INTRO TO PROGRAMMING

REMOTE SERVER

Use putty on Windows and `ssh` on Apple devices to connect to the Linux server

host: <your_first_name>@104.236.69.159

password: <your_first_name>

```
# replace my username with yours and run ssh
# then enter password from email
ssh imclaughlin@104.236.69.159
```

Once on the server:

```
passwd  # run to change your password
tmux -S /var/pair attach -r
```

ASSIGNMENT REVIEW

We will first focus on finishing old assignments

Email your repository URL to im60@nyu.edu

ERRORS

- many languages have explicit error handling mechanisms that provide helpful functionality
- two main types of errors
  - Syntax Error - invalid source code that the interpreter can't make sense of
  - Exceptions - some critical condition has been detected, if not addressed program will abort with description of problem

ERRORS

```
# code in 'try' block runs until an error occurs
# if error occurs, 'except' block is run
try:
    print(1/0)
except:
    print("Oops! Something happened")

# can optionally specify what type of exception to handle
import sys

try:
    f = open('myfile.txt')
    s = f.readline()
    i = int(s.strip())
except OSError as err:
    print("OS error: {0}".format(err))
except ValueError:
    print("Could not convert data to an integer.")
except:
    print("Unexpected error:", sys.exc_info()[0])
    raise
```

CUSTOM ERRORS

```
# most common errors from language itself and standard library
# also possible to create your own

class MyError(Exception):
    def __init__(self, value):
        self.value = value
    def __str__(self):
        return repr(self.value)

try:
    raise MyError("code lacked an example error")
except MyError as e:
    print("Exception detected, message: ", e.value)
```

ERRORS

```
try:
    print(unsetVariable)
except NameError asd e:
    print("Exception detected, message: ", e.value
finally:
    print("this code will always be executed, fail or pass")
```

ERRORS: DEMO

COMPLEXITY THEORY

- Time complexity - roughly correlates with the number of instructions executed for the program
- Space complexity - amount of memory needed to keep track of relevant state
- Big O - a mathematical represenation of worst case performance

BIG O CLASSES

- O(1) - Constant: 1, 1, 1, 1...
- O(log N) - Logarithmic: 1 , 0.3 , 0.48 , 0.60, 0.70 ...
- O(N) - Linear: 1, 2 , 3 , 4 ...
- O(N^2) - Polynomial: 1, 4, 9, 16, 25, 36, 49 ...
- O(2^N) - Exponential: 2, 4, 8, 16, 32, 64, 128 ...

https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/

RESOURCES

- Bash tutorials
  - https://linuxconfig.org/bash-scripting-tutorial
  - http://ryanstutorials.net/bash-scripting-tutorial/bash-script.php
- Python tutorials
  - https://docs.python.org/3.4/tutorial/index.html
- Git Tutorials
  - https://www.atlassian.com/git/tutorials/what-is-version-control

END OF COURSE 1

Congratulations, you have completed the first course!

Make sure the instructor has the link to your assignment repo and you will receive instructions for the next course shortly