Types of Errors

Syntax Errors

The compiler or interpreter can only read a program if the program is syntactically correct; otherwise, the process fails and returns an error message. Syntax refers to the structure of a program and the rules about that structure.

For example, in English, a sentence must begin with a capital letter and end with a period. You could also think of these as spelling errors.

```
>>> 3) + 2 * 4
SyntaxError: invalid syntax

>>> 12 = x
SyntaxError: can't assign to literal
```

Exceptions (Runtime and Semantic Errors)

Next, we look at errors that produce an exception. They may result from improper use of the statements or variables. In this case the syntax of the programming language is correct but the problem could result from the use of wrong variables, wrong operations or in the wrong order. These errors may also result from applying an operator not intended for the variable type, calling a function with the wrong number of arguments or with arguments of the wrong type, etc. These errors are exceptions because they usually indicate that something exceptional (and bad) has happened, and usually begin with *Traceback*.

```
>>> 89.4 / 0
Traceback (most recent call last):
  File "", line 1, in
    89.4 / 0
ZeroDivisionError: float division by zero
```

Logical errors

A logical error is when the wrong output is produced due to a miscalculation or misunderstanding of the requirements. These type of errors are generally the most difficult to fix because the code will execute without crashing. There are no error messages produced. Identifying logical errors can be tricky because it requires you to work backward by looking at the output of the program and trying to figure out what it is doing. Another method of debugging logical errors is to step through the program one instruction at a time to figure out where things go wrong.

```
>>> fahrenheit = 71.6
>>> celsius = fahrenheit - 32 * 5/9
>>> celsius
53.822222222222216
```

The above example is missing brackets, should be:

```
celsius = (fahrenheit - 32) * 5/9
```

Some more examples of logical errors are:

- using the wrong variable name
- indentation
- using integer division instead of floating-point division
- getting operator precedence wrong

As we go through the different topics of basic programming we will highlight some of the common mistakes made by beginners, for now just try to understand the definitions.

Testing Repeatedly

You should test your code **as you write it**. The reason is that the more you write the more likely it is that you will make a mistake. If you have not been testing and fixing your code along the way, at some point it will become analogous to finding a needle in a haystack.

Readability and Comments

Making it obvious what is happening in your code is a good thing.
Writing code is as much a communication exercise as it is a calculation exercise.

Readability

Much like we use spaces in the English language to separate words in a sentence to make them easier to read, so too, we should use spaces to separate our operators. For example:

```
>>> -100-25*3%4
>>> 3+2+1-5+4%2-1/4+6
```

Would be a lot easier to read if we included spaces.

```
>>> -100 - 25 * 3 % 4
>>> 3 + 2 + 1 - 5 + 4 % 2 - 1 / 4 + 6
```

For ambiguous operations it would be helpful to include parentheses to remove any doubts.

```
>>> -100 - (25 * 3) % 4
>>> 3 + 2 + 1 - 5 + (4 % 2) - (1 / 4) + 6
```

It is better to have more spaces and lines if it makes your code easier to read.

The Why and How of Comments

As your programs get longer and more complicated, some additional English explanation can be used to help you and other programmers read your code. These explanations called *comments* document your code (much the way *docstrings* document your functions – more on that later).

A comment begins with the number sign character (#) and goes until the end of the line. One name for this character is the hash character. Python ignores any lines that start with this character.

Comments are intended for programmers to read, and are usually there to explain the purpose of a function, as well as to describe relationships between your variables. Comments are to help you, and anyone else who is reading/using your code, to remember or understand the purpose of a given variable or function in a program.

Example:

```
fahrenheit = 212
celsius = (fahrenheit - 32) * 5 / 9          # Convert degrees
Fahrenheit to Celsius.

# calculate the area of a triangle
base = 20
height = 12
area = base * height / 2
```

- Analyze the **Problem**
- Determine **Specifications**
- *Refine the* ~~Create a~~ **Design**
- **Implement**
- Test & Debug

iterate many times