Questions and Answers

1. What will be the output of the following code?

```
i = 1
while i <= 3:
    print(i)
    i += 1
```

Answer:

The output will be:

```
1
2
3
```

Explanation: The `while` loop prints the value of `i` from 1 to 3. The loop continues as long as the condition `i <= 3` is true, and `i` is incremented by 1 after each iteration.

2. Write a `for` loop to print each character of the string `"hello"`. What will be the output?

```
for char in "hello":
    print(char)
```

Answer:

The output will be:

```
h
e
l
l
o
```

Explanation: The `for` loop iterates over each character in the string `"hello"` and prints them one by one.

3. How can you use the `range()` function to print numbers from 5 to 9?

Answer:

```python
for i in range(5, 10):
    print(i)
```

Explanation: The `range(5, 10)` generates numbers from 5 up to (but not including) 10. The loop prints each number in this range.

4. What will the following code do?

```python
for i in range(4):
    if i == 2:
        continue
    print(i)
```

Answer:

The output will be:

```
0
1
3
```

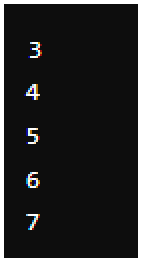
Explanation: The `continue` statement skips the current iteration of the loop when `i` is 2, so 2 is not printed.

5. Write a `while` loop that prints numbers from 3 to 7. What will be the output?

```python
i = 3
while i <= 7:
    print(i)
    i += 1
```

Answer:

The output will be:



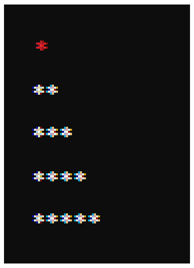
```
3
4
5
6
7
```

Explanation: The `while` loop prints the value of `i` from 3 to 7. The loop continues as long as `i <= 7`, and `i` is incremented by 1 after each iteration.

Different Types of Triangle Patterns in Python

Here are some examples of different triangle patterns and their Python code implementations using loops.

1. Right-Angled Triangle

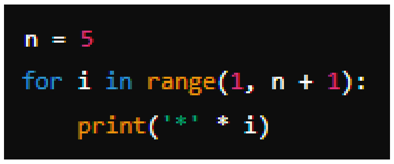


```
*
**
***
****
*****
```

Explanation:

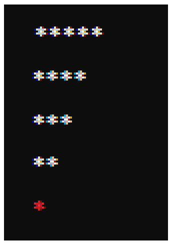
- Use a loop to iterate from 1 to `n` (number of lines).
- Print `i` number of `*` on each line.

Code:



```python
n = 5
for i in range(1, n + 1):
    print('*' * i)
```

2. Inverted Right-Angled Triangle

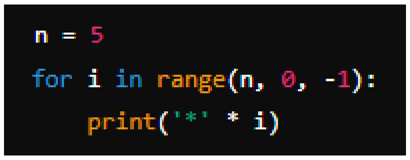


```
*****
****
***
**
*
```

Explanation:

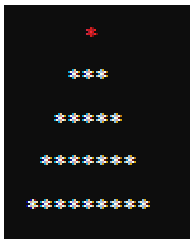
- Use a loop to iterate from `n` down to 1.
- Print `i` number of `*` on each line.

Code:



```python
n = 5
for i in range(n, 0, -1):
    print('*' * i)
```

3. Equilateral Triangle

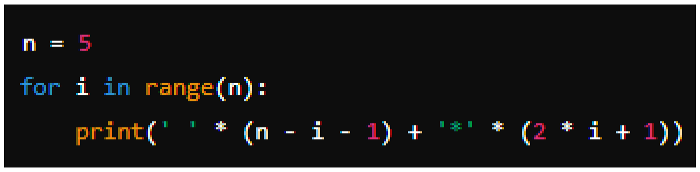


```
    *
   ***
  *****
 *******
*********
```

Explanation:

- Use a loop to iterate from 0 to `n-1`.
- Print spaces for padding on the left, then print `*` to form the triangle.
- The number of `*` increases by 2 for each line (1, 3, 5, etc.).

Code:



```python
n = 5
for i in range(n):
    print(' ' * (n - i - 1) + '*' * (2 * i + 1))
```

4. Inverted Equilateral Triangle

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

Explanation:

- Use a loop to iterate from `n` down to 1.
- Print spaces for padding on the left, then print `*` to form the inverted triangle.
- The number of `*` decreases by 2 for each line (9, 7, 5, etc.).

Code:

```python
n = 5
for i in range(n, 0, -1):
    print(' ' * (n - i) + '*' * (2 * i - 1))
```

5. Right-Angled Triangle with Numbers

```
1
22
333
4444
55555
```

Explanation:

- Use a loop to iterate from 1 to `n`.
- Print `i` number of `i` on each line.

Code:

```python
n = 5
for i in range(1, n + 1):
    print(str(i) * i)
```

6. Floyd's Triangle

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Explanation:

- Use a nested loop.
- Outer loop controls the number of lines.
- Inner loop prints numbers starting from 1 and increasing sequentially.

Code:

```python
n = 5
num = 1
for i in range(1, n + 1):
    for j in range(1, i + 1):
        print(num, end=' ')
        num += 1
    print()
```

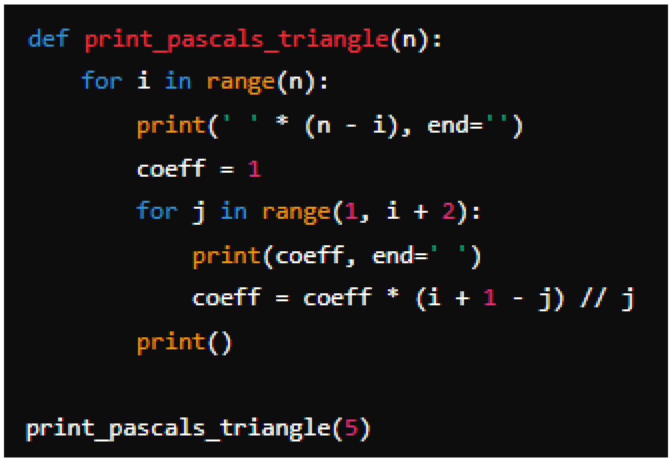
7. Pascal's Triangle

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

Explanation:

- Use a nested loop.
- Outer loop for each line.
- Inner loop to calculate and print coefficients using binomial theorem.

Code:



```python
def print_pascals_triangle(n):
    for i in range(n):
        print(' ' * (n - i), end='')
        coeff = 1
        for j in range(1, i + 2):
            print(coeff, end=' ')
            coeff = coeff * (i + 1 - j) // j
        print()

print_pascals_triangle(5)
```