>>>> DAY 7:

🔹 1. `range()` Function in Python

- Generates a **sequence of numbers** for looping.
- Syntax: `range(start, stop, step)`
- `stop` is **exclusive**.
- Doesn't create a list; returns a **range object** (lazy).
- Common uses:
  - `range(5)` → 0 to 4
  - `range(1, 6, 2)` → 1, 3, 5
  - `range(10, 0, -1)` → reverse loop

🔹 2. `end` Parameter in `print()`

- Default: `print()` ends with a newline (`\n`).
- `end` changes what happens after print.
  - `end=" "` → stay on same line, add space
  - `end=""` → stay on same line, no space

Example:

```
for i in range(5):
    print("*", end="")
```

- → prints all stars in one line.

🔹 3. Pyramid Pattern Logic

You understood how nested loops build visual patterns.

Code:

```
for i in range(1, 6):
    for j in range(5 - i): print(" ", end="")
    for k in range(2 * i - 1): print("*", end="")
    print()
```

Logic:

- Outer loop → controls rows
- First inner loop → prints spaces (for alignment)
- Second inner loop → prints stars
- Final `print()` → moves to next line
  → Builds a centered pyramid:

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

🔹 4. Pascal's Triangle

- Triangular arrangement of numbers.
- Each number = **sum of the two above it**.
- Formula for element:
  C(n,k)=n!k!(n−k)!C(n, k) = \frac{n!}{k!(n-k)!}C(n,k)=k!(n−k)!n!
- Represents binomial coefficients → used in (a+b)n(a + b)^n(a+b)n.

Example rows:

```
 1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
```

-
- Python logic (iterative generation shown).

🔹 5. Harmonic Series

- Series:
  1+12+13+…+1n1 + \frac{1}{2} + \frac{1}{3} + … + \frac{1}{n}1+21+31+…+n1
- No closed formula; use summation.

- Approximation:
  Sn≈ln(n)+0.5772S_n ≈ \ln(n) + 0.5772Sn≈ln(n)+0.5772
  (Euler–Mascheroni constant)

- Divergent — grows infinitely, but slowly.

- Python code for sum:

```
s = sum(1/i for i in range(1, n+1))
```