

Lecture 23: Errors and Exceptions

CS 51P

December 5, 2022

Announcements

- Final Project due Friday 12/9 at 5pm
- Final Exam next Tuesday (12/13), 7-10pm
 - Will take place in Edmunds 101/114
 - Practice exams will be released on Slack tomorrow
- Lots of help available
 - Lots of office hours/mentor sessions this week
 - Review session TBA (this weekend)
 - One-on-one tutoring available through QSC

MY CODE DOESNT WORK



I GOT NO IDEA WHY

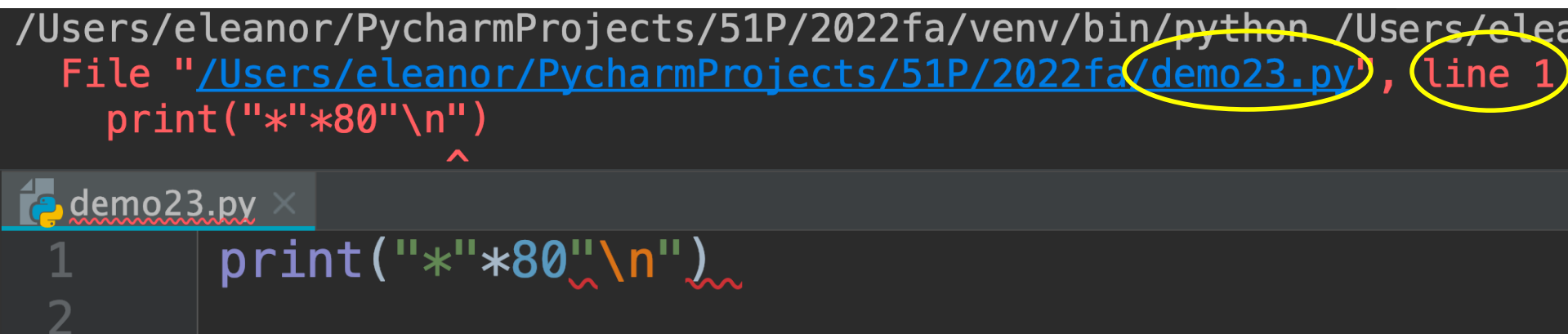
Common Types of Errors

- **Syntax Errors:** there is something wrong with the structure of the program, and Python doesn't understand it
- **Runtime Errors:** something goes wrong while the program is running (Exceptions)
- **Semantic Errors:** the program runs, but it doesn't do what you want it to do

Handling Syntax Errors

1. Find the bug

```
/Users/eleanor/PycharmProjects/51P/2022fa/venv/bin/python /Users/elea
File "/Users/eleanor/PycharmProjects/51P/2022fa/demo23.py", line 1
print("*"*80"\n")
      ^
```



```
demo23.py x
1 print("*"*80"\n")
2
```

2. Do you see the problem?

1. If yes, fix it!
2. If no, try running through the list of common syntax bugs
3. If still no, check your class notes, discuss the problem abstractly with a friend ("what's the right syntax for..."), or ask a TA/instructor (it's ok to get help!)

Common Syntax Errors

- Misspelling a variable name or a function name
- Missing quotation marks around a string
- Mismatched parentheses or quotation marks
- Missing a colon at the end of an if/while/for statement
- Using = instead of ==
- Using a Python keyword as a variable name

Make sure you remembered to save your file
after making your changes!

Example

```
in = int(input("Pick a number\n")) ← SyntaxError
if in = 13: ← SyntaxError
    print("I am also fond of the number 13!")
elif in > 13:
    print("I am fond of the number 13, which is "
          + str(in-13) + " less than " + str(in)) ← SyntaxError
else ← SyntaxError
    print("I am fond of the number 13, which is "
          + str(13-in) + " more than " + str(in)) ← SyntaxError
in2 = input("Do you like tea?) ← SyntaxError
while in2 != "yes" and != "no": ← SyntaxError
    in2 = input("Please answer yes or no. Do you like tea?")
if in2 == "yes":
    print("Great!")
else:
    print("That's too bad.")
print("Bye!") ← SyntaxError
```

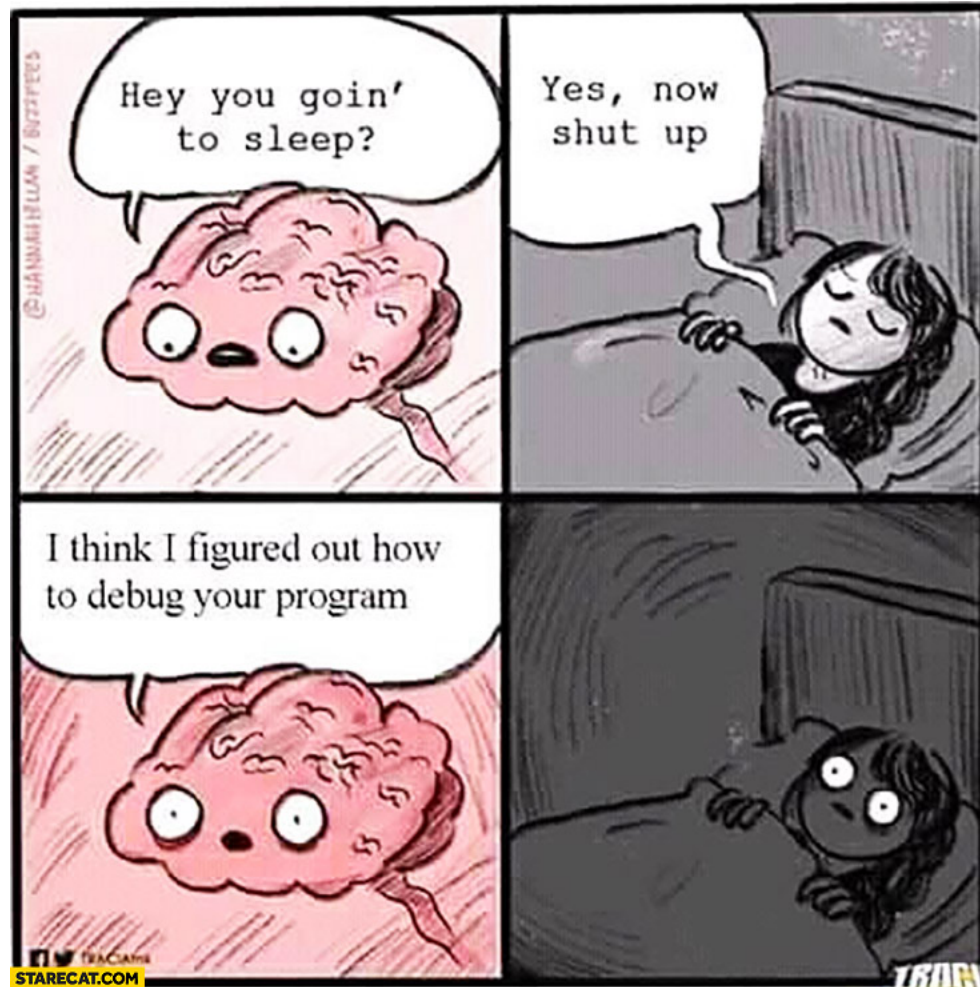
Can you find the
the **mistake**?

1 2 3 4 5 6 7 8 9

Handling Runtime Errors: Exceptions

- **NameError:** Python doesn't recognize a (variable) name
 - Find the bug!
 - Did you forget quotation marks around a string?
 - Did you misspell a variable name? Make a typo?
 - Is the variable you are trying to use in scope? Use before define?
- **TypeError:** Python can't perform that operation/function on that type
 - Find the bug!
 - Are the types that the error reports the type you expected?
 - Add a print statement on the previous line and print out all the variables/values on that line. Are they what you expect?
- **ValueError:** Python can't perform that operation/function on that value
 - Find the bug!
 - Add a print statement on the previous line and print out all the variables/values on that line. Are they what you expect?

Take a break



What if your code depends on inputs you don't control?

```
def example2(filename):  
    s = 0  
  
    file = open(filename, "r")  
    for i in file:  
        s = s + int(i)  
    file.close()  
  
    return s
```

- what if the file doesn't exist?
- what if it does exist but you don't have access permissions?
- what if the file exists and you can open it for reading, but it doesn't contain integers?

What if your code depends on inputs you don't control?

- Best answer: check whether input will work before using it
 - `if str.isdigit(user_in)`
 - `if k in dict.keys()`
- Alternate answer: try it and crash if an error occurs
- Alternate answer: try it and recover if an error occurs
 - Warning: very inefficient

Exception Handling

- A flexible mechanism for handling errors

```
try:  
    # code to execute  
except:  
    # what to do if there's an error
```

Example: Exception Handling

```
def exception_v0(filename):  
    s = 0  
  
    try:  
        file = open(filename, "r")  
        for i in file:  
            s = s + int(i)  
        file.close()  
    except:  
        print("An error occurred")  
        s = -1  
  
    print(s)
```

Exercise

- Write a function `return_int` that asks the user to enter an integer. If the user enters an integer, the function returns that integer. If the user does not enter an integer, the function returns 0.
- Use exceptions to handle the case where the user does not enter an integer. Do not use an if statement.

Exceptions

- A flexible mechanism for handling errors

```
try:
    try:
        # code to execute
    except <Error1>:
        # what to do if Error1 occurs
    except <Error2>:
        # what to do if Error 2 occurs
```



```
def exception_v1(filename):
    s = 0

    try:
        file = open(filename, "r")
        for i in file:
            s = s + int(i)

        file.close()
    except IOError:
        print("problem opening file")
        s = -1
    except ValueError:
        print("problem with non-integer")
        file.close()
        s = -1

    print(s)
```

Exceptions

- A flexible mechanism for handling errors

```
try:
    try:
        # code to execute
    except <Error1>:
        # what to do if Error1 occurs
    except <Error2>:
        # what to do if Error 2 occurs
    else:
        # additional code if no errors
```

```
def exception_v2(filename):
    s = 0
    try:
        file = open(filename, "r")
    except IOError:
        print("problem opening file")
        s = -1
    else:
        try:
            for i in file:
                s = s + int(i)
        except ValueError:
            print("problem with non-integer")
            s = -1
        file.close()

    print(s)
```

```
def f(x,y):
    try:
        try:
            for i in x:
                print(int(i))
            print(x[y])
        except ValueError:
            print(1)
        except TypeError:
            print(2)
        except IndexError:
            print(3)
        else:
            print(4)
        print(5)
    except:
        print(6)
```

- What happens when you evaluate `f("abc", "a")`?
- What happens when you evaluate `f([1, 2, 3], 4)`?
- What happens when you evaluate `f({"1": "2"}, 1)`?

Exercise

- Write a function `always_return_int` that repeatedly asks the user to enter an integer. Once the user enters an integer the function returns that integer.
- Use exceptions to handle the case where the user does not enter an integer. Do not use an if statement.

Debugging...

