**DEBUGGING:**

**Possible errors with Variables:**

1. Variable names can contain both letters and numbers. But they cannot begin with numbers.
2. Both upper case and lower-case letters can be used.
3. Variable names are case sensitive. For example, midname, midName, MidName are different variables.
4. Variable name cannot be any one of the keywords.
5. Variable names cannot have space in between.

**i) Syntax Error:**

The text of the program does not comply with the rules of the language is known as a syntax error. 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 syntax error you are most likely to make is:

1. An illegal variable name, like class and yield, which are keywords.
2. Variable names such as odd~job and US$, which contain illegal characters.
3. If you put a space in a variable name, Python thinks it is two operands without an operator:

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

**ii) Runtime Error:**

The error which occurs when the program is running is known as runtime error. The runtime error you are most likely to make is

1. "use before def;" that is, trying to use a variable before you have assigned a value.
2. Runtime error can also happen if you spell a variable name wrong:

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

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

**Possible errors with Statements:**

**Semantic error:**

A semantic error occurs when a statement is syntactically valid, but does not do what the programmer wants. The semantic error you are most likely to make is,

To evaluate 1 / 2π, you might be tempted to write

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

But the division happens first, so you would get π/2, which is not the same thing! So, in this case you don't get an error message; you just get the wrong answer. To get right answer, use

```
>>> 1.0 / ( 2.0 * pi )
```