try, except, or else!

WE'LL COVER THE FOLLOWING

- Wrapping Up

The **try/except** statement also has an **else** clause. The **else** will only run if there are no errors raised. We will spend a few moments looking at a couple examples:

```
my_dict = {"a":1, "b":2, "c":3}

try:
    value = my_dict["a"]
except KeyError:
    print("A KeyError occurred!")
else:
    print("No error occurred!")
```

Here we have a dictionary with 3 elements and in the **try/except** we access a key that exists. This works, so the **KeyError** is **not** raised. Because there is no error, the **else** executes and “No error occurred!” is printed to the screen. Now let’s add in the **finally** statement:

```
my_dict = {"a":1, "b":2, "c":3}

try:
    value = my_dict["a"]
except KeyError:
    print("A KeyError occurred!")
else:
    print("No error occurred!")
finally:
    print("The finally statement ran!")
```

If you run this example, it will execute the **else** and **finally** statements. Most of the time, you won't see the **else** statement used as any code that follows a **try**/**except** will be executed if no errors were raised. The only good usage of the **else** statement that I've seen mentioned is where you want to execute a **second** piece of code that can **also** raise an error. Of course, if an error is raised in the **else**, then it won't get caught.

Wrapping Up

Now you should be able to handle exceptions in your code. If you find your code raising an exception, you will know how to wrap it in such a way that you can catch the error and exit gracefully or continue without interruption.

Now we're ready to move on and learn about how to work with files in Python.