

## 2.13: DEBUGGING



Contributed by [Chuck Severance](#)  
Clinical Associate Professor (School of Information) at [University of Michigan](#)

At this point, the syntax error you are most likely to make is an illegal variable name, like `class` and `yield`, which are keywords, or `odd~job` and `US$`, which contain illegal characters.

If you put a space in a variable name, Python thinks it is two operands without an operator:

```
>>> bad name = 5
SyntaxError: invalid syntax
```

```
>>> month = 09
      File "<stdin>", line 1
        month = 09
                ^
SyntaxError: invalid token
```

For syntax errors, the error messages don't help much. The most common messages are `SyntaxError: invalid syntax` and `SyntaxError: invalid token`, neither of which is very informative.

The runtime error you are most likely to make is a "use before def;" that is, trying to use a variable before you have assigned a value. This can happen if you spell a variable name wrong:

```
>>> principal = 327.68
>>> interest = principle * rate
NameError: name 'principle' is not defined
```

Variables names are case sensitive, so `LaTeX` is not the same as `latex`.

At this point, the most likely cause of a semantic error is the order of operations. For example, to evaluate  $1/2\pi$ , you might be tempted to write

```
>>> 1.0 / 2.0 * pi
```

But the division happens first, so you would get  $\pi/2$ , which is not the same thing! There is no way for Python to know what you meant to write, so in this case you don't get an error message; you just get the wrong answer.