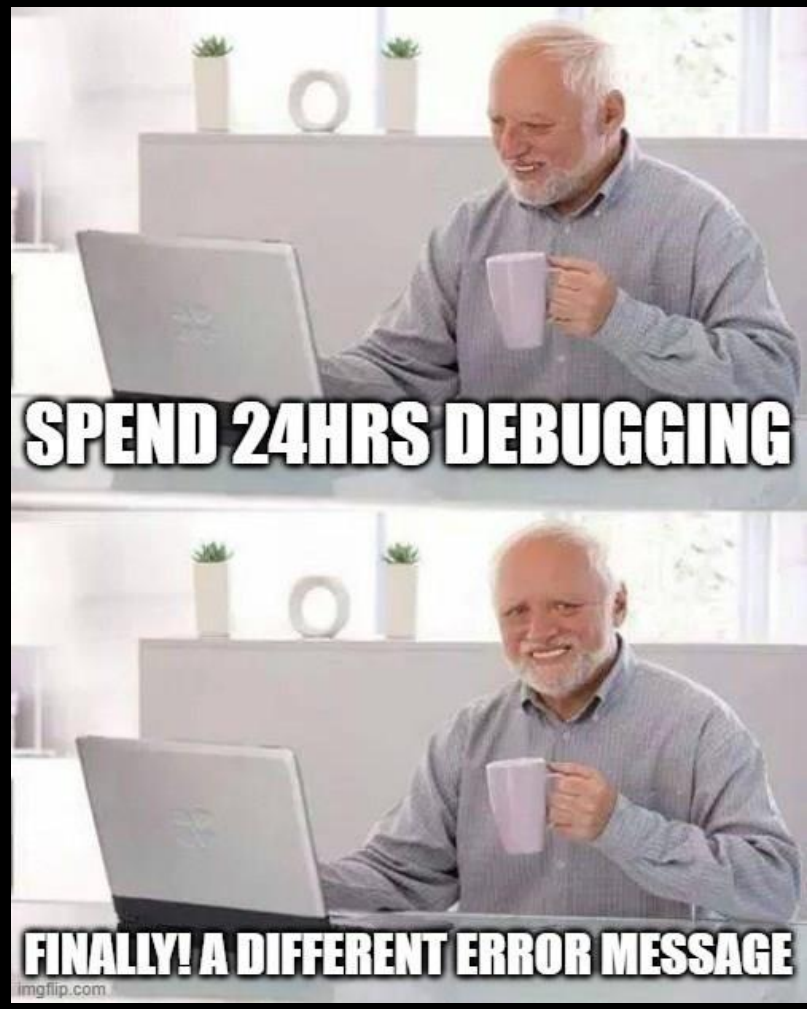
Testing!

- The more lines of code you write, the more likely it is that you will make a mistake and the harder it will be to find the mistake
  - "like finding a needle in a haystack"
- Test your code as you write it
  - Requires you understanding what specific output an input will provide
- "Modular code"
  - Test in small chunks or "modules"
  - Put a test input into the beginning where you know what the output is and see what you get!

Golden Rule: Never spend more than 15 minutes programming without testing

Error Reduction vs Debugging

- It is pretty much impossible to write code without errors.
  - *Error Reduction*: techniques we can use to reduce the number and severity of errors.
  - *Debugging*: techniques for identifying and correcting errors



Types of Errors

- Syntax error
- Semantic error
- Logical error
- Runtime error

Syntax Errors

- *Syntax error*: results when the programming language cannot understand your code.
- Examples: missing an operator or two operators in a row, illegal character in a variable name, missing a parentheses or bracket etc.
- In English, a syntax error is like a spelling error

```
>>> 3) + 2 * 4
Syntax Error: unmatched ')':  line 1, pos 2
```

Semantic Errors

- *Semantic error*: results from improper use of the statements or variables.
- Examples: using an operator not intended for the variable type, calling a function with the wrong argument type, or wrong number of arguments, etc.
- In English, a semantic error is like a grammar error

```
>>> “Hello” - 4
TypeError: unsupported operand type(s) for -: 'str' and 'int’
>>> number = number * 2
NameError: name 'number' is not defined
```

Runtime Errors

- *Runtime error*: is an error that occurs during the execution (runtime) of a program. Generally do not occur in simple programs.
- The code could run fine most of the time, but in certain circumstances the program may encounter an unexpected error and crash.
- Examples: infinite loops, attempting to access an index out of bounds, etc.

```
>>> x = 10
>>> while x > 0:
        print("This is the song that never ends")
```

Logical Errors

- *Logical Error*: results from unintended result due to a miscalculation or misunderstanding of specifications.
- Examples: miscalculation, typo, misunderstanding of requirements, indentation mistakes, operator precedence, integer instead of floating-point division, etc.
- **Most difficult to fix** because the code will execute without crashing. There are no error messages produced.

Logical Error Examples

71.6 degrees F is about 22 degrees C

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

Correct logic: celsius = (fahrenheit – 32) * 5/9

```
>>> fahrenheit = 716
>>> celsius  = (fahrenheit – 32) * 5/9
>>> Celsius
380.0
```

Whoops, typo! Forgot the decimal.