Exception Handling: Takeaways

Syntax

- Handling an exception using a try-except block:

```
try:
    impossible_value = int("Not an integer")
except ValueError:
    print("Cannot convert string to integer")
```

- Catching multiple types of exceptions:

```
try:
    f = open("data.txt", "r")
    s = f.readline()
    i = float(s)
except ValueError:
    print("Cannot convert data to floating point value")
except IOError:
    print("Could not read file")
```

Concepts

- Errors can be quite useful to us because they tell us what went wrong with our code.
- Exception handling comes into play when we want to handle errors gracefully so our program doesn't crash.
- An exception is a broad characterization of what can go wrong with a program. Exceptions occur during the execution of a program whereas syntax errors will cause your code not to run at all.
- In a `try-except` block, Python will attempt to execute the try section of the statement. If Python raises an exception, the code in the `except` statement will execute.

- While you have the ability to catch any exception without specifying a particular error in the `except:` section, not specifying an error is sometimes dangerous as you won't be able to execute exception-specific logic.

Resources

- Why a try-except block is useful in Python
- Errors and Exceptions