Debugging in Python

Debugging in Python

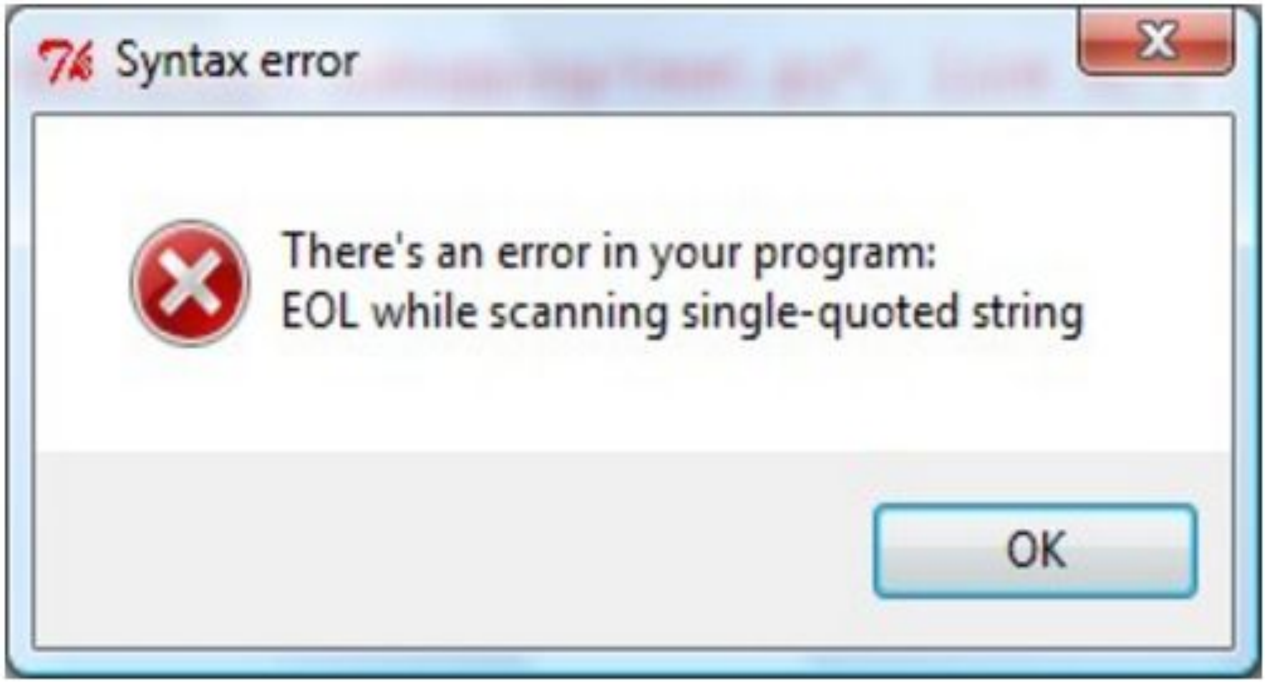
- **Syntax Errors:**
- These types of errors are usually **typing mistakes**, but more generally **it means that there is some problem with the structure of your program.**

- **Syntax errors** in Python will pop up a dialog box like the one below.

- The message in this box is *Syntax Error.*

- *There was an error in your program: EOL while scanning single-quoted string.*

print "hello world



```
Syntax error
There's an error in your program:
EOL while scanning single-quoted string
```

- EOL stands for ***End Of Line***.
- This error means that there was an open quote somewhere, but the line ended before a closing quote was found.

a = 3 + 5 7

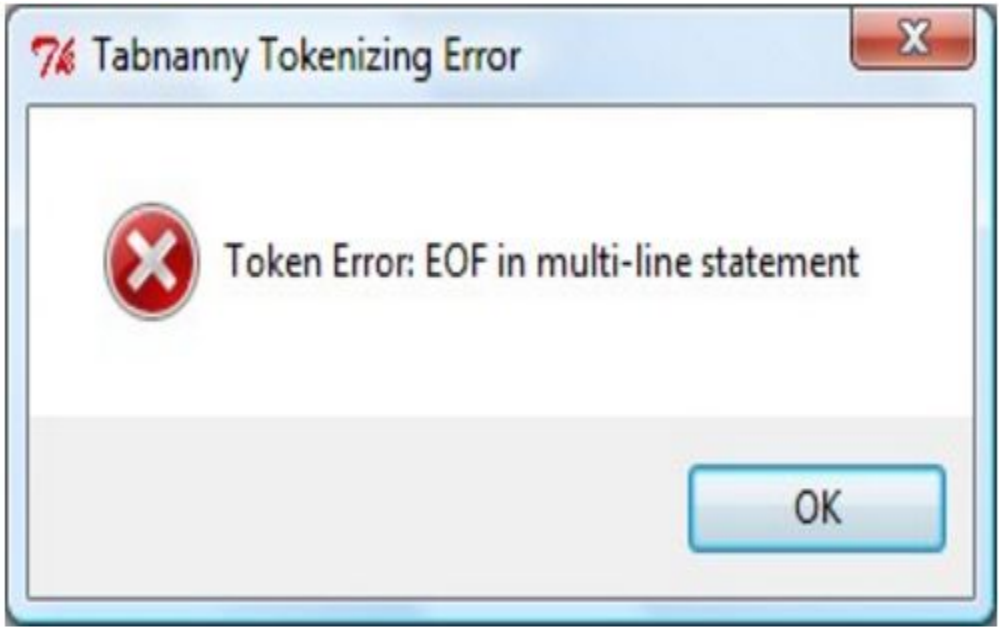
- Another type of syntax error will simply say ***invalid syntax***.
- **An invalid syntax error means that there is a line that python doesn't know what to do with.**
- The last common type of syntax error you will likely encounter has to do **with indention**.

Solution:

- When you press OK on the dialog box. **Python will attempt to highlight the offending line in your source code**.
- You should use this location as a hint for where to start looking for your problem.
- First check the area highlighted.
- Then check the entire line.
- Lastly, check the line or lines before the line highlighted.
- If you get an indention error, you should check that all of your lines of code are properly aligned in the correct columns.

Token Error (missing parenthesis a = 3 + (4 + 5

- Token errors in Python will pop up a dialog box like the one below.



Token Error: EOF in multi-line statement

- **This error usually means that there was an open parenthesis somewhere on a line, but not a matching closing parenthesis.**

- **Python reached the end of the file while looking for the closing parenthesis.**

Solution:

- When you press OK on the dialog box.

- Python will attempt to highlight the offending line in your source code.

- However, since it had reached the end of the file, it will highlight the last line in the file!

Runtime Errors

- **Runtime errors occur as your program executes.**

- Since Python is an interpreted language, these errors will not occur until the flow of control in your program reaches the line with the problem.

print hello

- Traceback (most recent call last): File "C:/Users/John/Documents/Teaching-BU/Python-debugging/test.py", line 7, in main() File
- "C:/Users/John/Documents/Teaching-BU/Python-debugging/test.py", line 7, in main print hello
- NameError: global name 'hello' is not defined

- **The first part tells you which file had the error.**
- In the example above, the **file is *test.py*** and **the error occurs on line 7**.
- **The next line shows the actual line of code where the error occurred.**
- This line executes the main() function.
- Similarly, the next two lines say that the error occurred on line 7, within main , and that the line with the error is print hello.
- Lastly, the actual **NameError** says that *global name* ***'hello'*** *is not defined*.

- **Usual Causes**:

- A mistyped variable or function name.
- Using a variable before it is defined.
- The name was intended to be enclosed in quotes.

Logic (semantic) errors

- Semantic or logic errors are problems with the design of your program.
- **These usually do not produce any error message, but instead cause your program to behave incorrectly.**
- These errors are often caused by accidentally using one variable in a place where a different variable is intended, or by simply doing some math incorrectly.

```
apples = 0
pickedApples = input("Pick some apples: ")
apples = apples + pickedApples

pickedApples = input("Pick some more apples: ")
apples = apples + pickedApples
print "You have a total of %d apples"
%pickedApples
```

- The above code will execute, but it will not output the total number of apples picked.
- Instead, it will output the amount that was picked the last time!
- This simple example is easy to fix, but in a more complicated program it can be difficult to find such problems.

Experimental Debugging

- debugging is one of the most challenging, and interesting parts of programming.
- Debugging is also like an experimental science.

- Once you have an idea what is going wrong, you modify your program and try again.

- If your hypothesis was correct, then you can predict the result of the modification, and you take a step closer to a working program.
- If your hypothesis was wrong, you have to come up with a new one.
- That is, programming is the process of gradually debugging a program until it does what you want.