HUMAN-CENTERED DESIGN IN CAPE TOWN

- **COMP 380H**
  Technology, Ethics, & Culture
- **Fall Semester 2025**
- **with Kris Jordan**

INFO SESSION
FEB 13 | 3:30 PM
GRAHAM MEMORIAL 039

Study abroad in Cape Town this Fall!

- Courses:
  - **Technology, Ethics, and Culture** with a UNC CS Prof.
  - **South African Urbanism** with a U of Cape Town Prof.
- Internship:
  - Intern with a non-profit in Cape Town!
- Details:
  - Program is run by Honors Carolina, but you don’t have to be in Honors Carolina to pursue it

Come to the info session to learn more and meet Kris!

CL12 – Quiz Practice

Today will be a pencil and paper/tablet kind of day!

Reminders

Quiz 01 Prep:

- Review session TODAY (Feb 12) at 6:15pm in Fred Brooks (FB) 007
- Visit Office Hours and Tutoring prior to the quiz – we *want* to help you!
    - Office Hours:
        - 11am–5pm every weekday in SN008
    - Tutoring:
        - Tonight (Wed): 5-7pm in Sitterson (SN) 011
        - Tomorrow (Thurs): 3-5pm in Fred Brooks (FB) 331

Unit 01 Content

- Review of relational operators
- Logical operators (**`and, not, or`**)
- Conditionals (**`if, if-else, if-elif-else`**)
- Positional and keyword arguments
- Named constants
- Default parameters
- Recursion

Answer the following:

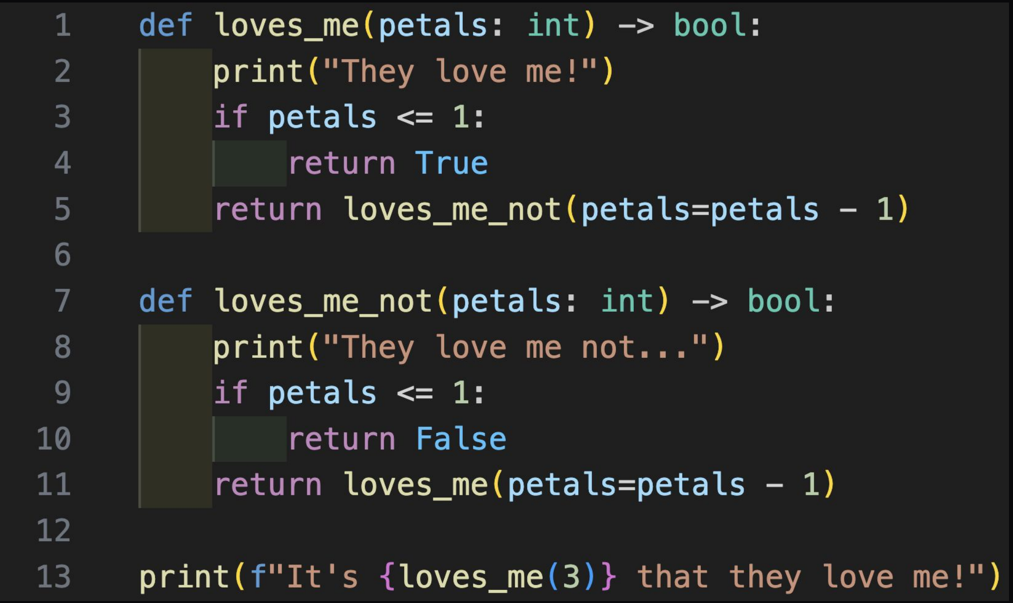
```
 1  def loves_me(petals: int) -> bool:
 2      print("They love me!")
 3      if petals <= 1:
 4          return True
 5      return loves_me_not(petals=petals - 1)
 6
 7  def loves_me_not(petals: int) -> bool:
 8      print("They love me not...")
 9      if petals <= 1:
10          return False
11      return loves_me(petals=petals - 1)
12
13  print(f"It's {loves_me(3)} that they love me!")
```

Which lines are the **base case(s)** on?

Which lines are the **recursive case(s)** on?

Which lines are the **function call(s)** on?

Your warm-up: Diagram this!



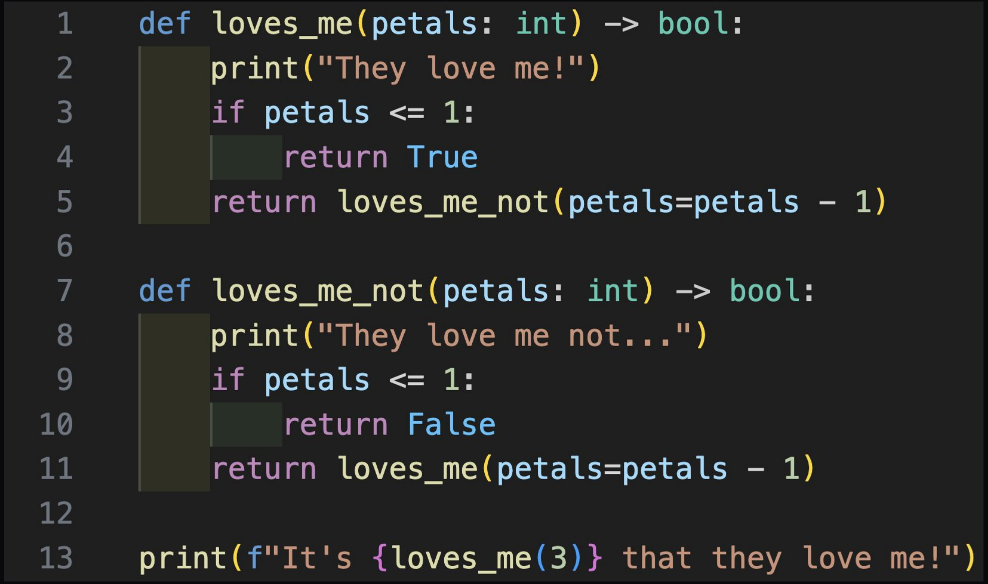
```python
def loves_me(petals: int) -> bool:
    print("They love me!")
    if petals <= 1:
        return True
    return loves_me_not(petals=petals - 1)

def loves_me_not(petals: int) -> bool:
    print("They love me not...")
    if petals <= 1:
        return False
    return loves_me(petals=petals - 1)

print(f"It's {loves_me(3)} that they love me!")
```

After you finish your diagram, ponder:

What are these functions *mathematically doing* (in addition to determining someone’s love)?

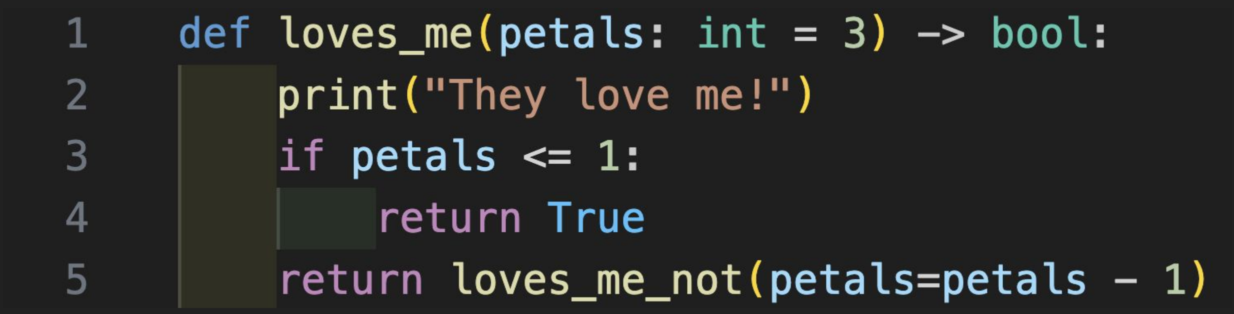


```python
def loves_me(petals: int) -> bool:
    print("They love me!")
    if petals <= 1:
        return True
    return loves_me_not(petals=petals - 1)

def loves_me_not(petals: int) -> bool:
    print("They love me not...")
    if petals <= 1:
        return False
    return loves_me(petals=petals - 1)

print(f"It's {loves_me(3)} that they love me!")
```

How could we call the `loves_me` function when `petals` is assigned a default value?



```python
def loves_me(petals: int = 3) -> bool:
    print("They love me!")
    if petals <= 1:
        return True
    return loves_me_not(petals=petals - 1)
```

Hand-writing code: An adaptation of `fizzbuzz`

A group of students start counting up from 1, taking turns saying either a number or a phrase.

If their number is divisible by 3, the student says “fizz” rather than the number.

If their number is divisible by 5, they say “buzz” rather than the number.

If their number is divisible by both 3 and 5, they say “fizzbuzz”

Example:

1, 2, fizz, 4, buzz, fizz, 7, 8, fizz, buzz, 11, fizz, 13, 14, fizzbuzz, 16, …

Hand-writing code: An adaptation of `fizzbuzz`

Our function definition should meet the following specifications:

- The function should be named **`fizzbuzz`**, have one **`int`** parameter named **`n`**, and return a **`str`**
- If **`n`** is divisible by 3 and not 5, the function should print "fizz"
- If **`n`** is divisible by 5 and not 3, the function should print "buzz"
- If **`n`** is divisible by 3 AND 5, the function should print "fizzbuzz"
- If **`n`** is not divisible by 3 OR 5, the function should print **`n`** as a string
- The function should keep calling itself, increasing the argument by 1 each time, until we finally reach a “fizzbuzz” number, when we’ll return “fizzbuzz”
- Explicitly type your parameter and return type.

One solution (note that there are many correct ways to write it!)

```python
def fizzbuzz(n: int) -> str:
    if n % 3 == 0:  # If n is divisible by 3...
        if n % 5 == 0:  # If n is also divisible by 5...
            print("fizzbuzz")
            return "fizzbuzz"  # Base case!
        else:
            print("fizz")  # Print this if n is divisible by 3 but NOT 5
    elif n % 5 == 0:
        print("buzz")  # Print this if n is divisible by 5 but NOT 3
    else:
        print(str(n))  # If not divisible by 3 or 5, just print n as a string
    return fizzbuzz(n + 1)  # Recursive case progresses toward the base case
```

Why does this recursive case work? If we keep increasing n by 1 with each recursive function call, n will eventually be a number that's divisible by 3 AND 5 (e.g., when n is 15). When this happens, the base case will be reached and “fizzbuzz” will be returned.