

## Lab 8

- Reading, writing files
- Modules
- Exception Handling
- Using lists to solve problems

## Lab 8: Pair Programming

Findley	Jordan
Parker	Margaret
Ryan	Chas
Lizzie	Robert
Olivia	Anna
Lindsey	Ben
Alison	Rachel
Kalady	Mary-Frances
Joseph	Andrew
Harris	Davis
Chase	Ian

Jordan	Findley
Margaret	Parker
Chas	Ryan
Robert	Lizzie
Anna	Olivia
Ben	Lindsey
Rachel	Alison
Mary-Frances	Kalady
Andrew	Joseph
Davis	Harris
Ian	Chase

## Pair Programming

- Two of you
  - double check problem requirements
  - Push each other: better tests, better comments
  - Iterate

## Compare Solutions

```
words = sentence.split()
shorthandList = []
for word in words:
    shorthandList.append(word[0])
shorthand = "".join(shorthandList)
shorthand = shorthand.lower()
print("Shorthand is:", shorthand)
```

```
words = sentence.split()
shorthand=""
for word in words:
    shorthand += word[0]
shorthand = shorthand.lower()
print("Shorthand is:", shorthand)
```

## Compare Solutions

```
words = sentence.split()

shorthandList = []
for word in words:
    shorthandList.append(word[0])

shorthand = "".join(shorthandList)
shorthand = shorthand.lower()
print("Shorthand is:", shorthand)
```

In general, look for less complex solutions.

Both are valid solutions. I'm not sure which is more efficient in practice.

However, the solution at left has more conceptual complexity (appending to a list and then converting to a string, as opposed to just creating the string).

```
words = sentence.split()

shorthand=""
for word in words:
    shorthand += word[0]

shorthand = shorthand.lower()
print("Shorthand is:", shorthand)
```

March 13, 2018

Sprenkle - CSCI111

5

## Comment Example

```
def encodeLetter(char, key):
    """Encodes a single character.
    PRE: Input parameters are a single, lowercase
    character string (char) and an integer key
    (between -25 and 25, inclusive)
    POST: returns the encoded character as a str"""
```

- Does not say *who* called function, where parameters came from, or where returned to
  - Any code can call the function and pass in input from anywhere (e.g., hardcoded, from user input, test function, ...)
- Does not say variable name returned

March 13, 2018

Sprenkle - CSCI111

6

## Review Caesar Cipher

- Consider the following solutions

```
for char in message:
    asciiVal = ord(char)
    if asciiVal == 32:
        ...
    else:
        ...
```

```
for char in message:
    if char == " ":
        ...
    else:
        ...
```

## Review Caesar Cipher

- Consider the following solutions

```
for char in message:
    asciiVal = ord(char)
    if asciiVal == 32:
        ...
    else:
        ...
```

I know what " " means.  
I don't immediately know  
what 32 means.

**Lesson: prefer words  
over numbers.**

```
for char in message:
    if char == " ":
        ...
    else:
        ...
```




## Review

- What are things we can do to lists?
- How do we work with files?
- What is the structure we use to do exception handling?

## CREATING MODULES

## Where are Functions Defined?

- Functions can go inside of program script
  - Defined before use/called (if no `main()` function)
  - Or, below the `main()` function (*preferred*)
- Functions can go inside a separate **module** 

## Creating Modules

- Modules group together related functions and constants
- Unlike functions, no special keyword to define a module
  - A module is named by its filename
- You've used modules in the past
  - `graphics.py`
  - `game.py`

Just a  
Python file!

## Typical Use of Modules

- Put your reusable code in a module that can be shared with others
- Example: `game.py`
  - `rollDie(sides)`
  - `rollMultipleDice(numDice, sides)`
- Call `import game` in Python interpreter
  - What happened?


## Creating Modules

- Then, to call `rollDie` function
  - `game.rollDie(6)`
- To access a defined constants
  - Example: `game.SIDES`

## Creating Modules

- So that our program doesn't execute code automatically when it is **imported** in a program, at bottom, add

```
if __name__ == '__main__':  
    testRollDie()  
    testRollMultipleDice()
```



Not important how this works;  
just know when to use

- Note the sub-directories now listed in the directory

## Benefits of Defining Functions in Separate Module

- Reduces code in **primary** driver script
- Easier to reuse by importing from a module
- Maintains the “black box”
  - **Abstraction**
- Isolates testing of function
- Write “test driver” scripts to test functions separately from use in script



## Lab 8 Overview

- Modules
- Reading Files
- Writing Files
- Exception handling
- Functions/Lists