

Reading and Writing in Python

Data Boot Camp

Lesson 3.2



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.



Create and use Python functions.



Activity: Python Check-Up

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

Suggested Time:

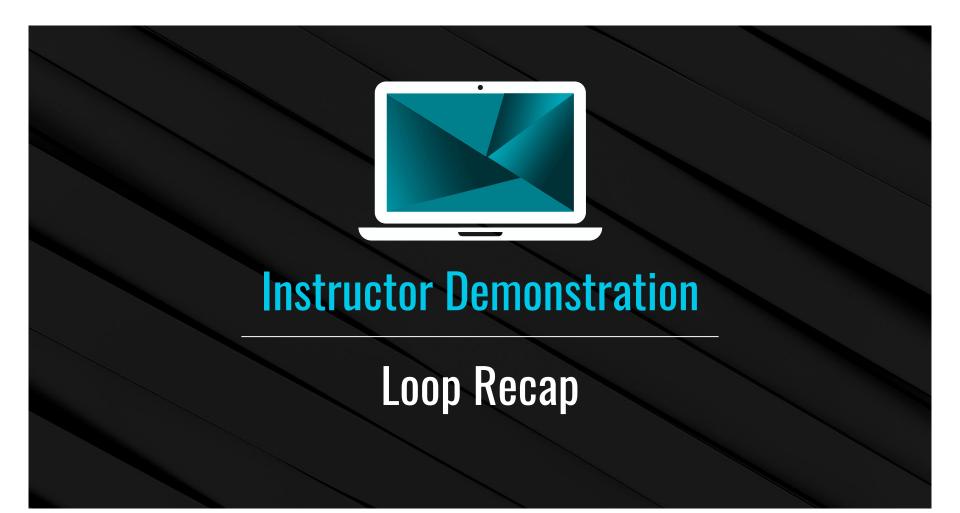
10 minutes

Activity: Python Check-Up

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







What Is a for Loop?

Loops through a range of numbers, the letters in a string, or the elements within a list one by one.

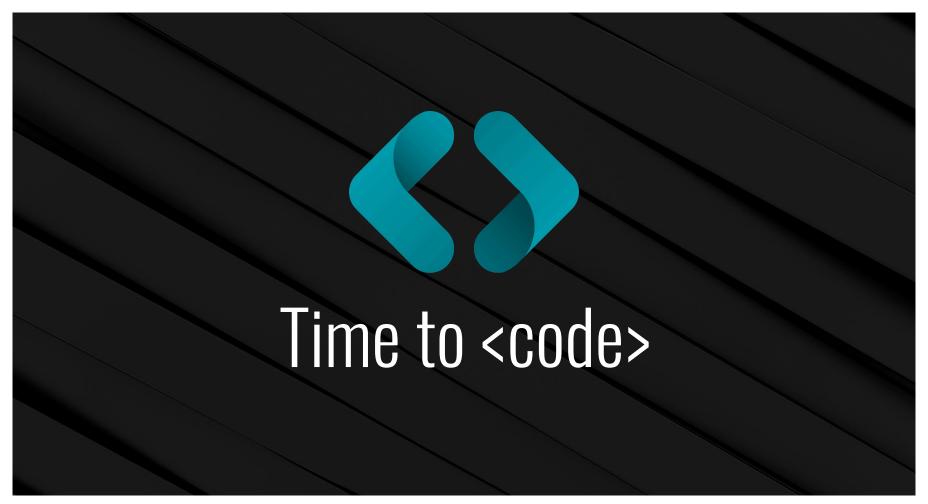
```
jacob@DESKTOP-0ICJMMD MINGW64 ~/OneDrive/Documents/WorkAndSchool/TeachingAssistant
/DataViz/DataViz-Lesson-Plans/01-Lesson-Plans/03-Python/2/Activities/02-Ins_Simple
Loops (Scramble-Branch)
$ python SimpleLoops.py
```

7

What Is a while Loop?

Loops through the code contained inside of it until some condition is met.

```
jacob@DESKTOP-0ICJMMD MINGW64 ~/OneDrive/Documents/WorkAndSchool/TeachingAssistant/
DataViz/DataViz-Lesson-Plans/01-Lesson-Plans/03-Python/2/Activities/02-Ins_SimpleLo
ops (Scramble-Branch)
$ python SimpleLoops.py
```





Activity: Kid in a Lolly Shop

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

Suggested Time:

15 minutes

Activity: Kid in a Lolly Shop

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] Fantales"

Create a second loop that runs for a 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 a matching index to the variable **lolly_cart**.

For example: If the user enters "0" as their input, "Fantales" should be added into the lolly_cart list.

Use another loop to print all of the selected lollies to the terminal.

Bonus

Create a version of the code that allows a user to select as many lollies as they want, up until they say they do not want any more.





Activity: House of Pies

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

Suggested Time:

20 minutes

Activity: House of Pies

Part 1 instructions:



Create an order form that displays a list of pies to the user in the following way:

Welcome to the House of Pies Here are our pies:

- (1) Aussie Beef, (2) Steak and Kidney, (3) Chicken, (4) Shepherd's, (5) Spinach and Feta, (6) Curry,
- (7) Lamb and Rosemary, (8) Steak and Mushroom, (9) Apple, (10) Lemon Meringue



Then, prompt the user to enter the number for the pie they'd like to order.



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



Once the user is done purchasing pies, print the total number of pies ordered.

Activity: House of Pies

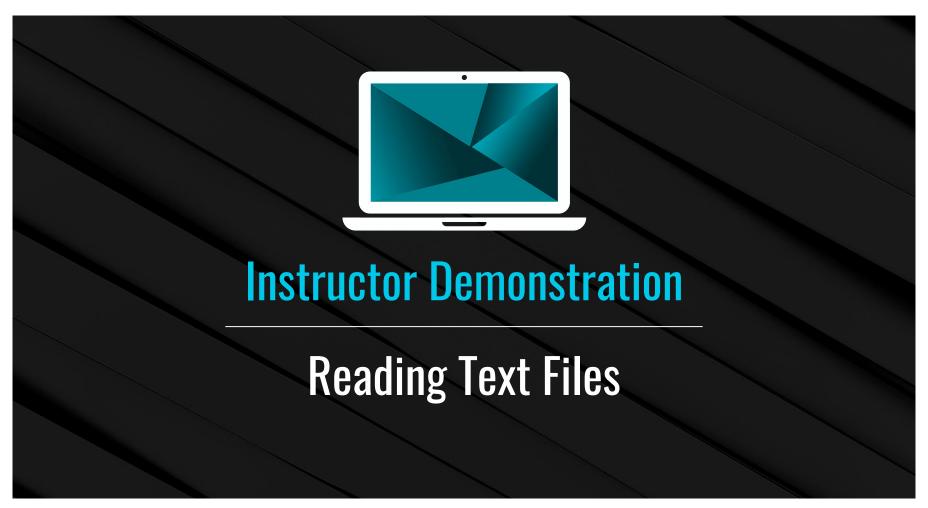
Bonus: Modify the application so that at the conclusion of the transaction, the user's purchases are listed out, with the total pie count broken by each pie.

For example:

You purchased:

- 0 Aussie Beef
- 0 Steak and Kidney
- 0 Chicken
- 2 Shepherd's
- 0 Spinach and Feta
- 0 Curry
- 0 Lamb and Rosemary
- 0 Steak and Mushroom
- 0 Apple
- 1 Lemon Meringue







Reading Text Files

We all need directions to go from point A to point B, and Python is no different when dealing with external files. It requires very precise directions about what path to follow to reach the desired file.

In this case, the desired file is located within a subfolder called "Resources," so the path we need to provide Python would be "Resources/FileName.txt".

Note: Different operating systems set their paths in different ways.

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

Reading Text Files



with is a special syntax block that allows us to perform operations that require a safety clean-up after the code block is completed.



open<File Path>, <Read/Write> is the function that Python uses to open a file. By specifying either 'r', 'w', or 'rw', we can read from a text file, write to a text file, or perform both operations.

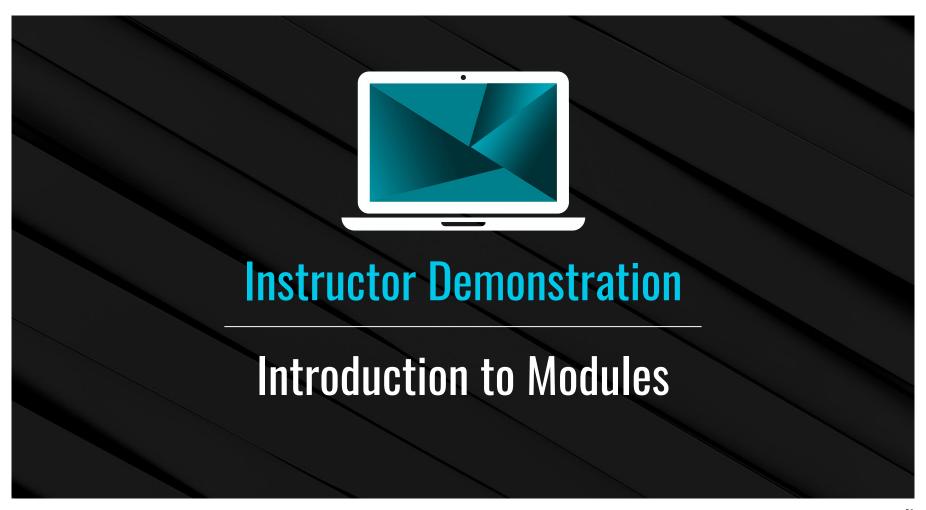


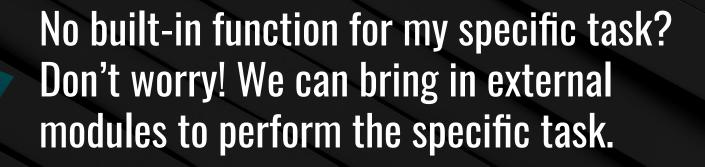
text.read() reads the entire file and converts it to a string type.

```
# 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

Import Modules

The string module contains many helpful constants and methods that pertain to strings. For example, we can use string.ascii_letters, and Python will instantly grab a reference to every ASCII character.

```
jacob@DESKTOP-@ICJMMD MINGW64 ~/OneDrive/Documents/WorkAndSchool/TeachingAssistant
/DataViz/DataViz-Lesson-Plans/01-Lesson-Plans/03-Python/2/Activities/06-Ins_Module
s (Scramble-Branch)
$ python imports.py
```

```
# Import the String Module
import string

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

Introduction to Modules

Import Modules

The random module does exactly what we might expect: it allows Python to randomly select values from set ranges, lists, or even strings.

```
TOP-0ICJMMD MINGW64 ~/OneDrive/Documents/WorkAndSchool/TeachingAssistant
/DataViz/DataViz-Lesson-Plans/01-Lesson-Plans/03-Python/2/Activities/06-Ins Module
(Scramble-Branch)
python imports.py
```





Activity: Module Playground

In this activity, you will explore some Python modules.

Suggested Time:

5 minutes

Activity: Module Playground

Instructions	There are tons of built-in modules for Python, and there is no possible way that a single class could cover all of them.
	Take a moment to explore a built-in Python module, then share what you uncovered.
Hint	Use your expert Google skills!









Reading in CSV Files

Comma Separated Values

- CSV stands for comma separated values. This
 file type is essentially a table that has been
 converted into text format with each row and
 column separated by specified symbols.
- More often than not, each row is located on a new line, and each column is separated by a comma, as indicated in the name "CSV."

First Name, Last Name, Phone Janetta, Bolduc, 499-820-0212 Bent, Hanburry, 125-890-6291 Kath, Beeres, 807-511-9864 Clarisse, Surgeon, 499-224-5982 Hae-Won, Park, 727-224-1623 Rodd, Camier, 199-541-8033 Javier,Martinez,543-206-4422 Patty, De'Ath, 950-579-4341 Charlie, Clewlow, 874-246-8418 Izel,Xiu,362-965-1637 Zechariah, Spikings, 570-486-2219 Stephie, Tootal, 326-912-0003

Python has a module called csv that pulls in data from external CSV files and allows users to perform operations on the data.

Reading in CSV Files

os module

This module allows Python programmers to easily create dynamic paths to external files that will function across different operating systems.

```
# First we'll import us module
# This will allow us to create file path across operating systems
import os
# Module for reading CSV files
import csv
csvpath = os.path.join('..', 'Resources', 'contacts.csv')
```

Reading in CSV Files

csv reader

Instead of text.read(), this new code instead uses csv.reader() to translate the object being opened by Python. Note the delimiter="csv.reader() to translate the object being opened by Python. Note the delimiter="csv.reader() to translate the object being opened by Python. Note the delimiter="csv.reader() parameter, which tells Python that each comma within the CSV, should be seen as moving into a new

column for a row.

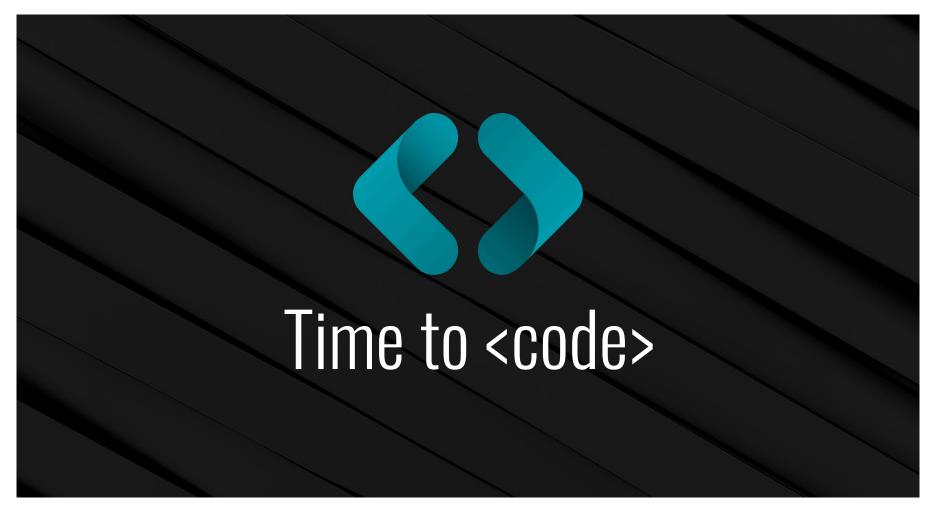
```
with open(csvpath) as csvfile:

# 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 now 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: 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.

Suggested Time:

15 minutes

Activity: Reading Comic Book Data

Instructions

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

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

If the CSV contains the user's book, then print out the title, the publisher's 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 then 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.







Writing CSV Files

Lost in Translation?

- os.path.join("..", "output",
 "new.csv") tells Python the file to
 write to while assigning it to the
 variable output_path.
- with open(output_path, 'w') as csvfile: tells Python to open the file by using write mode, while holding the contents in output_path.
- csv.writer() tells Python that this application will write code into an external CSV file.
- csv.writerow() is the code to write a new row into a CSV file.

```
import os
import csv
output_path = os.path.join("..", "output", "new.csv")
with open(output_path, 'w') as csvfile:
    csvwriter = csv.writer(csvfile, delimiter=',')
    csvwriter.writerow(['First Name', 'Last Name', 'SSN'])
    csvwriter.writerow(['Caleb', 'Frost', '505-80-2901'])
```



Python users can write data into a new CSV file more efficiently by using the zip() function.

Zipping Lists

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

By zipping these lists together, there is now a single, joined list whose indexes reference all three of the lists inside.

Each zipped object can be used only once. For example, you can write the zipped object to a CSV or print to the terminal, but not both.

```
$ python zipper.py
(1, 'Micheal', 'Boss')
(2, 'Dwight', 'Sales')
(4, 'Meredith', 'Sales')
(4, 'Kelly', 'HR')
```

```
# Three Lists
indexes = [1, 2, 3, 4]
employees = ["Micheal", "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 employees in roster:
   print(employee)
```





Activity: Absenteeism at Work Zip

In this activity, you will be provided with a large dataset related to absenteeism at work. Your task is to clean up this dataset and create a new CSV file that is easier to comprehend.

Suggested Time:

20 minutes

Activity: Absenteeism at Work Zip

Instructions

- Create a Python application that reads in the data.
- Then, store the contents of Reason for Absence, Distance From Work, Service Time, Age, and Absenteeism Time in Hours 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.

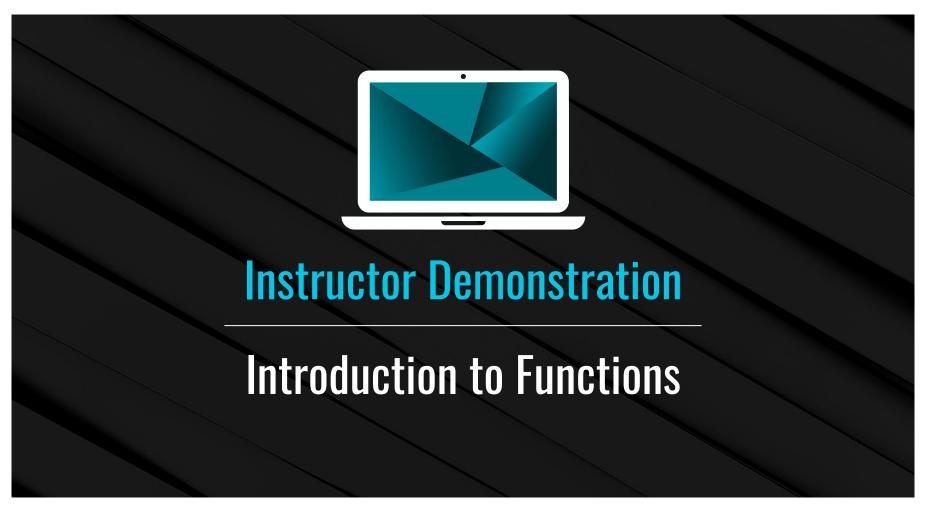
Hints

- Windows users may get a
 UnicodeDecodeError
 . To avoid this, pass in encoding=cutf8 as an additional parameter when reading in the file.
 - Use the the list provided in the README as a guide to the columns in the CSV file.

Bonus

- Find the percentage of work
 missed given an eight hour work
 day. 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.







Introduction to Functions

Prevent repetition with frequent use of Python functions.

- A function is a block of organised, 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.
- 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.
- To run the code stored within a function, the function itself must be called within the program. Functions will not run unless called upon.
- 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.

```
def print_hello():
  print(f"Hello!")
print_hello()
def print_name(name):
    print("Hello " + name + "!")
print_name("Bob Smith")
```

