INFO5002: Intro to Python for Info Sys

Error Handling

PCC 192-199

Northeastern University

Slides created by: Zachary Doucet

Sometimes our code crashes

- When executing some code Python may not know what to do and throw an exception.
- If an exception is **not caught**, then Python crashes the runtime.

```
x = 0
y = 10 / x
```

```
ZeroDivisionError: division by zero
```

We protect our at risk with try-except.

- If an exception may be thrown it is best to surround it with a try-except block specifying the expected exception.

```
try:
    x = int(input("Enter a number: "))
    y = 10 / x
except ZeroDivisionError:
    print("Thou shall not giveth a 0")
```

Use else to set what happens on non exception.

- If we have specific code we only want to run if the try block did not produce an error, we can use an else block.

```
try:
    x = int(input("Enter a number: "))
    y = 10 / x
except ZeroDivisionError:
    print("Thou shall not giveth a 0")
else:
    print(f"Your code is {y}")
```

You can fail silently if you want

- Sometimes you do not want to inform the user that an error was produced, so you can supply pass to the except block.

```
try:
    x = int(input("Enter a number: "))
    y = 10 / x
except ZeroDivisionError:
    pass
else:
    print(f"Your code is {y}")
```

Use exception if unsure type

- If you are unsure of the type, or have potentially multiple exception types that you want to act similarly, use Exception as type.

```
try:
    x = int(input("Enter a number: "))
    y = 10 / x
except Exception:
    pass
else:
    print(f"Your code is {y}")
```

Let’s practice

- Create a function **process_user** that will safely divide two user provided numbers, and will continuously prompt until success. The function takes no arguments.

  *Hint: you can get user data with the **input** function*