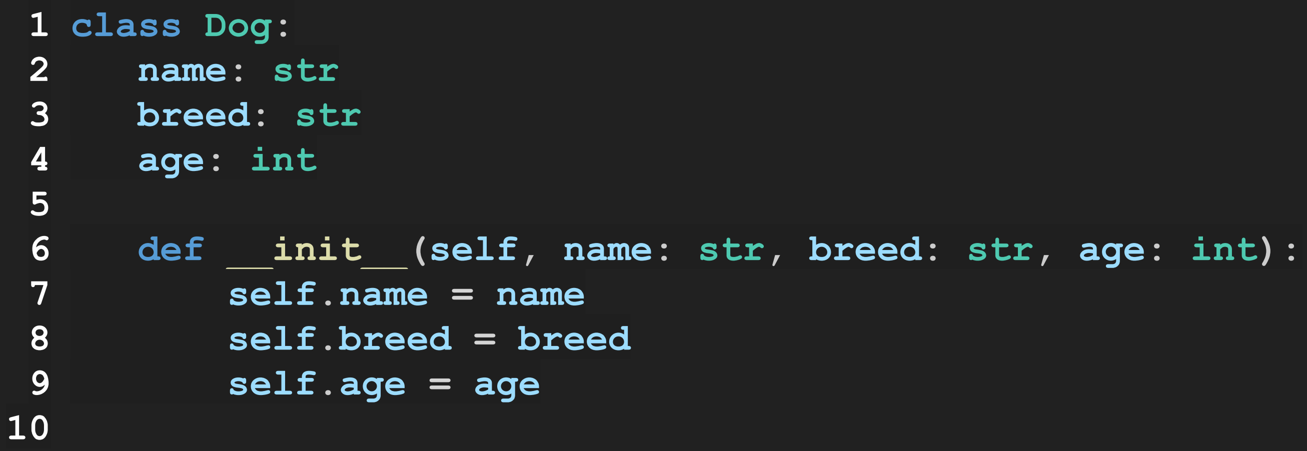
COMP110

🪄Magic Method🪄 Practice and Recursion Review

Announcements

EX06 due tomorrow at 11:59pm!

Warm-up: Consider this Dog class



```python
class Dog:
    name: str
    breed: str
    age: int

    def __init__(self, name: str, breed: str, age: int):
        self.name = name
        self.breed = breed
        self.age = age

```

With a partner:

Step 1: Write a __str__ magic method.

Step 2: Write a __repr__ magic method.

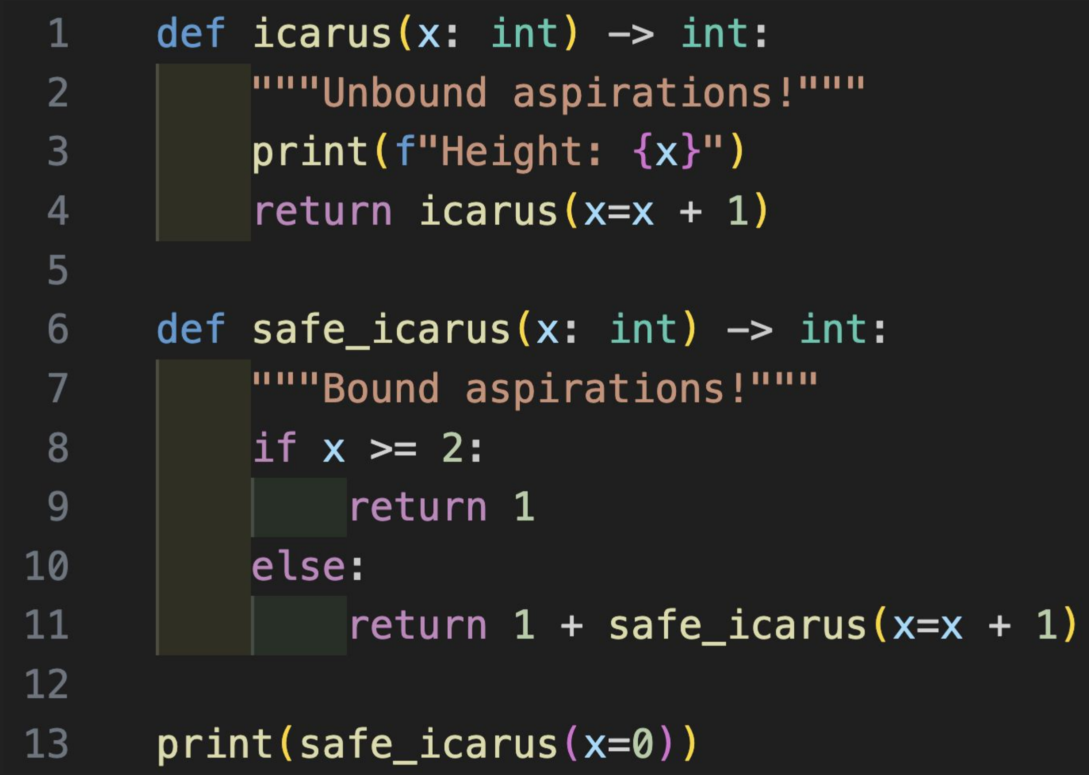
Warm-up: Consider this Dog class

```python
class Dog:
    name: str
    breed: str
    age: int

    def __init__(self, name: str, breed: str, age: int):
        self.name = name
        self.breed = breed
        self.age = age

    def __str__(self) -> str:
        """Returns a string representation (for humans)."""
        return ______________________

    def __repr__(self) -> str:
        """Returns a string representation (for debugging)."""
        return ______________________
```

Let’s go over it together! →

Shifting gears… remember recursion?

Recall these functions: what was the issue with the icarus function?



```python
def icarus(x: int) -> int:
    """Unbound aspirations!"""
    print(f"Height: {x}")
    return icarus(x=x + 1)

def safe_icarus(x: int) -> int:
    """Bound aspirations!"""
    if x >= 2:
        return 1
    else:
        return 1 + safe_icarus(x=x + 1)

print(safe_icarus(x=0))
```

The dreaded **`Recursion Error`**!

Stack Overflow and Recursion Errors

When a programmer writes a function that calls itself indefinitely (*infinitely*), the **function call stack** will *overflow…*

This leads to a **`Stack Overflow`** or **`Recursion Error`**:

```
RecursionError: maximum recursion depth exceeded while
calling a Python object
```

Recursive function checklist:

Base case:

- ❑ Does the function have a clear base case?
    - ❑ Ensure the base case returns a result directly (without calling the function again).
- ❑ Will the base case *always* be reached?

Recursive case:

- ❑ Does the function have a recursive case that *progresses toward the base case*?
    - ❑ Does the recursive call have the right arguments? The function should call itself on a simpler or smaller version of the problem.
- ❑ Have you tested your function with multiple cases, including edge cases?

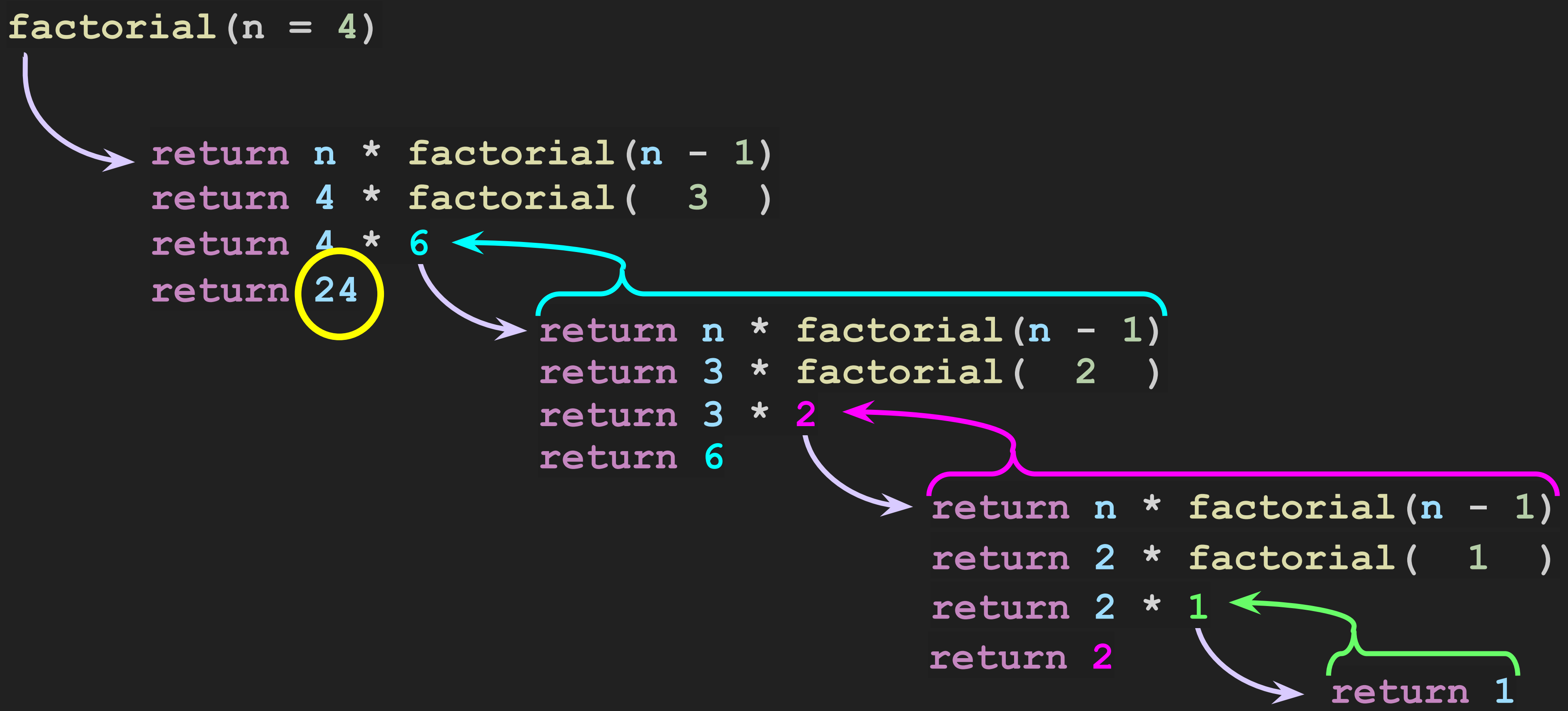
Another example of recursion: factorial!

To calculate the factorial of an int, `n`, we would multiply `n` by `(n-1)`, then `(n-2)`, and so on, until we reach `1`.

For instance, to calculate `5!`, we would do: `5 * 4 * 3 * 2 * 1`, which would evaluate to `120`.

```python
def factorial(n: int) -> int:
    # Base case: factorial of 0 or 1 is 1
    if n <= 1:
        return 1
    # Recursive case: n! = n × (n-1)!
    return n * factorial(n - 1)
```

Visualizing recursive calls to `factorial`



```
factorial(n = 4)

        return n * factorial(n - 1)
        return 4 * factorial(  3  )
        return 4 * 6
        return 24

                return n * factorial(n - 1)
                return 3 * factorial(  2  )
                return 3 * 2
                return 6

                        return n * factorial(n - 1)
                        return 2 * factorial(  1  )
                        return 2 * 1
                        return 2

                                return 1
```

Recursion: defining an operation/object in terms of itself

A real-world phenomenon! Examples:

- **You** have **parents**, who have **parents**, who have **parents**, who have **parents**, who…
  … were the **first humans**
- A **tree** has **branches**, which have **branches**, which have **branches**, which…
  … have **leaves**