PYTHON · COMS AND COMA · NESTED LOOPS

# Pattern Printing with Loops

*From one line of stars to full pyramids, one loop at a time.*



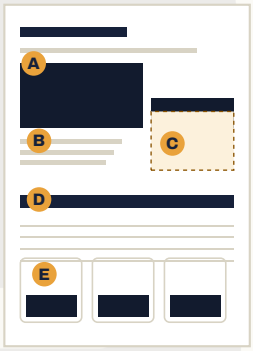
```
print(" " * (n - i) + "* " * i)
```

| LEVEL | LANGUAGE | LENGTH | PRACTICE |
|---|---|---|---|
| Beginner | Python 3 | 8 pages | 17 questions |

## HOW TO USE THIS GUIDE

1. **Read the idea**
   Start with the Key idea box. It is the one sentence to remember.
2. **Run the solution**
   Type the code, run it, and compare with the sample run.
3. **Follow the trace**
   The table shows which line runs next and how i and j change.
4. **Try it yourself**
   Solve the questions at the foot of the page, then check page 8.

## ANATOMY OF A GUIDE PAGE




- A Solution and sample run
- B How it works, line by line
- C Figure of the idea
- D Trace table: the next line
- E Try it yourself

## CONTENTS


| NO. | SECTION | WHAT YOU LEARN | PAGE |
|---|---|---|---|
| 01 | **The Foundation** | Rows and columns, print(), end, for and range(), the 3-step method | 2 |
| 02 | **One-Dimensional Patterns** | A single row: one loop, one print | 3 |
| 03 | **Rectangle and Square** | Nested loops: rows outside, columns inside | 4 |
| 04 | **Triangle Patterns** | Make the inner loop depend on the row number. Practical list Pattern-1, 2, 3 | 5 |
| 05 | **Pyramid Patterns** | Spaces first, then stars, for a centred shape | 6 |
| 06 | **Hollow and Special Patterns** | if-else inside the loops; border patterns | 7 |
| 07 | **Answer Key and Cheat Sheet** | Solutions to all 17 questions, and a recipe table | 8 |


**BEFORE YOU START**

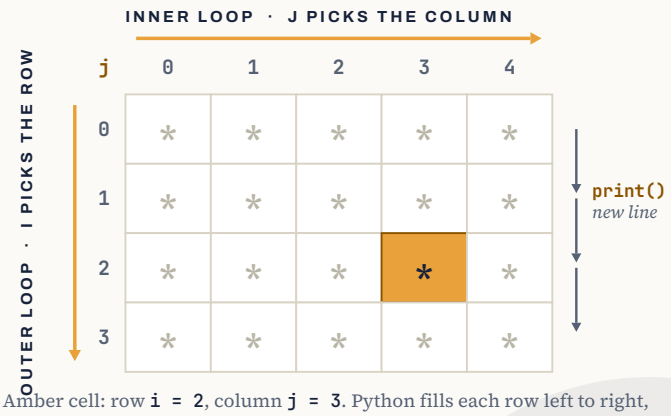
You need Python 3 and a place to run it, such as IDLE, VS Code or any online Python editor. Every program in this guide was run, and every output you see is the real output.

SECTION 01 · THE FOUNDATION

# Every Pattern Is a Grid

A pattern is rows of characters. Python prints one character at a time and one row at a time, so you need only three tools: `print()`, a `for` loop and `range()`.

FIGURE 1 · THE ROWS-AND-COLUMNS MODEL




Amber cell: row `i = 2`, column `j = 3`. Python fills each row left to right, then moves down.

TOOL 1 · PRINT() AND ITS END

| CODE | WHAT IT DOES |
|---|---|
| `print("*")` | Prints * then moves to the next line. |
| `print("*", end=" ")` | Prints * and a space, then stays on the same line. |
| `print()` | Prints nothing. Only moves to the next line. |

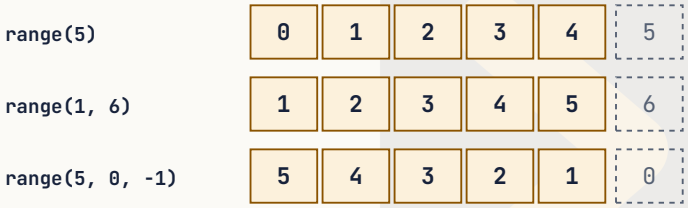
TOOL 2 · FOR AND RANGE()

A `for` loop repeats its **indented** lines once for every value that `range()` hands out. The loop variable (`i` or `j`) holds the current value, so it tells you which row or column you are on.

**INDENTATION**

Python uses spaces at the start of a line to know what is inside the loop. Keep the same 4 spaces for every line in one block.

FIGURE 2 · RANGE() HANDS OUT NUMBERS




`range(stop)` starts at 0.

`range(start, stop)` starts at start.

`range(start, stop, step)` moves by step; a negative step counts down.

The dashed box is the **stop** value. It is **never** used.

THE 3-STEP METHOD FOR ANY PATTERN

**1 Count the rows**

How many lines does the pattern have? That is the outer loop: `for i in range(...)`.

**2 Count the items in a row**

How many characters does row i have? That is the inner loop. Spaces and stars may need one loop each.

**3 End every row**

Put `print()` after the inner loop, level with the inner for, to start a new line.

**COMMON MISTAKE**

Writing `print()` inside the inner loop gives one character per line; leaving it out gives one very long line. Also remember that `range(5)` stops at 4, not 5.

TRY IT YOURSELF

ANSWERS ARE ON PAGE 8.

TRY IT · Q1

**A column of stars**

Print 4 stars, one under the other.

```
*
*
*
*
```

TRY IT · Q2

**Hello on one line**

Print Hello! three times on one line.

```
Hello! Hello! Hello!
```

SECTION 02 · ONE-DIMENSIONAL PATTERNS

# A Single Row

A 1D pattern has one row, so it needs only one loop and no `print()` at the end. Print the numbers 1 to 5 on one line.

**KEY IDEA**

One loop is one row. The loop variable `i` is the value you print.

### SOLUTION · PYTHON

```
1 n = 5
2 for i in range(1, n + 1):
3     print(i, end=" ")
```

### HOW IT WORKS · LINE BY LINE

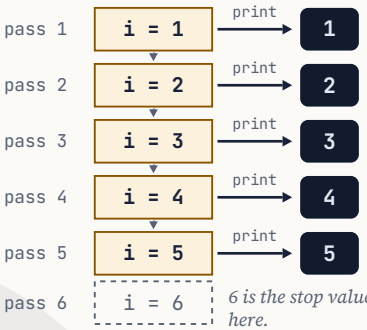
**L1** `n = 5` stores how many numbers we want.

**L2** `for i in range(1, n + 1):` hands `i` the values 1, 2, 3, 4, 5, one for each pass. The stop value 6 is never used.

**L3** `print(i, end=" ")` shows `i` and a space. `end` keeps the cursor on the same line.

Line 3 is indented, so it is the loop body and runs five times. There is no `print()` because there is only one row.

### FIGURE 3 · EACH PASS OF THE LOOP




### SAMPLE RUN · REAL OUTPUT

```
1 2 3 4 5
```

**SHORTCUT**

`print("* " * 5)` prints a row of five stars with no loop at all. Loops are still needed when each item changes.

### TRACE TABLE · WHICH LINE RUNS NEXT

| STEP | LINE | LOOP VALUES | WHAT HAPPENS | NEXT | OUTPUT SO FAR |
|---|---|---|---|---|---|
| 1 | 1 | - | Store 5 in n. | 2 | |
| 2 | 2 | i=1 | range gives the next value: i = 1. | 3 | |
| 3 | 3 | i=1 | Print 1 and a space. The cursor stays on the line. | 2 | 1· |
| 4 | 2 | i=2 | range gives the next value: i = 2. | 3 | 1· |
| 5 | 3 | i=2 | Print 2 and a space. The cursor stays on the line. | 2 | 1·2· |
| 6 | 2 | i=3 | range gives the next value: i = 3. | 3 | 1·2· |
| 7 | 3 | i=3 | Print 3 and a space. The cursor stays on the line. | 2 | 1·2·3· |
| 8 | 2 | i=4 | range gives the next value: i = 4. | 3 | 1·2·3· |
| 9 | 3 | i=4 | Print 4 and a space. The cursor stays on the line. | 2 | 1·2·3·4· |
| 10 | 2 | i=5 | range gives the next value: i = 5. | 3 | 1·2·3·4· |
| 11 | 3 | i=5 | Print 5 and a space. The cursor stays on the line. | 2 | 1·2·3·4·5· |
| 12 | 2 | - | No values left in range: the loop ends. | end | 1·2·3·4·5· |

Output column: · is a space. All 12 steps are shown. A bold amber NEXT number means the program jumps back to repeat the loop.

### TRY IT YOURSELF

ANSWERS ARE ON PAGE 8.

**TRY IT · Q3**

**A row of stars**

Print 5 stars on one line, with a space between them.

```
* * * * *
```

**TRY IT · Q4**

**Countdown**

Print the numbers 5 down to 1 on one line.

```
5 4 3 2 1
```

**TRY IT · Q5**

**Alphabet row**

Print A to E on one line. Hint: chr(65) gives A.

```
A B C D E
```

SECTION 03 · TWO-DIMENSIONAL PATTERNS

# Rectangle and Square

A 2D pattern has rows and columns, so it needs two loops, one inside the other. Print a rectangle of stars with 3 rows and 4 columns.

**KEY IDEA**

Outer loop = rows. Inner loop = columns. The inner loop finishes all its columns before the outer loop moves to the next row.

## SOLUTION · PYTHON

```
1 rows = 3
2 cols = 4
3 for i in range(rows):
4     for j in range(cols):
5         print("*", end=" ")
6     print()
```

## HOW IT WORKS · LINE BY LINE

**L1-2** `rows = 3` and `cols = 4` store the size of the rectangle.

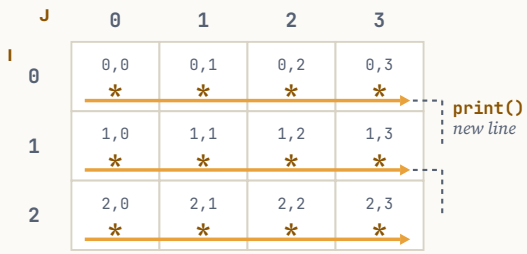
**L3** `for i in range(rows):` is the **outer** loop. `i` is 0, 1, 2: one pass for each row.

**L4** `for j in range(cols):` is the **inner** loop. `j` is 0 to 3: one pass for each column, restarted for every row.

**L5** `print("*", end=" ")` prints one star and a space on the same line.

**L6** `print()` is indented under the outer loop only. It runs when the inner loop is finished and starts a new line.

## FIGURE 4 · THE ORDER OF VISITS




Amber arrows: the inner loop runs along one row. Dashed line: `print()` then starts the next row.

## SAMPLE RUN · REAL OUTPUT

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

**ANALOGY**

A clock: the inner loop is the minute hand and the outer loop is the hour hand. The minute hand goes all the way round before the hour hand moves once.

## TRACE TABLE · WHICH LINE RUNS NEXT

| STEP | LINE | LOOP VALUES | WHAT HAPPENS | NEXT | OUTPUT SO FAR |
|---|---|---|---|---|---|
| 3 | 3 | i=0 | Outer loop starts row `i = 0`. | 4 | |
| 4 | 4 | i=0, j=0 | Inner loop starts column `j = 0`. | 5 | |
| 5 | 5 | i=0, j=0 | Print * and a space, on the same line. | 4 | *· |
| 6 | 4 | i=0, j=1 | Inner loop starts column `j = 1`. | 5 | *· |
| 7 | 5 | i=0, j=1 | Print * and a space, on the same line. | 4 | *·*· |
| 8 | 4 | i=0, j=2 | Inner loop starts column `j = 2`. | 5 | *·*· |
| 9 | 5 | i=0, j=2 | Print * and a space, on the same line. | 4 | *·*·*· |
| 10 | 4 | i=0, j=3 | Inner loop starts column `j = 3`. | 5 | *·*·*· |
| 11 | 5 | i=0, j=3 | Print * and a space, on the same line. | 4 | *·*·*·*· |
| 12 | 4 | i=0 | All columns done: leave the inner loop. | 6 | *·*·*·*· |
| 13 | 6 | i=0 | `print()` only starts a new line: the row is done. | 3 | *·*·*·*·/ |
| 14 | 3 | i=1 | Outer loop starts row `i = 1`. | 4 | *·*·*·*·/ |
| 15 | 4 | i=1, j=0 | Inner loop starts column `j = 0`. | 5 | *·*·*·*·/ |

Steps 1 and 2 only store rows and cols, so the table starts at step 3. It shows steps 3 to 15 of 36; rows i = 1 and i = 2 repeat the same steps. Output column: · is a space, / is a new line.

## TRY IT YOURSELF

ANSWERS ARE ON PAGE 8.

TRY IT · Q6

### Square of hashes

Print a 4 by 4 square of # symbols.

```
# # # #
# # # #
# # # #
# # # #
```

TRY IT · Q7

### Number grid

Print 3 rows, each showing 1 2 3 4.

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

TRY IT · Q8

### Row numbers

Fill each row with its own row number.

```
1 1 1 1
2 2 2 2
3 3 3 3
```

SECTION 04 · TRIANGLE PATTERNS

# The Right-Angled Triangle

In a triangle the number of items changes from row to row. Print Pattern-1 from the Class XII practical list: 1 star in row 1, 2 stars in row 2, up to 5 stars in row 5.

**KEY IDEA**

Make the inner loop depend on the outer loop. Row `i` has `i` stars, so the inner loop is `range(i)`.

### SOLUTION · PYTHON

```
1 n = 5
2 for i in range(1, n + 1):
3     for j in range(i):
4         print("*", end="")
5     print()
```

### HOW IT WORKS · LINE BY LINE

**L1** `n = 5` is the number of rows.

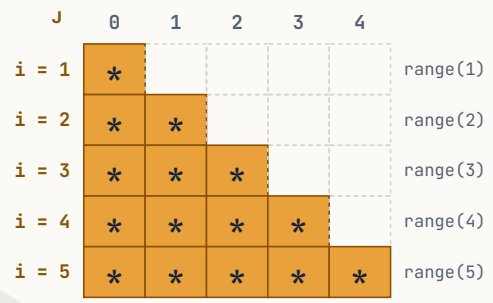
**L2** `for i in range(1, n + 1):` is the outer loop. `i` is the row number, 1 to 5.

**L3** `for j in range(i):` runs `i` times, so row `i` gets `i` stars. This is the only change from the rectangle: `range(cols)` became `range(i)`.

**L4** `print("*", end="")` prints a star with nothing after it, so the stars touch, as in the practical list.

**L5** `print()` starts a new line after each row.

### FIGURE 5 · ROW I FILLS I CELLS




Row `i` fills the first `i` cells: `j` runs from 0 to `i - 1`. Dashed cells are never printed.

### SAMPLE RUN · REAL OUTPUT

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

**SPOT THE DIFFERENCE**

Rectangle: `range(cols)` gives the same count in every row. Triangle: `range(i)` gives a count that grows with the row.

### TRACE TABLE · WHICH LINE RUNS NEXT

| STEP | LINE | LOOP VALUES | WHAT HAPPENS | NEXT | OUTPUT SO FAR |
|---|---|---|---|---|---|
| 2 | 2 | i=1 | Outer loop: row `i = 1`. | 3 | |
| 3 | 3 | i=1, j=0 | Inner loop: `j = 0` (`range(1)` gives 1 value). | 4 | |
| 4 | 4 | i=1, j=0 | Print one * with nothing after it. | 3 | * |
| 5 | 3 | i=1 | `range(1)` is used up: leave the inner loop. | 5 | * |
| 6 | 5 | i=1 | New line: the row is finished. | 2 | */ |
| 7 | 2 | i=2 | Outer loop: row `i = 2`. | 3 | */ |
| 8 | 3 | i=2, j=0 | Inner loop: `j = 0` (`range(2)` gives 2 values). | 4 | */ |
| 9 | 4 | i=2, j=0 | Print one * with nothing after it. | 3 | */* |
| 10 | 3 | i=2, j=1 | Inner loop: `j = 1` (`range(2)` gives 2 values). | 4 | */* |
| 11 | 4 | i=2, j=1 | Print one * with nothing after it. | 3 | */** |
| 12 | 3 | i=2 | `range(2)` is used up: leave the inner loop. | 5 | */** |

The table skips step 1 (storing n) and shows steps 2 to 12 of 47: row i = 1 in full and most of row i = 2. Output column: · is a space, / is a new line.

### TRY IT YOURSELF

ANSWERS ARE ON PAGE 8.

TRY IT · Q9

**Inverted triangle**

Print 5 stars in row 1, then 4, 3, 2, 1.

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

TRY IT · Q10

**Pattern-2 (practical list)**

Print this number triangle.

```
12345
1234
123
12
1
```

TRY IT · Q11

**Pattern-3 (practical list)**

Print this alphabet triangle.

```
A
AB
ABC
ABCD
ABCDE
```

SECTION 05 · PYRAMID PATTERNS

# The Pyramid

A pyramid is a triangle pushed to the centre. Each row has two parts: spaces first, then stars. Print a pyramid with 4 rows.

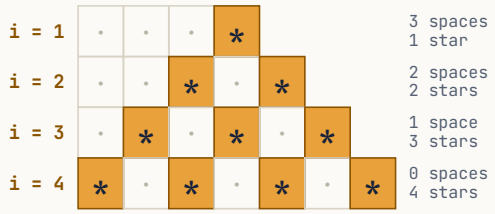
**KEY IDEA**

Row `i` needs `n - i` spaces, then `i` stars, each followed by a space. The spaces shrink as the stars grow.

### SOLUTION · PYTHON

```
1 n = 4
2 for i in range(1, n + 1):
3     for s in range(n - i):
4         print(" ", end="")
5     for k in range(i):
6         print("* ", end="")
7     print()
```

### FIGURE 6 · SPACES, THEN STARS




Each row is `n - i` spaces (dotted cells), then `i` stars with one space between them.

### HOW IT WORKS · LINE BY LINE

- **L1** `n = 4` is the number of rows.
- **L2** `for i in range(1, n + 1):` `i` is the row number, 1 to 4.
- **L3** `for s in range(n - i):` is the spaces loop. It runs `n - i` times: 3, 2, 1, 0.
- **L4** `print(" ", end="")` prints one space and stays on the line.
- **L5** `for k in range(i):` is the stars loop. It runs `i` times.
- **L6** `print("* ", end="")` prints a star and a space. The gaps make the sloping sides.
- **L7** `print()` starts a new line.

### SAMPLE RUN · REAL OUTPUT

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

**SHORTCUT**

`print(" " * (n - i) + "* " * i)` gives the same pyramid in one line, using string repetition.

### TRACE TABLE · WHICH LINE RUNS NEXT

| STEP | LINE | LOOP VALUES | WHAT HAPPENS | NEXT | OUTPUT SO FAR |
|---|---|---|---|---|---|
| 2 | 2 | i=1 | Outer loop: row `i = 1`. | 3 | |
| 3 | 3 | i=1, s=0 | Spaces loop: `s = 0` (runs 3 times). | 4 | |
| 4 | 4 | i=1, s=0 | Print one space. | 3 | · |
| 5 | 3 | i=1, s=1 | Spaces loop: `s = 1` (runs 3 times). | 4 | · |
| 6 | 4 | i=1, s=1 | Print one space. | 3 | ·· |
| 7 | 3 | i=1, s=2 | Spaces loop: `s = 2` (runs 3 times). | 4 | ·· |
| 8 | 4 | i=1, s=2 | Print one space. | 3 | ··· |
| 9 | 3 | i=1 | 3 spaces printed: on to the stars. | 5 | ··· |
| 10 | 5 | i=1, k=0 | Stars loop: `k = 0` (runs 1 time). | 6 | ··· |
| 11 | 6 | i=1, k=0 | Print * and a space. | 5 | ···*· |
| 12 | 5 | i=1 | 1 star printed: finish the row. | 7 | ···*· |
| 13 | 7 | i=1 | New line: the row is finished. | 2 | ···*·/ |
| 14 | 2 | i=2 | Outer loop: row `i = 2`. | 3 | ···*·/ |

The table shows steps 2 to 14 of 50: row i = 1 in full (3 spaces, 1 star) and the start of row 2. Output column: · is a space, / is a new line.

### TRY IT YOURSELF

ANSWERS ARE ON PAGE 8.

TRY IT · Q12

**Inverted pyramid**

Print an upside-down pyramid of 4 rows.

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

TRY IT · Q13

**Number pyramid**

Print a pyramid of numbers with 4 rows.

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

TRY IT · Q14

**Diamond**

Print a diamond: a 3-row pyramid, then 2 rows back down.

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

SECTION 06 · HOLLOW AND SPECIAL PATTERNS

# The Hollow Square

A hollow pattern prints stars on the border and blanks inside. Add an `if-else` inside the inner loop to decide, cell by cell, what to print. Print a hollow square of side 4.

**KEY IDEA**

A cell is on the border when it is in the first or last row, or in the first or last column.

## SOLUTION · PYTHON

```
1 n = 4
2 last = n - 1
3 for i in range(n):
4     for j in range(n):
5         if i == 0 or i == last or j == 0 or j == last:
6             print("*", end=" ")
7         else:
8             print(" ", end=" ")
9     print()
```

## HOW IT WORKS · LINE BY LINE

**L1-2** `n = 4` is the side. `last = n - 1` is 3: the last row and column, because counting starts at 0.

**L3-4** Outer loop `i` is the row, inner loop `j` is the column: 4 by 4 cells.

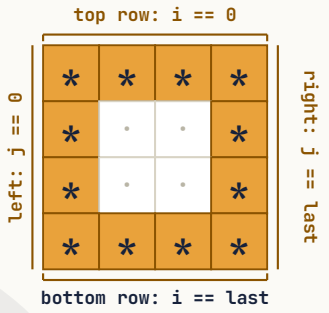
**L5** `if i == 0 or i == last or j == 0 or j == last:` is True for the first or last row, or the first or last column: the border.

**L6** Border cell: print a star and a space.

**L7-8** `else:` is an inside cell. Print a blank and a space so the columns stay lined up.

**L9** `print()` starts a new line after each row.

## FIGURE 7 · WHICH CELLS ARE THE BORDER




Amber = border cell: print a star. White = inside: print a blank. With n = 4, last is 3.

## SAMPLE RUN · REAL OUTPUT

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

**READING THE TEST**

`or` needs only one part to be True. A cell is inside only when all four parts are False.

## TRACE TABLE · WHICH LINE RUNS NEXT

| STEP | LINE | LOOP VALUES | WHAT HAPPENS | NEXT | OUTPUT SO FAR |
|---|---|---|---|---|---|
| 18 | 3 | i=1 | Outer loop: row `i = 1`. | 4 | `*·*·*·*·/` |
| 19 | 4 | i=1, j=0 | Inner loop: column `j = 0`. | 5 | `*·*·*·*·/` |
| 20 | 5 | i=1, j=0 | Border cell: `j == 0` is True, so run line 6. | 6 | `*·*·*·*·/` |
| 21 | 6 | i=1, j=0 | Print * and a space. | 4 | `*·*·*·*·/*·` |
| 22 | 4 | i=1, j=1 | Inner loop: column `j = 1`. | 5 | `*·*·*·*·/*·` |
| 23 | 5 | i=1, j=1 | Inside cell: all four tests are False, so jump to line 8. | 8 | `*·*·*·*·/*·` |
| 24 | 8 | i=1, j=1 | Print a blank and a space, to keep columns lined up. | 4 | `*·*·*·*·/*···` |
| 25 | 4 | i=1, j=2 | Inner loop: column `j = 2`. | 5 | `*·*·*·*·/*···` |
| 26 | 5 | i=1, j=2 | Inside cell: all four tests are False, so jump to line 8. | 8 | `*·*·*·*·/*···` |
| 27 | 8 | i=1, j=2 | Print a blank and a space, to keep columns lined up. | 4 | `*·*·*·*·/*·····` |
| 28 | 4 | i=1, j=3 | Inner loop: column `j = 3`. | 5 | `*·*·*·*·/*·····` |

The table follows row i = 1 only (steps 18 to 28 of 63): the first row that has a gap, up to its third column. Rows 0 and 3 print stars in every cell. Line 7 (`else`) never runs as a step of its own; the test on line 5 jumps straight to line 8. · is a space, / a new line.

## TRY IT YOURSELF

ANSWERS ARE ON PAGE 8.

TRY IT · Q15

### Hollow rectangle

Print a hollow rectangle, 4 rows by 6 columns.

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

TRY IT · Q16

### Floyd's triangle

Count 1, 2, 3, ... across the rows of a triangle.

```
1
2 3
4 5 6
7 8 9 10
```

TRY IT · Q17

### Hollow triangle

Print a hollow triangle of 5 rows.

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

SECTION 07 · ANSWER KEY AND CHEAT SHEET

# Check Your Work

Compare your code with the answers below. Different code that prints the same output is also correct.

### PATTERN RECIPES

| PATTERN | OUTER LOOP | INNER LOOP | REMEMBER |
|---|---|---|---|
| **Row (1D)** | none | `for i in range(1, n + 1): print(i, end=" ")` | One loop. No print() at the end. |
| **Rectangle** | `for i in range(rows)` | `for j in range(cols)` | Same count in every row. |
| **Triangle** | `for i in range(1, n + 1)` | `for j in range(i)` | Row i has i items. |
| **Inverted triangle** | `for i in range(n, 0, -1)` | `for j in range(i)` | Count the outer loop down. |
| **Pyramid** | `for i in range(1, n + 1)` | `range(n - i) spaces, range(i) stars` | Two inner loops: spaces, then stars. |
| **Hollow** | `for i in range(n)` | `for j in range(n) + if / else` | Test for the border cells. |

### ANSWER KEY · QUESTIONS 1 TO 17

**Q1 A column of stars** page 2

```
for i in range(4):
    print("*")
```

**Q2 Hello on one line** page 2

```
for i in range(3):
    print("Hello!", end=" ")
```

**Q3 A row of stars** page 3

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

**Q4 Countdown** page 3

```
for i in range(5, 0, -1):
    print(i, end=" ")
```

**Q5 Alphabet row** page 3

```
for i in range(5):
    print(chr(65 + i), end=" ")
```

**Q6 Square of hashes** page 4

```
for i in range(4):
    for j in range(4):
        print("#", end=" ")
    print()
```

**Q7 Number grid** page 4

```
for i in range(3):
    for j in range(1, 5):
        print(j, end=" ")
    print()
```

**Q8 Row numbers** page 4

```
for i in range(1, 4):
    for j in range(4):
        print(i, end=" ")
    print()
```

**Q9 Inverted triangle** page 5

```
for i in range(5, 0, -1):
    for j in range(i):
        print("*", end="")
    print()
```

**Q10 Pattern-2 (practical list)** page 5

```
for i in range(5, 0, -1):
    for j in range(1, i + 1):
        print(j, end="")
    print()
```

**Q11 Pattern-3 (practical list)** page 5

```
for i in range(1, 6):
    for j in range(i):
        print(chr(65 + j), end="")
    print()
```

**Q12 Inverted pyramid** page 6

```
n = 4
for i in range(n, 0, -1):
    print(" " * (n - i) + "* " * i)
```

**Q13 Number pyramid** page 6

```
n = 4
for i in range(1, n + 1):
    print(" " * (n - i), end="")
    for j in range(1, i + 1):
        print(j, end=" ")
    print()
```

**Q14 Diamond** page 6

```
n = 3
for i in range(1, n + 1):
    print(" " * (n - i) + "* " * i)
for i in range(n - 1, 0, -1):
    print(" " * (n - i) + "* " * i)
```

**Q15 Hollow rectangle** page 7

```
for i in range(4):
    for j in range(6):
        if i == 0 or i == 3 or j == 0 or j == 5:
            print("*", end=" ")
        else:
            print(" ", end=" ")
    print()
```

**Q16 Floyd's triangle** page 7

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

**Q17 Hollow triangle** page 7

```
n = 5
for i in range(1, n + 1):
    for j in range(i):
        if j == 0 or j == i - 1 or i == n:
            print("*", end="")
        else:
            print(" ", end="")
    print()
```