

Programming errors are often called “bugs” and the process of tracking them down is often called “debugging”.

Correcting errors

In Python programming there are three types of error that can occur. It is useful to recognize the different error types so they can be corrected more easily:

- **Syntax Error** – occurs when the interpreter encounters code that does not conform to the Python language rules. For example, a missing quote mark around a string. The interpreter halts and reports the error without executing the program.
- **Runtime Error** – occurs during execution of the program, at the time when the program runs. For example, when a variable name is later mis-typed so the variable cannot be recognized. The interpreter runs the program but halts at the error and reports the nature of the error as an “Exception”.
- **Semantic Error** – occurs when the program performs unexpectedly. For example, when order precedence has not been specified in an expression. The interpreter runs the program and does not report an error.

Correcting syntax and runtime errors is fairly straightforward, as the interpreter reports where the error occurred or the nature of the error type, but semantic errors require code examination.

- 1 Launch a plain text editor then add a statement to output a string that omits a closing quote mark
`print('Python in easy steps)`
- 2 Save the file in your scripts directory then open a Command Prompt window there and run this program – to see the interpreter report the syntax error and indicate the position in the code where the error occurs



Typically, the syntax error indicator points to the next character after an omission in the code.

```

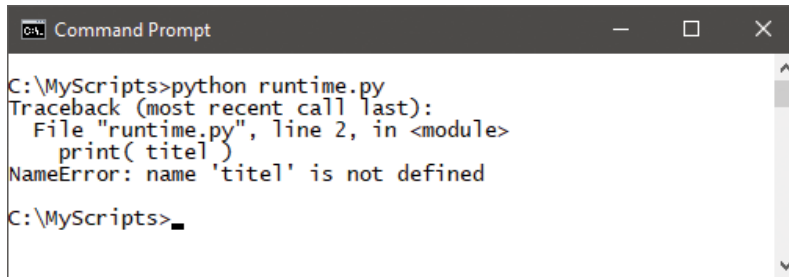
Command Prompt
C:\MyScripts>python syntax.py
File "syntax.py", line 1
  print( 'Python in easy steps )
                                     ^
SyntaxError: EOL while scanning string literal
C:\MyScripts>

```

...cont'd

- 3 Insert a quote mark before the closing parenthesis to terminate the string, then save the file and run the program again – to see the error has been corrected
- 4 Next, begin a new program by initializing a variable, then try to output its value with an incorrect variable name – to see the interpreter report a runtime error


```
title = 'Python in easy steps'
print( titel )
```



```

C:\MyScripts>python runtime.py
Traceback (most recent call last):
  File "runtime.py", line 2, in <module>
    print( titel )
NameError: name 'titel' is not defined
C:\MyScripts>

```

- 5 Amend the variable name to match that in the variable declaration, then save the file and run the program again – to see the error has been corrected
- 6 Now, begin a new program by initializing a variable, then try to output an expression using its value without explicit precedence – to see a possibly unexpected result of 28


```
num = 3
print( num * 8 + 4 )
```



```

C:\MyScripts>python semantic.py
28
C:\MyScripts>

```

- 7 Add parentheses to group the expression as `3 * (8 + 4)`, then save the file and run the program again – to see the expected result of 36, correcting the semantic error