Algorithms and Data Structures for Data Science

lab_debug

CS 277

Brad Solomon

September 10, 2021

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Learning Objectives

Practice identifying and correcting errors in code

Review understanding of Python fundamentals

Introduce some useful Python shorthand

You encounter an error…

```
Traceback (most recent call last):
  File “.../cs277/assignments/lab_debug/code/examples.py”, line 2, in <module>
    x += 5
NameError: name 'x' is not defined
```

You encounter an error without a clear cause

```
Traceback (most recent call last):
  File "../cs277/assignments/mp_racing/code/wrong_main.py", line 8, in <module>
    rb = racingBot(fName)
  File "../cs277/assignments/mp_racing/code/wrong_racingBot.py", line 21, in
__init__
    self.processHidden(inFile)
  File "../cs277/assignments/mp_racing/code/wrong_racingBot.py", line 245, in
processHidden
    with open(hiddenFile) as myFile:
FileNotFoundError: [Errno 2] No such file or directory: '../autograder/tests/
data/track1_obj.txt'
```

1. Read the error message!

Error messages will tell you where the error was discovered

Python error messages will often give key information

NameError: Problem with a variable

TypeError: Problem with a variable's type

IndentationError: Problem with whitespace in code

AttributeError: Object doesn't have a variable or function being called

When in doubt — Google is your friend!

Not every error has an error message!

```
1
2  def getReverseEvens(n):
3      outList = []
4      for i in range(n, 0, -2):
5          outList.append(i)
6
7  n = 4
8  ans = [4, 2, 0]
9
10 if (getReverseEvens(n)==ans):
11     print("Correct!")
12 else:
13     print("Incorrect!")
14
15
16
17
18
19
20
21
22
23
```

1. Understand the System

getGrade()

```
bool getGrade(int score)
# INPUT:
# score is a number representing a student's score from 0 to 1000 (int)
# OUTPUT:
# A single character with the letter grade based on their score (str)
```

```
def getGrade(score):
    # ranges based on https://courses.grainger.illinois.edu/cs277/fa2021/policy/syllabus/
    if score > 600:
        return "D"
    elif score > 700:
        return "C"
    elif score > 800:
        print("B")
    elif score > 900:
        return "A"
    else:
        return "F"
```

```
print("Brad got 799 points and got a ", getGrade(799))
print("Harsh got 800 points and got a", getGrade(800))
print()
```

2. Make it Fail

Identify what settings, values, or steps led to the failure-state

Make sure you can precisely repeat those steps to cause a failure!

If the failure is intermittent, find the uncontrolled cause.

3. Quit Thinking and Look

```
1
2  def getReverseEvens(n):
3      outList = []
4      for i in range(n, -1, -2):
5          outList.append(i)
6
7  n = 4
8  ans = [4, 2, 0]
9
10 if (getReverseEvens(n)==ans):
11     print("Correct!")
12 else:
13     print("Incorrect!")
14
15
16
17
18
19
20
21
22
23
```

print()

break / return

3. Quit Thinking and Look

```
breakpoint()
```

https://realpython.com/lessons/getting-started-pdb/

4. Divide and Conquer

Solve one error at a time

Determine its location in the workflow

Identify the code or interaction which is causing a problem

5. Change One Thing at a Time

Identify key factors but adjust them one at a time

Test each change as you make them

Compare a ‘bad’ result with a good result to figure out the difference

6. Keep an Audit Trail

Write down the details — great for office hours!

Git commit early and often (with messages!)

7. Check the Plug

Question your assumptions

Start at the beginning

Test your testing

8. Get a Fresh View

Go to office hours

Talk to your lab partner or class peer

Post on Campuswire

Coding the lab

1) Treat each function as its own independent problem

2) Identify what the function should be doing

3) Correct any errors that are preventing the code from running

4) Correct any errors where the function output is wrong

5) Be aware of edge cases and test your solution thoroughly!