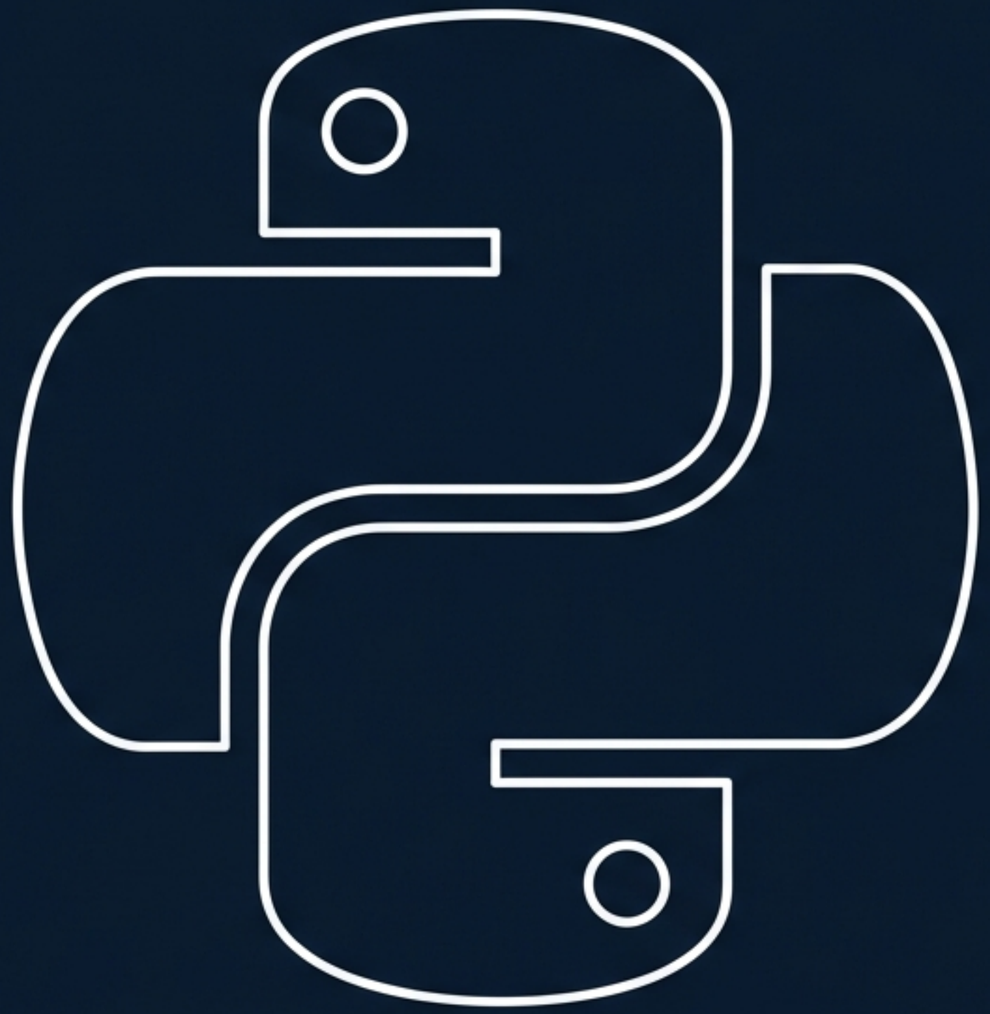


Structuring Data & Logic

Lists, Loops, and Functions

Python Fundamentals Course

The Limitation of Single Variables

The Problem

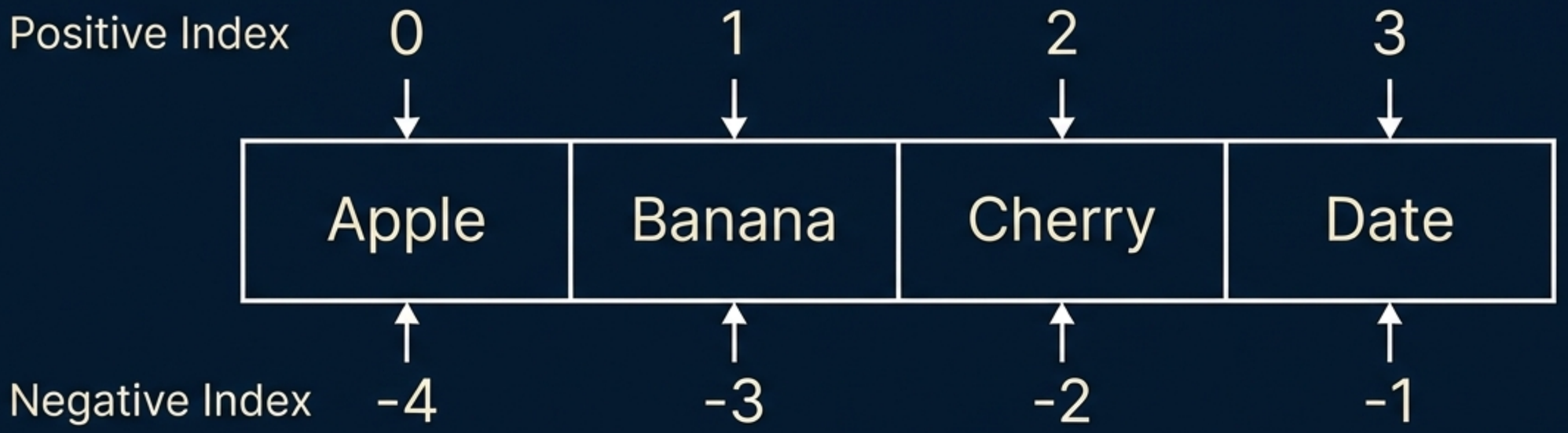
```
s1 = 85
s2 = 92
s3 = 78
s4 = 95
...
```

The Solution

```
scores = [85, 92, 78, 95]
```

- Variables fail at scale.
- The List: An ordered collection under one name.

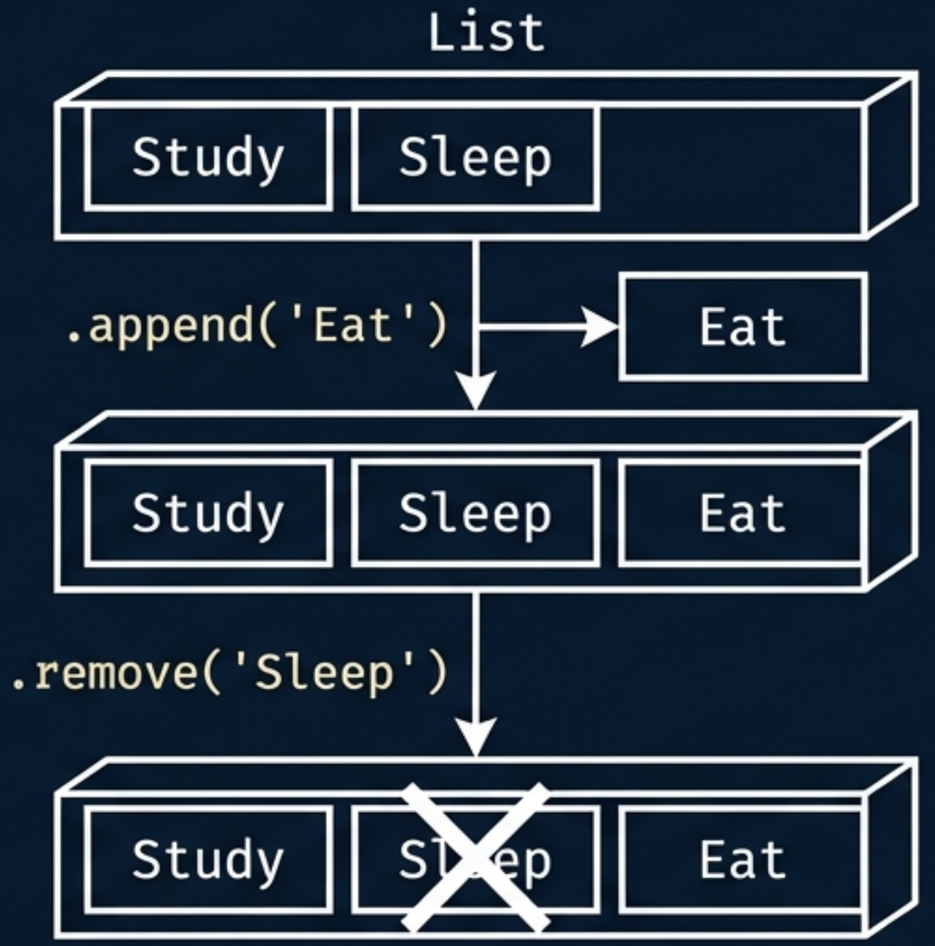
Precise Access: Indexing & Slicing




```
fruits = ['Apple', 'Banana', 'Cherry', 'Date']
print(fruits[0])     # Output: Apple
print(fruits[1:3])   # Output: ['Banana', 'Cherry']
```

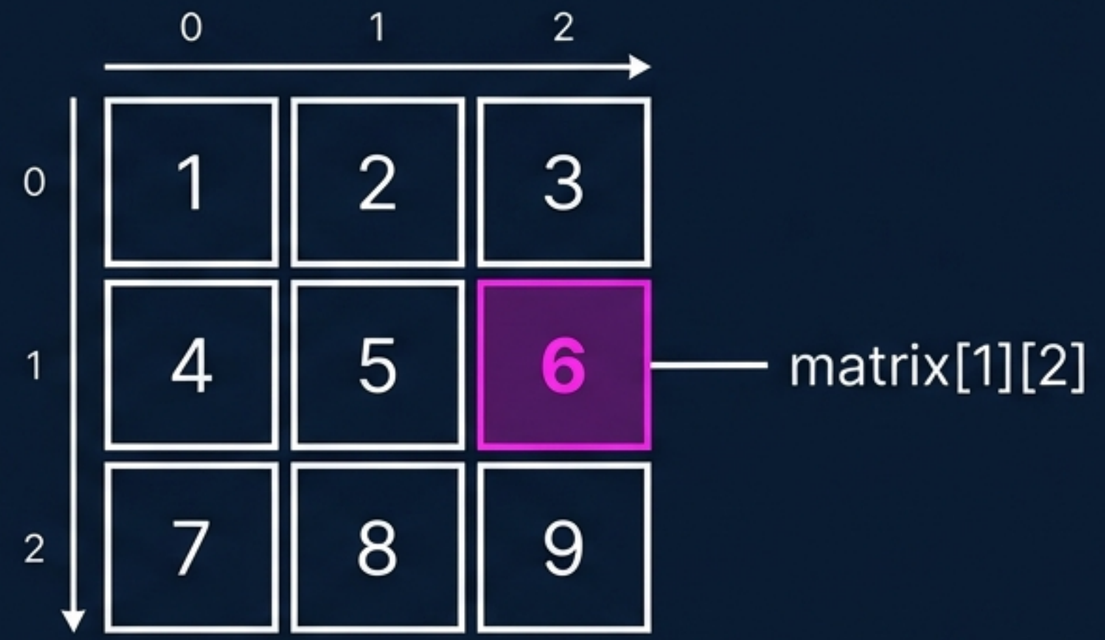
Living Data: List Methods

```
todos = ['Study', 'Sleep']
todos.append('Eat')
todos.remove('Sleep')
```

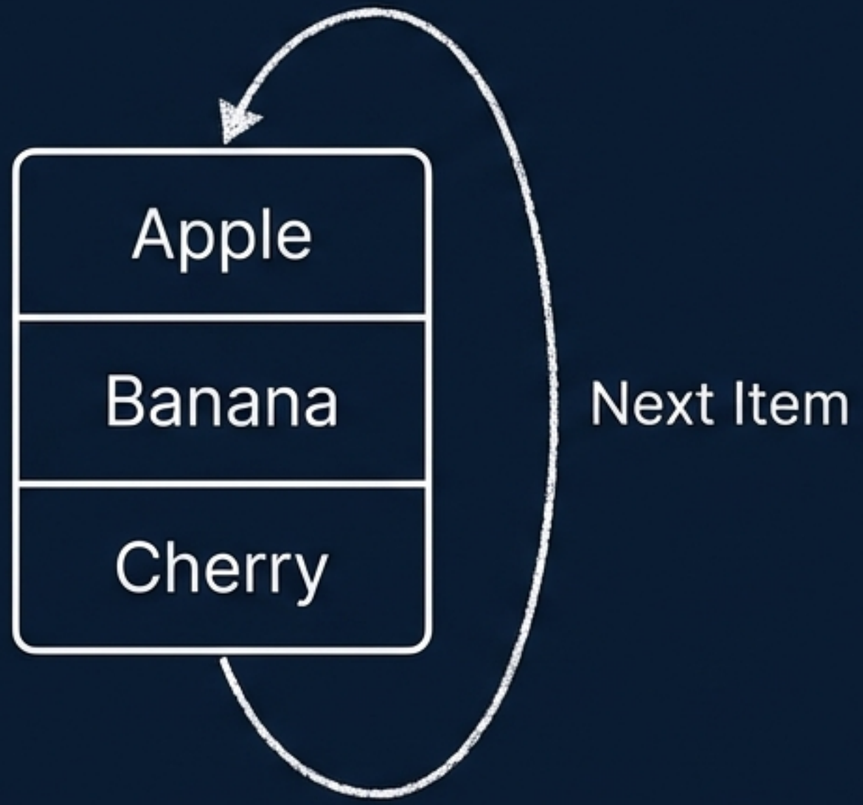


The Next Dimension: 2D Lists

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
print(matrix[1][2]) # Output: 6
```



Automating Tasks: The For Loop




```
fruits = ['Apple', 'Banana', 'Cherry']

for fruit in fruits:
    print('I like ' + fruit)
```

Variable that holds the current item

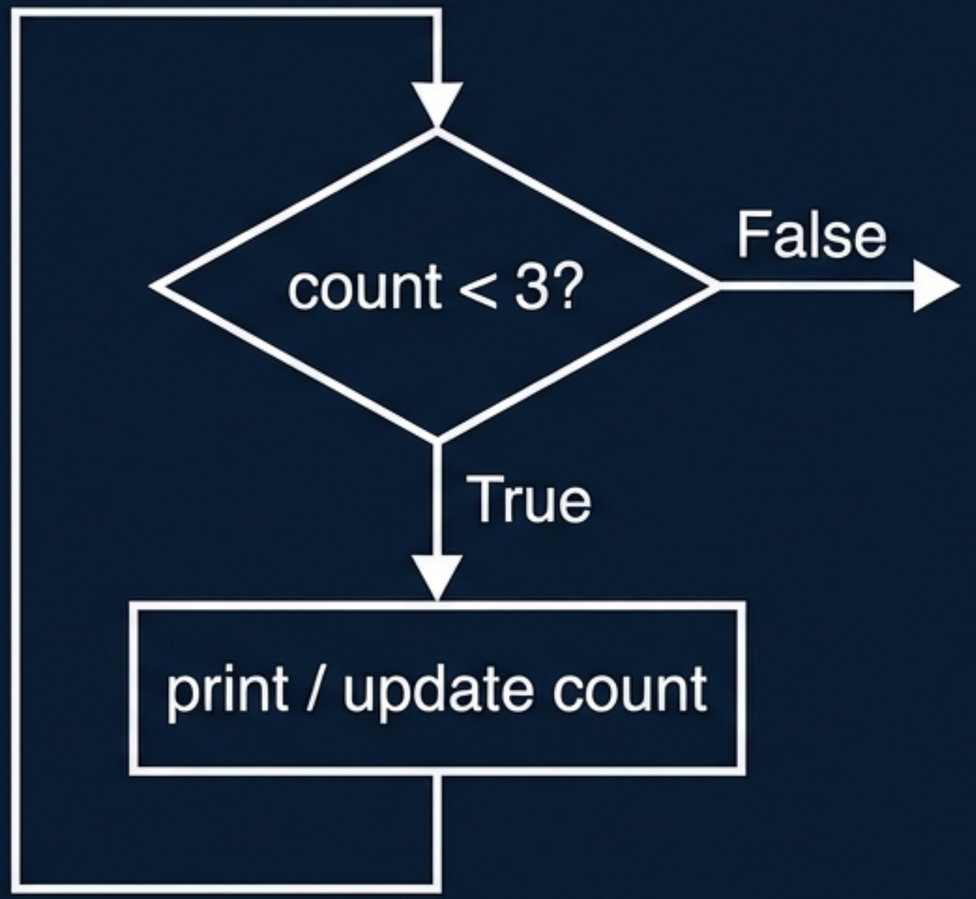
Nested Loops & Grid Traversal


```
for row in matrix:
    for num in row:
        print(num)
```

Inner loop runs fully for every single step of the outer loop.

Looping with Uncertainty: The While Loop

```
count = 0
while count < 3:
    print('Hello')
    count = count + 1
```



The Infinite Loop


```
while True:
    print('This runs forever...')
    # No update to condition!
```

Modularizing Logic: Functions

Functions are logical groupings of statements to achieve a task.

Before Functions

```
data = [10, 20, 30]
total = 0
for x in data:
    total += x
average = total / len(data)
print(average)
```

```
data = [10, 20, 30]
total = 0
for x in data:
    total += x
average = total / len(data)
print(average)
```

```
data = [10, 20, 30]
total = 0
for x in data:
    total += x
average = total / len(data)
print(average)
```

Define Once,
Call Many Times

After Functions

```
def calculate_average(data):
    total = 0
    for x in data:
        total += x
    return total / len(data)
```

calculate_average([10, 20, 30])

calculate_average([10, 20, 30])

calculate_average([10, 20, 30])

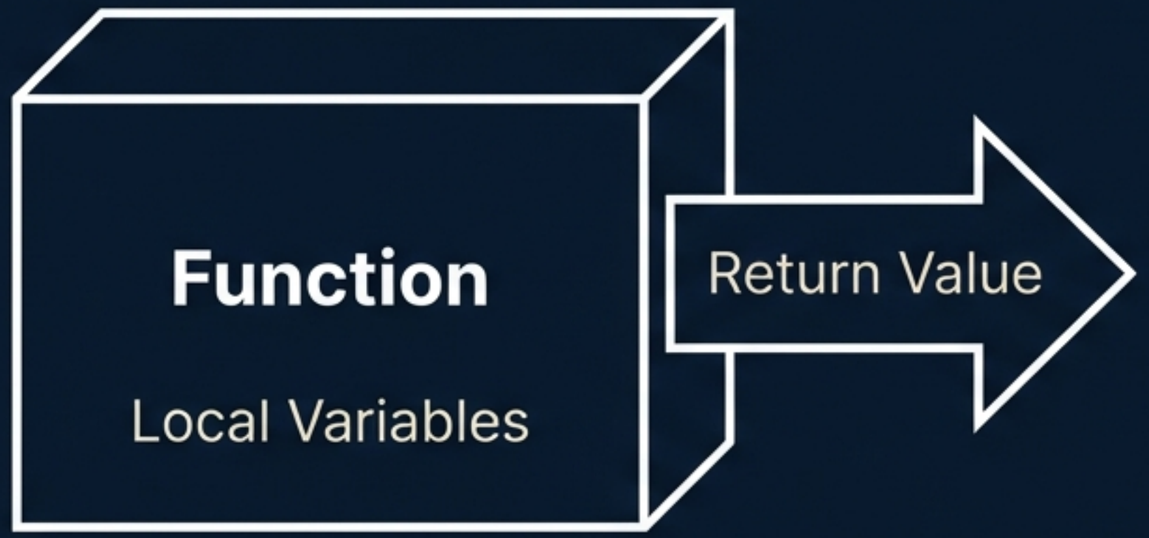
Anatomy of a Function

Parameters: Passing Data In


```
def square(number):
    print(number * number)

square(4)   # 16
square(10)  # 100
```

Return Values: Getting Data Out



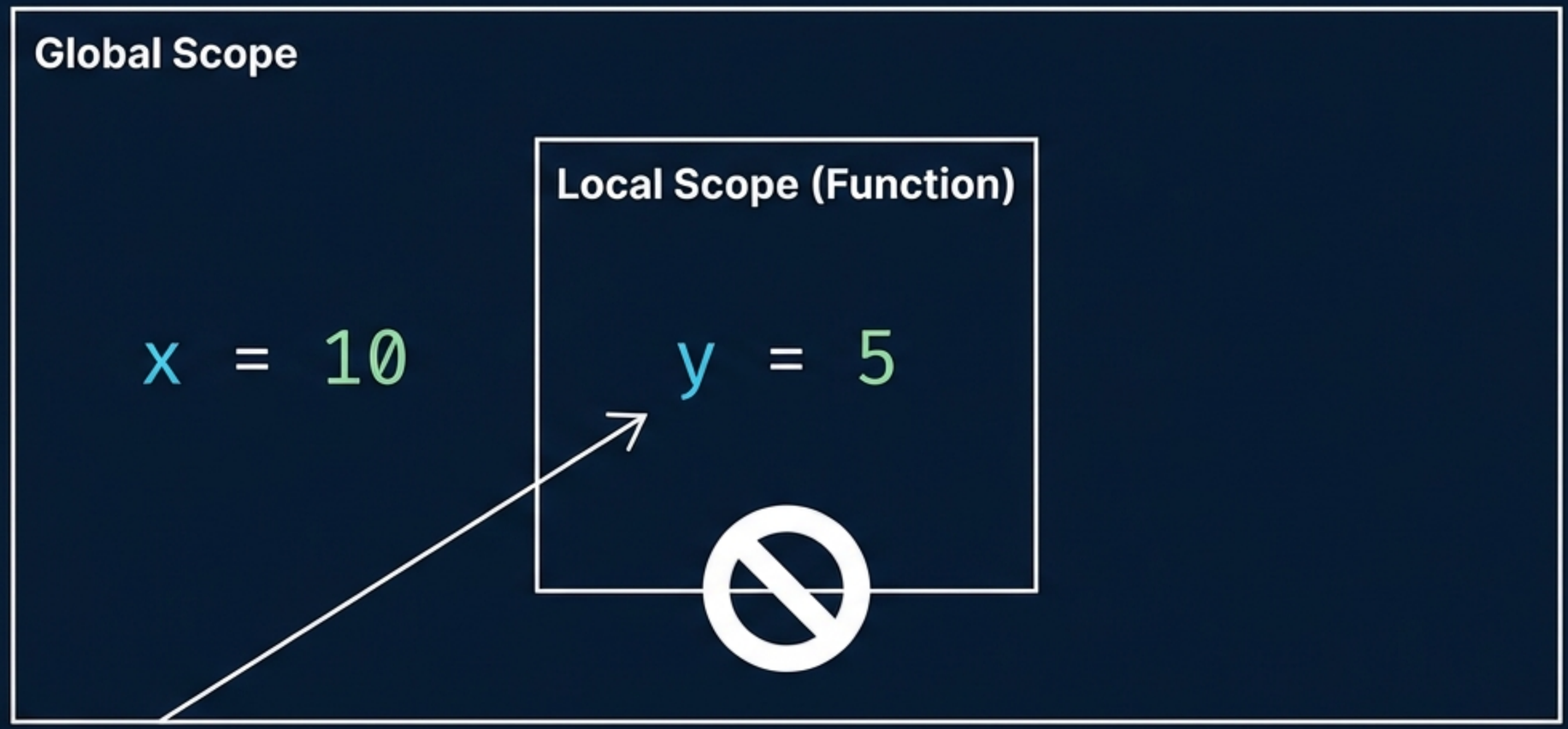
Stops execution and sends data back.

```
def add(a, b):
    return a + b

result = add(5, 3)
```

8

Variable Scope: Local vs. Global



What happens in the function stays in the function.

The Python Toolkit

- ☑ Lists: Ordered collections
- ☑ 2D Lists: Grids & Matrices
- ☑ Loops: Automation (For/While)
- ☑ Functions: Reusable Logic

Strategy: Decomposition. Break big problems into small functions.