Lecture 32:

HANDLING EXCEPTIONS

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

Announcements 1/3

- Additional office hours this week:
  - ✓Wednesday from 2 to 4
  - Friday (today) from 1:30 to 3
    - (16 to 0 for talking about projects instead of the midterm)

Announcements 2/3

interested in

CS Grad School & REUs?

Alvitta Ottley
WashU

Shannon Roberts
UMass

Angus Forbes
UCSC

Samantha Williams
UVM

Charles Perin
U. Victoria

Leilani Battle
UMD

Lane Harrison
WPI

TODAY!

Friday
Nov. 30th
4pm
Ford 241
Panel+Chat+Cookies

Announcements 3/3

Thanksgiving Welcome Back Game Night

Friday, 11/30 7-9pm

Gaming Lab Hillyer 326 (meet in Hillyer Atrium)

TONIGHT!

Having a rough semester and need an extra break after Thanksgiving?
Want to bond with students and get to know the CS community better?
Come by our Game Night and have a heartwarming time with us!

Smithies in CS

Coming up

- ✓Announcements
- ✓Final Project Proposal Recap
- ✓Working with files
  - ✓what is a file?
  - ✓reading data in
  - ✓writing data out
- ✓Intro to Algorithms
- **Recap of Lab: Sorting**
- Handling `Exceptions`
- Final Project Workshop

Discussion

What part of Lab 10 was

the **most challenging**?

Takeaways from Lab 10

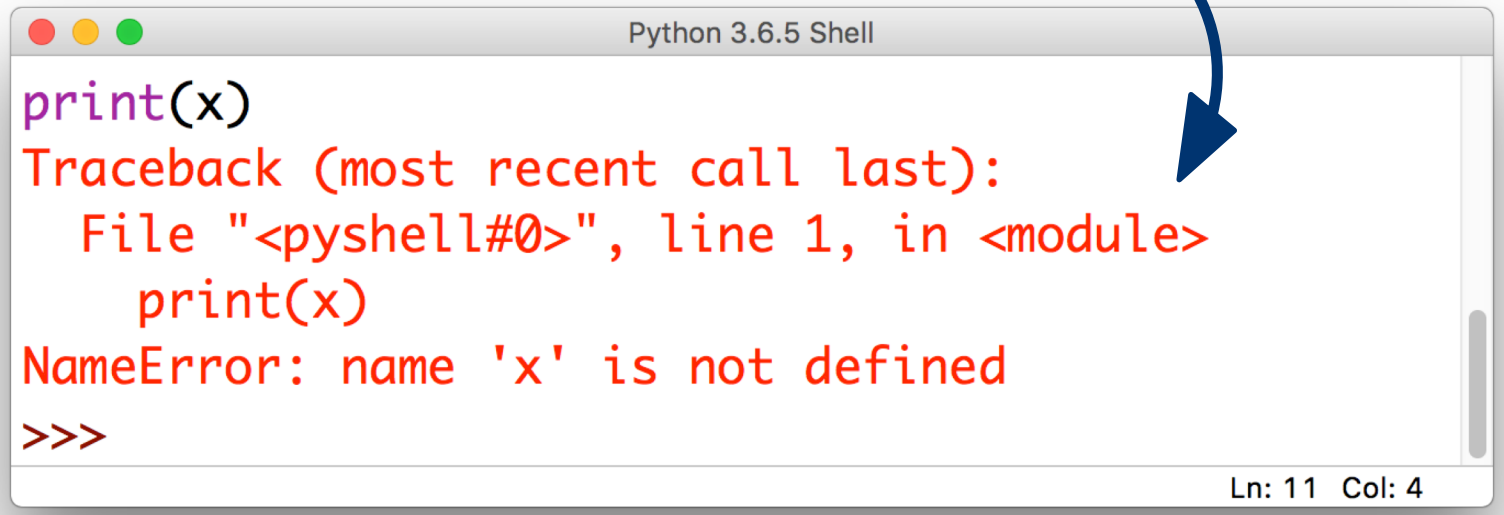
- Algorithmic thinking is **hard** on multiple levels
  - Figuring out how to **describe** the process
  - Translating that **process** into code
  - Recognizing when the code **isn't doing what you want**
  - Evaluating **efficiency**
  - …and much more…
- Yesterday was just a **small taste** (it's okay if you didn't finish)
- Many more opportunities to master this skill, will be a **recurring theme** in future courses
  - CSC212
  - CSC250
  - CSC252

Coming up

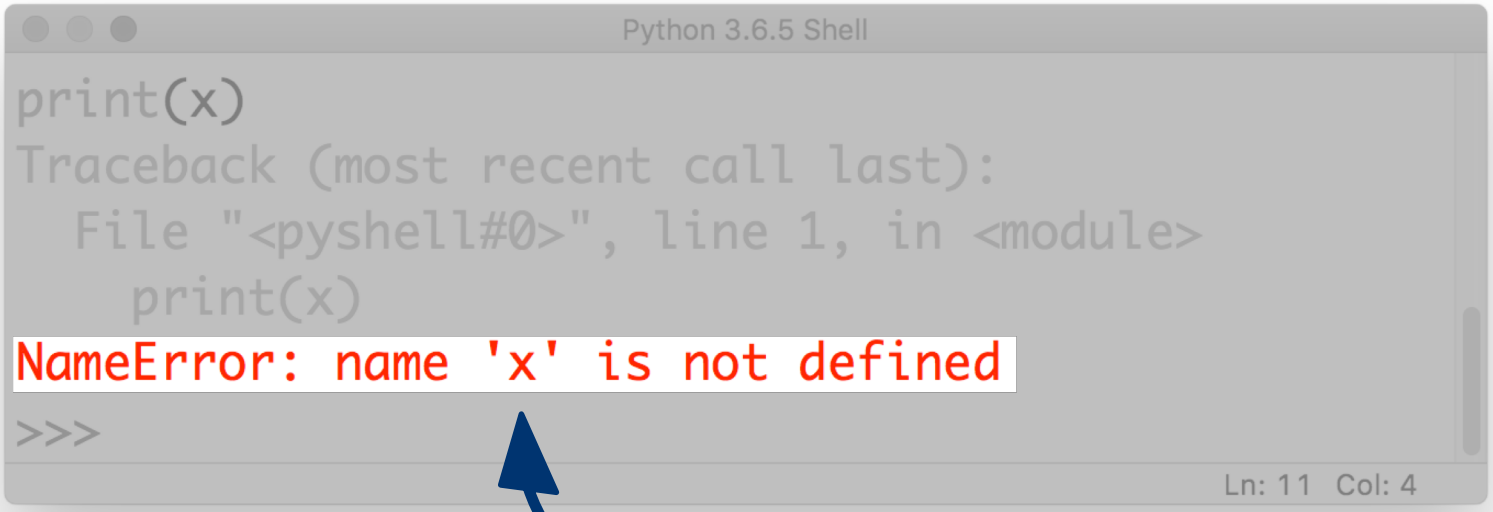
- ✓Announcements
- ✓Final Project Proposal Recap
- ✓Working with files
  - ✓what is a file?
  - ✓reading data in
  - ✓writing data out
- ✓Intro to Algorithms
- ✓Recap of Lab: Sorting
- Handling `Exceptions`
- Final Project Workshop

Lecture 10: some problems are obvious

this is called
an **`Exception`**



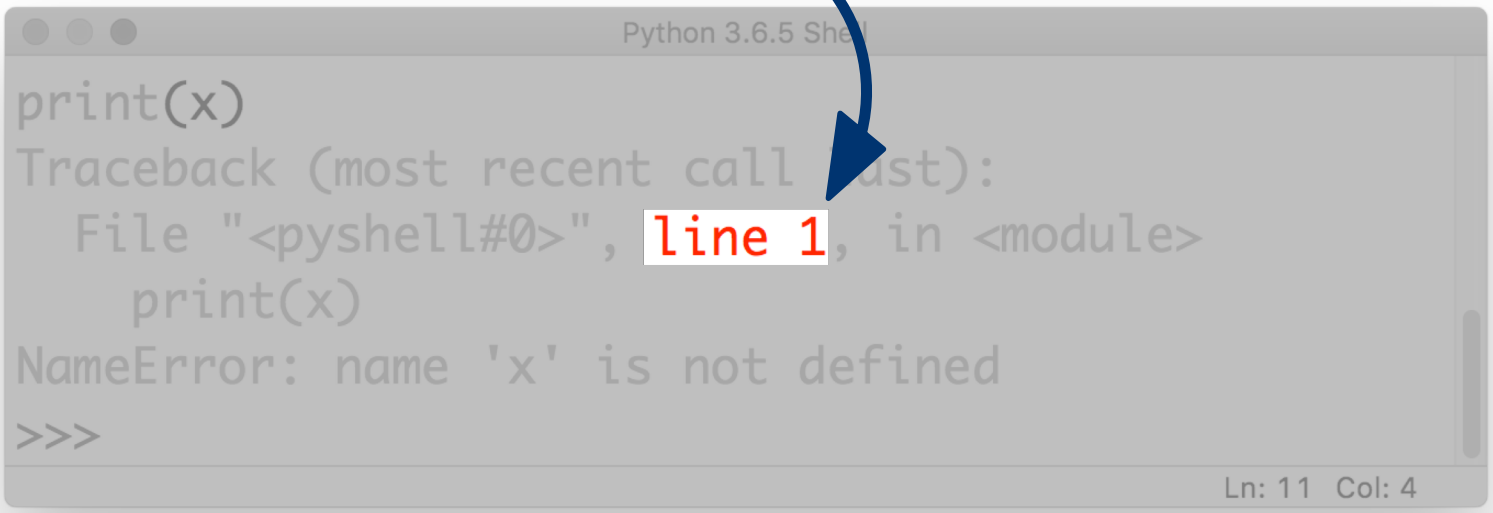
Lecture 10: some problems are obvious



the kind of error gives you
a **clue** about what the problem is

Lecture 10: some problems are obvious

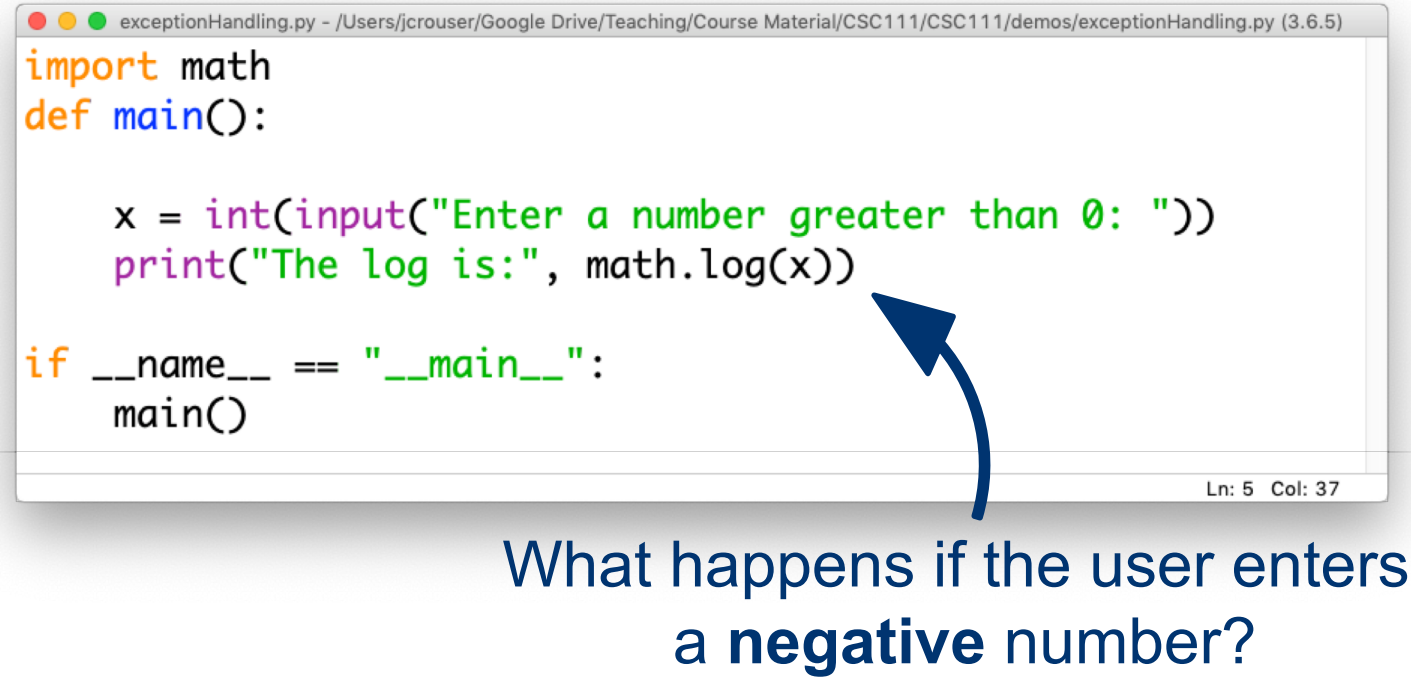
it also tells you **where** the problem is
(but be careful!)



Discussion

But there's a drawback to when your program throws an **`Exception`**…

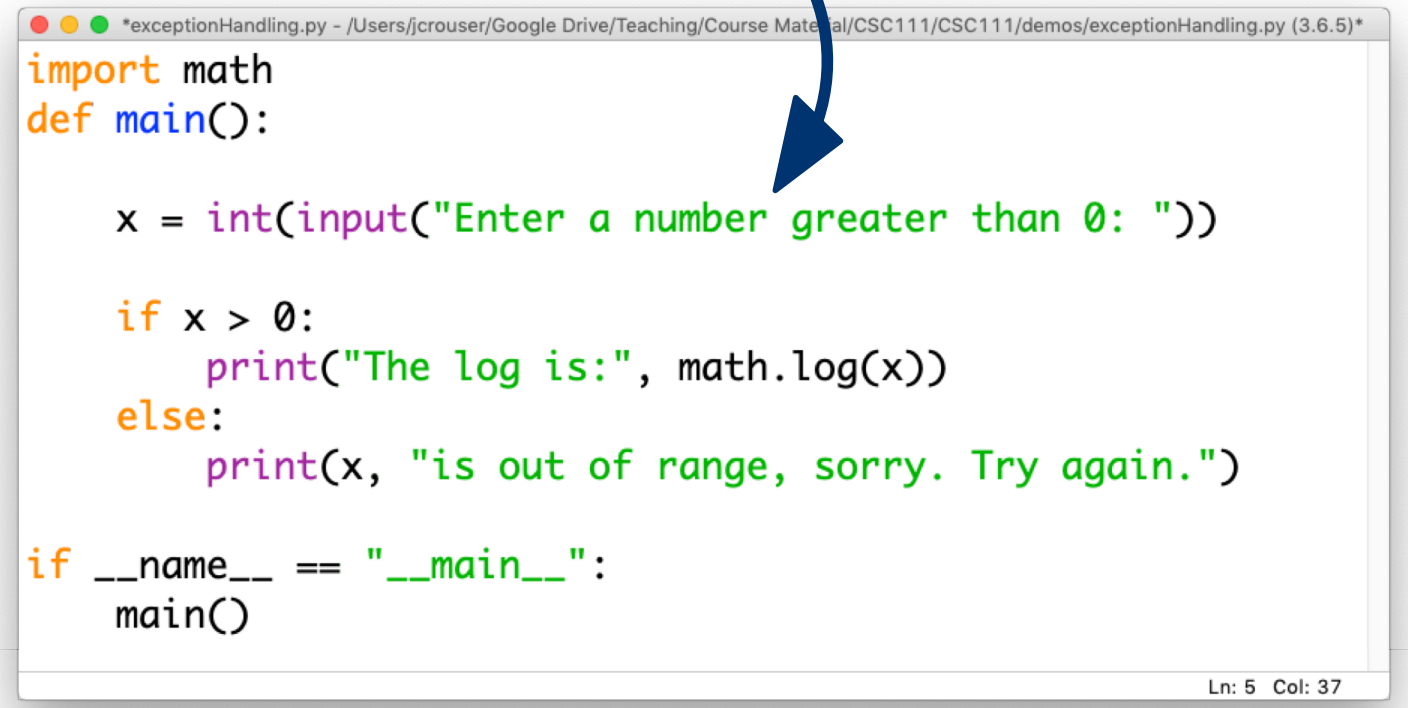
An example



What happens if the user enters a **negative** number?

An example

What happens if the user enters a **string**?



The `try...except` block

- There are some cases where avoiding an **`Exception`** isn't possible
- In this case, we want tell Python:
  - what we **want** to happen
  - how to **handle** it if things go wrong

An example

exceptionHandling.py - /Users/jcrouser/Google Drive/Teaching/Course Material/CSC111/CSC111/demos/exceptionHandling.py (3.6.5)

```
import math
def main():

    try:
        x = int(input("Enter a number greater than 0: "))
        print("The log is:", math.log(x))

    except ValueError:
        print("Not a valid input.")

if __name__ == "__main__":
    main()
```

Ln: 6 Col: 36

Takeaways

- Even if you can't avoid all errors, you can design your program to **fail gracefully**
- You can handle multiple different kinds of **`Exceptions`**, and you can handle them differently
- Think about **edge cases** to provide specific feedback about what went wrong