CS5001 HW5

Instructions

- Write your name as well as your NU ID on your assignment. Please number your problems.
- Submit both results and your code.
- Give complete answers. Do not just give the final answer; instead, show steps you went through to get there and explain what you are doing. Do not leave out critical intermediate steps.
- This assignment must be submitted electronically through Gradescope by October 21st, 2024 (Monday) by 11:59PM.
- All of your codes must be commented.

Written Questions

1. **What is loop?**
2. **What is "for else" statement?**
3. **What is the infinite loop problem? When you might run into the problem? How to avoid or fix it?**

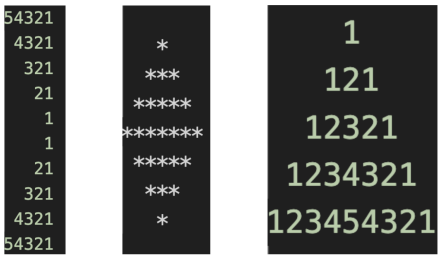
Coding Questions

1. Intensive Practice of For-Else

- `Check if a list is sorted`: Write a program that checks if a list of numbers is sorted in ascending order. If any number is out of order, print "List is not sorted" and stop the loop. If the loop completes without finding any issues, print "List is sorted" using for-else.
- `Check if a string contains only vowels`: Write a program that checks if a string contains only vowels (a, e, i, o, u). If a non-vowel character is found, print "Non-vowel character found" and stop. If the loop completes without finding a non-vowel, print "All vowels" using for-else.

- `Search for the longest word in a list`: Write a program that searches for the longest word in a list of words. If all words have the same length, print "All words are of equal length." Otherwise, print the longest word using a for-else structure.
- `Find the hidden treasure`: Write a program that simulates a treasure hunt on a 2D grid. The grid is represented as a list of lists, with one of the grid cells marked as the treasure ('T'). The program should search the grid row by row. If the treasure is found, print "Treasure found!" and stop. If the search completes and the treasure is not found, print "Treasure not found, keep looking!" using for-else.
- `Alien invasion detection`: Write a program that simulates an alien defense system. The system scans the sky and checks for any UFOs in a list of sky objects. If a UFO ('UFO') is detected, print "Alien invasion detected, prepare defenses!" and break the loop. If no UFOs are detected after scanning all sky objects, print "Sky is clear, no alien activity" using for-else.

2. Write a python code to print the following pattern with loop.



3. Loops in Autonomous Driving System Using Bounding Box Detection

In an autonomous driving system, bounding boxes are often used to detect and locate objects within the camera's field of view. A bounding box is defined by two pairs of coordinates `(x1, y1)` and `(x2, y2)`, where `(x1, y1)` represents the top-left corner of the box and `(x2, y2)` represents the bottom-right corner. These coordinates define the area in which an object, such as a pedestrian or vehicle, is detected.

In this exercise, you will keep creating a Python program that manages detected objects using bounding boxes.

Bounding Box Data Structure

Each detected object will be stored in a dictionary, where the key is the object's unique identifier (such as an integer ID) and the value is a dictionary representing the bounding box:

```
bounding_boxes = {
    1: {'x1': 50, 'y1': 100, 'x2': 200, 'y2': 250},
    2: {'x1': 300, 'y1': 400, 'x2': 350, 'y2': 450},
}
```

- **Calculate the Area of Bounding Boxes**: Write a Python program that iterates through a dictionary of bounding boxes and calculates the area for each one. The area of a bounding box can be calculated as:

$$\text{Area} = (x_2 - x_1) \times (y_2 - y_1)$$

  Store the areas in a new dictionary where the keys are the object IDs and the values are the calculated areas. Use a `for` loop to iterate through the bounding boxes.

- **Detect Overlapping Bounding Boxes**: Write a Python program that detects if any two bounding boxes overlap. For two bounding boxes to overlap, the following conditions must be met:

$$x_{11} < x_{22} \text{ and } x_{21} > x_{12} \text{ and } y_{11} < y_{22} \text{ and } y_{21} > y_{12}$$

  Use nested `for` loops to compare each bounding box with every other bounding box in the dictionary. If an overlap is detected, print a message indicating which two bounding boxes overlap and stop checking further.

- **Track Moving Objects**: Write a Python program that simulates tracking objects over multiple frames. In each frame, the position of each bounding box is updated based on a given movement vector for that object (e.g., {`'dx': 10, 'dy': -5`}). Use a `while` loop to simulate multiple frames, updating the bounding boxes in each iteration. After each frame, print the updated bounding box coordinates. Stop the loop after simulating 5 frames.

- **Merge Overlapping Bounding Boxes**: Write a Python program that merges any two overlapping bounding boxes into a single larger bounding box. The new bounding box should have its top-left corner as the minimum of (x_1, y_1) values and its bottom-right corner as the maximum of (x_2, y_2) values from the overlapping bounding boxes. Use nested `for` loops to compare all bounding boxes, and if an overlap is detected, create a new bounding box and update the dictionary.