

Class

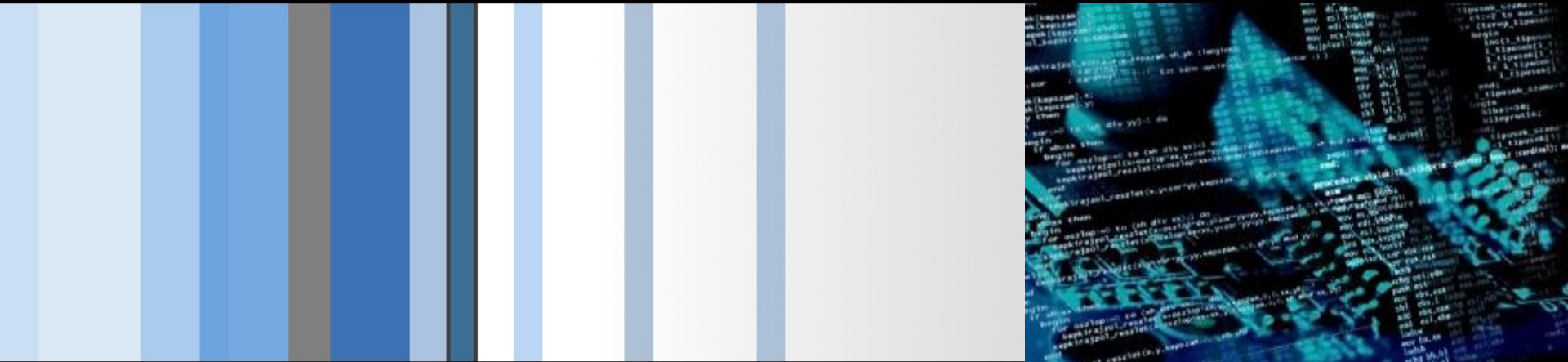
11

Function design and debugging

CSC 141 · Class 11

Thursday, October 1, 2026

Dr. Si Chen



Today's learning goals

- Recap local scope from a caller's perspective.
- Design a function's signature before writing its body.
- Test each function alone before combining them.
- Trace a bug that spans more than one function call.

Recap: local scope in one line

- A variable created inside a function exists only inside that call.
- Two functions can reuse the same variable name with no conflict.
- A local variable used before any path assigns it raises `UnboundLocalError`.

Parameters are local too

```
def add_tax(price):  
    price = price * 1.06  
    return price
```

```
cost = 10  
new_cost = add_tax(cost)  
print(cost, new_cost)
```

- Reassigning price inside the function does not change cost outside.
- price is a local copy of the value passed in.
- **Output:** 10 10.6

Designing a function before writing it

- Name the function for what it computes or does.
- **Decide the parameters:** what information does it need to receive?
- **Decide the return value:** what does it hand back to the caller?
- Write one sentence describing the job — before writing any code.

Design first: pick the signature

```
def shipping_cost(weight, rush):  
    # TODO: $0.50/pound, +$8 if rush  
    pass
```

- **Two inputs:** weight, rush.
- **One output:** a dollar amount.
- The signature is a promise about what goes in and what comes back — write it before the body.

Then write the body

```
def shipping_cost(weight, rush):  
    base = weight * 0.5  
    return base + 8 if rush else base  
  
print(shipping_cost(4, False))  
print(shipping_cost(4, True))
```

- 4 pounds, no rush: \$2.00.
- 4 pounds, rush: \$10.00.
- Test the function alone — the same habit from Class 9's `average_score(100)` check.

A bug that spans two functions

```
def tax_amount(subtotal):  
    rate = 0.06  
    amount = subtotal * rate  
    return amount  
  
def total_with_tax(subtotal):  
    return subtotal + rate  
  
print(total_with_tax(50))
```

- rate exists only inside tax_amount.
- total_with_tax cannot see it.
- **NameError:** name 'rate' is not defined.

The fix: call it, don't reach into it

```
def tax_amount(subtotal):  
    return subtotal * 0.06  
  
def total_with_tax(subtotal):  
    return subtotal + tax_amount(subtotal)  
  
print(total_with_tax(50))
```

- `total_with_tax` calls `tax_amount` to get the value.
- It does not try to reach inside `tax_amount`'s local variables.
- **Output:** 53.0

A systematic way to debug several functions

- Test each function alone with a known input first.
- If a function fails its own test, the bug is inside that function.
- If every function passes alone but the combined program is wrong, check how they are composed — the calls and the arguments passed between them.
- Trace one function call at a time; do not guess across all of them at once.

Guided build: a parking fee calculator

- Write two functions.
- `hourly_fee(hours)` returns \$2.50 per hour, capped at \$20.00 for the day.
- `member_total(hours, is_member)` returns `hourly_fee(hours)` with 10% off for members, full price otherwise.
- **Combine them:** print the total for a case you choose.

Parking fee test cases

hours	is_member	expected
3	False	\$7.50
10	False	\$20.00
10	True	\$18.00
0	True	\$0.00

Parking fee solution

```
def hourly_fee(hours):  
    return min(hours * 2.5, 20.0)  
  
def member_total(hours, is_member):  
    fee = hourly_fee(hours)  
    if is_member:  
        return fee * 0.9  
    return fee  
  
print(member_total(10, True))
```

- **hourly_fee is tested alone first:**
hourly_fee(10) should be 20.0.
- member_total then calls it — it never recomputes the cap itself.

Independent practice: a print shop calculator

- **Write `page_cost(pages)`:** \$0.10 per page.
- Write `rush_total(pages, rush)`: `page_cost(pages)` plus a flat \$5.00 if `rush` is `True`, otherwise unchanged.
- Test 20 pages not rushed, and 20 pages rushed.

Explain your design

- **With a partner:**
- Name your two functions and what each one returns.
- Point to the line where one function's result is used by the other.
- Explain why a bug inside `page_cost` could not accidentally change a variable inside `rush_total`.

Exit ticket

- Why couldn't `total_with_tax` directly use `tax_amount`'s local variable `rate`?
- What is the first step when a multi-function program produces a wrong number?
- Name one question you would ask yourself before writing a function's body.

Before next class

- Exam 1 is Thursday, October 8. Review begins Tuesday, October 6 — bring specific questions, not just "review everything."
- Save and rerun your print shop functions once more without looking at your notes.
- **Next class:** iteration — while, for, and range.