Lecture 31:

HANDLING EXCEPTIONS

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

Outline

- ✓Monday: Interaction
- Debrief of Lab 10: Interactive Fish Tank
- Wednesday: Handling Exceptions
- Friday: Special Topic – Ethical Issues in Tech
- FP2: Persona, Paper Prototype, and Architecture Diagram

Discussion

What parts of Lab 10 were the **most challenging**?

Takeaways from Lab 10

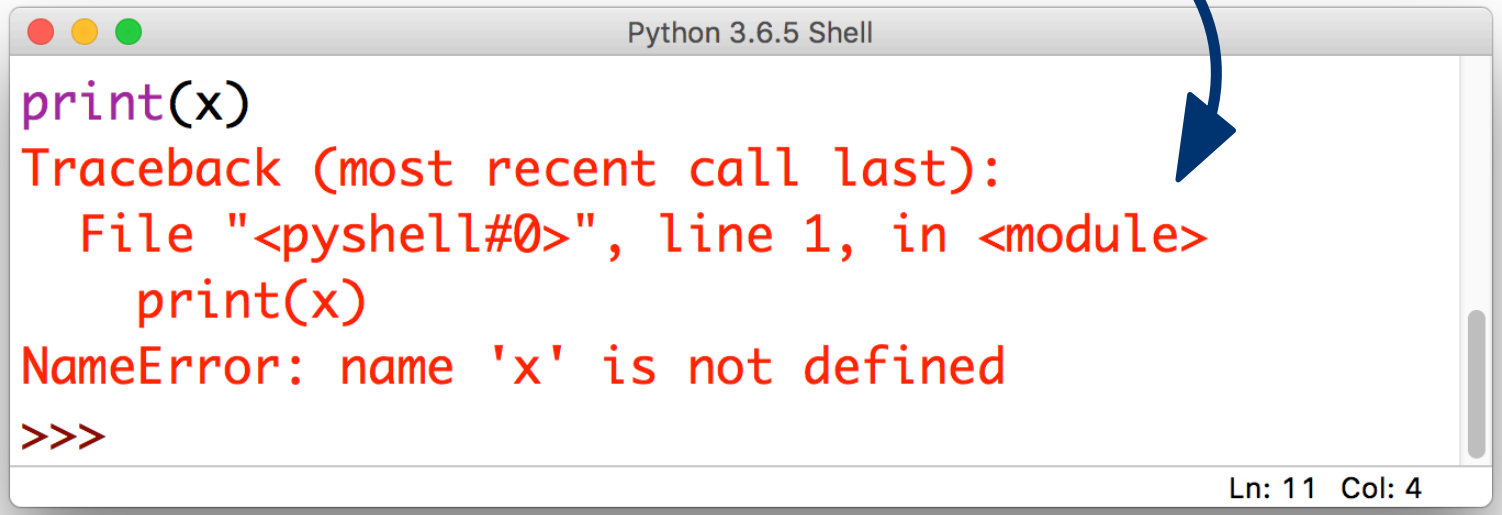
- Algorithmic thinking is **hard** on multiple levels
  - Figuring out how to **describe** the process you want to happen
  - Translating that **process** into code
  - Recognizing when the code **isn't doing what you want**
  - Evaluating **efficiency**
  - …and much more…
- Yesterday was just a **small 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

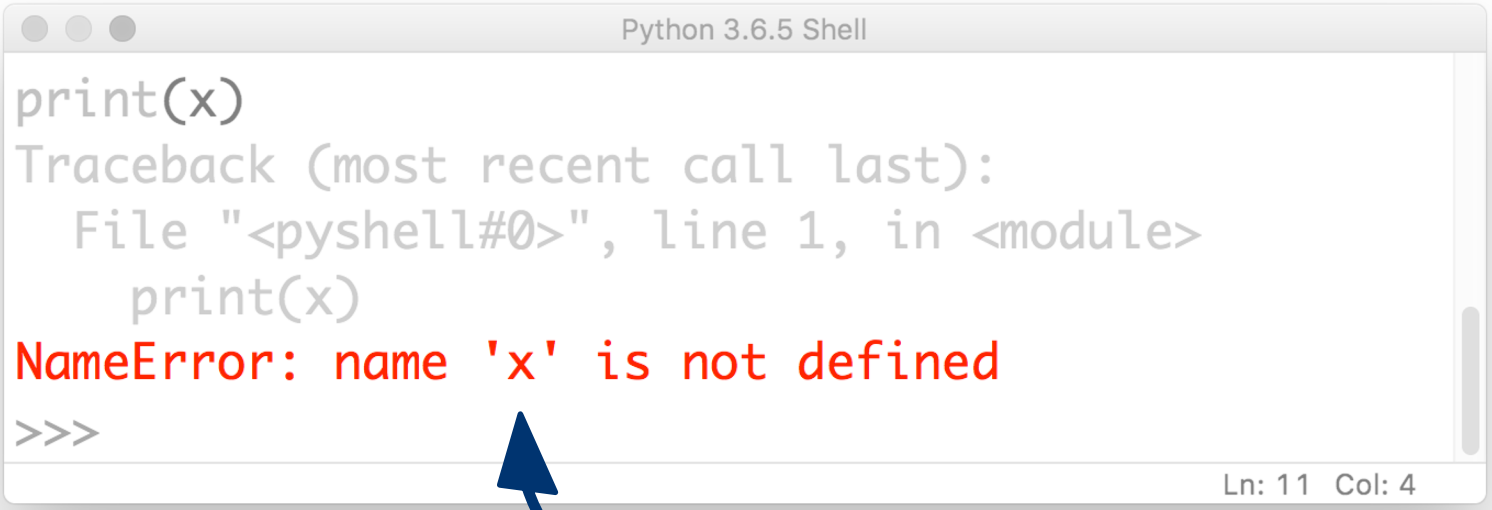
- ✓Announcements
- ✓Final Project Proposal Recap
- ✓Working with files
  - ✓what is a file?
  - ✓reading data in
  - ✓writing data out
- ✓Intro to Algorithms
- ✓Recap of Lab: Sorting
- Handling `Exceptions`
- Final Project Workshop

Lecture 10: some problems are obvious

this is called
an **`Exception`**



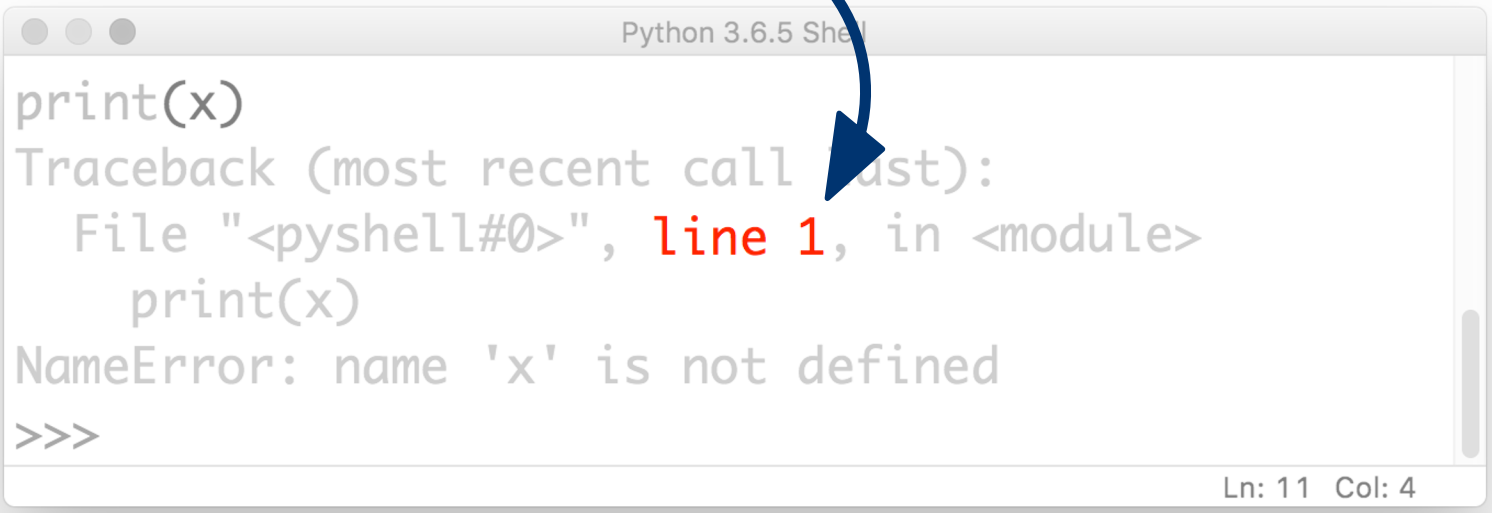
Lecture 10: some problems are obvious



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?

```
1   import math
2
3   def main():
4
5     x = int(input("Enter an integer greater than 0: "))
6
7     print("The log is:", math.log(x))
8
9     print("Have a nice day!")
10
11  if __name__ == "__main__":
12    main()
```

An example

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

```
1   import math
2
3   def main():
4
5     x = int(input("Enter an integer greater than 0: "))
6
7     if x < 0:
8       print("Cannot take the log of a negative number.")
9     else:
10      print("The log is:", math.log(x))
11
12    print("Have a nice day!")
13
14  if __name__ == "__main__":
15    main()
```

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 (what to **`try`**)
  - how to **handle** it if things go wrong (**`except`**)

An example


```
1   import math
2
3   def main():
4
5     try:
6       x = int(input("Enter an integer greater than 0: "))
7       print("The log is:", math.log(x))
8
9     except:
10      print("Sorry, that is not a valid input.")
11
12    print("Have a nice day!")
13
14  if __name__ == "__main__":
15    main()
```

An example

```
1   import math
2
3   def main():
4
5     try:
6       x = int(input("Enter an integer greater than 0: "))
7       print("The log is:", math.log(x))
8
9     except:
10      print("Sorry, that is not a valid input.")
11
12    print("Have a nice day!")
13
14  if __name__ == "__main__":
15    main()
```

except if you can't;
then do this instead

Your turn!

Given: a (brittle) solution to A2: Clunky Calculator

Objective: find any places that might throw **`Exceptions`**, and handle them so the program doesn't crash!

https://repl.it/@JordanCrouser/handling-exceptions-activity

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

Up next

- ✓Monday: Interaction
- ✓Debrief of Lab 10: Interactive Fish Tank
- ✓Wednesday: Handling Exceptions
- Friday: Special Topic – Ethical Issues in Tech
- FP2: Persona, Paper Prototype, and Architecture Diagram