23rd SEPTEMBER NOTES

#write a syntax of while loop inside for loop

for var in range():

initialization of while #outer loop

while(condition): #inner loop

statements of while loop

inc/dec of while loop

statements of for loop

Diamond number pattern code.

```
# Upper part of the diamond
for i in range(1, 6):
    # Print leading spaces
    for j in range(5 - i):
        print(" ", end="")
    # Print numbers
    for k in range(1, i + 1):
        print(i, end=" ")
    print()

# Lower part of the diamond
for i in range(4, 0, -1):
    # Print leading spaces
    for j in range(5 - i):
        print(" ", end="")
    # Print numbers
    for k in range(1, i + 1):
        print(i, end=" ")
    print()
```

```
    1
   2 2
  3 3 3
 4 4 4 4
5 5 5 5 5
 4 4 4 4
  3 3 3
   2 2
    1
```

Code Explanation

Upper Part of the Diamond

for i in range(1, 6):

- This loop runs from i = 1 to i = 5.
- Each i represents a row in the upper half.

for j in range(5 - i):

print(" ", end="")

- Prints spaces before the numbers, so that the numbers are centered.
- Example:
    - For i = 1, it prints 4 spaces.
    - For i = 2, it prints 3 spaces.
    - ... until i = 5, where it prints 0 spaces.

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

print(i, end=" ")

- Prints the number i, repeated i times, with a space after each number.
- Example:
    - For i = 1, prints 1.
    - For i = 2, prints 2 2.
    - For i = 5, prints 5 5 5 5 5.

print()

- Moves to the next line after each row.

Lower Part of the Diamond

for i in range(4, 0, -1):

- Runs i from 4 down to 1 (reverse order).
- This creates the bottom half of the diamond.

for j in range(5 - i):

print(" ", end="")

- Same as above: spaces for alignment.

```
for k in range(1, i + 1):
```

```
  print(i, end=" ")
```

- Prints the number i repeated i times.

```
print()
```

- Moves to the next line.

#write a syntax of for loop inside while loop

```
initialisation of while loop
while(condition):  #outerloop
  for var in range():
    statements of for loop
  statements of while loop
  inc/dec of while loop
```

half pyramid pattern

```
# Function to print a half pyramid pattern
i=1

def half_pyramid(n):
    num=2
    for i in range(1, n + 1):
        for j in range(1, i + 1):
            print(num,end=" ")
            num=num+2
        print("")

# Example: Print a half pyramid with 5 rows
n = 5
half_pyramid(n)
```

```
2
4 6
8 10 12
14 16 18 20
22 24 26 28 30
```

Code Explanation

i = 1 # not really used, since you redefine i in the loop

- This line isn't needed, because inside the function you already use for i in range(...).

---

def half_pyramid(n):

num = 2

- Defines a function half_pyramid with parameter n (the number of rows).
- num is initialized to 2, so the pattern starts with 2.

---

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

- Outer loop runs from 1 to n.
- Each value of i represents the current row number.

```
for j in range(1, i + 1):
    print(num, end=" ")
    num = num + 2
```

- Inner loop runs i times.
- Prints the current num value followed by a space (end=" " keeps printing on the same line).
- After printing, num increases by 2.
- This ensures the next printed number is the next even number.

```
print("")
```

- After finishing one row, move to the next line.

```
# Example: Print a half pyramid with 5 rows
n = 5
half_pyramid(n)
```

- Calls the function with n = 5, so the pyramid has 5 rows.

In short:

- Each row adds one more number.
- Numbers start at 2 and increase by 2 each time.
- The result is a half-pyramid made of **even numbers**.

half pyramid of even numbers using a while

```
i=1
num=2

while(i<=5):
    for j in range(1,i+1,1):
        print(num,end=" ")
        num=num+2
    print()
    i=i+1
```

```
2
4 6
8 10 12
14 16 18 20
22 24 26 28 30
```

Code Explanation

i = 1

num = 2

- i keeps track of the current row (starting from row 1).
- num starts at 2, so the first number printed will be 2.

---

while(i <= 5):

- Outer loop runs as long as i is less than or equal to 5.
- This means it will create **5 rows**.

---

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

print(num, end=" ")

num = num + 2

- Inner loop prints i numbers on the current row.
- end=" " keeps printing on the same line with spaces.

- After printing a number, num increases by 2 (so only even numbers appear).

---

print()

- After finishing one row, moves to the next line.

---

i = i + 1

- Moves to the next row by incrementing i.

Prime half pyramid

```
# Check if a number is prime
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, num):   # check divisibility
        if num % i == 0:
            return False
    return True


n = 5      # number of rows
num = 2    # start with the first prime
i = 1      # row counter

while i <= n:       # loop for rows
    count = 0       # primes printed in this row
    while count < i:    # print i primes
        if is_prime(num):
            print(num, end=" ")
            count += 1  # one prime printed
        num += 1        # check next number
    print()             # new line after each row
    i += 1              # go to next row
```

```
2
3 5
7 11 13
17 19 23 29
31 37 41 43 47
```

Code Explanation

1. Prime Checker

```
def is_prime(num):
  if num < 2:
    return False
  for i in range(2, num):   # check divisibility
    if num % i == 0:
      return False
  return True
```

- A function that checks if a number is **prime**.
- Numbers < 2 are not prime.
- Loop through all numbers from 2 to num-1 and check divisibility.
- If divisible → not prime.
- Otherwise, return True.

---

2. Initialization

```
n = 5     # number of rows in the pyramid
num = 2   # start with the first prime (2)
i = 1     # row counter
```

- We want a pyramid with 5 rows.
- First prime number is 2.
- i will keep track of which row we're printing.

---

3. Outer Loop → Rows

```
while i <= n:
```

- Runs until all rows (n = 5) are printed.
- Each iteration represents **one row**.

---

4. Inner Loop → Numbers in Each Row

```
count = 0      # reset counter for each row
while count < i:    # print i primes in row i
  if is_prime(num):
    print(num, end=" ")
    count += 1
  num += 1
```

- At the start of each row, count = 0.
- Keep checking numbers until we print exactly i primes.
- If a number is prime → print it and increase count.
- Always move num to the next number.

---

5. Line Break and Row Increment

```
print()    # go to next line after finishing a row
i += 1     # move to next row
```

📌 **Dry Run (Row by Row)**

- **Row 1 (i=1)**
  Prints 2
- **Row 2 (i=2)**
  Prints 3 5
- **Row 3 (i=3)**
  Prints 7 11 13
- **Row 4 (i=4)**
  Prints 17 19 23 29
- **Row 5 (i=5)**
  Prints 31 37 41 43 47

Final Output

2

3 5

7 11 13

17 19 23 29

31 37 41 43 47