MOD 3 Day 2 - Reading, Writing, and Pyrithmetic

Monday, September 25, 2023 9:28 AM

Class Objectives

By the end of today's class, you will be able to:

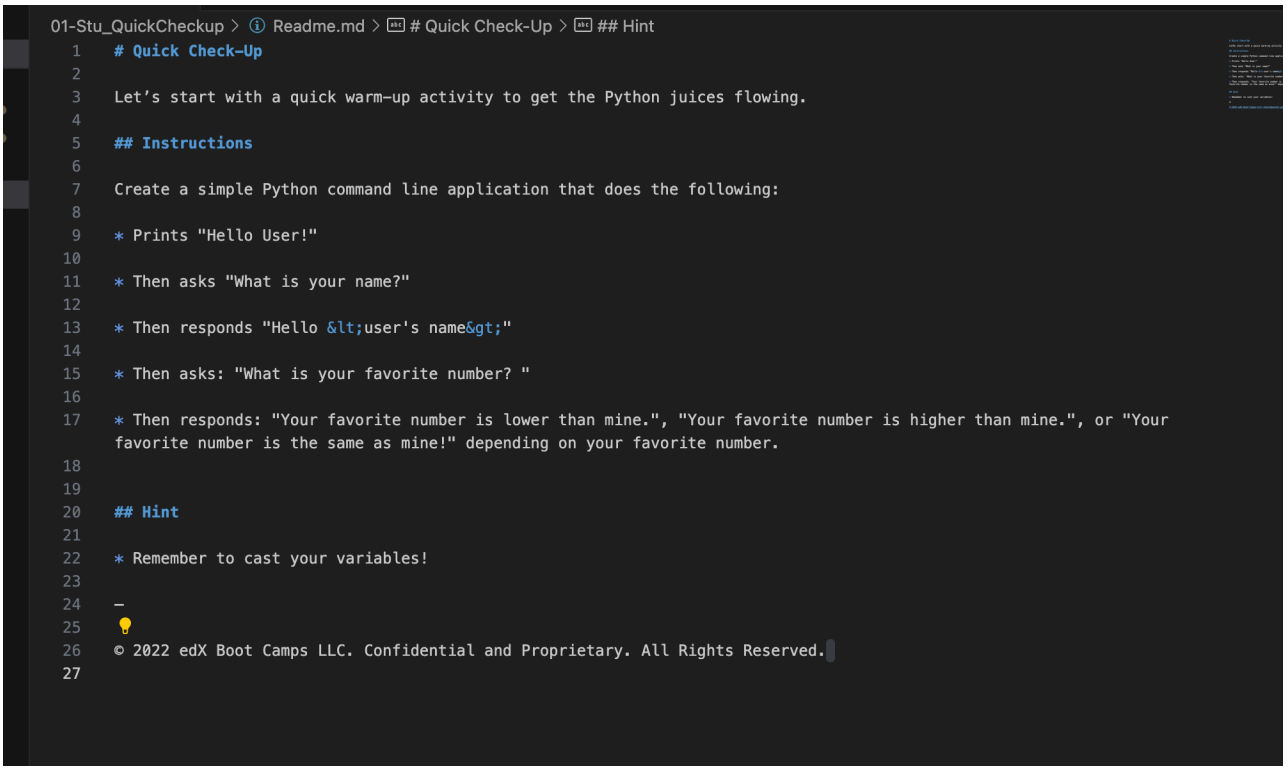
- Read data into Python from CSV files.
- Write data from Python to CSV files.
- Zip two lists together and know when this is helpful.
- Create and use Python functions.

Activity: Python Check-Up

Create a simple Python command line application that does the following:

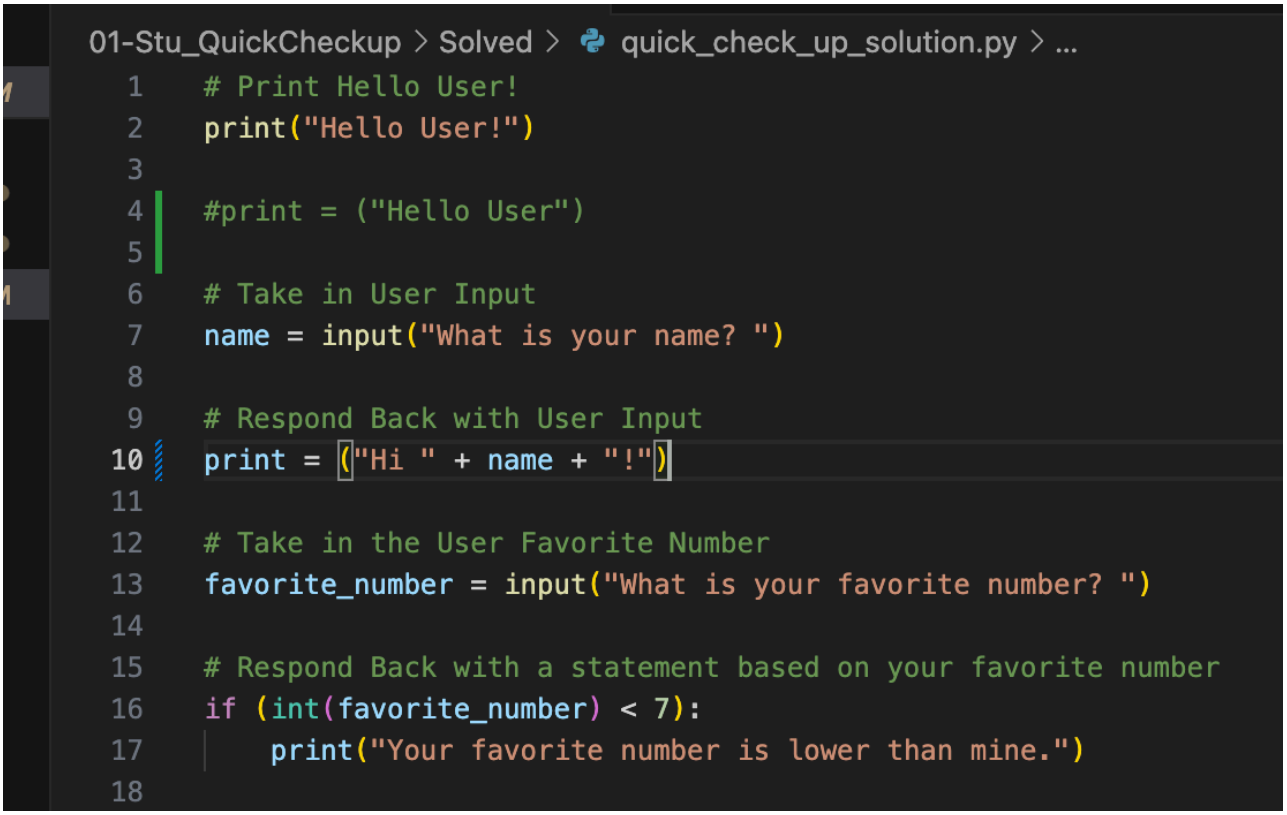
- Prints "Hello User!"
- Then asks "What is your name?"
- Then responds "Hello <user's name>"
- Then asks "What's your favorite number?"
- Then responds: "Your favorite number is lower than mine.", "Your favorite number is higher than mine.", or "Your favorite number is the same as mine!" depending on your favorite number.

Activity 1
0:13:00



01-Stu_QuickCheckup > Readme.md > # Quick Check-Up > ## Hint

```
# Quick Check-Up

Let's start with a quick warm-up activity to get the Python juices flowing.

## Instructions

Create a simple Python command line application that does the following:

* Prints "Hello User!"

* Then asks "What is your name?"

* Then responds "Hello &lt;user's name&gt;"

* Then asks: "What is your favorite number? "

* Then responds: "Your favorite number is lower than mine.", "Your favorite number is higher than mine.", or "Your favorite number is the same as mine!" depending on your favorite number.


## Hint

* Remember to cast your variables!

-

💡
```

01-Stu_QuickCheckup > Solved > quick_check_up_solution.py > ...

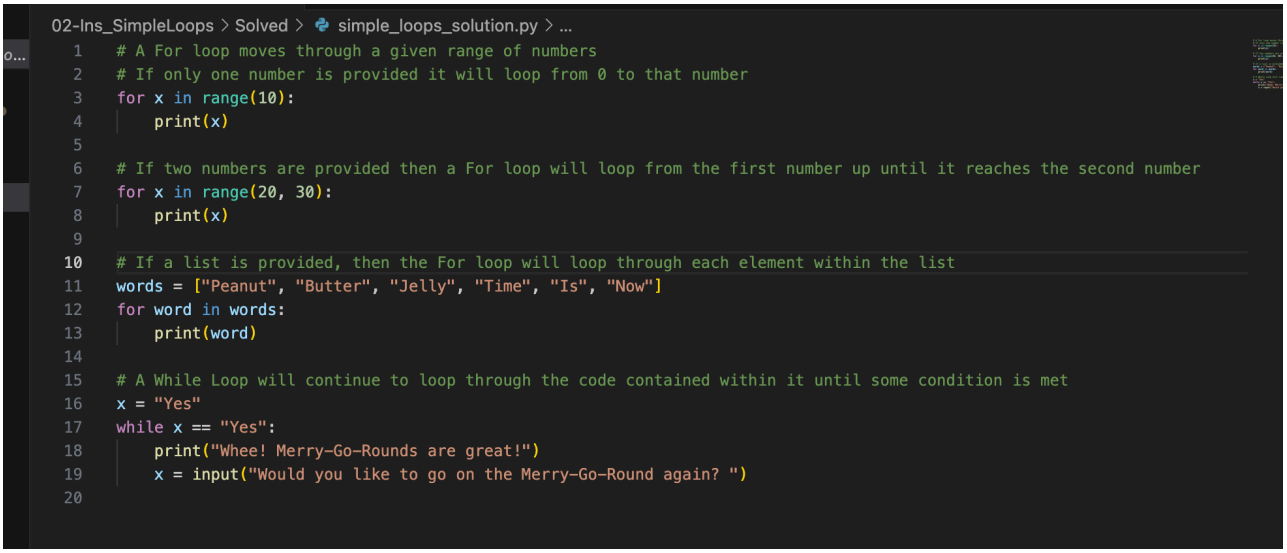
```python
# Print Hello User!
print("Hello User!")

#print = ("Hello User")

# Take in User Input
name = input("What is your name? ")

# Respond Back with User Input
print = ("Hi " + name + "!")

# Take in the User Favorite Number
favorite_number = input("What is your favorite number? ")

# Respond Back with a statement based on your favorite number
if (int(favorite_number) < 7):
    print("Your favorite number is lower than mine.")

```

```
19    elif (int(favorite_number) == 7):
20        print("Your favorite number is the same as mine!")
21
22    else:
23        print("Your favorite number is higher than mine.")
24
```

Activity 2
0:17:00

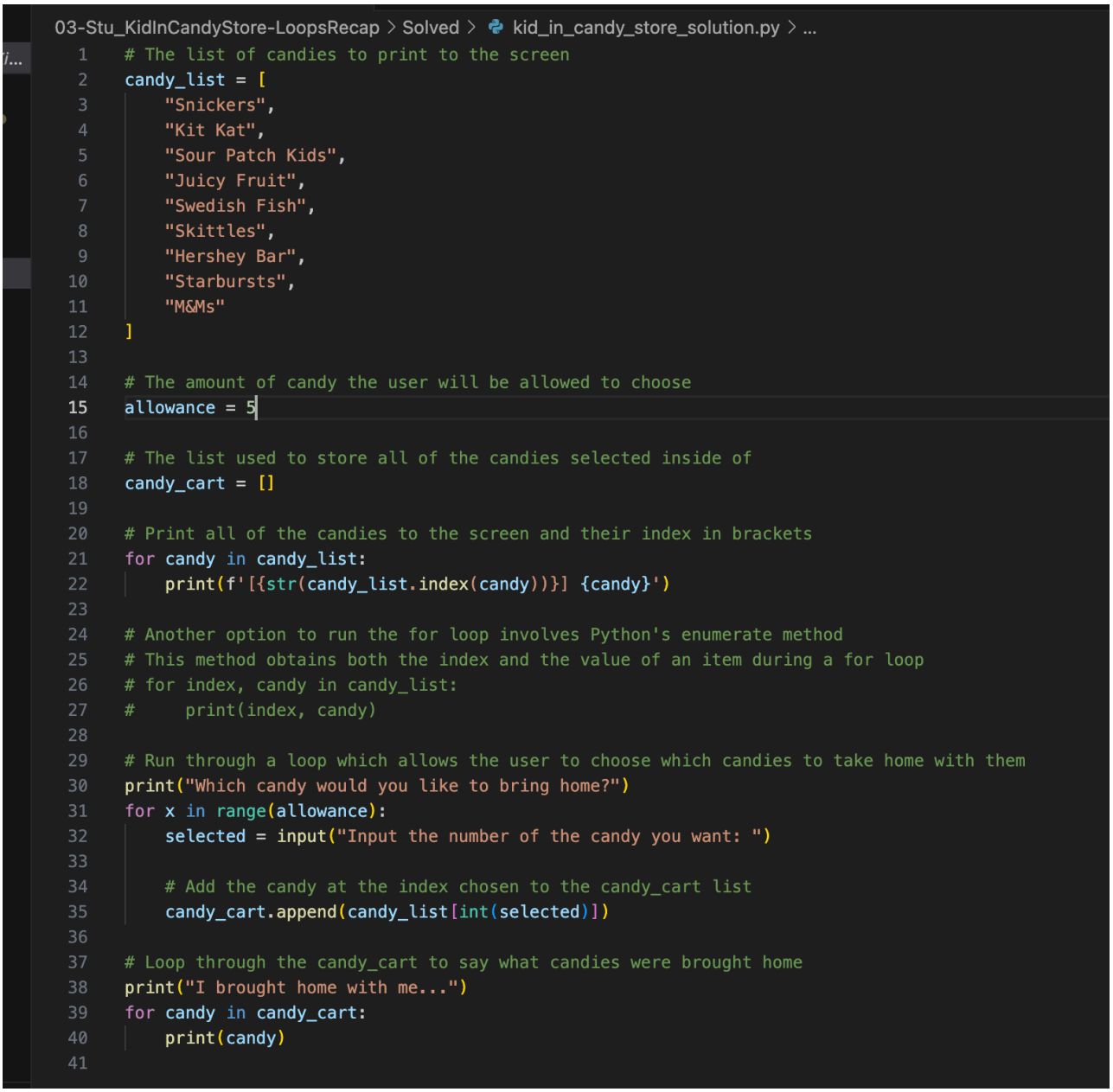
We use " while "instead of " if "because while will loop and if will loop once.

```
# A For loop moves through a given range of numbers
# If only one number is provided it will loop from 0 to that number

# If two numbers are provided then a For loop will loop from the first
number up until it reaches the second number


# If a list is provided, then the For loop will loop through each
element within the list


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



Activity 3
Kid in candy store
0:27:00

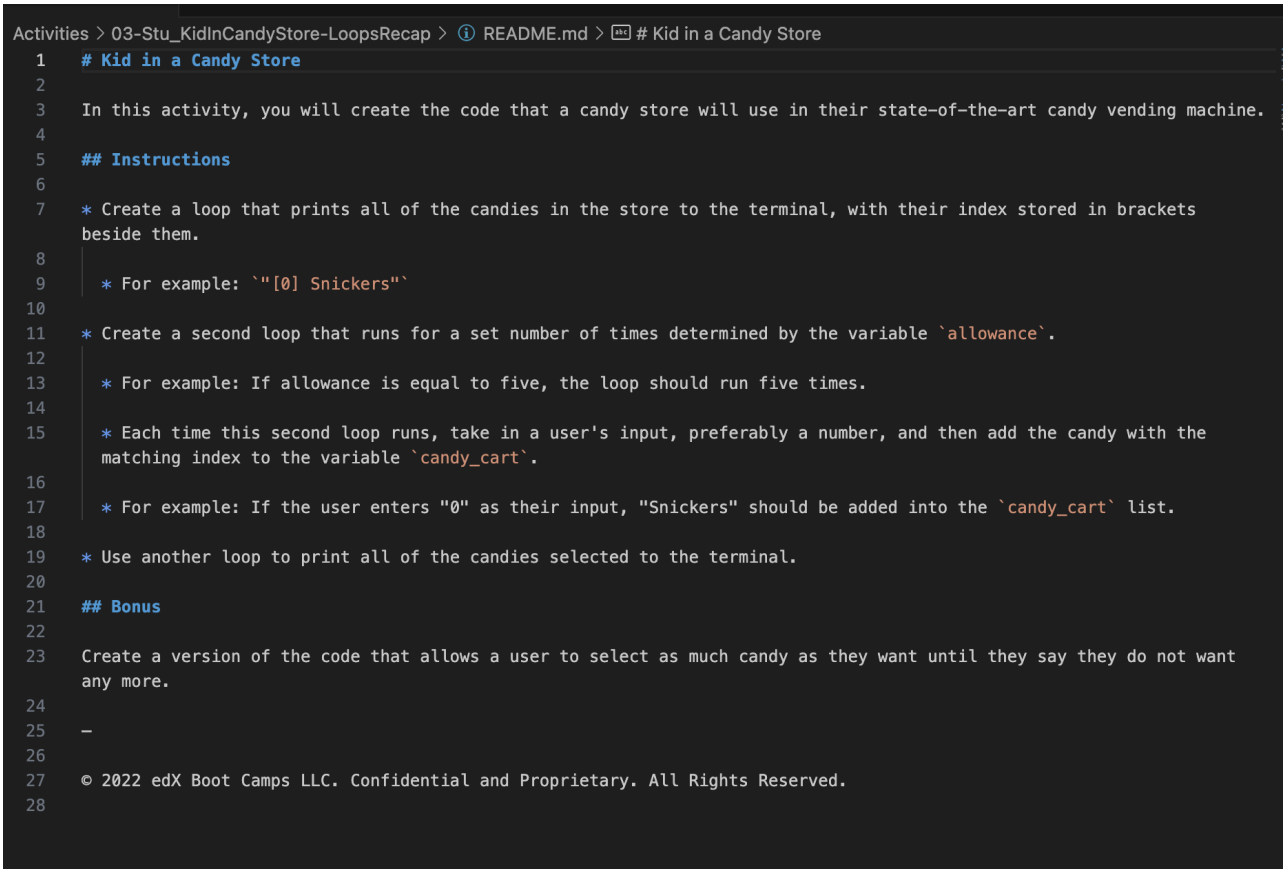
3 (multiple) loops being used
Using select variable



Activity 3

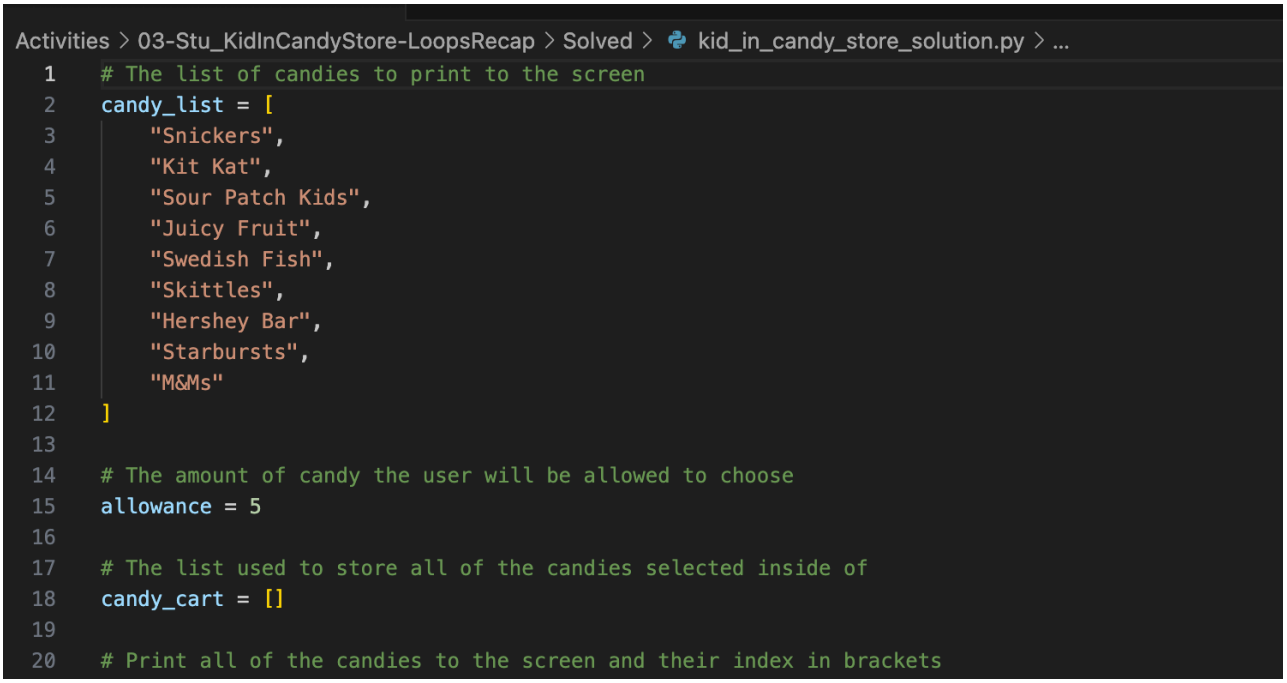
0:43:00
Kid in candy store

This code has 3 for loops



Activities > 03-Stu_KidInCandyStore-LoopsRecap > README.md > # Kid in a Candy Store

```
# Kid in a Candy Store

In this activity, you will create the code that a candy store will use in their state-of-the-art candy vending machine.

## 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"`

* Create a second loop that runs for a set number of times 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 the 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.

* Use another loop to print all of the candies selected to the terminal.

## Bonus

Create a version of the code that allows a user to select as much candy as they want until they say they do not want any more.

-

© 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.
```



Activities > 03-Stu_KidInCandyStore-LoopsRecap > Solved > kid_in_candy_store_solution.py > ...

```python
# 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 candy in candy_list:
    print(f'[{str(candy_list.index(candy))}] {candy}')

# Another option to run the for loop involves Python's enumerate method
# This method obtains both the index and the value of an item during a for loop
# for index, candy in candy_list:
#     print(index, candy)

# 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)
```

Activities > 03-Stu_KidInCandyStore-LoopsRecap > Solved > kid_in_candy_store_bonus_solution.py > ...

```
# 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])


# Set answer to "yes" for while loop
answer = "yes"

while answer == "yes":

    # Ask which candy the user would like to bring ho
    print("Which candy would you like to bring home?")
    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)])
```

```
35
36      # ask the user if they want more candy
37      answer = input("Would you like to make another selection? ('yes' or 'no') ")
38
39
40  # Loop through the candy_cart to say what candies were brought home
41  print("I brought home with me...")
42  for candy in candy_cart:
43      print(candy)
44
```

Ln 7, Col 20 Spaces: 4 UTF-8

Activity 4
House of pies
0:56:00
1:16:00

In this activity, you will build an order form that displays a list of pies and then prompts users to make a selection. The form will continue to prompt for selections until the user decides to end the process.

Activities > 04-Stu_HouseOfPies-AdvancedLoops > README.md > # House of Pies

```
1   # House of Pies
2
3   In this activity, you will build an order form that displays a list of pies and then prompts users to make a
    selection. The form will continue to prompt for selections until the user decides to end the process.
4
5   ## Instructions
6
7   * Create an order form that displays a list of pies to the user in the following way:
8
9
10  ```
11  Welcome to the House of Pies! Here are our pies:
12
13  ----------------------------------------------------------------------
14  (1) Pecan, (2) Apple Crisp, (3) Bean, (4) Banoffee,  (5) Black Bun, (6) Blueberry, (7) Buko, (8) Burek,  (9) Tamale,
    (10) Steak
15  ```
16
17  * Then, prompt the user to enter the number for the pie they would like to order.
18
19  * Immediately follow up their 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.
20
21  * Once the user is done purchasing pies, print the total number of pies ordered.
22
23  ## Bonus
24
25  * Modify the application so that at the conclusion of the transaction, the user's purchases are listed out, with the
    total pie count broken by _each_ pie. For example:
26
27  ```
28  You purchased:
29  0 Pecan
30  0 Apple Crisp
31  0 Bean
```

```
2 Banoffee
0 Black Bun
0 Blueberry
0 Buko
0 Burek
0 Tamale
1 Steak
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Markd

Activities > 04-Stu_HouseOfPies-AdvancedLoops > Solved > house_of_pies_solution.py > ...

```python
# 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)) + ".")
```

Activities > 04-Stu_HouseOfPies-AdvancedLoops > Solved > house_of_pies_bonus_solution.py > ...

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

# List to track pie purchases
pie_purchases = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

# 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:")
```

```
13
14  # While we are still shopping...
15  while shopping == "y":
16
17      # Show pie selection prompt
18      print("-------------------------------------------------------------------------")
19      print("(1) Pecan, (2) Apple Crisp, (3) Bean, (4) Banoffee, " +
20            " (5) Black Bun, (6) Blueberry, (7) Buko, (8) Burek, " +
21            " (9) Tamale, (10) Steak ")
22
23      pie_choice = input("Which would you like? ")
24
25      # Get index of the pie from the selected number
26      choice_index = int(pie_choice) - 1
27
28      # Add pie to the pie list by finding the matching index and adding one to its value
29      pie_purchases[choice_index] += 1
30
31      print("-------------------------------------------------------------------------")
32
33      # Inform the customer of the pie purchase
34      print("Great! We'll have that " + pie_list[choice_index] + " right out for you.")
35
36      # Provide exit option
37      shopping = input("Would you like to make another purchase: (y)es or (n)o? ")
38
39  # Once the pie list is complete
40  print("-------------------------------------------------------------------------")
41
42  # Count instances of each pie
43  print("You purchased: ")
44
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF {}

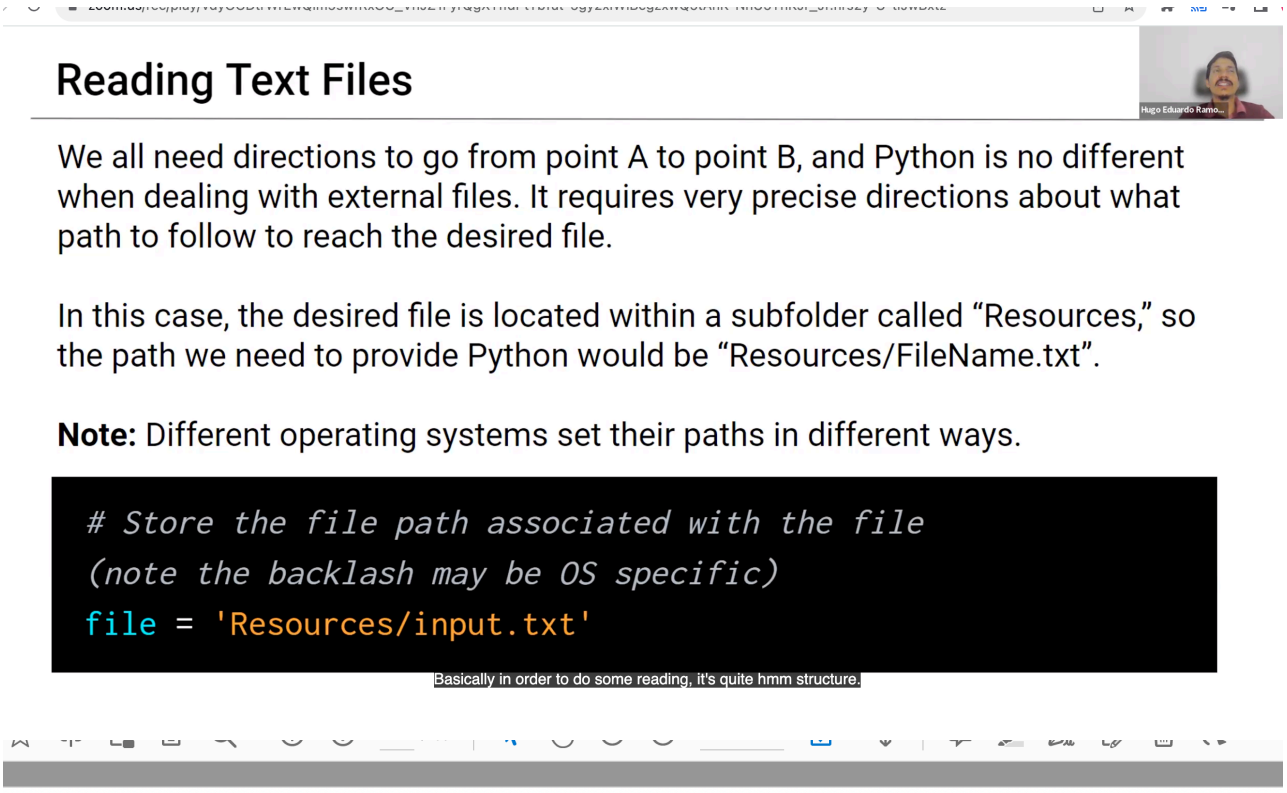
Activities > 04-Stu_HouseOfPies-AdvancedLoops > Solved > house_of_pies_bonus_solution.py > ...

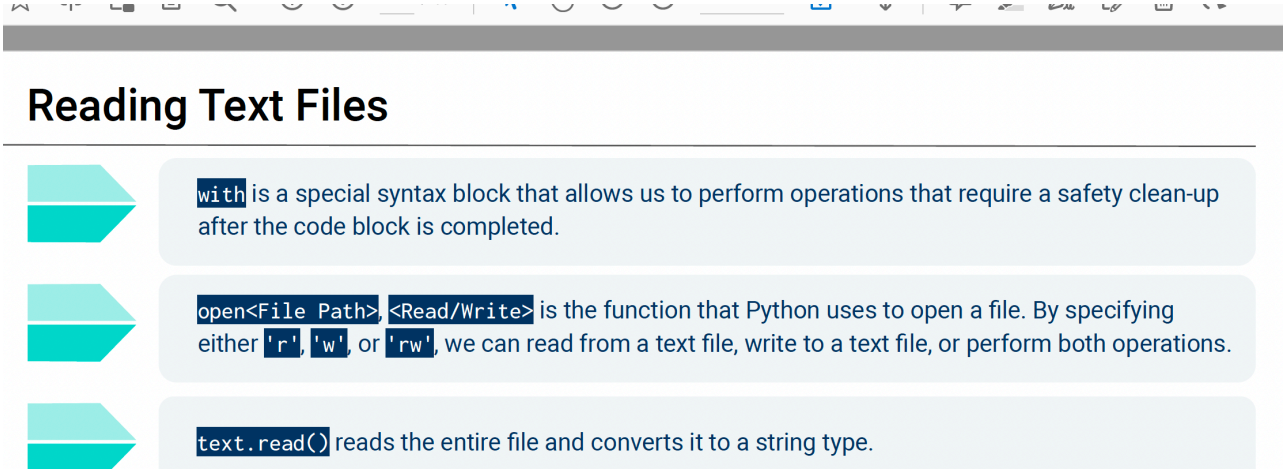
```
12  print("Welcome to the House of Pies! Here are our pies:")
13
14  # While we are still shopping...
15  while shopping == "y":
16
17      # Show pie selection prompt
18      print("-------------------------------------------------------------------------")
19      print("(1) Pecan, (2) Apple Crisp, (3) Bean, (4) Banoffee, " +
20            " (5) Black Bun, (6) Blueberry, (7) Buko, (8) Burek, " +
21            " (9) Tamale, (10) Steak ")
22
23      pie_choice = input("Which would you like? ")
24
25      # Get index of the pie from the selected number
26      choice_index = int(pie_choice) - 1
27
28      # Add pie to the pie list by finding the matching index and adding one to its value
29      pie_purchases[choice_index] += 1
30
31      print("-------------------------------------------------------------------------")
32
33      # Inform the customer of the pie purchase
34      print("Great! We'll have that " + pie_list[choice_index] + " right out for you.")
35
36      # Provide exit option
37      shopping = input("Would you like to make another purchase: (y)es or (n)o? ")
38
39  # Once the pie list is complete
```

```
41
42  # Count instances of each pie
43  print("You purchased: ")
44
45  # Loop through the full pie list
46  for pie_index in range(len(pie_list)):
47      pie_count = str(pie_purchases[pie_index])
48      pie_name = str(pie_list[pie_index])
49
50      # Gather the count of each pie in the pie list and print them alongside the pies
51      print(pie_count + " " + pie_name)
52
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF

1:27:00





```
# 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)
```

1:29:00
Activity 5
Basic read

We open external files
We input form a text files called input.txt
We read from text file

Activities > 05-Ins_BasicRead > Solved > read_file_solution.py > ...

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

# Open the file in "read" mode ('r') and store the contents in the variable "text"
with open(file, 'r') as text:

    # This stores a reference to a file stream
    print(text)

    # Store all of the text inside a variable called "lines"
    lines = text.read()

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

```

1:35:00
Activity 6

What are modules stings and random

Activities > 06-Ins_Modules > Solved > imports_solution.py > ...

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

# Utilize the string module's custom method: ".ascii_letters"
```

```
6   print(string.ascii_letters)
7
8   # Utilize the random module's custom method randint
9   for x in range(10):
10      print(random.randint(1, 10))
11
```

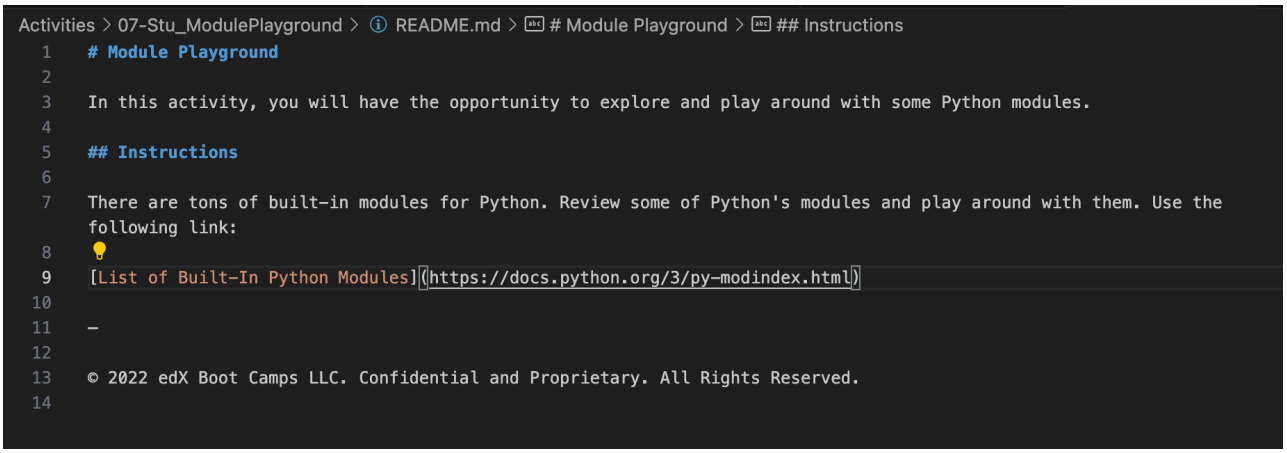
1:40:00
Activity 7

These are built in python modules

THIS WEBSITE

https://docs.python.org/3/py-modindex.html

```
# Module Playground
In this activity, you will have the opportunity to explore and play
around with some Python modules.
## Instructions
There are tons of built-in modules for Python. Review some of Python's
modules and play around with them. Use the following link:
[List of Built-In Python Modules](https://docs.python.org/3/py-
modindex.html)
—
© 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights
Reserved.
```



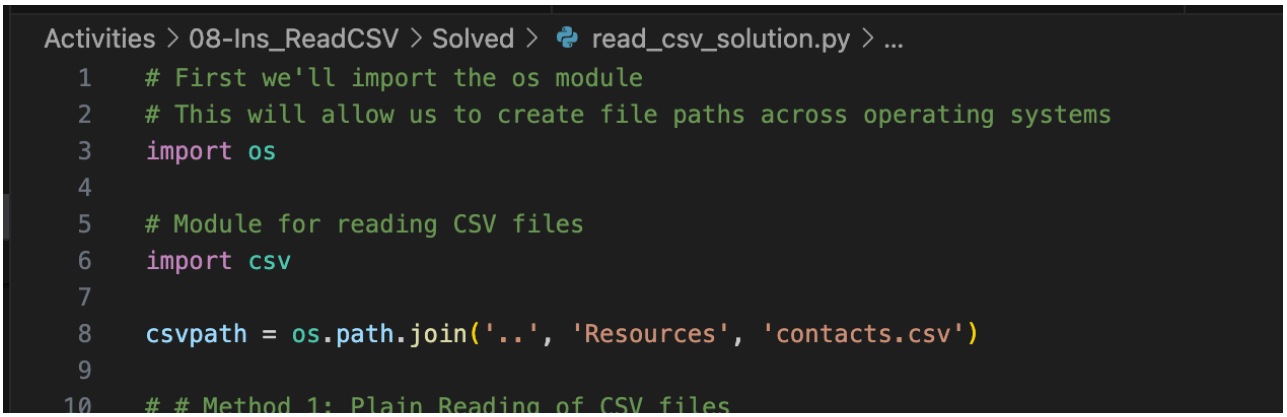
02:00:00

Remember to open the file and the resources file in the same folder to run code .

What are csv
How to read csv
What is a delimiter

Activity 8

2:03:00



```
# with open(csvpath, 'r') as file_handler:
#     lines = file_handler.read()
#     print(lines)
#     print(type(lines))


# Method 2: Improved Reading using CSV module

with open(csvpath) as csvfile:

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

    print(csvreader)

    # Read the header row first (skip this step if there is no header)
    csv_header = next(csvreader)
    print(f"CSV Header: {csv_header}")

    # Read each row of data after the header
    for row in csvreader:
        print(row)

```

Activity 09
2:12:05
ReadComicsBooks

2:23:00
CONTINUE

Activities > 09-Stu_ReadComicBooksCSV > README.md > # Reading Comic Book Data > ## Instructions

```
# Reading Comic Book Data

In this activity, you will create an application that searches the provided CSV file for a specific graphic novel title and then returns the title, publisher's name, and the year it was published.

## Instructions

* Prompt the user for the book title they'd like to search.

* Read in the CSV using the UTF-8 encoding, due to foreign characters.

* Search through the `comic_books.csv` to find the user's book.

* If the CSV contains the user's title, then print out the title, the publisher name, and the year it was published.

    * For example: `"Alien was published by DC Comics in 2015"`.

* If the CSV does not contain the user's title, then print out a message telling them that their book could not be found.

    * Set a variable to `False` to check if we found the comic book.
```

```
    * In the `for` loop, change the variable to confirm that the comic book is found.

## References

Data modified from "Comic books CSV" Updated April 2021. Initially released in 2014 to accompany the British Library's exhibition Comics Unmasked. [https://www.bl.uk/collection-metadata/downloads](https://www.bl.uk/collection-metadata/downloads)

-


```

Activities > 09-Stu_ReadComicBooksCSV > Solved > comicbooks_solution.py > ...

```python
# Modules
import os
import csv

# Prompt user for title lookup
book = input("What title are you looking for? ")

# Set path for file
csvpath = os.path.join("..", "Resources", "comic_books.csv")

# Set variable to check if we found the video
found = False

# Open the CSV using the UTF-8 encoding
with open(csvpath, encoding='UTF-8') as csvfile:
    csvreader = csv.reader(csvfile, delimiter=",")

    # Loop through looking for the video
    for row in csvreader:
        if row[0] == book:
            print(row[0] + " was published by " + row[8] + " in " + row[9])

            # Set variable to confirm we have found the video
            found = True

    # If the book is never found, alert the user
    if found is False:
        print("Sorry about this, we don't seem to have what you are looking for!")

```

2:30:00
Activity 10

Activities > 10-Ins_WriteCSV > Solved > write_solution.py > ...

```python
# Dependencies
import os
import csv

```

```python
# 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') 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'])

```

Activity 11
2:35:00

Activities > 11-Ins_Zip > Solved > zipper_solution.py > ...

```python
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)

# Print the contents of each row
for employee in roster:
    print(employee)

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

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

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

    writer.writerows(roster)


# # to print out to terminal:
# #comment out above code and run the code below
# for employee in roster:
#     print(employee)

```

2:39:00
Activity 13 (we skipped 12)

Functions helps us not to repeat code

Activities > 13-Ins_Functions > Solved > functions_solution.py > ...

```python
# Define the function and tell it to print "Hello!" when called
def print_hello():
    print(f"Hello!")


# Call the function within the application to ensure the code is run
print_hello()
# -------------#


# Functions that take in and use parameters can also be defined
def print_name(name):
    print("Hello " + name + "!")


# When calling a function with a parameter, a parameter must be passed into the function
print_name("Bob Smith")
# -------------#

# The prime use case for functions is in being able to run the same code for different values
list_one = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list_two = [11, 12, 13, 14, 15]


def list_information(simple_list):
    print(f"The values within the list are...")
    for value in simple_list:
        print(value)
    print("The length of this list is... " + str(len(simple_list)))


list_information(list_one)
list_information(list_two)
```

Acticity 12
2:59:00

Activities > 12-Stu_CensusZip > README.md > # U.S. Census Breakdown > ## Instructions

```
# U.S. Census Breakdown

In this activity, you will be provided with a large dataset from the 2019 U.S. Census. Your task is to clean up this
dataset and create a new CSV file that is easier to comprehend.

```

```
## Instructions

* Create a Python application that reads in the data from the 2019 U.S. Census.

* Then, store the contents of `Place`, `Population`, `Per Capita Income`, and `Poverty Count` 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.

## Bonus

* Find the poverty rate (percentage of population living in poverty). Include this in your final output, converting the rate to a string and including a "%" at the end of the string.

* Parse the string associated with `Place`, separating it into `County` and `State`, so we can store both in separate columns.

## Hints

* Windows users may get a `UnicodeDecodeError`. To avoid this, pass in `encoding="utf8"` as an additional parameter when reading in the file.

* As with many datasets, the file does not include the header line. Use the following list as a guide to the columns: "Place,Population,Median Age,Household Income,Per Capita Income,Employed Civilians,Unemployed Civilians,People in the Military,Poverty Count"

## References

Data Source: [U.S. Census API - ACS 5-Year Estimates 2019](https://www.census.gov/data/developers/data-sets/census-microdata-api.ACS_5-Year_PUMS.html)

-
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

Activities > 12-Stu_CensusZip > Solved > census_solution.py > ...

```python
import os
import csv

census_csv = os.path.join("..", "Resources", "census_starter.csv")

# Lists to store data
place = []
population = []
income = []
poverty_count = []
poverty_rate = []
county = []
state = []

# with open(udemy_csv, encoding='utf-8') as csvfile:
with open(census_csv) as csvfile:
    csvreader = csv.reader(csvfile, delimiter=",")
    for row in csvreader:
        # Add place
        place.append(row[0])

        # Add population
        population.append(row[1])

        # Add per capita income
        income.append(row[4])

        # Add poverty count
        poverty_count.append(row[8])

        # Determine poverty rate to 2 decimal places, convert to string
        percent = round(int(row[8]) / int(row[1]) * 100, 2)
        poverty_rate.append(str(percent) + "%")
```

```
        # Split the place into county and state
        split_place = row[0].split(", ")
        county.append(split_place[0])
        state.append(split_place[1])

# Zip lists together
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

Activities > 12-Stu_CensusZip > Solved > census_solution.py > ...

```
        place.append(row[0])

        # Add population
        population.append(row[1])

        # Add per capita income
        income.append(row[4])

        # Add poverty count
        poverty_count.append(row[8])

        # Determine poverty rate to 2 decimal places, convert to string
        percent = round(int(row[8]) / int(row[1]) * 100, 2)
        poverty_rate.append(str(percent) + "%")

        # Split the place into county and state
        split_place = row[0].split(", ")
        county.append(split_place[0])
        state.append(split_place[1])

# Zip lists together
cleaned_csv = zip(place, population, income, poverty_count, poverty_rate, county, state)

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

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

    # Write the header row
    writer.writerow(["Place", "Population", "Per Capita Income", "Poverty Count", "Poverty Rate",
                     "County", "State"])

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

[Reading, Writing, and Pyrithmetic]