

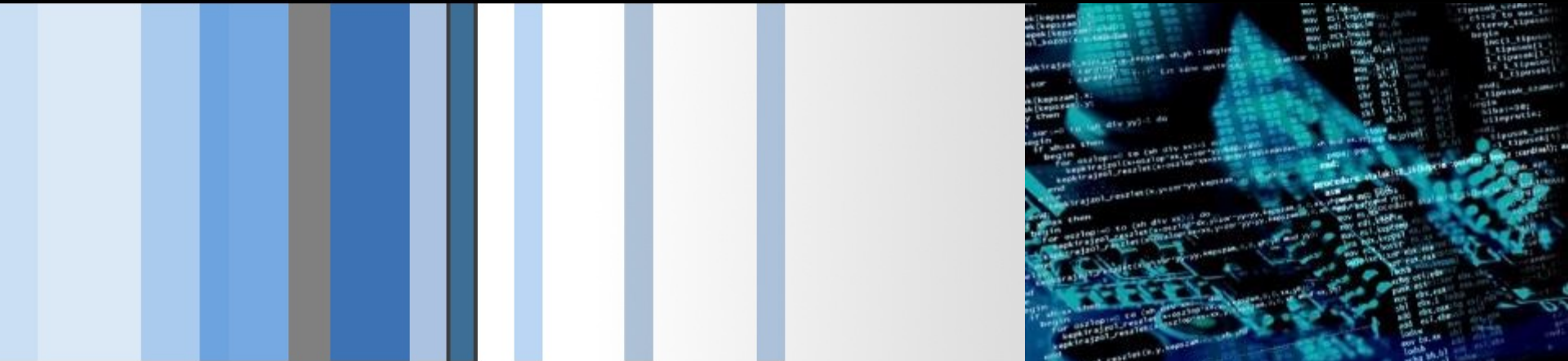
Class 10

Local scope

CSC 141 · Class 10

Tuesday, September 29, 2026

Dr. Si Chen



Quiz 2

- 11:00–11:15 AM
- Covers Classes 6–7: nested decisions, testing, and debugging.
- Closed notes. Work independently.
- **Online:** wcu.ninja/csc141/quiz2
- Use the paper version if directed.

Today's learning goals

- Explain what "local" means for a variable created inside a function.
- Predict whether a name defined in one function is visible in another.
- Recognize UnboundLocalError as a scope mistake, not a random bug.
- See why keeping functions independent makes them easier to test.

A function's variables stay inside it

```
def rate_label(hour):  
    shift = "day" if 6 <= hour < 18 else "night"  
    return shift  
  
rate_label(9)  
print(shift)
```

- shift is created only while rate_label runs.
- Once the function returns, shift is gone.
- NameError, not a typo — Python is saying the name never existed out here.

Trace: two independent calls

```
def add_bonus(points):  
    bonus = points // 10  
    return points + bonus  
  
total1 = add_bonus(47)  
total2 = add_bonus(82)  
print(total1, total2)
```

- Each call gets its own fresh bonus.
- The 4 computed for the first call is gone before the second call starts.
- **Output:** 51 90

Same name, two functions, two boxes

```
def double_it(n):  
    result = n * 2  
    return result  
  
def square_it(n):  
    result = n * n  
    return result  
  
print(double_it(5), square_it(5))
```

- Both functions use the name result.
- Python keeps a separate box per function call, so they never collide.
- **Output:** 10 25

A common mistake: used before it exists

```
def compute_total(price, qty):  
    if qty > 10:  
        total = price * qty * 0.9  
    return total  
  
print(compute_total(5, 3))
```

- total is only assigned inside the if.
- For qty = 3, the if never runs, so total was never created.
- **UnboundLocalError:** local variable 'total' referenced before assignment.

Why this matters

- Functions can be tested and reasoned about on their own.
- A bug inside one function cannot silently change a variable in another.
- This is part of why Class 9 could test `average_score` and `letter_grade` separately.
- Local scope is a feature, not a limitation — it keeps functions predictable.

Exit ticket

- If a variable is created inside a function, where does it exist?
- What error appears when a local variable is used before any path assigns it?
- Can two different functions safely use the same variable name? Why?

Before Thursday

- Programming Assignment 2 is due tonight, 11:59 PM in D2L.
- Bring one scope question, or a program where a variable seemed to "disappear."
- **Next class:** designing functions on purpose, and debugging programs with several functions.