CSI 102: Lab 04

Lists

Anthony Cavallo

10.07.2025

I certify that this lab report is entirely my own work.

Introduction

This lab focused on using lists in Python to store and manipulate data. The primary goals were to: use single-dimensional lists, add/remove/search data, iterate through lists, use nested lists for 2D structures, and practice writing and executing Python programs.

Methods

Task 1: High Scores (highScoreUpdate.py)

The program replaces separate variables for top scores with two lists: one for scores and one for player initials. Input is accepted repeatedly; new scores are inserted into the correct position in the top-three list using element shifts. If a score does not beat existing top three, it is discarded.

Here's a snippet of what the code looks like:

```
for i in range(len(scores)):
  if score > scores[i]:
    for j in range(len(scores)-1, i, -1):
      scores[j] = scores[j-1]
    scores[i] = score
```

```python
# Author: Anthony Cavallo
# Date: 10/07/2025
# Description: Update high scores using lists instead of individual variables
# Code of honesty: I certify that this lab is entirely my own work.

# create lists for scores and initials
scores = [0, 0, 0]
initials = ["", "", ""]

print("Root Beer Tapper - High Scores (type 'exit' to stop)")
print("Current High Scores:")
for i in range(3):
    print(i + 1, ".", initials[i], "-", scores[i])

entry = ""

while entry.lower() != "exit":
    entry = input("Enter a new score (or 'exit' to finish): ")
    if entry.lower() == "exit":
        break

    if not entry.isdigit():
        print("Please enter a number or 'exit'.")
        continue

    score = int(entry)
    player = input("Enter the player's initials (max 3 letters): ")[:3]

    # check where to place new score
    if score > scores[0]:
        scores[2] = scores[1]
        scores[1] = scores[0]
        scores[0] = score

        initials[2] = initials[1]
        initials[1] = initials[0]
        initials[0] = player
    elif score > scores[1]:
        scores[2] = scores[1]
        scores[1] = score

        initials[2] = initials[1]
        initials[1] = player
    elif score > scores[2]:
        scores[2] = score
        initials[2] = player
    else:
        print("That score did not make the top three.")

    print("\nUpdated High Scores:")
    for i in range(3):
        print(i + 1, ".", initials[i], "-", scores[i])
    print()

print("Final High Scores:")
for i in range(3):
    print(i + 1, ".", initials[i], "-", scores[i])
print("Goodbye!")
```

Task 2: Inventory Manager (invManager.py)

A looped menu allows adding (up to 5 items), removing, displaying, and exiting. Removal searches the list safely and reports if the item was not found.

Here’s a snippet of what the code looks like:

```
if len(rucksack) >= max_items:
   print("Your rucksack is full.")
else:
   rucksack.append(item)
```

```python
# Author: Anthony Cavallo
# Date: 10/07/2025
# Description: Simple Inventory Manager using lists
# Code of honesty: I certify that this lab is entirely my own work.

rucksack = []
max_items = 5
choice = ""

while choice != "4":
    print("\nInventory Manager")
    print("1. Add an Item")
    print("2. Remove an Item")
    print("3. Display the Contents")
    print("4. Leave the Inventory Manager")

    choice = input("Choose an option (1-4): ")

    if choice == "1":
        if len(rucksack) >= max_items:
            print("Your rucksack is full. You can't add more items."
        else:
            item = input("Enter the item to add: ")
            if item == "":
                print("You didn't type anything.")
            else:
                rucksack.append(item)
                print(item, "has been added.")
    elif choice == "2":
        if len(rucksack) == 0:
            print("Your rucksack is empty.")
        else:
            item = input("Enter the item to remove: ")
            if item in rucksack:
                rucksack.remove(item)
                print(item, "has been removed.")
            else:
                print("That item is not in your rucksack.")
    elif choice == "3":
        if len(rucksack) == 0:
            print("Your rucksack is empty.")
        else:
            print("\nItems in your rucksack:")
            for i in range(len(rucksack)):
                print(i + 1, ".", rucksack[i])
    elif choice == "4":
        print("Leaving the Inventory Manager. Goodbye!")
    else:
        print("Invalid option, please choose between 1 and 4.")
```

Task 3: Highs and Lows (elevation.py)

A 7x7 two-dimensional list was created and filled with random integers 10-99. The program iterates to find and record the highest and lowest values and their coordinates.

Here’s a snippet of what the code looks like:

```
for r in range(size):
  for c in range(size):
    grid[r][c] = random.randint(10,99)
```

```python
# Author: Anthony Cavallo
# Date: 10/07/2025
# Description: Creates a 7x7 grid with random elevations and finds the highest and lowes
# Code of honesty: I certify that this lab is entirely my own work.

import random

rows = 7
cols = 7
elevations = []

# create 7x7 list filled with 0
for r in range(rows):
    row = []
    for c in range(cols):
        row.append(0)
    elevations.append(row)

# fill with random numbers between 10 and 99
for r in range(rows):
    for c in range(cols):
        elevations[r][c] = random.randint(10, 99)

# find highest and lowest
highest = -1
low = 999
high_x = 0
high_y = 0
low_x = 0
low_y = 0

for r in range(rows):
    for c in range(cols):
        if elevations[r][c] > highest:
            highest = elevations[r][c]
            high_x = r
            high_y = c
        if elevations[r][c] < low:
            low = elevations[r][c]
            low_x = r
            low_y = c

print("Highest elevation:", highest, "at row", high_x, "column", high_y)
print("Lowest elevation:", low, "at row", low_x, "column", low_y)
print()

print("Elevation Map:")
for r in range(rows):
    for c in range(cols):
        print(elevations[r][c], end=" ")
    print()

```

Results

High Scores: Completed highScoreUpdate.py which maintains top three scores and initials stored in lists. The program inserts new scores in order and displays the current top three.

```
Current High Scores:
1 .  - 0
2 .  - 0
3 .  - 0
Enter a new score (or 'exit' to finish):
```

Inventory Manager: Created invManager.py implementing add/remove/display with a maximum capacity of five.

```
Inventory Manager
1. Add an Item
2. Remove an Item
3. Display the Contents
4. Leave the Inventory Manager
Choose an option (1-4):
```

Highs and Lows: Created elevation.py which prints the 7x7 elevation grid and reports highest and lowest values with coordinates.

```
Highest elevation: 99 at row 5 column 0
Lowest elevation: 12 at row 2 column 0

Elevation Map:
45 33 98 62 54 70 35
45 53 79 71 14 93 71
12 45 61 77 45 79 65
97 94 53 16 36 84 30
33 15 91 22 39 83 35
99 28 37 46 92 20 26
```

Discussion

This lab reinforced list manipulation in Python. Using lists simplified operations such as inserting and removing elements as compared to many individual variables. The two-dimensional list was useful for representing grid-like data. I decided to be a bit more creative in this and personalized the code as separate mini projects to give them my own style. These concepts can be applied to leaderboards, inventory systems in games, and map-based analyses in future projects.

Challenges

One challenge was ensuring that shifting scores does not overwrite values; this was handled by shifting elements from the end toward the insertion point. Another consideration was safe removal of items from lists without causing runtime exceptions.

Conclusion

The lab successfully demonstrated the power and flexibility of lists in Python. Implementing the three programs showed practical ways lists make data management easier and less error-prone.

Appendix

Files included: highScoreUpdate.py, invManager.py, elevation.py.