Debugging & Errors
"Why you were up till 2AM"

Dan Fleck

Fall 2008

Coming up: Running programs from the command line…

Running programs from the command line…

- In windows just double-click on the "sliders.py" file
- In Unix/Mac (or Windows also) run the command:
  - python sliders.py
- This is needed to verify that the clear screen works.
- Sliders grading criteria posted to sliders assignment in Blackboard

Coming up: Debugging

Debugging

- Definition: The process of finding problems (bugs) in programs and removing them

Coming up: General Steps

General Steps

- ✓ Recognize that a bug exists
- ✓ Isolate the source of the bug
- ✓ Identify the cause of the bug
- ✓ Determine a fix for the bug
- ✓ Apply the fix and test it

Coming up: Isolating source of the bug

Isolating source of the bug

Once found, and understood, most bugs are easy to fix. We will concentrate only on finding the bug

Coming up: Confirm your beliefs

Confirm your beliefs

Finding your bug is a process of confirming the many things you believe are true, until you find one which is not true

- ✓ You believe that at a certain point in your source file, a certain variable has a certain value
- ✓ You believe that in a given if-then-else statement, the “else” part is the one that is executed
- ✓ You believe that when you call a certain function, the function receives its parameters correctly

Confirm your beliefs

So the processing of finding the location of a bug consists of confirming all these things!

Check everything!

Okay… How?

One way

- Put lots of print statements in at each point that says “I am in method X with parameters set to Y,Z”
- Note: If you do this, be able to turn them on/off easily!
  - DEBUG = True;
  - If (DEBUG) print…..

Coming up: Often a better way

Often a better way

- Use the debugger
  - Set a breakpoint at a point in the code
  - Step through the code
  - Look at the variables to see ones that are not as you expected

The debugger is usually better/faster… but print statements are also easy to turn on/off as a whole to make a DEBUG mode for your program. Which is often useful.

Coming up: The Most Important Thing

The Most Important Thing

- ✓ To do ANY debugging you must understand what the program should do
- ✓ What each method should do
- ✓ What each if statement should do

If your beliefs are wrong, confirming your beliefs will not help!

The Most Important Thing

- ✓ Add beliefs into your code… beliefs are otherwise known as COMMENTS!

Coming up: Terminology

Terminology

- Breakpoint – A marker in a program that signals the debugger to stop when execution reaches that point. Code beyond the breakpoint is not executed until further instructions are provided.
- Step through your code – execute your program a single line (or “step”) at a time stopping after each line to inspect values.

Coming up: Lets try out the IDLE debugger

Lets try out the IDLE debugger

- Startup IDLE
- Open a module
- Select DEBUG->Debugger
  - You should see [DEBUG ON] in the shell
- Run your program

More info: http://www.python.org/idle/doc/idle2.html#Debugger

Lets try out the IDLE debugger

- ✓ Next – go to the next line in the current function
- ✓ Step – go to the next line that executes (usually to step into a function)
- ✓ Out – run until the current function ends or a return statement is reached

IDLE does support breakpoints in the recent versions!
Right-click on your source vode to set a breakpoint!

Coming up: Lets work some examples

Lets work some examples

- ✓ See debugExamples.py
- ✓ See debugExamples2.py

Coming up: COMMENT YOUR CODE!

COMMENT YOUR CODE!

COMMENT! COMMENT! COMMENT! COMMENT!

COMMENT! COMMENT! COMMENT! COMMENT!

COMMENT! COMMENT!

COMMENT! COMMENT! COMMENT!

COMMENT! COMMENT! COMMENT! COMMENT!

COMMENT!

COMMENT! COMMENT! COMMENT!

COMMENT! COMMENT! COMMENT!

COMMENT!

COMMENT!

COMMENT! COMMENT! COMMENT!

COMMENT! COMMENT! COMMENT!

COMMENT! COMMENT! COMMENT!

COMMENT! COMMENT! COMMENT!

Coming up: Whoa there cowboy… I can’t even run my program!

Whoa there cowboy… I can't even run my program!

- First, try never to get here. Run early, run often. If you can't run at any point STOP… fix the code before adding any new code.
- Review the error, read it carefully
- Check your book and online resources for examples and see what may be wrong
- Ask questions on Blackboard, to your TA or Professor.

Coming up: The Most Important Thing (again)

The Most Important Thing (again)

- ✓ Add beliefs into your code… beliefs are otherwise known as COMMENTS!

Coming up: More Information

More Information

- List of Python debuggers:
  - http://wiki.python.org/moin/PythonDebuggers
- Other Python editors
  - http://wiki.python.org/moin/PythonEditors

Basic Error Types

- Syntax errors - incorrect syntax due to spelling errors, missing operators, etc. , (easiest to fix)
- Run- time errors - compiled/interpreted, but crashes when encounters certain data sets (harder to fix)
- Logic error - executes, but provides incorrect or inconsistent outputs (hardest)

Coming up: Syntax Error

Syntax Error

```
def main():
    test_str = "FAC50000BC4A01015CC01010
    for i in range(len(test_str)):
        if i % 8 == 0:
            print " " + test_str[i:i+4] + \
                  "   " + test_str[i+4:i+8]

main()
```

```
>>> import errors_1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/rheishma/GMU/Classes/CS-112/errors_1.py", line 3
    test_str = "FAC50000BC4A01015CC01010
                                       ^
SyntaxError: EOL while scanning single-quoted string
```

Syntax Error

```
def main():
   test_str = "FAC50000BC4A01015CC01010”
   for i in range(len(test_str)):
      if i % 8 == 0:
         print " ”  test_str[i:i+4] + \
                  "   " + test_str[i+4:i+8]

main()
```

```
>>> import errors_1
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
   File "/Users/rheishma/GMU/Classes/CS-112/errors_1.py", line 8
      print " " test_str[i:i+4] + \
                    ^
SyntaxError: invalid syntax
```

Syntax Error

```
def main():
   test_str = "FAC50000BC4A01015CC01010”
   fir i in range(len(test_str)):
      if i % 8 == 0:
         print " ”  + test_str[i:i+4] + \
                  "   " + test_str[i+4:i+8]

main()
```

```
>>> import errors_1
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
   File "/Users/rheishma/GMU/Classes/CS-112/errors_1.py", line 5
      fir i in range(len(test_str)):
        ^
SyntaxError: invalid syntax
```

Coming up: Run-time error

Run-time error

```
# Runtime Error Example
# runtimeError.py
def func1(name, num):

    print "%s has a favorite number: %d" %(name, int(num))

def main():
    name = raw_input("What is your name?")
    num = raw_input("What is your favorite number?")
    func1(name, num)

main()
```

Input: Dan, 5
Output: Dan has a favorite number: 5

Coming up: Run-time error

Run-time error

```
# Runtime Error Example
# runtimeError.py
def func1(name, num):

    print "%s has a favorite number: %d" %(name, int(num))

def main():
    name = raw_input("What is your name?")
    num = raw_input("What is your favorite number?")
    func1(name, num)

main()
```

Input: Dan, A2

```
Traceback (most recent call last):
File "/Users/dfleck/Documents/gmuwebsite/classes/cs112/spring08/
samplecode/runtimeError.py", line 6, in func1

    print "%s has a favorite number: %d" %(name, int(num))

ValueError: invalid literal for int() with base 10: 'A2'
```

Coming up: Logic Error

Logic Error

```
def add_nums(op_1,op_2):
        return op_1 - op_2

def main():
        x,y = input("Enter (#,#): ")
        z = add_nums(x,y)
        print z

main()
```

Enter (#,#): 10, 5

Output: 5

Coming up: References

References

- http://heather.cs.ucdavis.edu/~matloff/UnixAndC/CLanguage/Debug.html#tth_sEc2
- http://en.wikipedia.org/wiki/Debugging

End of presentation