8a

Working with Lists I - Solutions

Introduction

list_tasks_start.py

```
import random as rnd

# generate some random data to play with
rnd.seed(42)
data = rnd.choices(range(100), k=10)

print("Generated data is")
print(data)
```

```
Generated data is

[63, 2, 27, 22, 73, 67, 89,
8, 42, 2]
```

- All problems use the same dataset, which is generated by the code shown in `list_tasks_start.py`. Changing the random seed, we can get different, but repeatable datasets.
- Since all solutions will start with the code in `list_tasks_start.py` those lines are not repeated in the solutions here to save space. But the code need to be in the solution scripts AS IS, and coders append their solution.
- The intention here is to develop the understanding of loops and conditionals, and the difference between indexes and the data values in lists. So while many questions have a builtin function answer **`min`**, `reverse` etc. we are only interested in solutions that avoid their use.
- Recap of Python Loops:
  - Use **`for`** `d` **`in`** `data:`
    To have a loop variable, `d`, that runs over the data values in the list `data`.
  - Use **`for`** `k` **`in`** **`range`**`(`**`len`**`(data)):`
    To have a loop variable, `k`, that runs over the indexes needed to access all the data values in list `data`. Then `k` is the index, while `data[k]` is the data value at that index.
  - Use **`for`** `k,d` **`in`** **`enumerate`**`(data):`
    To have two loop variables, `k` and `d`, that runs over the indexes and the data values in the list `data`. So `k` and `d` satisfy property `data[k]==d`.
- The first few are warm up exercises and should be very doable, but near the end they get very hairy. But it depends on the coder — for those that need help I get them to redo problems using each of the different for loop patterns above (if applicable). This will be of benefit to them later.

CoderDojo, Tramore, Waterford. (kmurphy@wit.ie)

Programming Tasks

Using the generated dataset:

[63, 2, 27, 22, 73, 67, 89, 8, 42, 2]

list_tasks_1

Save your `list_tasks_start.py` as `list_tasks_1.py` and add code to print out the data value at index 5.

Full solution is

list_task_1.py

```
1  import random as rnd
2
3  # generate some random data to play with
4  rnd.seed(42)
5  data = rnd.choices(range(100), k=10)
6
7  print("Generated data is")
8  print(data)
9
10 print(data[5])
```

```
Generated data is
[63, 2, 27, 22, 73,
   67, 89, 8, 42, 2]
67
```

But only displaying required solution by coder we have

list_task_1.py

```
10 print(data[5])
```

```
67
```

list_tasks_2

Save your `list_tasks_start.py` as `list_tasks_2.py` and add code to print out the data values, with one value per line — you need a **for** loop.

list_task_2.py

```
10 for d in data:
11     print(d)
12
13 # OR
14
15 for k in range(len(data)):
16     print(data[k])
17
18 # OR
19
20 for k,d in enumerate(data):
21     print (d)
```

```
63
2
27
22
73
67
89
8
42
2
```

While any of the three approaches given above work. The first is the python preferred due to being the simplest. However, the second approach is the one needed for the next question.

list_tasks_3

Save your `list_tasks_start.py` as `list_tasks_3.py` and add code to print out the data values, with one value per line but in reverse order.

The sequence of suggestions that I would give here are:

1. Output list elements is in current order using second alternative of question 1.
2. Replace `data[k]` with `k` to output the index instead of the data values. This will produce `0,1,2,...`.
3. What mathematical operation do I need to change `0,1,2,...` to `9,8,7,...`?
   — Replace `k` by `9-k` or even better `len(data)-1-k`.
4. So now we have the index in reverse order how do we get the corresponding data values?
   — Wrap `len(data)-1-k` by `data[len(data)-1-k]`.

list_task_3.py

```
10 for k in range(len(data)):
11     print(data[len(data)-1-k])
```

```
2
42
8
89
67
73
22
27
2
63
```

Python list have method `reverse` or we could use features of the `range` function which will greatly simplify this task but we want solutions that avoid such shortcuts.

list_tasks_4

Save your `list_tasks_start.py` as `list_tasks_4.py` and add code to print out the maximum data value in the list.

```
89
```

The sequence of suggestions that I would give here are:

1. First we should use the `for d in data` loop since we are just interested in data values not their index/positions.
2. Do a thought experiment where I am showing you a deck of cards — one at a time – how do you determine what is my max card?
   — keep a record of max value seen so far.
3. And since computers are (fast but) stupid we need to explicitly set the value of our "max so far" before we start our loop (i.e., start showing each individual card).

4. We have a few choices for the initial value of `myMax`. Setting it equal to the first element in the list is simplest (and we will do this here) but this can cause problems. Why?

— what happens if the list is empty?

list_task_4.py

```
10 myMax = data[0]
11 for d in data:
12     if d>myMax:
13         myMax = d
14 print(myMax)
```

```
89
```

list_tasks_5

Save your `list_tasks_start.py` as `list_tasks_5.py` and add code to print out the minimum data value in the list.

```
2
```

1. This problem is here to make sure that coders understood the previous task. The only difference here is keeping a record of the smallest value seen so far instead of the largest value.
2. Make sure good variable names are used `myMin` not `myMax`

list_task_5.py

```
10 myMin = data[0]
11 for d in data:
12     if d<myMin:
13         myMin = d
14 print(myMin)
```

```
2
```

list_tasks_6

Save your `list_tasks_start.py` as `list_tasks_6.py` and add code to print out the difference between the maximum value and the minimum data value in the list.

1. Start by getting the code for keeping track of the max and the min seen so far working.
2. Should do this in a single foo loop.
3. Use `print(myMax,myMin)` to output both values to help debugging.
4. Finally edit print function to output difference.

```
87
```

list_task_6.py

```
10 myMax = data[0]
11 myMin = data[0]
12 for d in data:
13     if d>myMax:
14         myMax = d
15     if d<myMin:
16         myMin = d
17 print(myMax-myMin)
```

```
87
```

list_tasks_7

Save your `list_tasks_start.py` as `list_tasks_7.py` and add code to print out the index of the maximum value in the list.
If the maximum value appears more than once, then print out index of first occurrence.

```
6
```

1. Now these tasks are beginning to get a little harder. Here we want both data values (to see what is the max value) and indexes/positions (to see where the max value is). So use the third loop alternative given in task 1.
2. Using the `for k,d in enumerate(data)` loop we find the max value as usual.
3. Next we add code to keep a record of the index as well, don't forget to initialise this variable before the loop.

list_task_7.py

```
myMax = data[0]
myMaxIndex = 0
for k, d in enumerate(data):
    if d>myMax:
        myMax = d
        myMaxIndex = k
print(myMaxIndex)
```

```
6
```

list_tasks_8

Save your `list_tasks_start.py` as `list_tasks_8.py` and add code to print out the index of the minimum value in the list.
If the minimum value appears more than once, then print out index of first occurrence.

```
0
```

1. Like tasks 4 and 5, this is very similar to task 7 to check understanding.
2. You could ask how to modify solution so that it outputs position of last min value.

list_task_8.py

```
myMin = data[0]
myMinIndex = 0
for k, d in enumerate(data):
    if d<myMin:
        myMin = d
        myMinIndex = k
print(myMinIndex)
```

```
1
```

list_tasks_9

Save your `list_tasks_start.py` as `list_tasks_9.py` and add code to print out the total of the data values in the list.

```
395
```

1. Since we only need the data values, we can go back to using the easier `for d in data` type loop.
2. We need to keep a running total of the data values seen so far.

list_task_9.py

```
total = 0
for d in data:
    total = total + d
print(total)
```

```
395
```

list_tasks_10

Save your `list_tasks_start.py` as `list_tasks_10.py` and add code to print out the total of all of the even data values minus the total of the odd data values in the list.

```
-243
```

1. First modify the previous solution so that the total only involves the even numbers. To check if data value `d` is even we use `d%2==0`.
2. Next modify to have update a second total, the total of odd numbers.
3. Finally print out the difference.
4. Note that there is a solution using only one total and does addition/subtraction within the loop.

list_task_10.py

```
evenTotal = 0
oddTotal = 0
for d in data:
    if d%2==0:
        evenTotal = evenTotal + d
    else:
        oddTotal = oddTotal + d
print(evenTotal-oddTotal)

# OR

total = 0
for d in data:
    if d%2==0:
        total = total + d
    else:
        total = total - d
print(total)
```

```
-243
```

list_tasks_11

Save your `list_tasks_start.py` as `list_tasks_11.py` and add code to print out the second largeest data value in the list.

```
73
```

1. We need to keep track of two data values — the largest found so far and the second largest found so far.
2. We update the second largest when we get a data value (`d`) larger than the second largest found so far (`mySecondMax`) but less than the max found so far (`myMax`).

3. We update both the second largest and the largest when we get a value that is larger than the max found so far.

list_task_11.py

```
myMax = 0
mySecondMax = 0
for d in data:
    if d>myMax:
        mySecondMax = myMax
        myMax = d
    elif d>mySecondMax:
        mySecondMax = d
print(mySecondMax)
```

```
73
```