# Welcome Class

Monday, September 23, 2019 5:00 PM

## Overview

Today we will be diving into reading and writing data from/to external CSV files. We will also be diving into Python dictionaries, zipping lists, and functions

## Class Objectives

- Reading data into Python from CSV Files
- Writing data from Python into CSV Files
- Zipping lists together
- Understanding how to create and use Python functions

# Exercise: Rock, Paper, Scissors [12 minutes]

Sunday, September 22, 2019 5:40 PM

## Overview

Students will be creating a simple Rock, Papers, Scissors that will run within the console

**Instructor Demo** [RPS_Solved.py]

**File Needed**: RPS_Unsolved.py

This starter code imports a module called **random** that will allow the computer to make a choice randomly from a list of actions. We'll talk about modules later

## Starter Code

- In the starter code, **random.choice** will pick a random choice within the **options** list for the computer and store its pick within a variable called **computer_choice**.
- The application prompts the user for their option and stores it within a variable called **user_choice**

```
# Incorporate the random library
import random

# Print Title
print("Let's Play Rock Paper Scissors!")

# Specify the three options
options = ["r", "p", "s"]

# Computer Selection
computer_choice = random.choice(options)

# User Selection
user_choice = input("Make your Choice: (r)ock, (p)aper, (s)cissors? ")
```

Random documentation:
https://docs.python.org/3.6/library/random.html

**Instructions**

- Using the terminal, take an input of r, p or s which will stand for rock, paper, and scissors.
- Have the computer randomly pick one of these three choices.
- Compare the user's input to the computer's choice to determine if the user won, lost, or tied.

**Hints**

- Look into this stackoverflow question for help on using the random module to select a value from a list
    - https://stackoverflow.com/questions/306400/how-to-randomly-select-an-item-from-a-list

# Review: Rock, Paper, Scissors

Sunday, September 22, 2019 5:40 PM

## Starter Code

- In the starter code, **random.choice** will pick a random choice within the **options** list for the computer and store its pick within a variable called **computer_choice**.
- The application prompts the user for their option and stores it within a variable called **user_choice**

```
# Incorporate the random library
import random

# Print Title
print("Let's Play Rock Paper Scissors!")

# Specify the three options
options = ["r", "p", "s"]

# Computer Selection
computer_choice = random.choice(options)

# User Selection
user_choice = input("Make your Choice: (r)ock, (p)aper, (s)cissors? ")
```

- Knowing that rock beats scissors, scissors beats paper, and paper beats rock the code can be organized into a series of conditional statements to compare the user's choice to the computer's choice

```
if (user_choice == "r" and computer_choice == "p"):
  print("You chose rock. The computer chose paper.")
  print("Sorry. You lose.")

elif (user_choice == "r" and computer_choice == "s"):
  print("You chose rock. The computer chose scissors.")
  print("Yay! You won.")

elif (user_choice == "r" and computer_choice == "r"):
  print("You chose rock. The computer chose rock.")
  print("A smashing tie!")

elif (user_choice == "p" and computer_choice == "p"):
  print("You chose paper. The computer chose paper.")
  print("A smashing tie!")
```

# Loop Recap

Monday, September 23, 2019 5:02 PM

Open Simple_loops.py

```
# A For loop moves through a given range of numbers
# If only one number is provided it will loop from 0 to that number
for x in range(10):
    print(x)

# If two numbers are provided then a For loop will loop from the first number up until it reaches the second number
for x in range(20, 30):
    print(x)

# If a list is provided, then the For loop will loop through each element within the list
words = ["Peanut", "Butter", "Jelly", "Time", "Is", "Now"]
for word in words:
    print(word)

# A While Loop will continue to loop through the code contained within it until some condition is met
x = "Yes"
while x == "Yes":
    print("Whee! Merry-Go-Rounds are great!")
    x = input("Would you like to go on the Merry-Go-Round again? ")
```

- For loops move through a given range of numbers
- If only one number is provided, it will loop from 0 to that number
- What does this block of code print out?

```
# A For loop moves through a given range of numbers
# If only one number is provided it will loop from 0 to that number
for x in range(10):
    print(x)

# If two numbers are provided then a For loop will loop from the first number up until it reaches the second number
for x in range(20, 30):
    print(x)

# If a list is provided, then the For loop will loop through each element within the list
words = ["Peanut", "Butter", "Jelly", "Time", "Is", "Now"]
for word in words:
    print(word)

# A While Loop will continue to loop through the code contained within it until some condition is met
x = "Yes"
while x == "Yes":
    print("Whee! Merry-Go-Rounds are great!")
    x = input("Would you like to go on the Merry-Go-Round again? ")
```

- If two numbers are provided, then a for loop will loop form the first number up until it reaches the second number
- What does this block print out?

```
# A For loop moves through a given range of numbers
# If only one number is provided it will loop from 0 to that number
for x in range(10):
    print(x)

# If two numbers are provided then a For loop will loop from the first number up until it reaches the second number
for x in range(20, 30):
    print(x)

# If a list is provided, then the For loop will loop through each element within the list
words = ["Peanut", "Butter", "Jelly", "Time", "Is", "Now"]
for word in words:
    print(word)

# A While Loop will continue to loop through the code contained within it until some condition is met
x = "Yes"
```

```
# A For loop moves through a given range of numbers
# If only one number is provided it will loop from 0 to that number
for x in range(10):
    print(x)

# If two numbers are provided then a For loop will loop from the first number up until it reaches the second number
for x in range(20, 30):
    print(x)

# If a list is provided, then the For loop will loop through each element within the list
words = ["Peanut", "Butter", "Jelly", "Time", "Is", "Now"]
for word in words:
    print(word)

# A While Loop will continue to loop through the code contained within it until some condition is met
x = "Yes"
while x == "Yes":
    print("Whee! Merry-Go-Rounds are great!")
    x = input("Would you like to go on the Merry-Go-Round again? ")
```

- If a list is provided, then the for loop will loop through each element within the list
- What does this block print out?

```
# A For loop moves through a given range of numbers
# If only one number is provided it will loop from 0 to that number
for x in range(10):
    print(x)

# If two numbers are provided then a For loop will loop from the first number up until it reaches the second number
for x in range(20, 30):
    print(x)

# If a list is provided, then the For loop will loop through each element within the list
words = ["Peanut", "Butter", "Jelly", "Time", "Is", "Now"]
for word in words:
    print(word)

# A While Loop will continue to loop through the code contained within it until some condition is met
x = "Yes"
while x == "Yes":
    print("Whee! Merry-Go-Rounds are great!")
    x = input("Would you like to go on the Merry-Go-Round again? ")
```

- A while loop will continue to loop through the code contained within it until some condition is met
- This loop will keep running as long as X is equal to what?

```
# A For loop moves through a given range of numbers
# If only one number is provided it will loop from 0 to that number
for x in range(10):
    print(x)

# If two numbers are provided then a For loop will loop from the first number up until it reaches the second number
for x in range(20, 30):
    print(x)

# If a list is provided, then the For loop will loop through each element within the list
words = ["Peanut", "Butter", "Jelly", "Time", "Is", "Now"]
for word in words:
    print(word)

# A While Loop will continue to loop through the code contained within it until some condition is met
x = "Yes"
while x == "Yes":
    print("Whee! Merry-Go-Rounds are great!")
    x = input("Would you like to go on the Merry-Go-Round again? ")
```

# Exercise: Number Chain [10 minutes]

Sunday, September 22, 2019 5:41 PM

## Overview

This exercise will take user input and print out a string of numbers.

**Instructor Demo** [NumberChain_solved.py]

## Instructions

- Using a while loop, ask the user "How many numbers?", and then print out a chain of ascending numbers from 0 to the number input.

- After the results have printed, ask the user if they would like to continue. If "y" is entered, keep the chain running by inputting a new number and starting a new count from 0 to the number input. If "n" is entered, exit the application

# Review: Number Chain

Sunday, September 22, 2019 5:41 PM

Key Points

- The initial value for `user_play` is set to "y" so that the `while` loop will run initially. This loop will continue to run so long as the value of `user_play` is "y" at the end of the code block.

```
# Initial variable to track game play
user_play = "y"

# While we are still playing...
while user_play == "y":

    # Ask the user how many numbers to loop through
    user_number = input("How many numbers? ")

    # Loop through the numbers. (Be sure to cast the string into an integer.)
    for x in range(int(user_number)):

        # Print each number in the range
        print(x)

    # Once complete...
    user_play = input("Continue: (y)es or (n)o? ")
```

- An input number is asked for and then a `for` loop will then run to count from 0 to that number.

```
# Initial variable to track game play
user_play = "y"

# While we are still playing...
while user_play == "y":

    # Ask the user how many numbers to loop through
    user_number = input("How many numbers? ")

    # Loop through the numbers. (Be sure to cast the string into an integer.)
    for x in range(int(user_number)):

        # Print each number in the range
        print(x)

    # Once complete...
    user_play = input("Continue: (y)es or (n)o? ")
```

- The user is then prompted to either enter "y" if they would like to create a new number chain or "n" if they would like to terminate the application.

```
# Initial variable to track game play
user_play = "y"

# While we are still playing...
while user_play == "y":

    # Ask the user how many numbers to loop through
    user_number = input("How many numbers? ")

    # Loop through the numbers. (Be sure to cast the string into an integer.)
    for x in range(int(user_number)):

        # Print each number in the range
        print(x)

    # Once complete...
    user_play = input("Continue: (y)es or (n)o? ")
```

# Exercise: Kid in a Candy Store

Monday, September 23, 2019 5:02 PM

## Overview

Each student is being placed in the role a kid going to with their parents to the supermarket. After pestering their parents for a while, they finally are allowed to pick out some candy to take home

**Instructor Demo ** [kid_in_candy_story.py]

**Files**: KidInCandyStore_Unsolved.py

## Starter Code

```
# The list of candies to print to the screen
candy_list =["Snickers", "Kit Kat", "Sour Patch Kids", "Juicy Fruit", "Swedish Fish",
             "Skittles", "Hershey Bar", "Starbursts", "M&Ms"]

# The amount of candy the user will be allowed to choose
allowance = 5

# The list used to store all of the candies selected inside of
candy_cart = []

# Print all of the candies to the screen and their index in brackets
for i in range(len(candy_list)):
    print("[" + str(i) + "] " + candy_list[i])
```

## Instructions

- Create a loop that prints all of the candies in the store to the terminal with their index stored in brackets beside them.
    - For example: "[0] Snickers"
    - Starter Code does this
- Create a loop that runs for a number of times as determined by the variable allowance.
    - For example: If allowance is equal to five, the loop should run five times.
- Each time this second loop runs, take in a user's input - preferably a number - and then add the candy with a matching index to the

variable candy_cart.
    - For example: If the user enters "0" as their input, "Snickers" should be added into the candy_cart list.
    - Refer back to Lists.py, in the 09-Ins_List/Solved folder in GitLab
- Use another loop to print all of the candies selected to the terminal.

# Review: Kid in a Candy Store

Monday, September 23, 2019 5:02 PM

Key Points

- There are three For loops being used in this activity.
- One to print out the original candy list.

```
# The list of candies to print to the screen
candy_list =["Snickers", "Kit Kat", "Sour Patch Kids", "Juicy Fruit", "Swedish Fish",
             "Skittles", "Hershey Bar", "Starbursts", "M&Ms"]

# The amount of candy the user will be allowed to choose
allowance = 5

# The list used to store all of the candies selected inside of
candy_cart = []

# Print all of the candies to the screen and their index in brackets
for i in range(len(candy_list)):
    print("[" + str(i) + "] " + candy_list[i])

# Run through a loop which allows the user to choose which candies to take home with them
print("Which candy would you like to bring home?")
for x in range(allowance):
    selected = input("Input the number of the candy you want: ")


    # Add the candy at the index chosen to the candy_cart list
    candy_cart.append(candy_list[int(selected)])

# Loop through the candy_cart to say what candies were brought home
print("I brought home with me...")
for candy in candy_cart:
    print(candy)
```

- A second to collect all of the candy choices the user has.
- When adding candies into the candy_cart list, the selection variable has to be cast as an integer since all inputs are naturally set as strings.

```
# The list of candies to print to the screen
candy_list =["Snickers", "Kit Kat", "Sour Patch Kids", "Juicy Fruit", "Swedish Fish",
             "Skittles", "Hershey Bar", "Starbursts", "M&Ms"]

# The amount of candy the user will be allowed to choose
allowance = 5

# The list used to store all of the candies selected inside of
candy_cart = []

# Print all of the candies to the screen and their index in brackets
for i in range(len(candy_list)):
    print("[" + str(i) + "] " + candy_list[i])

# Run through a loop which allows the user to choose which candies to take home with them
print("Which candy would you like to bring home?")
for x in range(allowance):
    selected = input("Input the number of the candy you want: ")


    # Add the candy at the index chosen to the candy_cart list
    candy_cart.append(candy_list[int(selected)])

# Loop through the candy_cart to say what candies were brought home
print("I brought home with me...")
for candy in candy_cart:
    print(candy)
```

- And a third to print the final list of choices to the screen.

```
# The list of candies to print to the screen
candy_list =["Snickers", "Kit Kat", "Sour Patch Kids", "Juicy Fruit", "Swedish Fish",
             "Skittles", "Hershey Bar", "Starbursts", "M&Ms"]

# The amount of candy the user will be allowed to choose
allowance = 5

# The list used to store all of the candies selected inside of
candy_cart = []

# Print all of the candies to the screen and their index in brackets
for i in range(len(candy_list)):
    print("[" + str(i) + "] " + candy_list[i])

# Run through a loop which allows the user to choose which candies to take home with them
print("Which candy would you like to bring home?")
for x in range(allowance):
    selected = input("Input the number of the candy you want: ")


    # Add the candy at the index chosen to the candy_cart list
    candy_cart.append(candy_list[int(selected)])

# Loop through the candy_cart to say what candies were brought home
print("I brought home with me...")
for candy in candy_cart:
    print(candy)
```

# Exercise: House of Pies

Monday, September 23, 2019 5:03 PM

## Overview

In this activity, student will be constructing an order form that will display a list of pies and then prompt users to make a selection. It will continue

**Instructor Demo** [house_of_pies.py]

## Instructions

- Create an order form that will display a list of pies to the users in the following way:

Welcome to the House of Pies! Here are our pies:
---
(1) Pecan, (2) Apple Crisp, (3) Bean, (4) Banoffee, (5) Black Bun, (6) Blueberry, (7) Buko, (8) Burek, (9) Tamale, (10) Steak

```
# Show pie selection prompt
print("----------------------------------------------------------------------")
print("(1) Pecan, (2) Apple Crisp, (3) Bean, (4) Banoffee, " +
      " (5) Black Bun, (6) Blueberry, (7) Buko, (8) Burek, " +
      " (9) Tamale, (10) Steak ")
```

- Then prompt the user to select which pie they'd like to order via number
- Immediately after, follow the order with "Great! We'll have that <PIE NAME> right out for you" and then ask if they would like to make another order. If so, repeat the process
- Once the user is done purchasing pies, print the total number of pies ordered

Hints

- Code from previous exercise will be helpful

# Review: House of Pies

Monday, September 23, 2019 5:03 PM

## Key Points

- The primary loop being used is a While loop that is constantly checking whether the user's response to the question **Would you like to make another purchase?** ever changes from **y**

```
# Initial variable to track shopping status
shopping = 'y'

# List to track pie purchases
pie_purchases = []

# Pie List
pie_list = ["Pecan", "Apple Crisp", "Bean", "Banoffee", "Black Bun",
            "Blueberry", "Buko", "Burek", "Tamale", "Steak"]

# Display initial message
print("Welcome to the House of Pies! Here are our pies:")

# While we are still shopping...
while shopping == "y":

    # Show pie selection prompt
    print("-------------------------------------------------------------------")
    print("(1) Pecan, (2) Apple Crisp, (3) Bean, (4) Banoffee, " +
          " (5) Black Bun, (6) Blueberry, (7) Buko, (8) Burek, " +
          " (9) Tamale, (10) Steak ")

    pie_choice = input("Which would you like? ")

    # Add pie to the pie list
    pie_purchases.append(pie_choice)

    print("-------------------------------------------------------------------")

    # Inform the customer of the pie purchase
    print("Great! We'll have that " + pie_list[int(pie_choice) - 1] + " right out for you.")

    # Provide exit option
    shopping = input("Would you like to make another purchase: (y)es or (n)o? ")

# Once the pie list is complete
print("-------------------------------------------------------------------")
print("You purchased a total of " + str(len(pie_purchases)) + ".")
```

- Since the pie menu we created started with 1, the user's input must be subtracted by 1 one when referencing the pie's actual index. This is because all indices start at 0 naturally.

```
# Initial variable to track shopping status
shopping = 'y'

# List to track pie purchases
pie_purchases = []

# Pie List
pie_list = ["Pecan", "Apple Crisp", "Bean", "Banoffee", "Black Bun",
            "Blueberry", "Buko", "Burek", "Tamale", "Steak"]

# Display initial message
print("Welcome to the House of Pies! Here are our pies:")

# While we are still shopping...
while shopping == "y":

    # Show pie selection prompt
    print("----------------------------------------------------------------------")
    print("(1) Pecan, (2) Apple Crisp, (3) Bean, (4) Banoffee, " +
          " (5) Black Bun, (6) Blueberry, (7) Buko, (8) Burek, " +
          " (9) Tamale, (10) Steak ")

    pie_choice = input("Which would you like? ")

    # Add pie to the pie list
    pie_purchases.append(pie_choice)

    print("----------------------------------------------------------------------")

    # Inform the customer of the pie purchase
    print("Great! We'll have that " + pie_list[int(pie_choice) - 1] + " right out for you.")

    # Provide exit option
    shopping = input("Would you like to make another purchase: (y)es or (n)o? ")

# Once the pie list is complete
print("----------------------------------------------------------------------")
print("You purchased a total of " + str(len(pie_purchases)) + ".")
```

- The means by which the total number of pies is calculated in the original solution is by determining the length of the **pie_purchases** array.

```
# Initial variable to track shopping status
shopping = 'y'

# List to track pie purchases
pie_purchases = []

# Pie List
pie_list = ["Pecan", "Apple Crisp", "Bean", "Banoffee", "Black Bun",
            "Blueberry", "Buko", "Burek", "Tamale", "Steak"]

# Display initial message
print("Welcome to the House of Pies! Here are our pies:")

# While we are still shopping...
while shopping == "y":

    # Show pie selection prompt
    print("----------------------------------------------------------------------")
    print("(1) Pecan, (2) Apple Crisp, (3) Bean, (4) Banoffee, " +
          " (5) Black Bun, (6) Blueberry, (7) Buko, (8) Burek, " +
          " (9) Tamale, (10) Steak ")
```

```
    print("(1) Pecan, (2) Apple Crisp, (3) Bean, (4) Banoffee, " +
          " (5) Black Bun, (6) Blueberry, (7) Buko, (8) Burek, " +
          " (9) Tamale, (10) Steak ")

    pie_choice = input("Which would you like? ")

    # Add pie to the pie list
    pie_purchases.append(pie_choice)

    print("------------------------------------------------------------------------")

    # Inform the customer of the pie purchase
    print("Great! We'll have that " + pie_list[int(pie_choice) - 1] + " right out for you.")

    # Provide exit option
    shopping = input("Would you like to make another purchase: (y)es or (n)o? ")

# Once the pie list is complete
print("------------------------------------------------------------------------")
print("You purchased a total of " + str(len(pie_purchases)) + ".")
```

Friday, September 27, 2019 4:47 PM

# Break (15 minutes)

Monday, September 23, 2019 5:05 PM

# Reading Text Files

Monday, September 23, 2019 5:03 PM

Python is capable of reading data in form external text files and then performing some task

Open read_file.py and input.txt

- When dealing with external files, Python requires very precise directions on what path to follow to reach the desired file. As such, if the desired file is stored within a sub-folder called "Resources", the path needed would be "Resources/FileName.txt".
- Different operating systems use different paths to locate files. For example: Windows machines will often use forward slashes to separate folders while Mac devices will use backslashes.

```
# Store the file path associated with the file (note the backslash may be OS specific)
file = 'Resources/input.txt'
```

- The **with** statement is simply saying that, for as long as we are dealing with the code within the following block, save the text variable.
- **open(<File Path>, <Read/Write>)** is the function Python uses in order to open up and work with external text files.
- By specifying either **'r', 'w', or 'rw'**, users can use the **open()** function to either
    - read from a text file,
    - write to a text file, or
    - perform both operations in the code block that follows it.

```
# Open the file in "read" mode ('r') and store the contents in the variable "text"
with open(file, 'r') as text:
    # Store all of the text inside a variable called "lines"
    lines = text.read()

    # Print the contents of the text file
    print(lines)
```

- **text.read()** parses the data that is read in by **the open()** function and converts it into a string type.
- If this function was not used all that would be printed to the screen would be an unhelpful text wrapper object.


```
# Open the file in "read" mode ('r') and store the contents in the variable "text"
with open(file, 'r') as text:
    # Store all of the text inside a variable called "lines"
    lines = text.read()

    # Print the contents of the text file
    print(lines)
```

# Introduction to Modules

Monday, September 23, 2019 5:04 PM

- While Python includes many built-in functions, sometimes coders have to bring in external modules in order to perform specific tasks.
    - If a coder wanted a random number generator for a dice game they were making, for example, they would most likely want to use the **random** module instead of having to create the code from scratch.
- Importing modules into Python is actually very simple. All that is required is for the module to be installed and for the user to import the module into their code.
    - All of the modules for today come packaged with Python, so there is no need to install anything new.
- Open Imports.py

```
# Import the random and string Module
import random
import string

# Utilize the string module's custom method: ".ascii_letters"
print(string.ascii_letters)

# Utilize the random module's custom method randint
for x in range(10):
    print(random.randint(1, 10))
```

- The **string** module contains many helpful constants and methods which pertain to strings. For example, users can use **string.ascii_letters** and Python will instantly grab a reference to every ascii character.
- The **random** module allows Python to randomly select values from set ranges, lists, or even strings.

# Reading in CSV Files

Monday, September 23, 2019 5:05 PM

- While reading in text files can be useful in some circumstances, it is more likely within the data industry to run across files known as CSV files.
- CSV stands for **Comma Separated Values** and is essentially a table that has been converted into text format with each row and column being separated by specified symbols.
- More often than not each row is located on a new line and each column is separated by a comma.

```
First Name,Last Name,SSN
Tina,Fleming,619-16-7988
Erica,Shah,164-51-7615
Paula,Ortiz,051-83-3290
James,Hendricks,776-83-2884
Lauren,King,197-94-2398
David,Cowan,252-92-1832
Andrew,Burton,296-23-6842
Julian,Baker,337-40-7543
Scott,Castro,399-46-5595
Billy,Rodriguez,014-18-2503
Darrell,Leblanc,005-82-7918
David,Hammond,561-17-6312
```

- Open read_csv.py and accounting.csv
- The **OS** module allows Python programmers to very easily create dynamic paths to external files that function across different operating systems.

```
# First we'll import the os module
# This will allow us to create file paths across operating systems
import os
csvpath = os.path.join('Resources', 'accounting.csv')
```

- Python has a module called **csv** which allows its users to easily pull

in data from external CSV files and perform some operations upon them.

- We use the **with open()** syntax from earlier to ready in the file.
- The key difference here is that this code now includes the **newline=''** parameter.
- This tells Python that each time the CSV file goes down a line, it should be considered a new row

```
import csv
with open(csvpath, newline='') as csvfile:

    # CSV reader specifies delimiter and variable that holds contents
    csvreader = csv.reader(csvfile, delimiter=',')

    print(csvreader)

    #  Each row is read as a row
    for row in csvreader:
        print(row)
```

- Instead of **text.read()**, this new code instead utilizes csv.reader() to translate the object being opened by Python.
- The **delimiter=','** parameter being used as this tells Python that each comma within the CSV should be seen as moving into a new column for a row.

```
import csv
with open(csvpath, newline='') as csvfile:

    # CSV reader specifies delimiter and variable that holds contents
    csvreader = csv.reader(csvfile, delimiter=',')

    print(csvreader)

    #  Each row is read as a row
    for row in csvreader:
        print(row)
```

- Reminder: reading of the file must be done within **with open()** statement. Outside of that block of code, the variable csvreader will not be useful because the file will be closed when the with open() block ends.
- In a later example, we will see that one option for working with the

data outside of with open() is to append it to a list.

# Exercise: Reading Netflix

Monday, September 23, 2019 5:05 PM

## Overview

Student will be provided with a CSV file containing data taken from Netflix. They will then create an application which searches through data for a specific movie/show and returns the name, rating, and review score

**Instructor Demo** [netflix.py]

**File**: Netflix_Ratings.csv

## Instructions

- Prompt the user for what video they are looking for.
- Search through the **netflix_ratings.csv** to find the user's video.
- If the CSV contains the user's video then print out the title, what it is rated and the current user ratings.
    - For example: **'Grease is rated PG with a rating of 86'**

# Review: Reading Netflix

Monday, September 23, 2019 5:06 PM

## Key Points

- Imports both the **os** and **csv** modules for use later on. It is common practice to import all modules at the start of an application.

```
# Modules
import os
import csv

# Prompt user for video lookup
video = input("What show or movie are you looking for? ")

# Set path for file
csvpath = os.path.join("..", "Files", "netflix_ratings.csv")


# Open the CSV
with open(csvpath, newline="") as csvfile:
    csvreader = csv.reader(csvfile, delimiter=",")

    # Loop through looking for the video
    for row in csvreader:
        if row[0] == video:
            print(row[0] + " is rated " + row[1] + " with a rating of " + row[5])
```

- When opening up the CSV file, the code dictates that each new line in the file should be viewed as a new line of data to be read in.
- When reading the CSV file, the delimiter is set to **","** to ensure Python splits up the data into the proper columns whenever a comma is found.

```
# Modules
import os
import csv

# Prompt user for video lookup
video = input("What show or movie are you looking for? ")

# Set path for file
csvpath = os.path.join("..", "Files", "netflix_ratings.csv")


# Open the CSV
with open(csvpath, newline="") as csvfile:
    csvreader = csv.reader(csvfile, delimiter=",")

    # Loop through looking for the video
    for row in csvreader:
        if row[0] == video:
            print(row[0] + " is rated " + row[1] + " with a rating of " + row[5])
```

- The code loops through each row, searching for the row whose first value - index 0 - is equal to that of the search term entered.
- The rating of a video is at the index of 1 and the review score is located at the index of 5. For the bonus the break statement is added to end the loop once a movie is found.

# Writing CSV Files

Monday, September 23, 2019 5:07 PM

Python can write data into CSV files too. This allows users to easily modify and/or create datasets based up previous data

Open write.py

- The syntax for writing into a CSV file is similar to that used to read data in from an external file.
- First, the code references the path that will point into the CSV file the user would like to write to.

```
# Specify the file to write to
output_path = os.path.join('output', 'new.csv')

# Open the file using "write" mode. Specify the variable to hold the contents
with open(output_path, 'w', newline='') as csvfile:

    # Initialize csv.writer
    csvwriter = csv.writer(csvfile, delimiter=',')

    # Write the first row (column headers)
    csvwriter.writerow(['First Name', 'Last Name', 'SSN'])

    # Write the second row
    csvwriter.writerow(['Caleb', 'Frost', '505-80-2901'])
```

- Next, the **with open()** statement is used once more but with one significant difference. The parameter **'w'** is passed instead to inform Python to write to the file.

```
# Specify the file to write to
output_path = os.path.join('output', 'new.csv')

# Open the file using "write" mode. Specify the variable to hold the contents
with open(output_path, 'w', newline='') as csvfile:

    # Initialize csv.writer
    csvwriter = csv.writer(csvfile, delimiter=',')

    # Write the first row (column headers)
    csvwriter.writerow(['First Name', 'Last Name', 'SSN'])

    # Write the second row
    csvwriter.writerow(['Caleb', 'Frost', '505-80-2901'])
```

- Instead of **csv.reader()**, **csv.writer()** is used to inform Python that this application will be writing code into an external CSV file.

```
# Specify the file to write to
output_path = os.path.join('output', 'new.csv')

# Open the file using "write" mode. Specify the variable to hold the contents
with open(output_path, 'w', newline='') as csvfile:

    # Initialize csv.writer
    csvwriter = csv.writer(csvfile, delimiter=',')

    # Write the first row (column headers)
    csvwriter.writerow(['First Name', 'Last Name', 'SSN'])

    # Write the second row
    csvwriter.writerow(['Caleb', 'Frost', '505-80-2901'])
```

- To write a new row into a CSV file, simply use the **csv.writerow(<DATA LIST>)** function and pass in an array of data as the parameter

```
# Specify the file to write to
output_path = os.path.join('output', 'new.csv')

# Open the file using "write" mode. Specify the variable to hold the contents
with open(output_path, 'w', newline='') as csvfile:

    # Initialize csv.writer
    csvwriter = csv.writer(csvfile, delimiter=',')

    # Write the first row (column headers)
    csvwriter.writerow(['First Name', 'Last Name', 'SSN'])

    # Write the second row
    csvwriter.writerow(['Caleb', 'Frost', '505-80-2901'])
```

# Zipping Lists

Monday, September 23, 2019 5:07 PM

It is possible to write new rows of data into a CSV file using a bunch of `csv.writerow()` statements, Python users can far more efficiently write data into a new CSV file by using the `zip()` function.

`zip()` takes in a series of lists as its parameters and joins them together into a stack.

Open Zipper.py

- This application has three lists, all of which pertain to each other and are of the same length. By zipping these lists together, there is now a single joined list whose indexes reference all three of the lists inside

```
import csv
import os

# Three Lists
indexes = [1, 2, 3, 4]
employees = ["Michael", "Dwight", "Meredith", "Kelly"]
department = ["Boss", "Sales", "Sales", "HR"]

# Zip all three lists together into tuples
roster = zip(indexes, employees, department)

# save the output file path
output_file = os.path.join("..", "Files","output","output.csv")

# open the output file, create a header row, and then write the zipped object to the csv
with open(output_file, "w", newline="") as datafile:
    writer = csv.writer(datafile)

    writer.writerow(["Index", "Employee", "Department"])

    writer.writerows(roster)
```

# Exercise: Udemy Zip

Monday, September 23, 2019 5:07 PM

## Overview

Students will take a large dataset from Udemy, clean it up, and create a new CSV file that is far easier to comprehend

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Title | Course Price | Subscribers | Reviews Left | Percent of Reviews | Length of Course |
| 2 | Learn Web Designing & HTML5/CSS3 Essentials in 4-Hours | 75 | 43285 | 525 | 0.01 | 4 |
| 3 | Learning Dynamic Website Design - PHP MySQL and JavaScript | 50 | 47886 | 285 | 0.01 | 12.5 |
| 4 | ChatBots: Messenger ChatBot with API.AI and Node.JS | 50 | 2577 | 529 | 0.21 | 4.5 |
| 5 | Projects in HTML5 | 60 | 8777 | 206 | 0.02 | 15.5 |
| 6 | Programming Foundations: HTML5 + CSS3 for Entrepreneurs 2015 | 20 | 23764 | 490 | 0.02 | 5.5 |
| 7 | How To Make A Wordpress Website 2017 \| Divi Theme Tutorial | 40 | 3541 | 202 | 0.06 | 4 |
| 8 | Build Your Own Backend REST API using Django REST Framework | 50 | 2669 | 112 | 0.04 | 5.5 |
| 9 | Angular and Firebase - Build a Web App with Typescript | 150 | 1966 | 359 | 0.18 | 5 |
| 10 | Web Development Masterclass - Complete Certificate Course | 200 | 4090 | 178 | 0.04 | 19.5 |
| 11 | Spring Boot Tutorial For Beginners | 40 | 2578 | 210 | 0.08 | 5.5 |
| 12 | The Complete Bootstrap Masterclass Course - Build 4 Projects | 195 | 24978 | 540 | 0.02 | 7 |

**Instructor Demo** [web_starter.csv, web_solved.py]

**Files**: WebDevelopment.csv??

## Instructions

- Create a Python application that reads the data on Udemy Web Development offerings.
- Then store the contents of the Title, Price, Subscriber Count, Number of Reviews, and Course Length into Python Lists.
- Then zip these lists together into a single tuple.

  Finally, write the contents of your extracted data into a CSV. Make sure to include the titles of these columns in your CSV.

## Notes

- As, with many datasets, the file does not include the header line. Use the below as a guide on the columns: "id,title,url,isPaid,price,numSubscribers,numReviews,numPublishedLectures,instructionalLevel,contentInfo,publishedTime"

# Review: Udemy Zip

Monday, September 23, 2019 5:08 PM

## Key Points

- There are six empty lists created at the start of this application, all of which will be used to hold specific data taken from within the original CSV and ultimately be zipped together before being written to a new CSV file.
- For every new row read in from the original CSV file, new data is appended into the lists from earlier. In the cases of percent and length, the data is being altered before being placed into their respective list.

```
# Lists to store data
title = []
price = []
subscribers = []
reviews = []
review_percent = []
length = []

with open(udemy_csv, newline="") as csvfile:
    csvreader= csv.reader(csvfile, delimiter=",")
    for row in csvreader:
        # Add title
        title.append(row[1])

        # Add price
        price.append(row[4])

        # Add number of subscribers
        subscribers.append(row[5])

        # Add amount of reviews
        reviews.append(row[6])

        # Determine percent of review left to 2 decimal places
        percent = round(int(row[6])/int(row[5]), 2)
        review_percent.append(percent)

        # Get length of the course to just a number
        new_length = row[9].split(" ")
        length.append(new_length[0])
```

- Once all data has been read, the lists are zipped together and written into a new CSV file with a header row being written in beforehand.

```
# Zip lists together
cleaned_csv = zip(title, price, subscribers, reviews, review_percent,length)

# Set variable for output file
output_file = os.path.join("web_final.csv")

#  Open the output file
with open(output_file, "w", newline="") as datafile:
    writer = csv.writer(datafile)

    # Write the header row
    writer.writerow(["Title", "Course Price", "Subscribers", "Reviews Left",
                     "Percent of Reviews", "Length of Course"])

    # Write in zipped rows
    writer.writerows(cleaned_csv)
```

# Introduction to Functions

Monday, September 23, 2019 5:09 PM

A function is a block of organized, reusable code that is used to perform a single, related action. In other words, functions are placeable blocks of code that perform a specific action

Open functions.py

- To create a new function, simply use `def <Function Name>():` and then place the code that you would like to run within the block underneath it.
- In order to run the code stored within a function, the function itself must be called within the program. Functions will never run unless called upon.

```
def printHello():
  print(f"Hello!")

printHello()
```

- Functions that take in parameters can also be created by simply adding a variable into the parentheses of the function's definition. This allows specific data to be passed into the function for usage.

```
def printName(name):
    print("Hello " + name + "!")

printName("Bob Smith")
```