22nd September Notes

Pattern problems

Hollow Square Box with Equal Spacing

```
# 22/09/25

# Hollow square box with equal spacing

size = 7  # Change to any size you like

for i in range(size):
    for j in range(size):
        if i == 0 or i == size - 1 or j == 0 or j == size - 1:
            print("*", end=" ")
        else:
            print(" ", end=" ")
    print()
```

```
* * * * * * *
*           *
*           *
*           *
*           *
*           *
* * * * * * *
```

size = 7 # Change to any size you like

- **Purpose:** Sets the size of the square. Here, size = 7 means the square will be 7 rows × 7 columns.
- You can change this number to make a bigger or smaller square.

for i in range(size):

- **Outer loop (rows):** Loops through each row from 0 to size-1.
- i represents the **current row index**.

---

for j in range(size):

- **Inner loop (columns):** Loops through each column in the current row.
- j represents the **current column index**.

```
if i == 0 or i == size - 1 or j == 0 or j == size - 1:
    print("*", end=" ")
```

- **Condition:** Checks if the current position is on the **border of the square**.
    - i == 0 → top row
    - i == size - 1 → bottom row
    - j == 0 → first column
    - j == size - 1 → last column
- If **any of these** are true, print a *.
- end=" " ensures that the stars are printed on the same line with a space between them.

```
else:
    print(" ", end=" ")
```

- **Else part:** For all positions **not on the border**, print a space " ".
- This creates the **hollow effect** inside the square.

```
print()
```

- Moves to the **next row** after finishing all columns of the current row.

Output for size = 7:

```
* * * * * * *
*       *
*       *
*       *
*       *
```

* *

* * * * * * *

- The stars form the **border**, and spaces create the **hollow interior**.
- Each row and column is **evenly spaced** because of end=" ".

Pyramid with Equally Spaced Stars

```
 # Pyramid with equally spaced stars

rows = 5  # You can change this number

for i in range(rows):
    # Print leading spaces
    print("-" * (rows - i - 1), end="")

    # Print stars with spaces between
    for j in range(i + 1):
        print("*", end=" ")

    print()  # Move to next line
```

```
----*
---* *
--* * *
-* * * *
* * * * *
```

rows = 5 # You can change this number

Purpose: Sets the number of rows in the pyramid.

Changing this number will make the pyramid taller or shorter.

for i in range(rows):

Outer loop (rows): Loops through each row from 0 to rows - 1.

i represents the current row index.

```
print("-" * (rows - i - 1), end="")
```

Leading spaces: For each row, prints rows - i - 1

hyphens (-) to shift the stars to the center.

end="" ensures stars are printed on the same line after the spaces.

Example for row 0 (i = 0) when rows = 5: prints 4 hyphens before the first star.

```
for j in range(i + 1):
    print("*", end=" ")
```

Inner loop (columns/stars): Prints i + 1 stars for the current row.

end=" " adds a space between stars to keep them evenly spaced.

```
print()  # Move to next line
```

After finishing the stars in the current row, moves to the next line.

Half Diamond Shape Using Stars

```
# Diamond shape using stars

rows = 5  # You can change this number

# Top half
for i in range(1, rows + 1):
    print("*" * i)

# Bottom half
for i in range(rows - 1, 0, -1):
    print("*" * i)
```

```
*
**
***
****
*****
****
***
**
*
```

detailed explanation **diamond shape using stars** code:

Code Heading: Diamond Shape Using Stars

rows = 5 # You can change this number

- **Purpose:** Sets the number of rows for the **top half** of the diamond.
- Changing this number will make the diamond taller or shorter.

Top Half of Diamond

for i in range(1, rows + 1):

  print("*" * i)

- **Outer loop:** i goes from 1 to rows (inclusive).
- i represents **the number of stars in the current row**.
- print("*" * i) prints i stars consecutively on the same line.
- Example for rows = 5:

*

**

***

****

*****

- This creates the **top half of the diamond**.

Bottom Half of Diamond

for i in range(rows - 1, 0, -1):

  print("*" * i)

- **Outer loop:** i goes **downwards** from rows - 1 to 1.
- print("*" * i) prints i stars for each row.
- Example for rows = 5:

****

***

**

*

- This creates the **bottom half of the diamond**.

Complete Output (rows = 5):

*

**

***

****

*****

****

***

**

*

- The diamond is **made of stars**, but note that this version is **left-aligned**.
- Each row simply adds or removes stars without centering.

Centered Diamond Shape Using Stars

```python
n = 5  # number of rows for the top half

# Top half of the diamond
for i in range(1, n + 1):
    # Print leading spaces
    print(" " * (n - i), end=" ")
    # Print stars
    print("*" * (2 * i - 1))

# Bottom half of the diamond
for i in range(n - 1, 0, -1):
    # Print leading spaces
    print(" " * (n - i), end=" ")
    # Print stars
    print("*" * (2 * i - 1))
```

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

Detailed explanation of **centered diamond shape using stars** code:

Code Heading: Centered Diamond Shape Using Stars

n = 5 # number of rows for the top half

- **Purpose:** Sets the number of rows for the **top half** of the diamond.
- Changing this number will make the diamond taller or shorter.

Top Half of Diamond

for i in range(1, n + 1):

  # Print leading spaces

  print(" " * (n - i), end=" ")

```
    # Print stars
    print("*" * (2 * i - 1))
```

Explanation:

1. **Loop:** i goes from 1 to n (inclusive). Each i is the current row number.
2. **Leading spaces:**
3. print(" " * (n - i), end=" ")
    - Prints (n - i) spaces to **center the stars**.
    - The end=" " ensures the stars appear on the same line immediately after the spaces.
4. **Stars:**
5. print("*" * (2 * i - 1))
    - Prints an **odd number of stars** for each row: 1, 3, 5, ...
    - (2 * i - 1) ensures symmetry in the diamond.

Bottom Half of Diamond

```
for i in range(n - 1, 0, -1):
    # Print leading spaces
    print(" " * (n - i), end=" ")
    # Print stars
    print("*" * (2 * i - 1))
```

Explanation:

1. **Loop:** i goes **downward** from n - 1 to 1.
2. **Leading spaces:** Same logic as the top half. More spaces as you move down.
3. **Stars:** Prints (2 * i - 1) stars, decreasing each row.

- This creates the **bottom half of the diamond**, perfectly symmetric to the top half.

Output (n = 5):

```
    *
```

```
  ***
 *****
*******
*********
*******
 *****
  ***
   *
```

- The diamond is **centered** and each row has an **odd number of stars** for symmetry.
- Leading spaces ensure **even spacing** on both sides.