Practice with While Loops

Today starts as a Paper + Pencil or Tablet + Pencil day... please keep laptops stowed away!

COMP110 - CL09

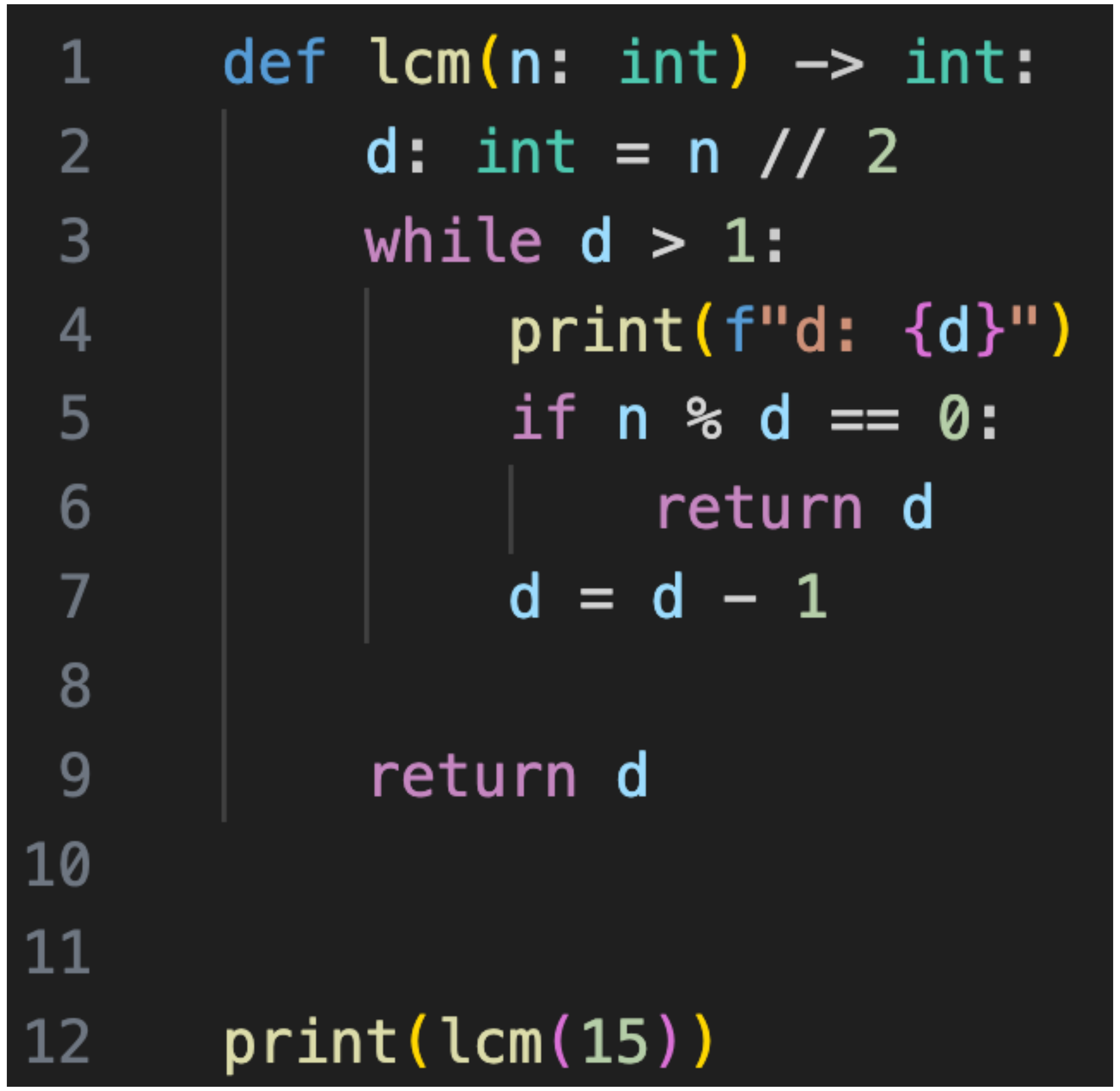
2024/02/22

Announcements

- EX03 - Wordle
  - Uses concepts through LS07
  - Implement the popular online game Wordle!

Warm-up

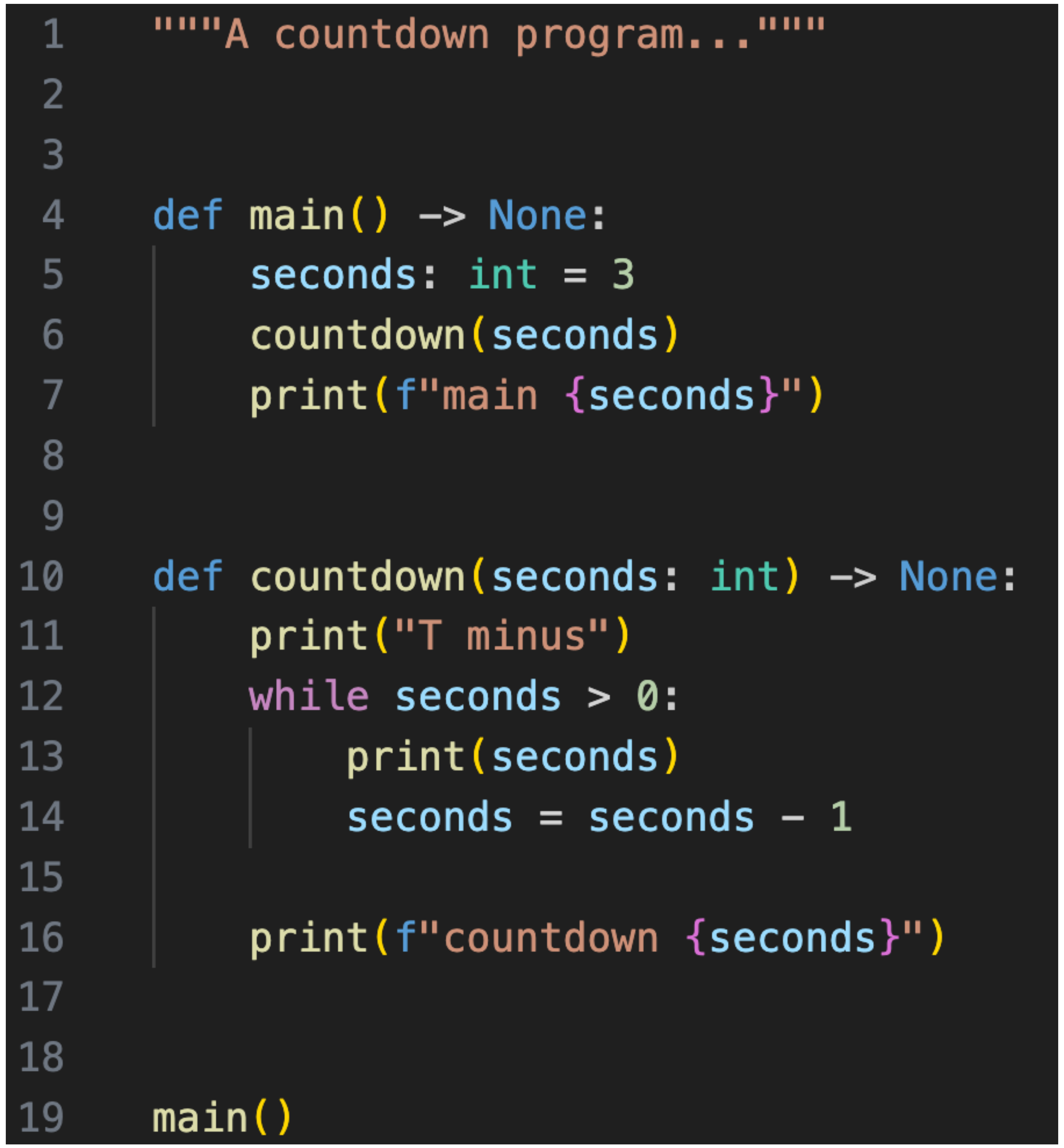
Trace a Memory Diagram



```python
def lcm(n: int) -> int:
    d: int = n // 2
    while d > 1:
        print(f"d: {d}")
        if n % d == 0:
            return d
        d = d - 1

    return d


print(lcm(15))
```

Warm-up

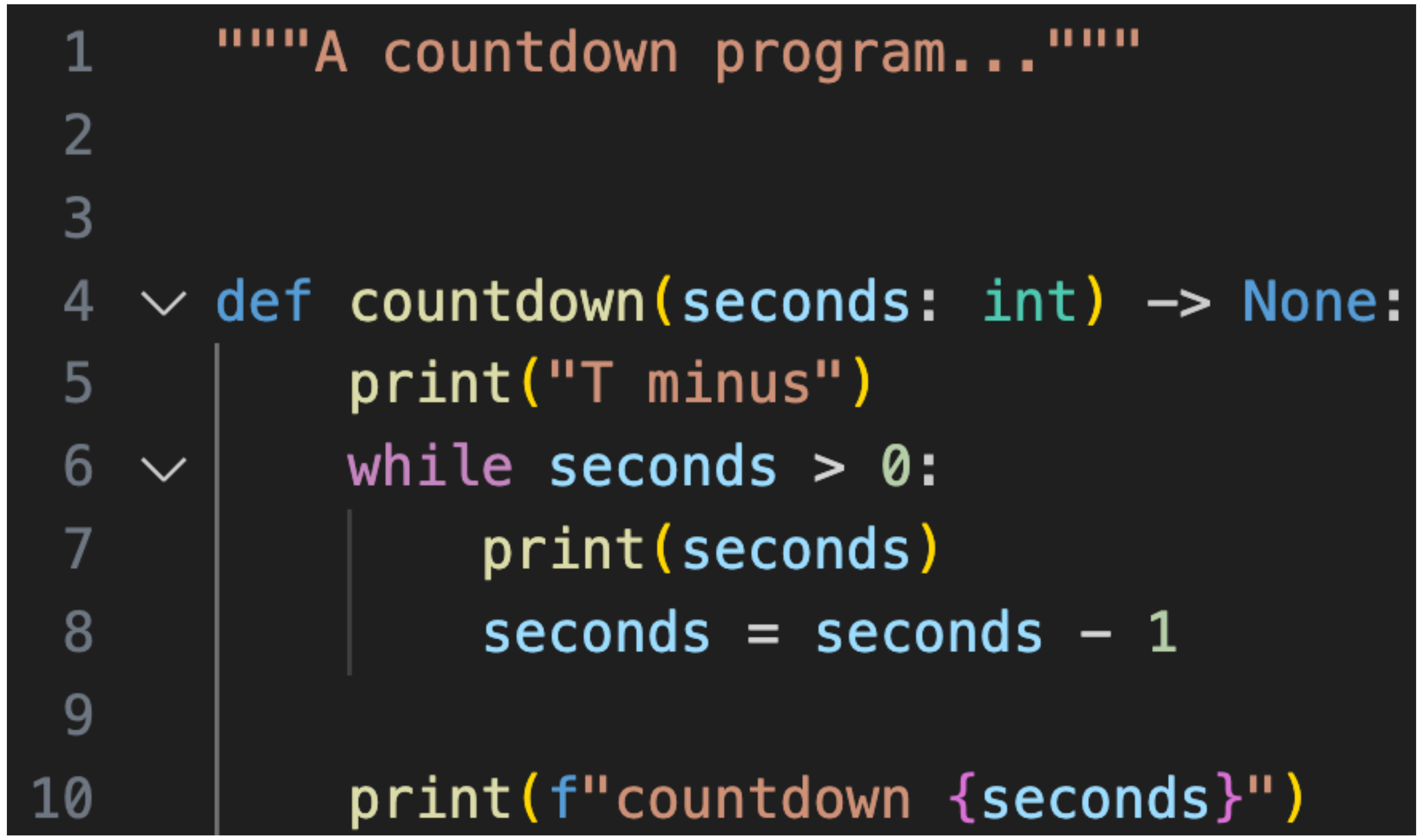
Trace a Memory Diagram



```python
"""A countdown program..."""


def main() -> None:
    seconds: int = 3
    countdown(seconds)
    print(f"main {seconds}")


def countdown(seconds: int) -> None:
    print("T minus")
    while seconds > 0:
        print(seconds)
        seconds = seconds - 1

    print(f"countdown {seconds}")


main()
```

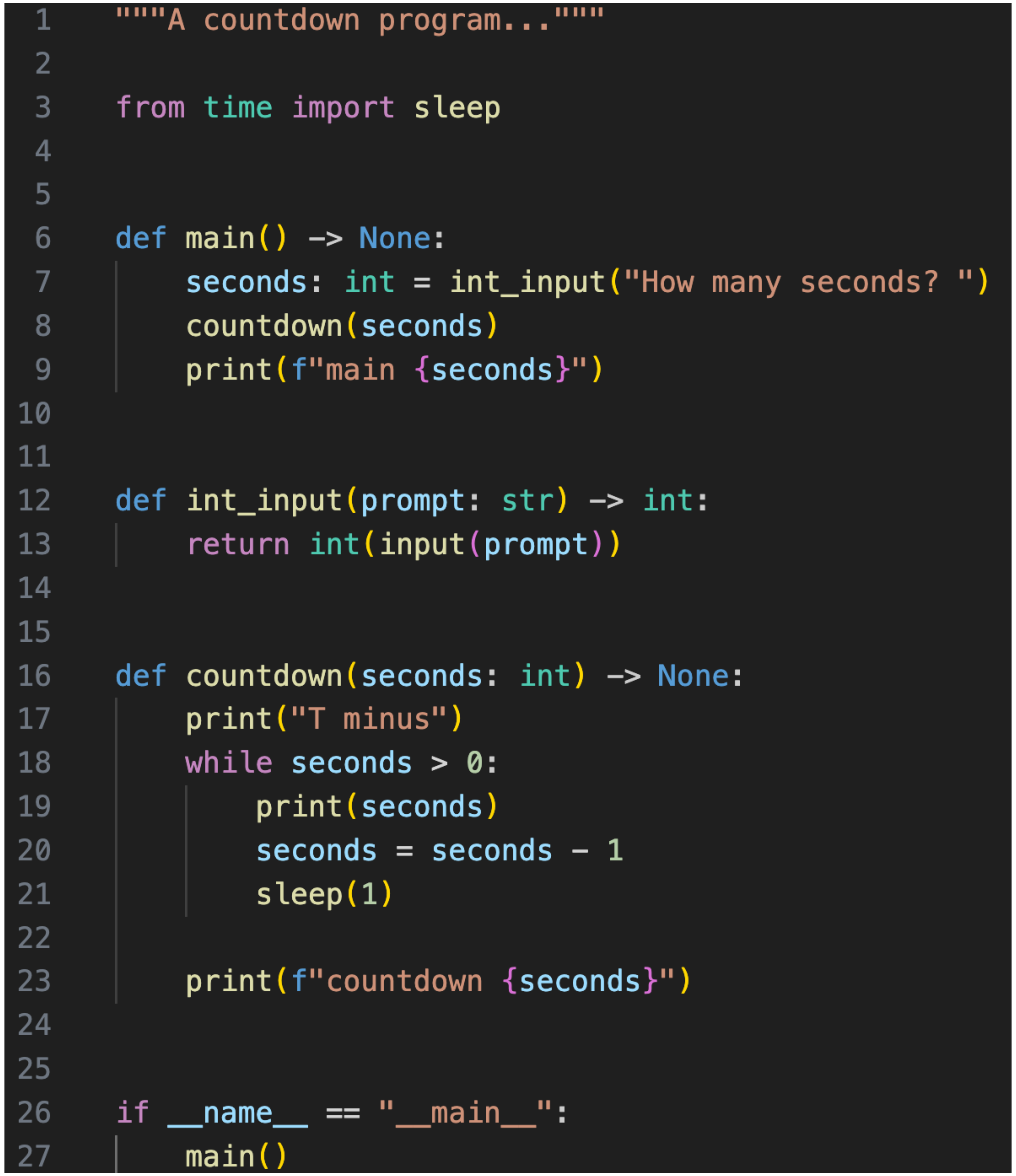
Code-Along

- Create a Directory named `lecture`, and in it a Python Module `cl08_countdown.py`
- This will be our starting point:



```python
"""A countdown program..."""


def countdown(seconds: int) -> None:
    print("T minus")
    while seconds > 0:
        print(seconds)
        seconds = seconds - 1

    print(f"countdown {seconds}")
```

Goal

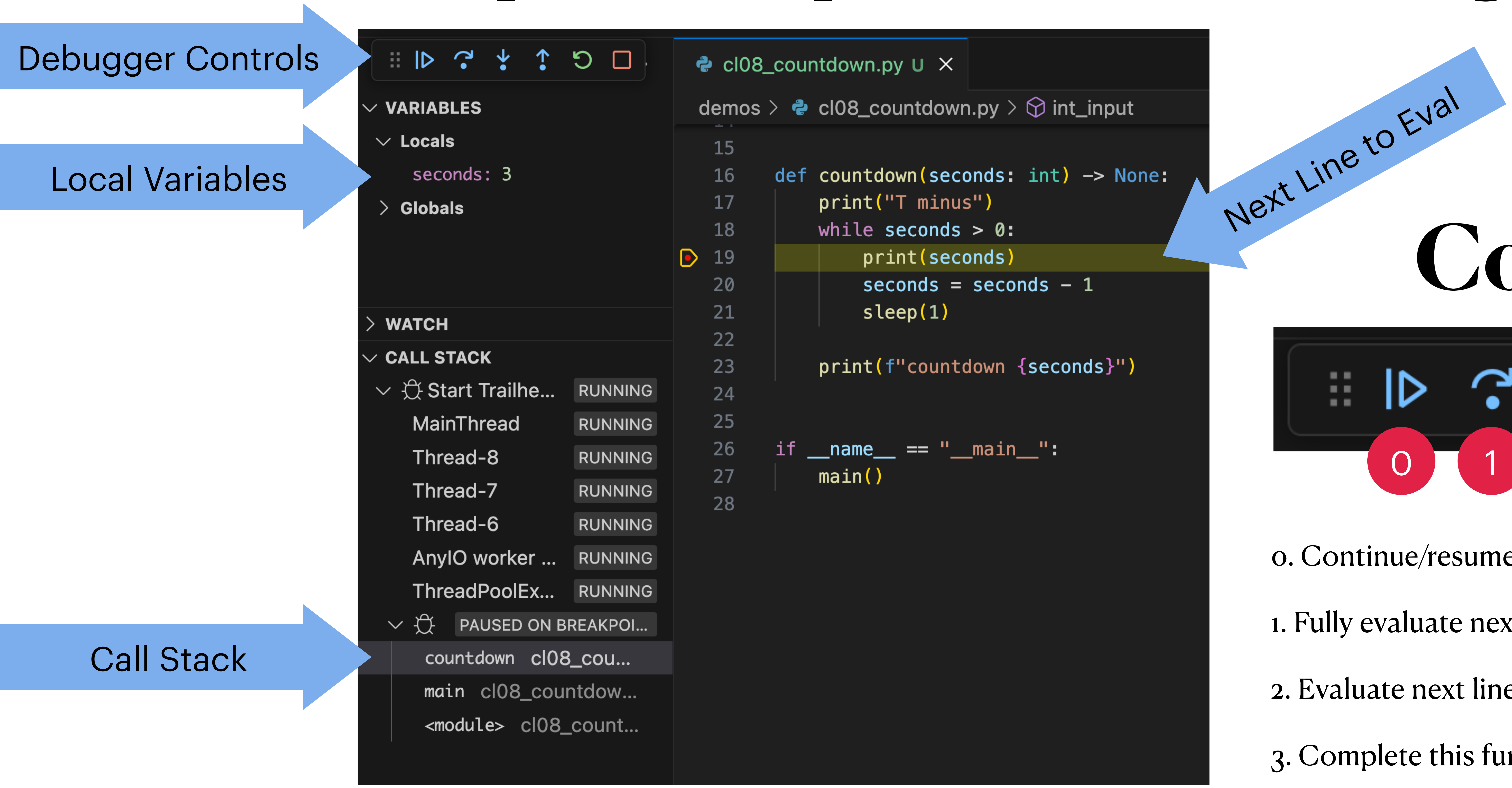

```python
"""A countdown program..."""

from time import sleep


def main() -> None:
    seconds: int = int_input("How many seconds? ")
    countdown(seconds)
    print(f"main {seconds}")


def int_input(prompt: str) -> int:
    return int(input(prompt))


def countdown(seconds: int) -> None:
    print("T minus")
    while seconds > 0:
        print(seconds)
        seconds = seconds - 1
        sleep(1)

    print(f"countdown {seconds}")


if __name__ == "__main__":
    main()
```

Introducing: Interactive Debugging

Pause your program at any point, inspect its state, and control execution!

- When Trailhead (or programs more generally) are run via run/debug in Code you have a programming superpower available: Interactive Debugging
- "Drop a **Breakpoint**" - A breakpoint marks a line of code the debugger will *pause at* when the Python interpreter reaches its evaluation of your program.
    - Right click on line number and "Add Breakpoint" or click red circle in line gutter
    - Run or use REPL to cause this code to evaluate
    - Your program pauses at this point!
- Let's explore the debugger on the next slide

Important parts of the debugger...



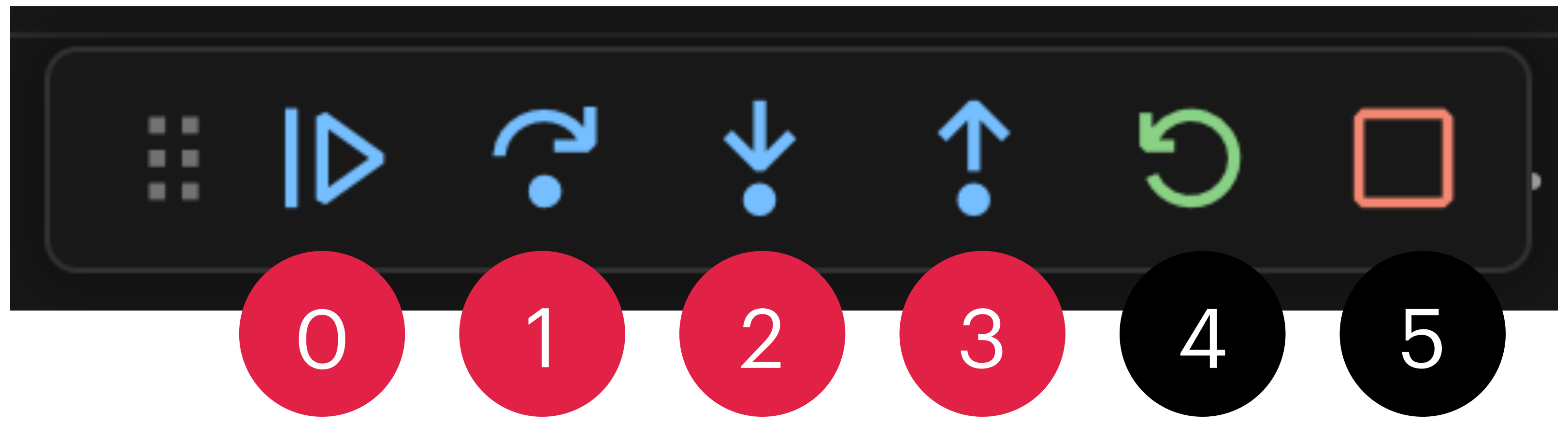
cl08_countdown.py U ×

demos > cl08_countdown.py > int_input

```
VARIABLES
  Locals
    seconds: 3
  Globals

WATCH
CALL STACK
  Start Trailhe...     RUNNING
    MainThread         RUNNING
    Thread-8           RUNNING
    Thread-7           RUNNING
    Thread-6           RUNNING
    AnyIO worker ...   RUNNING
    ThreadPoolEx...    RUNNING
  PAUSED ON BREAKPOI...
    countdown   cl08_cou...
    main   cl08_countdow...
    <module>   cl08_count...
```

```python

def countdown(seconds: int) -> None:
    print("T minus")
    while seconds > 0:
        print(seconds)
        seconds = seconds - 1
        sleep(1)

    print(f"countdown {seconds}")


if __name__ == "__main__":
    main()

```

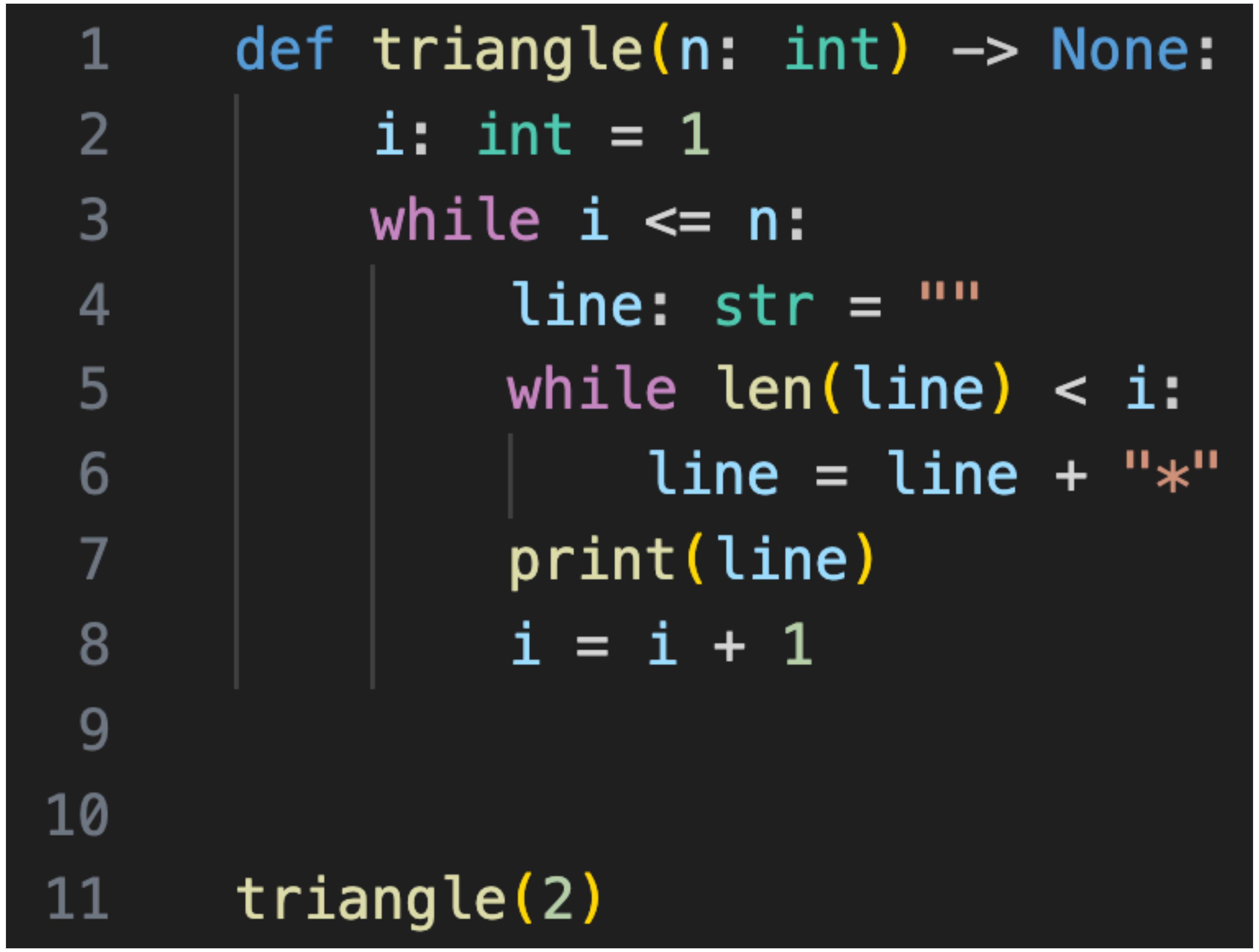
Controls



0. Continue/resume execution of program
1. Fully evaluate next line *jumping over function calls*
2. Evaluate next line and *jump into junction calls*
3. Complete this function and return to caller paused
4. Restart program*** (this restarts Trailhead)
5. Stop program*** (this stops Trailhead)

0 - 3 will pause for additional breakpoints if encountered.

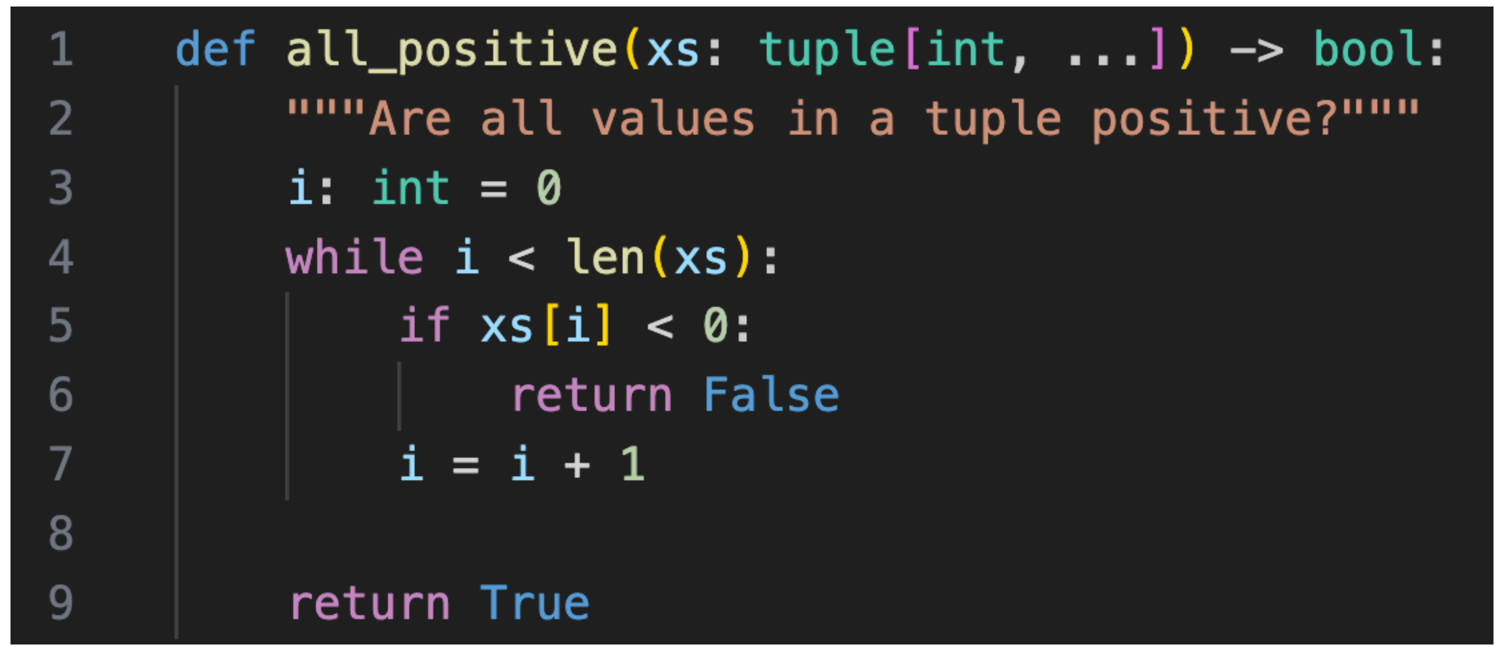
Trace a Memory Diagram



```python
def triangle(n: int) -> None:
    i: int = 1
    while i <= n:
        line: str = ""
        while len(line) < i:
            line = line + "*"
        print(line)
        i = i + 1


triangle(2)
```

Iterating N Times

... and over a Sequence.

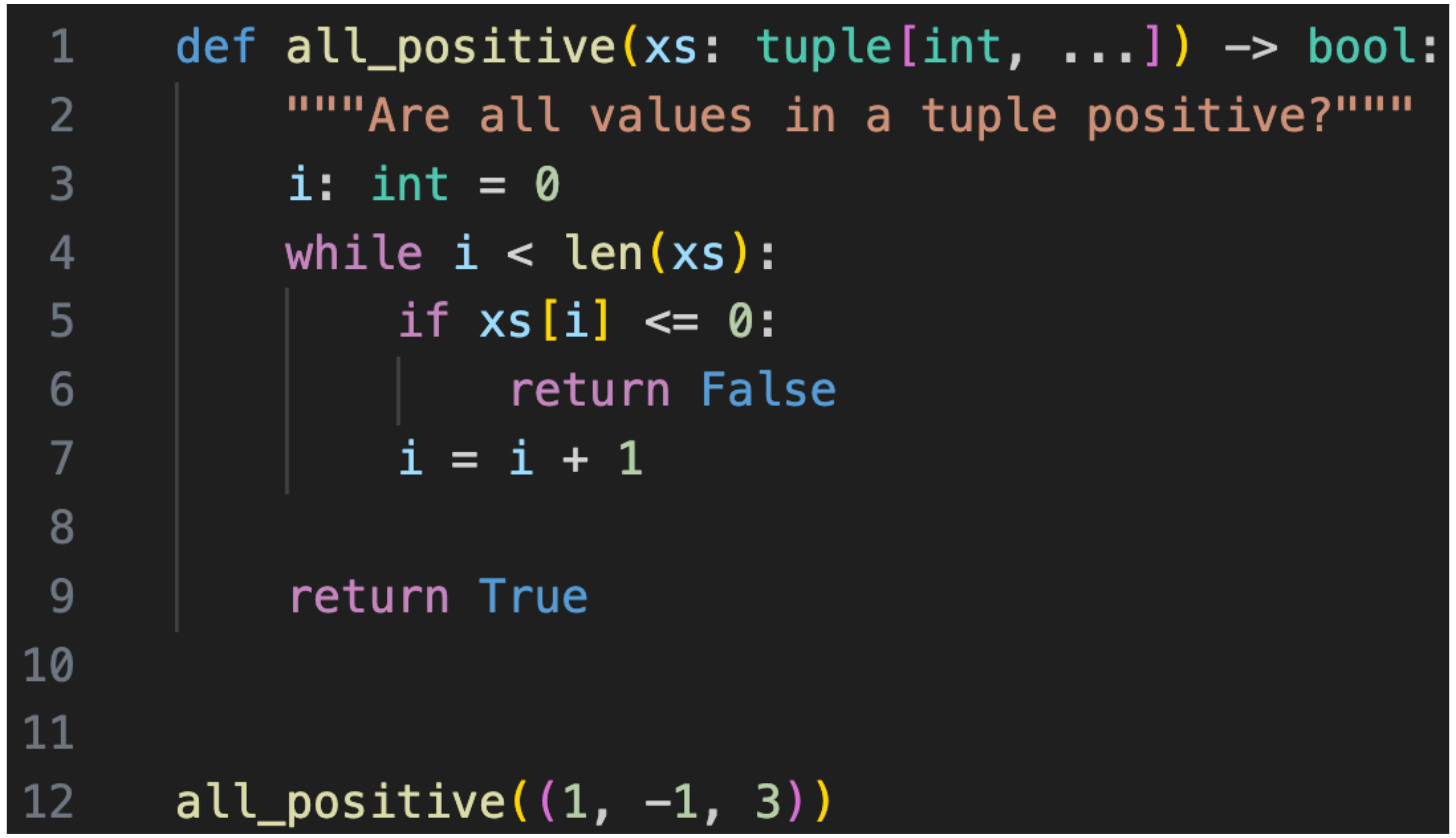


```python
def all_positive(xs: tuple[int, ...]) -> bool:
    """Are all values in a tuple positive?"""
    i: int = 0
    while i < len(xs):
        if xs[i] < 0:
            return False
        i = i + 1

    return True
```

Iterating N Times

… and over a Sequence.

Trace a Memory Diagram



```python
def all_positive(xs: tuple[int, ...]) -> bool:
    """Are all values in a tuple positive?"""
    i: int = 0
    while i < len(xs):
        if xs[i] <= 0:
            return False
        i = i + 1

    return True


all_positive((1, -1, 3))
```