Evaluating and creating expressions and simple statements [SOLUTIONS]

Section 1

1.1

ii. Assignment. Variable b is assigned to a list of integers, containing the integer values shown
iii. Variable le is assigned to the length of the string referred to by variable b.
iv. The header of a for loop. The loop is looping over the consecutive integers from zero up to one less than the value contained in variable le. The for loop variable is i
v. In the body of the for loop. The value in variable a is printed out, followed by a space character - this is determined by the optional parameter “end”.
vi. In an assignment, the variable a is updated to refer to its old value minus the value at the ith index of list b.
vii. Finally, the value in variable a is printed out.

1.2

i. Assignment. Variable b is assigned to a string as shown
ii. Variable le is assigned to half the length of string variable v’s value - note that integer division is used - //
iii. The header of a for loop. The loop is looping over the consecutive integers from zero up to one less than the value contained in variable le. The for loop variable is i
iv. The value at the ith position of string b is printed out, followed by a space
v. The value i positions from the end of the string is printed out - if i is zero, that is the last position, 1, then it’s the second last position, and so on.

1.3

i. A value is input from the user, with the prompt as shown, and stored into variable inputString
ii. The empty string is assigned to variable newString
iii. The length of inputString is determined using the len function, and assigned to the variable index
iv. A while loop header. The Boolean expression in between while and : is evaluated. If the result is true, the indented lines 5 and 6 coming immediately after this line are executed and after that, this line is executed once again. If the result is false, then execution will jump to just after the indented code - in this case, line 9.
v. The value in variable index is decremented by 1.
vi. A new string is created by concatenating the value currently in variable newString and the single character string found at position index in the string in variable inputString. This new string is then assigned to variable newString.

vii. The string in variable newString is printed to the screen.

1.4

i. Assignment statement. The variable b is assigned a list of integers.
ii. Variable u is assigned a boolean which indicates if the first element of b (b[0]) is less than the second element of b (b[1]).
iii. Variable p is assigned the value of the second element of b (b[1]).
iv. A for loop header. The loop is looping from 2 up to the length of the list b. The loop variable is i.
v. An if statement. This will be True if the boolean u is True and if p is less than the ith element of b.
vi. Prints the character "x" followed by the loop variable i minus 1. The ending character will not be the default newline character, but a space.
vii. The boolean u is flipped (if it were True it is now False and vice versa)
viii. An elif statement, leading on from the if statement in v., which will be True if u is False and p is less than the second element of b.
ix. A print function identical to the above, except that "y" is printed in place of "x"
x. The boolean u is flipped, as in vii.
xi. The variable p is assigned the value of the ith element of b

Section 2

2.1

i. 37
ii. Error - can’t concatenate a str to a list (only another list)
iii. Error - Python is case-sensitive (or). Otherwise True.
iv. 2
v. 3
vi. True
vii. Error - fred needs to be in quotes (will get an error saying ‘fred’ is not defined)

2.2

i. 3
ii. [7, 2]
iii. Error - Python is case-sensitive (or). Otherwise False.
iv. 4
v. Error - unsupported operand type(s) for +: 'int' and 'str'
vi. False
vii. 5

2.3

i. 1978
ii. 1984.0

iii. 5
iv. 4
v. 1983
vi. 12.0
vii. 7777
viii. 14.0
ix. “999"

2.4

i. True
ii. False
iii. True
iv. True
v. True

Section 3

1. `s[ : n ] + s [ n+1 : ]`
2. `range( 1, 101 )`
3. `range( len( a ) )`
4.
   i.
   ```
   for c in s:
       print( c )
   ```
   ii.
   ```
   for c in s:
       print( c, end = " " )
   ```
5. `v >= 0 and v < len( li )`
6. `[ 0 ] * 20`
7.
   i. `s[ -1 ]`
   ii. `s[ -2: ]`
   iii. `s[ : len( s ) // 2 + 1 ] + s [ len( s ) // 2 + 2 : ]`
   iv. `s[ 0 ] + s[ -1 ]`
   v. `s[ : -1 ]`
8.
   i. `lis[ -1 ] = lis[ -1 ] + 1` or `lis[ -1 ] += 1`
   ii.
   ```
   sum = 0
   for v in lis[ 1: ]:
       sum = sum + v
   lis[ 0 ] = lis[ 0 ] + sum
   ```
   iii. `lis = lis[ : -1 ]`
   iv. `lis = lis[ : 2 ] + lis[ -2 : ]`

Section 4

4.1

i. Open a file called someData.txt and assign the file handle/reference to variable myFile
ii. Read all the lines in the file referred to by myFile storing them in a list, each line in a separate list element. This list is assigned to variable lines.
iii. For loop header. Using the functions range and len, loop over the indices of the list, from 0 up to one less than the length of the list. The loop variable is i. The body of the loop is lines 4 and 5 - these are the code that is repeated.
iv. Print the value of i incremented by 1 as a string, followed by a space, and then
v. Print the value in the list lines at index position indicated by the value held in variable i.

4.2

i. Open a file, as above
ii. Read in the lines, as above
iii.
iv. Loop over all the values in the list lines. The loop variable line takes on each successful value (these are the lines from the file). Lines 5-9 are being repeated in the loop.
v. line[:-1] returns the same string as line referred to, only the final character is removed. In this case, that final character is the new line character in the file. This updated version of the line is stored back in the variable
vi. The string referred to by the line variable is split into substrings which are stored in a list. The dividing points are commas, and these are not included in the substrings created. This list is stored into variable items
vii. A for loop, looping over the substrings in the list referred to by items. The for loop variable is called piece.
viii. The single line body of the loop - the substring in variable piece is printed and then a space afterwards
ix. A blank line is printed

4.3

i. Open the file
ii. Read in the lines
iii. Create an empty list
iv. Loop over all the lines read in
v. Store the line as a list, with words split on ",", using [:-1] to remove the newline character
vi. Create a dictionary variable which has the name as the first element in the split list...
vii. And the age as the second item, after casting it to an int
viii. Add this dictionary record to the list we created earlier

ix. Loop over every entry in the table list
x. Check if the age is greater than or equal to 17
xi. If so, print the record's name

4.4

i.
```
dictList[7]["Quintin"]
```

ii.
```
dictList += [{"Jeremy": 128}]
```

iii.
```
myFile = open("file.txt")
lines = myFile.readlines()
print(lines)
```

iv.
```
if "CS" in d:
     d["CS"] += [7]
else:
     d["CS"] = [7]
```