**Day 11: Nested Loop Patterns**

**Example 1: Mixed Pattern ('+' and '5')**

**Definition:** This nested loop pattern uses the variable a=6.

* The outer loop (i) iterates from 1 to 5.
* The first inner loop (j) prints an increasing number of + symbols (from 1 to i).
* The second inner loop (k) prints a decreasing number of 5s (from i up to 5).

**Example Code:**

Python

a=6

for i in range(1,a,1):

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

print("+",end=" ")

for k in range(i,6,1):

print(5,end=" ")

print()

**Output:**

Plaintext

+ 5 5 5 5 5

+ + 5 5 5 5

+ + + 5 5 5

+ + + + 5 5

+ + + + + 5

**Example 2: Mixed Pattern ('+' and Decreasing Numbers)**

**Definition:** This pattern uses an outer loop (i) that counts *down* from 5 to 1.

* The first inner loop (j) prints an increasing number of + symbols.
* The second inner loop (k) prints a decreasing sequence of numbers (5 down to 1, then 5 down to 2, etc.).

**Example Code:**

Python

for i in range(5,0,-1):

for j in range(5,i-1,-1):

print("+",end=" ")

for k in range(5,5-i,-1):

print(k,end=" ")

print()

**Output:**

Plaintext

+ 5 4 3 2 1

+ + 5 4 3 2

+ + + 5 4 3

+ + + + 5 4

+ + + + + 5

**Example 3: Mixed Number Pattern**

**Definition:** A complex nested loop pattern where a=11.

* The outer loop (i) iterates from 0 to 4.
* The first inner loop (j) prints an increasing sequence of numbers (1, then 1 2, then 1 2 3, etc.).
* The second inner loop (k) prints a decreasing sequence of *even* numbers, with the starting point determined by a-(2\*i).

**Example Code:**

Python

a=11

for i in range(0,5,1):

for j in range(1,i+2,1):

print(j,end=" ")

for k in range(2,a-(2\*i)+1,2):

print(k,end=" ")

print()

**Output:**

Plaintext

1 2 4 6 8 10

1 2 2 4 6 8

1 2 3 2 4 6

1 2 3 4 2 4

1 2 3 4 5 2

**Using the chr() Function in Loops**

**Definition:** The chr() function returns the character (a string) represented by a specified Unicode integer. For example, chr(65) returns 'A'. This is commonly used to create letter-based patterns.

**Example 4: Simple chr() Loop**

**Definition:** A basic for loop that runs 4 times. In each iteration, it prints chr(65), which is 'A'. The end=" " keeps all the output on a single line.

**Example Code:**

Python

for i in range(1,5,1):

print(chr(65),end=" ")

**Output:**

Plaintext

A A A A

**Example 5: chr() with a Variable**

**Definition:** This loop runs 4 times, with i taking the values 1, 2, 3, and 4. It prints the character for 64 + i, which results in chr(65) ('A'), chr(66) ('B'), chr(67) ('C'), and chr(68) ('D').

**Example Code:**

Python

for i in range(1,5,1):

print(chr(64+i),end=" ")

**Output:**

Plaintext

A B C D

**Example 6: chr() in a Nested Loop (Square)**

**Definition:** A nested loop to print a 4x4 square of letters. The outer loop (i) controls the rows. The inner loop (j) runs 4 times for each row, printing 'A', 'B', 'C', 'D' each time.

**Example Code:**

Python

for i in range (1,5,1):

for j in range(1,5,1):

print(chr(64+j),end=" ")

print()

**Output:**

Plaintext

A B C D

A B C D

A B C D

A B C D

**Example 7: Mixed Pattern ('\*' and Letters)**

**Definition:** A nested loop pattern.

* The first inner loop (j) prints an increasing number of \* symbols.
* The second inner loop (k) prints a decreasing sequence of letters by using the expression chr(69-k) (e.g., when i=1, k goes 1, 2, 3, 4, printing chr(68)='D', chr(67)='C', etc.).

**Example Code:**

Python

for i in range (1,5,1):

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

print("\*",end=" ")

for k in range (i,5,1):

print(chr(69-k),end=" ")

print()

**Output:**

Plaintext

\* D C B A

\* \* C B A

\* \* \* B A

\* \* \* \* A

**Example 8: Letter Triangle**

**Definition:** This nested loop prints a right-angle triangle of letters. The inner loop (j) iterates from 1 up to i+1, so it prints 'A', then 'A B', then 'A B C', and so on.

**Example Code:**

Python

for i in range(1,5,1):

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

print(chr(64+j),end=" ")

print()

**Output:**

Plaintext

A

A B

A B C

A B C D

**Example 9: Diamond Letter Pattern**

**Definition:** This is a complex pattern that combines multiple loops to print a diamond shape of letters, padded with \* symbols.

1. The first loop prints a decreasing number of leading \*s.
2. The second loop prints letters in reverse (e.g., B, C B, D C B).
3. The print("A", ...) statement prints the center 'A' for each row.
4. The fourth loop prints letters in forward order (e.g., B, B C, B C D).
5. The final loop prints a decreasing number of trailing \*s.

**Note on File Discrepancy:** The code in the file for j in range(i, a): uses the variable a (which was set to 11 in a previous cell). This does *not* match the saved output. The output \* \* \* on the first line and \* \* on the second line strongly suggests the loop was for j in range(i, 4): when the cell was run, matching the first padding loop.

**Example Code (as written in file):**

Python

for i in range(1, 5):

for j in range(i, 4):

print("\*", end=" ")

for j in range(i, 1, -1):

print(chr(64 + j), end=" ")

print("A", end=" ")

for j in range(2, i + 1):

print(chr(64 + j), end=" ")

for j in range(i, a):

print("\*", end=" ")

print()

**Output (as saved in file):**

Plaintext

\* \* \* A \* \* \* \* \* B A B \* \* \* C B A B C \* D C B A B C D

**Example 10: Inverted '\*' Triangle (Partial)**

**Definition:** This loop prints an inverted right-angle triangle of \* symbols. The inner loop j runs from i up to 3.

* When i=1, range(1, 4) prints 3 stars.
* When i=2, range(2, 4) prints 2 stars.
* When i=3, range(3, 4) prints 1 star.
* When i=4, range(4, 4) is empty, so the print() creates a blank line.

**Example Code:**

Python

for i in range(1, 5):

for j in range(i, 4):

print("\*", end=" ")

print()

**Output:**

Plaintext

\* \* \* \* \* \* ```

---

## \*\*WHILE LOOP\*\*

\*\*Definition:\*\* A `while` loop repeatedly executes a block of code as long as a specified condition remains `True`.

### Example 1: Basic `while` Loop

\*\*Definition:\*\* This loop initializes `a` to 1. The loop continues to run \*while\* `a` is less than 5. Inside the loop, it prints the value of `a` and then increments `a` by 1 (`a += 1`). The loop stops when `a` becomes 5.

\*\*Example Code:\*\*

```python

a=1

while a<5 :

print(a,end=" ")

a+=1

**Output:**

Plaintext

1 2 3 4

**Example 2: Nested while and for Loops**

**Definition:** This shows a for loop nested inside a while loop.

* The outer while loop runs 4 times (for a = 1, 2, 3, 4).
* In each pass of the while loop, the inner for loop runs completely, printing the numbers 1, 2, 3, 4.
* The print() after the for loop moves to a new line before the while loop continues.

**Example Code:**

Python

a=1

while a<5 :

for i in range(1,5,1):

print(i,end=" ")

print()

a+=1

**Output:**

Plaintext

1 2 3 4

1 2 3 4

1 2 3 4

1 2 3 4

**Example 3: while Loop for Sorting Even/Odd**

**Definition:** This code initializes two empty lists, even and odd. The while loop runs as long as a is less than or equal to 100. Inside the loop:

* a % 2 == 0 checks if a is even.
* The **append()** method is used to add the number a to the end of the correct list.
* a is incremented, and the loop repeats.

**Example Code:**

Python

a=1

even=[]

odd=[]

while a<=100:

if a % 2 ==0:

even.append(a)

else:

odd.append(a)

a+=1

print(even)

print(odd)

**Output:**

Plaintext

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99]