Exceptions

Spring 2021

Exceptions

- A way to try code throughout Python but setting conditions where if there's an error, it displays a pre-determined error message instead of crashing
- Can be referred to as a "try/except" block
  - Exceptions can be specific (you determine what it is looking for)
  - They can also be general (it will catch *any* exception)

Exceptions: Examples

Examples	
Exception	**Description**
ValueError	Error raised when the wrong type is used (ex. trying to use a string in math)
ZeroDivisionError	Error raised when a number is divided by zero
IOError	Error raised when a file doesn’t exist and you try to open it to read
...and many more...	

Exceptions: Anatomy

- The code you want to try should be under a “try”
- Any exceptions you want should be through “except” blocks
  - Should include an error message
  - Can also have a general exception
    - Used for exceptions you’re not generally anticipating, but still don’t want the program to crash

```
# example of code we want to do
try:
    num1 = int(input("Enter in a number: "))
    num2 = int(input("Enter in another number: "))

    result = num1 / num2

    print("\nYour result is:", result)

# specific exception
except ZeroDivisionError:
    print("\nYou tried to divide by zero.")

# general exception
except:
    print("\nAn error occurred")
```

Exceptions: Anatomy

- Mirrors if/elif/else statements closely in how they work step wise
  - If the try is successful, it doesn't look at exceptions
    - Exceptions are only triggered if something goes wrong
  - Can have multiple *specific* exceptions per try (kind of like elif's)
  - Can have only *one* general exception per try (kind of like else's)
    - General exceptions must be last in a chain (like else's)
  - *Cannot* have exceptions without a "try"
  - If one exception is triggered, it doesn't look at the other ones

Exceptions: Example

```
# example of code we want to do
try:
    num1 = int(input("Enter in a number: "))
    num2 = int(input("Enter in another number: "))

    result = num1 / num2

    print("\nYour result is:", result)

# specific exception
except ZeroDivisionError:
    print("\nYou tried to divide by zero.")

# general exception
except:
    print("\nAn error occurred")
```

```
>>>
= RESTART: /home/kort/MEGA/teac
Enter in a number: 3
Enter in another number: 0

You tried to divide by zero.
>>>
= RESTART: /home/kort/MEGA/teac
Enter in a number: Hello

An error occurred
>>>
= RESTART: /home/kort/MEGA/teac
Enter in a number: 6
Enter in another number: 2

Your result is: 3.0
```

Exceptions: Notes

- Valid exceptions should highlight purple in IDLE (could be different in different IDEs or editors)
  - You don’t always need specific exceptions
  - A general “except” should be included in case the specific exceptions don’t catch every possible error
- Exceptions should include an error message of some sort to inform the user why the program isn’t continuing as excepted
- Cannot have multiple “try” blocks associated with the same exceptions

Multiple Exceptions

```
# example of code we want to do
try:
    num1 = int(input("Enter in a number: "))
    num2 = int(input("Enter in another number: "))

    result = num1 / num2

    print("\nYour result is:", result)

# specific exception
except ZeroDivisionError:
    print("\nYou tried to divide by zero.")

# specific exception
except ValueError:
    print("You entered in a non-integer value")

# general exception
except:
    print("\nAn error occurred")
```

For Reference

https://docs.python.org/3/library/exceptions.html

- Contains a list of valid Python 3 exceptions