ReDI School of Digital Integration

Debugging 

We use tech to connect human potential and opportunity with dignity & humility

9/9

0800 Arctan started

1000 " stopped - arctan ✓ { 1.2700 · 9.037 847 025
9.037 846 795 correct

13"wc (032) MP - MC ~~1.9826447000~~ ~~2.13047676915 (-3)~~ 4.61592505 9(-2)

(033) PRO 2 2.13047676415

correct 2.13067676415

Relays 6-2 in 033 failed special speed test
in relay " 10.00 test.

Relays changed

Relay 2145
Relay 3370

1700 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 

Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

~~1700~~ 1630 Arctangent started.

1700 closed down.

What does "debugging" mean?

- Even before computers were common, engineers used "debugging" to mean finding and fixing errors in systems, such as engines or cameras
- Why not just "find and fix errors"?
- There's usually a connotation that the system's input and output are not directly coupled (related), so it is not easily reproducible
- Therefore you need process and tools to find out where things went wrong
- Sometimes, you debug systems that you created in the first place!

https://en.wikipedia.org/wiki/Debugging

Real world bug examples

- The door of the U-Bahn train is defective and doesn’t open
  - System doesn’t work at all

Real world bug examples

- The door of the U-Bahn train is defective and doesn’t open
  - System doesn’t work at all
- You buy a Margherita pizza and they give you a Salami pizza
  - System works (you got a pizza) but wrong output

Real world bug examples

- The door of the U-Bahn train is defective and doesn’t open
  - System doesn’t work at all
- You buy a Margherita pizza and they give you a Salami pizza
  - System works (you got a pizza) but wrong output
- Another customer orders Hawaii and they give him your Margherita pizza
  - System works, wrong output for you and for another customer
  - The error in your case affected another case

Real world bug examples

- The door of the U-Bahn train is defective and doesn’t open
  - System doesn’t work at all
- You buy a Margherita pizza and they give you a Salami pizza
  - System works (you got a pizza) but wrong output
- Another customer orders Hawaii and they give him your Margherita pizza
  - System works, wrong output for you and for another customer
  - The error in your case affected another case
- You make an appointment at the doctor, you go there and they forgot about you (but the rest of the people are ok)
  - System works partially. Hard to identify because it only happens sometimes

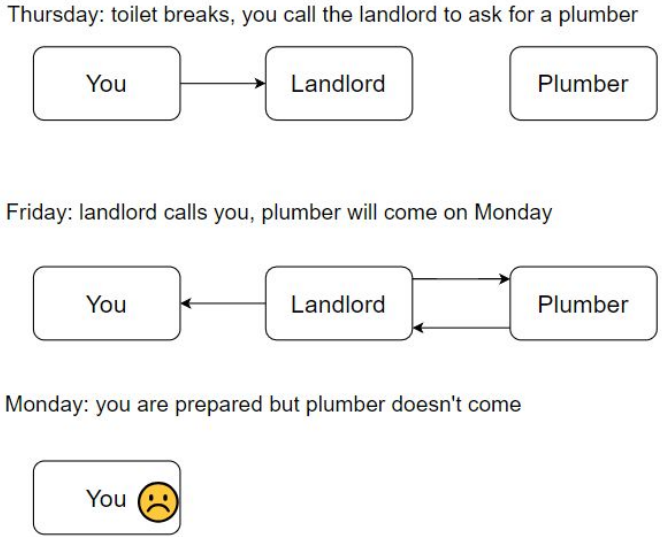
From Ane's slides last semester!

Real world bug examples

- Your toilet is broken. You call the landlord to ask for a plumber, he tells you plumber will be there next Monday.
- Plumber never comes
- What happened? What do you do?

Real world bug examples

- Your toilet is broken. You call the landlord to ask for a plumber, he tells you plumber will be there next Monday.
- Plumber never comes
- What happened? What do you do?



Also from Ane's slides last semester!

Real world bug examples

Step 1: toilet breaks, you call landlord to ask for a plumber

Step 2: landlord calls plumber, makes an appointment and tells you the day

Step 3: plumber cancels, landlord forgets to tell you

Step 4: the day comes, you wait and no one comes. Program failure

Steps

1. Reproduce the problem
   Find the right environment / inputs that this problem happens
   Define what output you are expecting
2. Narrow down / simplify the inputs
   Remove/control extraneous bits that don't cause the issue
3. Examine / trace program state to find source of issue
   Use tools such as logs / debuggers / metrics / LLMs to find the issue
4. Write a fix for it
5. Write a test for it
   To prevent it from coming back (regressions)!

Can you automate it? Read this article from Meta

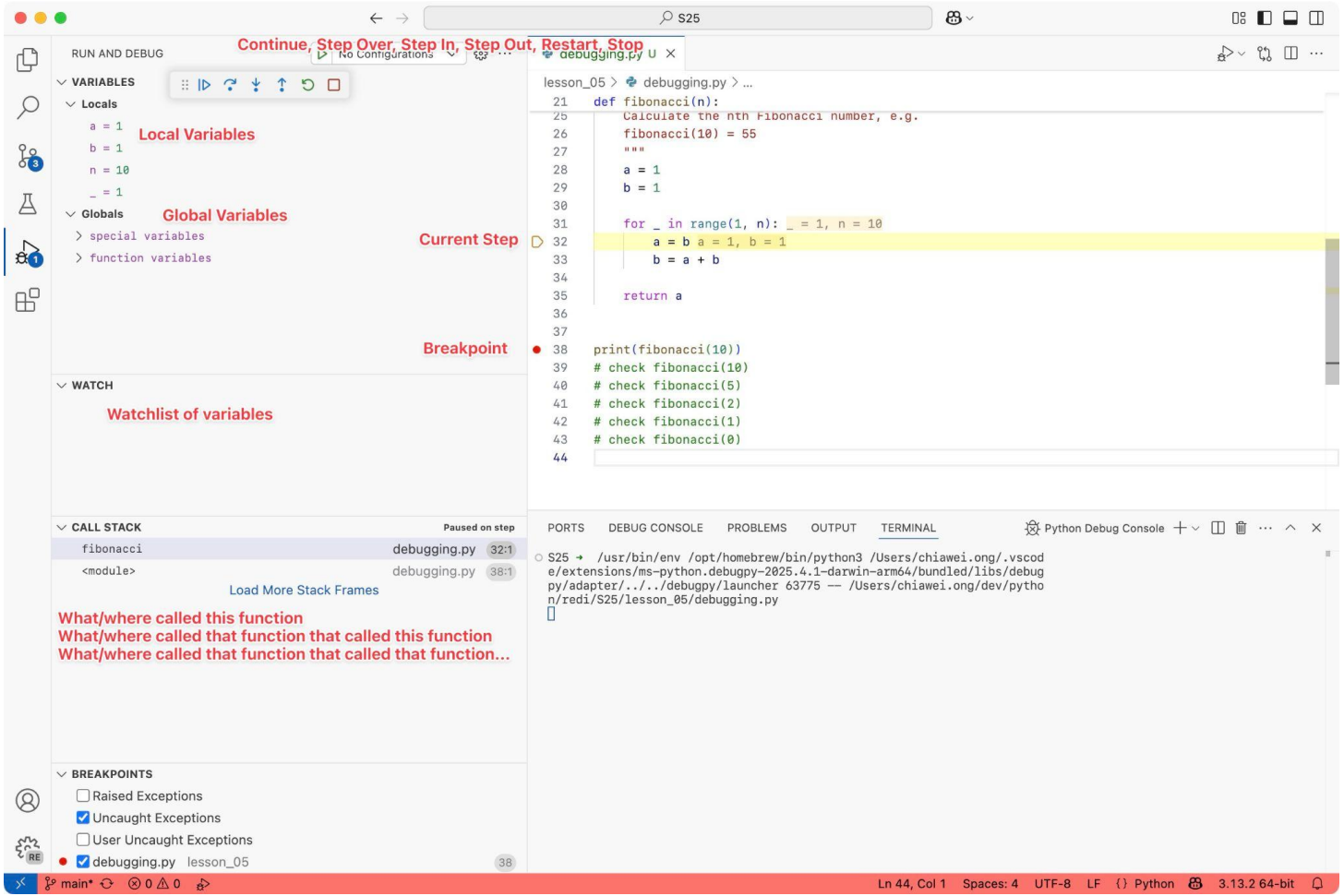
Logs / Print Debugging

- Add `print()` to your code to find out what it is doing
- Easy to start; can be difficult to write this over many functions and files, and cleanup after
- Frowned upon if done unnecessarily
- Necessary for tracing, especially in distributed systems (many machines)

```
for i in range(1, n):
    print(f"this is the {i} loop")
    print(f"a is {a}")
    print(f"b is {a}")
    a = b
    b = a + b
    print(f"(after) a is {a}")
    print(f"(after) b is {a}")
```

Debugger

- Can be hard to start / use at first
- Allow you to inspect every step / state of your program
- Breakpoints
  - Places where the debugger will stop at your code
- Step in / out / over
  - Go into functions, out of functions, over functions
- Variables (recall scopes from functions)
  - Local: variables defined/accessible in this function
  - Global: variables defined outside of this function


Continue, Step Over, Step In, Step Out, Restart, Stop
RUN AND DEBUG
No Configurations
debugging.py U
VARIABLES
Locals
a = 1
b = 1
n = 10
_ = 1
Local Variables
Globals
special variables
function variables
Global Variables
Current Step
Breakpoint
WATCH
Watchlist of variables
CALL STACK
Paused on step
fibonacci debugging.py 32:1
<module> debugging.py 38:1
Load More Stack Frames
What/where called this function
What/where called that function that called this function
What/where called that function that called that function...
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
debugging.py lesson_05 38
lesson_05 > debugging.py > ...
def fibonacci(n):
Calculate the nth Fibonacci number, e.g.
fibonacci(10) = 55
"""
a = 1
b = 1
for _ in range(1, n): _ = 1, n = 10
a = b a = 1, b = 1
b = a + b
return a
print(fibonacci(10))
# check fibonacci(10)
# check fibonacci(5)
# check fibonacci(2)
# check fibonacci(1)
# check fibonacci(0)
PORTS DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL
Python Debug Console
S25 → /usr/bin/env /opt/homebrew/bin/python3 /Users/chiawei.ong/.vscode/extensions/ms-python.debugpy-2025.4.1-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 63775 -- /Users/chiawei.ong/dev/python/redi/S25/lesson_05/debugging.py
main*
Ln 44, Col 1 Spaces: 4 UTF-8 LF Python 3.13.2 64-bit