

EGM722 – Programming for GIS and Remote Sensing

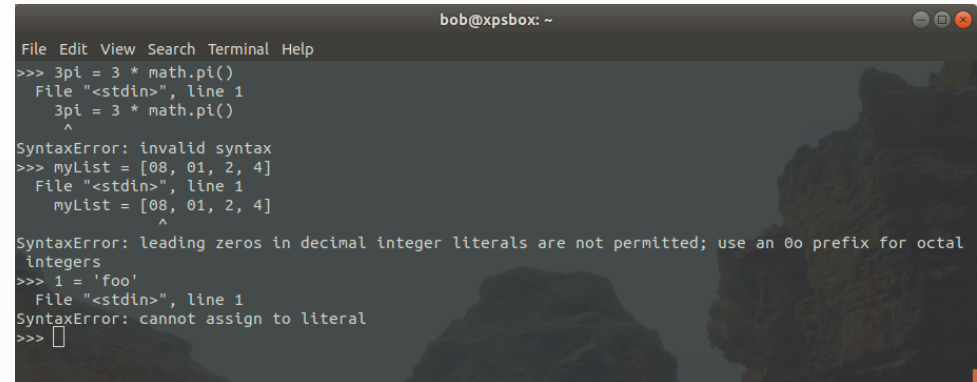
Week 2, Part 3: Errors and Debugging

When writing code, you **will** have errors

Bugs! Everywhere, bugs!

- Three main types of errors:
 - **Syntax** errors: produced when interpreter translates code
 - **Runtime** errors (**exceptions**): produced by interpreter if something goes wrong while the program is running
 - **Semantic** errors: no error message produced, but the program doesn't do what you expected
- **Debugging**: the process of identifying and fixing errors

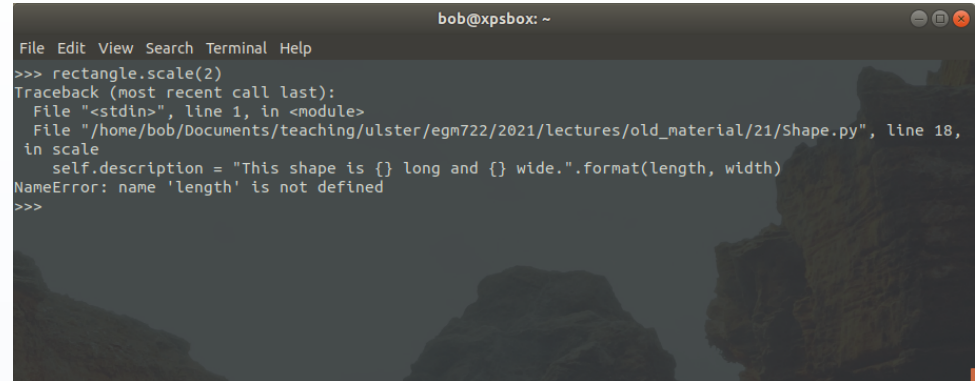
- Common messages:
 - Invalid syntax
 - Cannot assign to <something>
- Tips:
 - Avoid python keywords
 - Headers should end with colon
 - Check quotation marks, parentheses, brackets
 - Check indentation
- Most IDEs will alert you to syntax errors



```

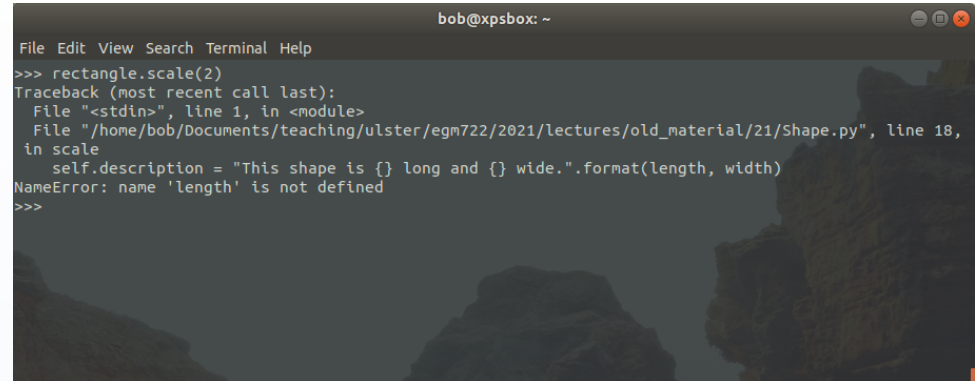
bob@xpsbox: ~
File Edit View Search Terminal Help
>>> 3pi = 3 * math.pi()
      File "<stdin>", line 1
      3pi = 3 * math.pi()
      ^
SyntaxError: invalid syntax
>>> myList = [08, 01, 2, 4]
      File "<stdin>", line 1
      myList = [08, 01, 2, 4]
              ^
SyntaxError: leading zeros in decimal integer literals are not permitted; use an 0o prefix for octal integers
>>> 1 = 'foo'
      File "<stdin>", line 1
      1 = 'foo'
      ^
SyntaxError: cannot assign to literal
>>> 
  
```

- Runtime error:
 - Program passes syntax checks (syntactically correct)
 - Fails during execution
- Also called **exceptions**
- Common causes:
 - Wrong variable names
 - Index/Key errors
 - Attribute errors



```
bob@xpsbox: ~  
File Edit View Search Terminal Help  
>>> rectangle.scale(2)  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
    File "/home/bob/Documents/teaching/ulster/egm722/2021/lectures/old_material/21/Shape.py", line 18,  
      in scale  
        self.description = "This shape is {} long and {} wide.".format(length, width)  
NameError: name 'length' is not defined  
>>>
```

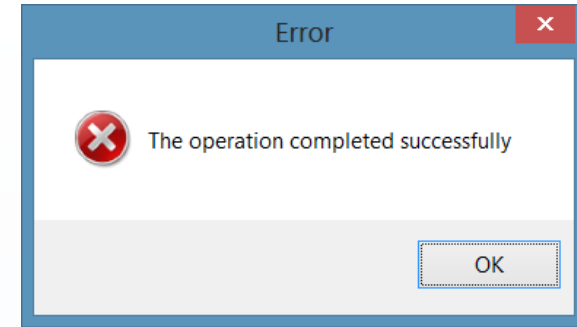
- Interpreter keeps track of function calls (**Traceback**)
- When an exception is raised, this prints to the screen
 - Provides function, line number, file where error occurred
 - Traces back through sequence of function calls
- Use this to examine where the error occurred



```

bob@xpsbox: ~
File Edit View Search Terminal Help
>>> rectangle.scale(2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "/home/bob/Documents/teaching/ulster/egn722/2021/lectures/old_material/21/Shape.py", line 18,
    in scale
        self.description = "This shape is {} long and {} wide.".format(length, width)
NameError: name 'length' is not defined
>>>
  
```

- Hardest errors to have to debug
- Program runs “successfully”
 - Interpreter provides no errors
 - Something isn’t right
- The issue: you did not write the program you wanted to write
- To debug, you have to work backward from the output



Being a Programmer

Mom said: "Please go to the shop and buy 1 bottle of milk. If they have eggs, bring 6"

I came back with 6 bottle of milk.

She said: "Why the hell did you buy 6 bottles of milk?"

I said: "BECAUSE THEY HAD EGGS"



WebDevelopersNotes.com

Out! Out, damned bug!

- Break complicated expressions into series of variables
- Use parentheses when unsure of order of operations
- Use a debugger to help pause the program
- Take a step back, grab a soda
- Call Joe.

- You will have to deal with errors/bugs
- You will have to deal with errors/bugs
- Different kinds of errors require different kinds of troubleshooting
- Debugging errors can require a fresh look
- You will have to deal with errors/bugs