checking to see if each value is greater than the average. If it is, the value is appended onto the "above" list.

	Answer	Additional notes
142	[1, 1, 0, 0, 0, 0, 1, 1]	The program begins by working backwards through list "one" appending each value in turn to list "two". List "two" is therefore the reverse of list "one". one = [3,4,5,1,5,1,4,3] two = [3,4,1,5,1,5,4,3] The code then loops again comparing list "one" to list "two". Where the two numbers are equal, the value 1 is appended to list "three". Where the values are not equal, 0 is appended.

Puzzle set 14 – Modular programming

	Answer	Additional notes
143	Volume = 24	The values 2, 3 and 4 are passed into module1 as parameters. Within the module these three values are assigned to the formal parameters first, second and third. The three values (2, 3 and 4) are then multiplied and the result displayed as part of a message.
144	4	The function module4 is called with the parameters 8,3. Within the function middle is calculated as the remainder of 8 divided by 3 = 2 Within the function, top is calculated as the integer of 8/3=2. The variable total is therefore =4 (2+2). The total (4) is returned and stored in "value". The variable value is then displayed by the main program.
145	6 16.0	Two functions are called by this program. The function module2 divides the parameter passed (num1) by 10 and returns the result (10.0). The function module3 loops from the first parameter to the second (num2,num3) or (6,9) adding 2 onto a running total each repetition. After three repetitions, total = 6. The variable total (6) is displayed at this point before the result is returned. The last line of code adds and displays the values returned from two functions 16.0 (10.0+6).
146	8 8	The program calls module2 with the parameters 5 and 10. The parameters are used to set the range of a fixed loop. A range of 5 to 10 means 4 repetitions. During each repetition 2 is added onto the "total" variable. After 4 repetitions "total" = 8. The result is displayed twice. Once by the function and once after the result is returned to the main program.
147	6 Volume = 48.0	This code calls module1 using three parameters, two of which are values returned from other functions. module3(2,5) displays and return 6 as the loop repeats 3 times. module2(5*8) returns 4. 5*8 = a parameter of 40 which is then divided by 10 and returned. module1(2,6,4) displays a volume of 48 (2*6*4 = 48).

	Answer	Additional notes
148	Password is edvotedvo	The function textChanger1 is passed the parameter "voted". The function performs three different tasks depending on the length of the word it is passed. As "voted" is 5 characters long, the function copies the first and last two characters into temporary variables before concatenating them (in a different order) to the from and back of "voted", ed + voted + vo
149	Password is loaded	The string "LOADED" is stored and then passed to the function textChanger2. The comparisons in the function test to see if the word is in lower or upper case. In the example "word.upper( )" will equal "word" because applying the upper( ) function to text that is already upper case will have no effect. As "word.upper() == word" is true word.lower( ) is returned as then displayed in the main program.
150	Password is 25Hidden30	Function textChanger3 multiplies the number of characters in the passed parameter by 5. So for "Hidden" length = 30 (6 characters * 5). This number is then changed to a string and concatenated onto the beginning and end of "Hidden" (–5 for the value at the beginning).
151	Password is 20mixed25	Dug is first passed to the function textChanger2 which returns "mixed" as Dug is neither completely upper or lower case. "mixed" is then passed to the function textChanger3. This multiplies the length of "mixed" by 5 and stores the value 25. The function returns a string comprising of: "25-5" + "mixed" + "25" The password "20mixed25" is then displayed.
152	Password is 105XU105X	The values returned from each function are as follows: textChanger3 – xu passed in, 5xu10 returned textChanger2 – 5xu10 passed in, 5XU10 returned textChanger1 – 5XU10 passed in, 105XU105X is returned and displayed.
153	Password is E2520G	The values returned from each function are as follows: textChanger3 – gnome passed in, 20gnome25 returned textChanger2 – 20gnome25 passed in, 20GNOME25 returned textChanger1 – 20GNOME25 passed in, E2520G returned (as "20GNOME25" is > 6 characters long) and displayed.
154	Password is 4020MIXED2545	The values returned from each function are as follows: textChanger2 – Bruv passed in, mixed returned textChanger2 – mixed passed in, MIXED returned textChanger3 – MIXED passed in, 20MIXED25 textChanger3 – 20MIXED25 passed in, 4020MIXED2545 and displayed.
155	Password is 35RDHARDHA40	When one function is placed inside another as a parameter, the function within the brackets should be evaluated first. This puzzle there can be worked out as: textChanger1 – hard passed in, rdhardha returned textChanger2 – rdhardha passed in, RDHARDHA returned textChanger3 – RDHARDHA passed in, 35RDHARDHA40 returned and then displayed.

Puzzle set 15 – File handling

	Answer	Additional notes
156	1. Perseid 2. Lomond 3. Retford 4. Timbuktu	Each item in the list is written to the file with the following additions: A "count" variable is used to concatenate a number onto the start of each line. The "count" is incremented in the loop so that the number is one greater each time a line is written. A new line character is concatenated to the end of each line.
157	10 20 12 23 16 8	The "sizes.txt" is created and each element of the "moreSizes" list is written to the file. The file is then opened again with an append "a" connection. The elements of the "sizes" list are now written to the end of the file.
158	asdf	There are two things to spot in this puzzle. Firstly, the second connection is also a write "w" connection meaning anything written to the file by the first loop will be deleted and over-written by the second loop. Secondly, when the elements of the "keysTwo" list are written to the file, no end-of-line character is included. This means that each of the list elements will appear on the same line of the file.
159	Oxygen Neon Mercury	This program simply reads and displays each line of the file, minus the end-of-line character. Note that if the end-of-line characters were not removed, the output would be: Oxygen Neon Mercury as both the /n character and the print( ) statement create new lines.
160	George Washington John Adams Thomas Jefferson James Madison	Each line is read from the file and stored in a list, without splitting the name in a list element. The list is then displayed one element at a time using a loop.
161	1. James Monroe 2. John Quincy Adams 3. George Washington 4. John Adams 5. Thomas Jefferson 6. James Madison	The program reads the four presidents names from the text file and appends them to the list of two presidents. It then displays the list, using the loop variable to number the presidents.

<table>
<tr><th></th><th>Answer</th><th>Additional notes</th></tr>
<tr><td>162</td><td>The scores for Alex were:
97.7
97.3
97.8</td><td>The program reads the data from the file "RCaverages.csv", loops through each line of the data and splits the data into a two-element list (for example ["Kevin","98.2"]).
If the name entered by the user matches the name in the list, the second element of the list is appended to the "scores" list.
In the example, Alex was entered. Each time Alex appeared in the file, their score was appended to "scores".
The program displays a message, along with Alex's three scores from the file.</td></tr>
<tr><td>163</td><td>Summa 8.11
Petra 7.92
Kylie 8.13</td><td>The program reads the data from the file and splits it using the "-" symbol: ["Summa","7.23","7.34","8.11","8.08"].
The code appends index 0 to the empty "names" list.
Using the max( ) function, the code appends the maximum value found in indexes 1, 2, 3 and 4 to the empty "jump" list.
A loop and print statement are used to display each name with the largest number found in that line of the file.
Summa-7.23-7.34-8.11-8.08
Petra-6.92-7.33-7.92-7.55
Kylie-7.55-7.41-7.99-8.13</td></tr>
<tr><td>164</td><td>Total = 43</td><td>The program begins by reading data from two files into two 2D lists: "temp1List" and "temp2List".
Two loops then use specific ranges and indexes to copy only the highlighted data into a third file called "num3".
<code>for loop in range(0,2):
numbersOut.write(str(temp1List[loop][0])+"\n")</code>
<table><tr><td>temp1List</td><td colspan="4">second index</td></tr><tr><td>first index</td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td>10</td><td>20</td><td>30</td><td>40</td></tr><tr><td>1</td><td>20</td><td>30</td><td>40</td><td>50</td></tr><tr><td>2</td><td>30</td><td>40</td><td>50</td><td>60</td></tr><tr><td>3</td><td>40</td><td>50</td><td>60</td><td>70</td></tr></table>
<code>for loop in range(2,4):
numbersOut.write(str(temp2List[loop][3])+"\n")</code>
<table><tr><td>temp2List</td><td colspan="4">second index</td></tr><tr><td>first index</td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr><tr><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr></table>
A file reads these values back out of file "num3" and uses a running total to up the values (10+20+6+7).
This total is displayed with a message.</td></tr>
</table>