Exception Introduction

Why is this important?

Exceptions are a way to handle errors in your code. You're probably familiar with errors in your code because they happen all the time.

You've probably seen errors like `NameError`, `TypeError`, `ValueError`, `SyntaxError`, etc. These are all examples of exceptions. Exceptions are a way to handle errors in your code so that your program doesn't crash.

Many times we don't want to crash our program when an error occurs. We want to handle the error and move on.

We're going to use `try except` blocks to handle exceptions in our code, so that we can handle the error and move on.

What are we going to do?

We're going to take a look at the edge cases of our golf score calculator and handle the errors that occur.

1. Let's make `golf_score.py` crash and observe the error.

- Enter "sixty-two" as the first score and let's see what the output is going to be.

```
$ python golf_scores.py
Golf Score Calculator
What was your most recent golf score? (enter 'quit' to stop) sixty-two
Traceback (most recent call last):
  File "C:\Users\dmouris\sdev1001-master-course\looping-and-
exceptions\exceptions_intro_start\golf_scores.py", line 11, in <module>
    total_score += int(user_input)
                   ^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'sixty-two'
```

- You can see that we get a `ValueError` because we're trying to convert the string "sixty-two" to an integer. This is not possible because "sixty-two" is not a number.
    - We can also see that this is happening on `line 11` of our code where we're trying to convert the user input to an integer.

2. Let's handle this error so that our program doesn't crash.

- We're going to add a `try except` block around the code that is causing the error on `line 11`.
- Let's take a look at the code.

```
print("Golf Score Calculator")
count = 0
```

```
total_score = 0
# removed active variable

while True:
    user_input = input("What was your most recent golf score? (enter
'quit' to stop) ")
    if user_input == 'quit': # quit if the user enters 'quit'
        break
    else: # add the score to the total and increment the count
        try:
            total_score += int(user_input)
            count += 1
        except ValueError as error_message:
            print(F"Could not convert input to a number. {error_message}")

# use the average.
average = total_score / count
print(f"Your average golf score is er{average}.")
```

- You can see here that we're using a `try except` block to handle the `ValueError` that is occurring on `line 11`.
    - We're also printing out the error message so that we can see what the error is.
    - When using `try except` blocks you can also use the `as` keyword to assign the error message to a variable. In this case we're assigning the error message to a variable named `error_message`.
- You want to keep the code that's inside of the `try` block as small as possible. This is because you want to only handle the error that is occurring in that block of code.
    - In this case we're only handling the error that is occurring on `line 11` of our code.
- Let's take a look at the output below of our program

```
$ python golf_scores.py
Golf Score Calculator
What was your most recent golf score? (enter 'quit' to stop) sixty-two
Could not convert input to a number. invalid literal for int() with base
10: 'sixty-two'
What was your most recent golf score? (enter 'quit' to stop) 87
What was your most recent golf score? (enter 'quit' to stop) 38
What was your most recent golf score? (enter 'quit' to stop) something
else
Could not convert input to a number. invalid literal for int() with base
10: 'something else'
What was your most recent golf score? (enter 'quit' to stop) 76
What was your most recent golf score? (enter 'quit' to stop) 99
What was your most recent golf score? (enter 'quit' to stop) quit
Your average golf score is 75.0.
```

- You can see now that eve though we entered "sixty-two" and "something else" our program didn't crash. It handled the error and moved on.

3. Let's handle another error that could occur, we don't put any scores in.

- Let's take a look at what happens when we don't put any scores in.

```
$ python golf_scores.py
Golf Score Calculator
What was your most recent golf score? (enter 'quit' to stop) quit
Traceback (most recent call last):
  File "C:\Users\dmouris\sdev1001-master-course\looping-and-
exceptions\exceptions_intro_start\golf_scores.py", line 19, in <module>
    average = total_score / count
              ~~~~~~~~~~~~^~~~~~~
ZeroDivisionError: division by zero
```

- You can see that we get a `ZeroDivisionError` because we're trying to divide by zero. This is because we didn't put any scores in so our `count` variable is still 0.
- This should not make our program crash! Let's add a `try except` block around this code handling the `ZeroDivisionError`.

```
# other code removed for brevity

try:
    average = total_score / count
    print(f"Your average golf score is {average}.")
except ZeroDivisionError:
    print("You didn't enter any scores!")
```

- You can see that we're handling the `ZeroDivisionError` that is occurring on `line 19` of our code, and you can see that we're not using an error message because we know what's going on with this error.

- Let's take a look at the output of our program now.

```
$ python golf_scores.py
Golf Score Calculator
What was your most recent golf score? (enter 'quit' to stop)
Could not convert input to a number. invalid literal for int() with base
10: ''
What was your most recent golf score? (enter 'quit' to stop) quit
You didn't enter any scores!
```

- Great! You can see that we didn't enter a score properly the first time and handled the `ValueError` error and notified the user. Then we quit without entering a score and we handled that

`ZeroDivisionError` error as well.

Conclusion

Handling exceptions is a tool that you can use to make your programs more robust. You can handle exceptions so that your program doesn't crash and you can notify the user of what's going on.

This is a technique that you'll use quite often when trying to connect or interface with some external system. We'll learn more about this in the future.