Try and Except

What happens when an error occurs?

- Code breaks and exits

```
a = 4
print("Your score is" + 4)
```

```
Traceback (most recent call last):
  File "/Users/katherinehansen/Chapman/Fall2023/CPSC230/Exception.py
", line 3, in <module>
    print("Your score is" + 4)
          ~~~~~~~~~~~~~~~~~~^~~
TypeError: can only concatenate str (not "int") to str
```

How can we prevent the code from stopping?

- Try and except blocks!

Syntax

```
try:
    # some code that you hope works
except:
    # code that will run if your above code causes an error
```

How can we prevent the code from stopping?

```
try:
    a = 4
    print("Your score is" + 4)
except:
    print("An error has happened")
```

```
MacBook-Pro-114:CPSC230 katherinehansen$ python3 Exception.py
An error has happened
MacBook-Pro-114:CPSC230 katherinehansen$
```

- Notice in the previous slide that the code did not break - it just executed the code in the except block instead!

Try/except + while loops

- We can use try/except blocks as a way to do error checking for user input
- Instead of code breaking when the user enters something incorrectly, we will just reprompt them

Code

```
while True:
    try: # code that we hope executes
        user_num = input("Please enter an integer: ")
        user_num = int(user_num)
        # ^ this line will cause an error if the user input is not an int
        print(user_num)
        break
    except:
        print("Uh oh! That wasn't an int. Try again.")
```

Terminal Output

```
MacBook-Pro-114:CPSC230 katherinehansen$ python3 Exception.py
Please enter an integer: t
Uh oh! That wasn't an int. Try again.
Please enter an integer: m
Uh oh! That wasn't an int. Try again.
Please enter an integer: 0.3
Uh oh! That wasn't an int. Try again.
Please enter an integer: 5
5
```

Multiple Exceptions

- We can specify that we want something specific to happen based on the kind of error that occurs
- Example: Write some code that prompts the user to enter a number and then print the result of 10/num
- There are two types of errors that can occur - a ValueError if the user enters a non int or a ZeroDivisionError if the user enters a 0

Code

```
while True:
    try: # code that we hope executes
        user_num = input("Please enter a non-zero integer: ")
        user_num = int(user_num)
        # ^ this line will cause an error if the user input is not an int
        print(10/user_num)
        break
    except ValueError:
        print("Uh oh! That wasn't an int. Try again.")
    except ZeroDivisionError:
        print("Uh oh! We can't divide by 0. Try again.")
```

Terminal Output

```
MacBook-Pro-114:CPSC230 katherinehansen$ python3 Exception.py
Please enter a non-zero integer: t
Uh oh! That wasn't an int. Try again.
Please enter a non-zero integer: 0
Uh oh! We can't divide by 0. Try again.
Please enter a non-zero integer: 9
1.1111111111111112
```

Raising Errors

- We can raise our own errors with personalized messages if we know we may be expecting errors
- Example: Write a function that asks the user for a direction. If the user enters an invalid direction, raise a ValueError

Code

```
def direction_getter():
    print("Hello! You will choose a direction to go")
    direction = input("Would you like to go N/S/E/W? ")
    if direction.upper() not in ["N", "S", "E", "W"]:
        raise ValueError("That is not an acceptable direction...")
    else:
        return direction

d = direction_getter()
```

Raising Errors

- exception is a variable of type string that holds the personalized message we made above
- So in this case, exception = “That is not an acceptable direction…”

Terminal Output w/ no error

```
MacBook-Pro-114:CPSC230 katherinehansen$ python3 Exception.py
Hello! You will choose a direction to go
Would you like to go N/S/E/W? N
```

Terminal Output w/ error

Note that the code broke because an error occurred, AND our personalized message was printed

```
MacBook-Pro-114:CPSC230 katherinehansen$ python3 Exception.py
Hello! You will choose a direction to go
Would you like to go N/S/E/W? G
Traceback (most recent call last):
  File "/Users/katherinehansen/Chapman/Fall2023/CPSC230/Exception.py", line 42, in <module>
    d = direction_getter()
        ^^^^^^^^^^^^^^^^^^
  File "/Users/katherinehansen/Chapman/Fall2023/CPSC230/Exception.py", line 38, in direction_getter
    raise ValueError("That is not an acceptable direction...")
ValueError: That is not an acceptable direction...
```

Raising Errors + Try & Except

- Using a try+except block to call a function that potentially raises an error
- We can print out our personalized message WITHOUT breaking our code

Code

```
try:
    d = direction_getter()
    print(d)
except ValueError as exception:
    print(exception)
```

Terminal Output w/ error

Note that the code did not break, because it was inside a try/except block, but our personalized message is still printed

```
MacBook-Pro-114:CPSC230 katherinehansen$ python3 Exception.py
Hello! You will choose a direction to go
Would you like to go N/S/E/W? G
That is not an acceptable direction...
```

Raising Errors + Try & Except + while loops

- We can use this combination to continuously prompt the user until they enter valid input without our code breaking while providing the personalized message

Code

```
while True:
    try:
        d = direction_getter()
        print(d)
        break
    except ValueError as exception:
        print(exception)
```

Terminal Output

```
MacBook-Pro-114:CPSC230 katherinehansen$ python3 Exception.py
Hello! You will choose a direction to go
Would you like to go N/S/E/W? J
That is not an acceptable direction...
Hello! You will choose a direction to go
Would you like to go N/S/E/W? G
That is not an acceptable direction...
Hello! You will choose a direction to go
Would you like to go N/S/E/W? N
N
```