Video 21 : Custom Exceptions

I cover exception handling in a previous video, but this time I'll show you how to create custom exceptions. I'll also cover how to use finally and else with exceptions.

Exceptions are triggered either when an error occurs or when you want them to.

We know exceptions are used to handle errors, execute specific code when code generates something out of the ordinary, or to always execute code when something happens (close a file that was opened),

When an error occurs you stop executing code and jump to execute other code that responds to that error.

In this example let's handle an IndexError exception that is triggered when you try to access an index in a list that doesn't exist.

CODE

```
# Surround a potential exception with try
try:
    a_list = [1, 2, 3]

    print(a_list[3])

# Catch the exception with except followed by the
# exception you want to catch

# You can catch multiple exceptions by separating them
# with commas inside parentheses
# except (IndexError, NameError):
except IndexError:
    print("Sorry that index doesn't exist")

# If the exception wasn't caught above this will
# catch all others
except:
    print("An unknown error occurred")
```

Custom Exceptions

Now let's create a custom exception. Lets trigger an exception if the user enters a name that contains a number. Although you won't commonly create your own exceptions
this is how you do it.

CODE

```
# Create a class that inherits from Exception
class DogNameError(Exception):

    def __init__(self, *args, **kwargs):
        Exception.__init__(self, *args, **kwargs)
```

```
try:
    dog_name = input("What is your dogs name : ")

    if any(char.isdigit() for char in dog_name):

        # Raise your own exception
        # You can raise the built in exceptions as well
        raise DogNameError

except DogNameError:
    print("Your dogs name can't contain a number")
```

Finally & Else

Finally is used when you always want certain code to execute whether an exception is raised or not. Else is only executed if no exception was raised.

CODE

```
num1, num2 = input("Enter to values to divide : ").split()

try:
    quotient = int(num1) / int(num2)
    print("{} / {} = {}".format(num1, num2, quotient))

except ZeroDivisionError:
    print("You can't divide by zero")

# else is only executed if no exception was raised
else:
    print("You didn't raise an exception")

finally:
    print("I execute no matter what")
```

Problem for you to Solve

1. Create a file named mydata2.txt and put data in it
2. Using what you learned in part 8 and Google to find out how to open a file without with try to open the file in a try block
3. Catch the FileNotFoundError exception
4. In else print the file contents
5. In finally close the file
6. Try to open the nonexistent file mydata3.txt and test to see if you caught the exception

SOLUTION

```
try:
    my_file = open("mydata2.txt", encoding="utf-8")

# We can use as to access data and methods in the
# exception class
except FileNotFoundError as ex:
```

```
    print("That file was not found")

    # Print out further data on the exception
    print(ex.args)

else:
    print("File :", my_file.read())
    my_file.close()

finally:
    print("Finished Working with File")
```

That's all for now. In the next video I'll show how to treat functions as objects, cover function annotations and provide a problem for you to solve.