**Week 3 - Python**

**first\_python.script.py**

(C:\Users\malin\Desktop\Bootcamp\MONU-VIRT-DATA-PT-11-2023-U-LOLC\03-Python\1)

first\_number = input("first number: ")

second\_number = input("Second number: ")

if(first\_number == second\_number):

    print("first number is equal to the second number")

else:

    print("first number is NOT equal to the second number")

**print("My name is " + name + " and I am " + str(age) + " years old")**

name = "Malini"

age = 25

print("My name is " + name + " and I am " + str(age) + " years old")

**Day1, Activity 3**

# Creates a variable with a string "Frankfurter"

title = "Frankfurter"

# Creates a variable with an integer 80

years = 80

# Creates a variable with the boolean value of True

expert\_status = True

# Prints a statement adding the variable

print("Nick is a professional " + title)

# Convert the integer years into a string and prints

print("He has been coding for " + str(years) + " years")

# Converts a boolean into a string and prints

print("Expert status: " + str(expert\_status))

# An f-string accepts all data types without conversion

print(f"Expert status: {expert\_status}")

**Day1, Activity 4**

 Create a variable called 'name' that holds a string

name = "Shannon Liu"

# Create a variable called 'country' that holds a string

country = "Australia"

# Create a variable called 'age' that holds an integer

age = 28

# Create a variable called 'hourly\_wage' that holds an integer

hourly\_wage = 25

# Calculate the daily wage for the user

daily\_wage = hourly\_wage \* 8

# Create a variable called 'satisfied' that holds a boolean

satisfied = True

# Print out "Hello <name>!"

print("Hello " + name + "!")

# Print out what country the user entered

print("You live in " + country)

# Print out the user's age

print("You are " + str(age) + " years old")

# With an f-string, print out the daily wage that was calculated

print(f"You make {daily\_wage} per day")

# With an f-string, print out whether the users were satisfied

print(f"Are you satisfied with your current wage? {satisfied}")

**Day1, Activity 5**

# Collects the user's input for the prompt "What is your name?"

name = input("What is your name? ")

# Collects the user's input for the prompt "How old are you?" and converts the string to an integer.

age = int(input("How old are you? "))

# Collects the user's input for the prompt "Is input truthy?" and converts it to a boolean. Note that non-zero,

#   non-empty objects are truth-y.

trueOrFalse = bool(input("Is the input truthy? "))

# Creates three print statements that to respond with the output.

print("My name is " + str(name))

print("I will be " + str(age + 1) + " next year.")

print("The input was converted to " + str(trueOrFalse))

**Day1, Activity 6**

# Take input of you and your neighbour

your\_first\_name = input("What is your name? ")

neighbour\_first\_name = input("What is your neighbour's name? ")

# Take how long each of you have been coding

months\_you\_coded = input("How many months have you been coding? ")

months\_neighbour\_coded = input("How many months has your neighbour been coding? ")

# Add total months

total\_months\_coded = int(months\_you\_coded) + int(months\_neighbour\_coded)

# Print results

print("I am " + your\_first\_name + " and my neighbour is " + neighbour\_first\_name)

print("Together we have been coding for " + str(total\_months\_coded) + " months!")

**Day1, Activity 7**

x = 1

y = 10

# Checks if one value is equal to another

if x == 1:

    print("x is equal to 1")

# Checks if one value is NOT equal to another

if y != 1:

    print("y is not equal to 1")

# Checks if one value is less than another

if x < y:

    print("x is less than y")

# Checks if one value is greater than another

if y > x:

    print("y is greater than x")

# Checks if a value is greater than or equal to another

if x >= 1:

    print("x is greater than or equal to 1")

# Checks for two conditions to be met using "and"

if x == 1 and y == 10:

    print("Both values returned true")

# Checks if either of two conditions is met

if x < 45 or y < 5:

    print("One or more of the statements were true")

# Nested if statements

if x < 10:

    if y < 5:

        print("x is less than 10 and y is less than 5")

    elif y == 5:

        print("x is less than 10 and y is equal to 5")

    else:

        print("x is less than 10 and y is greater than 5")

**Day1, Activity 8 – Conditionals**

# 1.

x = 5

if 2 \* x > 10:

    print("Question 1 works!")

else:

    print("oooo needs some work")

# 2.

x = 5

if len("Dog") < x:

    print("Question 2 works!")

else:

    print("Still missing out")

# 2. 1  -- San & Mal Test

x = 5

if len("Elephant") < x:

    print("Question 2 works!")

else:

    print("Still missing out")

# 3.

x = 2

y = 5

if (x \*\* 3 >= y) and (y \*\* 2 < 26):

    print("GOT QUESTION 3!")

else:

    print("Oh good you can count")

**# 3.1 -- San & Mal Test**

x = 1

y = 6

if (x \*\* 3 >= y) and (y \*\* 2 < 26):

    print("Both conditions are fullfilled")

elif (x \*\* 3 >= y) or (y \*\* 2 < 26):

    print("Only one of the conditions is met")

else:

    print("Both of the conditions are NOT met")

# 4.

name = "Dan"

group\_one = ["Greg", "Tony", "Susan"]

group\_two = ["Gerald", "Paul", "Ryder"]

group\_three = ["Carla", "Dan", "Jefferson"]

if name in group\_one:

    print(name + " is in the first group")

elif name in group\_two:

    print(name + " is in group two")

elif name in group\_three:

    print(name + " is in group three")

else:

    print(name + " does not have a group")

**# 4.1 - Mal & San Test**

name = input("What is your name? ")

group\_one = ["Greg", "Tony", "Susan"]

group\_two = ["Gerald", "Paul", "Ryder"]

group\_three = ["Carla", "Dan", "Jefferson"]

if name in group\_one:

    print(name + " is in the first group")

elif name in group\_two:

    print(name + " is in group two")

elif name in group\_three:

    print(name + " is in group three")

else:

    print(name + " does not have a group")

# 5.

height = 66

age = 16

adult\_permission = True

if (height > 70) and (age >= 18):

    print("Can ride all the roller coasters")

elif (height > 65) and (age >= 18):

    print("Can ride moderate roller coasters")

elif (height > 60) and (age >= 18):

    print("Can ride light roller coasters")

elif ((height > 50) and (age >= 18)) or ((adult\_permission) and (height > 50)):

    print("Can ride bumper cars")

else:

    print("Stick to lazy river")

**Day 1 , Activity 9**

# Create a variable and set it as an List

myList = ["Kirra", 25, "Ahmed", 80]

print(myList)

# Adds an element onto the end of a List

myList.append("Matt")

print(myList)

# Returns the index of the first object with a matching value

print(myList.index("Matt"))

# Changes a specified element within an List at the given index

myList[3] = 85

print(myList)

# Returns the length of the List

print(len(myList))

# Removes a specified object from an List

myList.remove("Matt")

print(myList)

# Removes the object at the index specified

myList.pop(0)

myList.pop(0)

print(myList)

# Creates a tuple, a sequence of immutable Python objects that cannot be changed

myTuple = ('Python', 100, 'VBA', False)

print(myTuple)

**Day 1, Activity 10 – Import Random (Computer 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? ")

# Run Conditionals

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!")

elif (user\_choice == "p" and computer\_choice == "s"):

    print("You chose paper. The computer chose scissors.")

    print("Sorry. You lose.")

elif (user\_choice == "p" and computer\_choice == "r"):

    print("You chose paper. The computer chose rock.")

    print("Yay! You won.")

elif (user\_choice == "s" and computer\_choice == "p"):

    print("You chose scissors. The computer chose paper.")

    print("Yay! You won.")

elif (user\_choice == "s" and computer\_choice == "s"):

    print("You chose scissors. The computer chose scissors.")

    print("A smashing tie!")

elif (user\_choice == "s" and computer\_choice == "r"):

    print("You chose scissors. The computer chose rock.")

    print("Sorry. You lose.")

else:

    print("I don't understand that!")

    print("Next time, choose from 'r', 'p', or 's'.")

**Day 1, Activity 10 – Loop**

# Loop through a range of numbers (0 through 4)

for x in range(5):

    print(x)

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

# Loop through a range of numbers (2 through 6) Start at 2 and count up

0, 1, 2, 3, 4, 5, 6 and the range to stop at 7, the count starts from 0

for x in range(2, 7):

    print(x)

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

The **range()** function in Python generates a sequence of numbers based on the specified parameters. When using **range(start, stop)**, it starts at the "start" value and goes up to, but does not include, the "stop" value. If only one parameter is provided, it assumes the range starts from 0.

# Iterate through letters in a string

word = "Peace"

for letter in word:

    print(letter)

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

# Iterate through a list

zoo = ["cow", "dog", "bee", "zebra"]

for animal in zoo:

    print(animal)

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

# Loop while a condition is being met

run = "y"

while run == "y":

    print("Hi!")

    run = input("To run again. Enter 'y': ")

**Day 2, Activity 1 – Recap**

# 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 Favourite Number

fave\_number = input("What is your favourite number? ")

# Respond Back with a statement based on your favourite number

if (int(fave\_number) < 7):

    print("Your favourite number is lower than mine.")

if (int(fave\_number) == 7):

    print("Your favourite number is the same as mine!")

if (int(fave\_number) > 7):

    print("Your favourite number is higher than mine.")

**Day 2, Activity 2 – Simple loop**

# 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? ")

**Day 2, Activity 3 – Lolly Shop (with allowance = 5 and F string)**

# The list of lollies to print to the screen

lolly\_list = [

    "Fantales",

    "Snakes Alive",

    "Minties",

    "Liquorice Allsorts",

    "Strawberries & Cream",

    "Life Savers",

    "Banana Lollies",

    "Fruit Pastilles",

    "Sherbies"

]

# The number of lollies the user will be allowed to choose

allowance = 5

# The list used to store all of the lollies selected inside of

lolly\_cart = []

# Print all of the candies to the screen and their index in brackets

for lolly in lolly\_list:

    print(f'[{str(lolly\_list.index(lolly))}] {lolly}')

# 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, lolly in lolly\_list:

#     print(index, lolly)

# Run through a loop which allows the user to choose which candies to take home with them

print("Which lolly would you like to bring home?")

for x in range(allowance):

    selected = input("Input the number of the lolly you want: ")

    # Add the lolly at the index chosen to the lolly\_cart list

    lolly\_cart.append(lolly\_list[int(selected)])

# Loop through the lolly\_cart to say what lollies were brought home

print("I brought home with me...")

for lolly in lolly\_cart:

    print(lolly)

**Day 2, Activity 3 – Lolly Shop (with no allowance)**

# The list of lollies to print to the screen

lolly\_list = [

    "Fantales",

    "Snakes Alive",

    "Minties",

    "Liquorice Allsorts",

    "Strawberries & Cream",

    "Life Savers",

    "Banana Lollies",

    "Fruit Pastilles",

    "Sherbies"

]

# The list used to store all of the lollies selected inside of

lolly\_cart = []

# Print all of the candies to the screen and their index in brackets

for i in range(len(lolly\_list)):

    print("[" + str(i) + "] " + lolly\_list[i])

# Set answer to "yes" for while loop

answer = "yes"

while answer == "yes":

    # Ask which lolly the user would like to bring home

    print("Which lolly would you like to bring home?")

    selected = input("Input the number of the lolly you want: ")

    # Add the lolly at the index chosen to the lolly\_cart list

    lolly\_cart.append(lolly\_list[int(selected)])

    # ask the user if they want more candy

    answer = input("Would you like to make another selection? ('yes' or 'no') ")

# Loop through the lolly\_cart to say what lollies were brought home

print("I brought home with me...")

for lolly in lolly\_cart:

    print(lolly)

**Day 2, Activity 4 – House of Pies**

# Initial variable to track shopping status

shopping = 'y'

# List to track pie purchases

pie\_purchases = []

# Pie List

pie\_list = ["Aussie Beef", "Steak and Kidney", "Chicken", "Shepherd's", "Spinach and Feta", "Curry", "Lamb and Rosemary", "Steak and Mushroom", "Apple", "Lemon Meringue"]

# 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) 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")

    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)) + ".")

**Day 2, Activity 4 – House of Pies (Bonus Question – Pie count)**

# 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 = ["Aussie Beef", "Steak and Kidney", "Chicken", "Shepherd's", "Spinach and Feta",

            "Curry", "Lamb and Rosemary", "Steak and Mushroom", "Apple", "Lemon Meringue"]

# 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) 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")

    pie\_choice = input("Which would you like? ")

    # Get index of the pie from the selected number

    choice\_index = int(pie\_choice) - 1

    # Add pie to the pie list by finding the matching index and adding one to its value

    pie\_purchases[choice\_index] += 1

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

    # Inform the customer of the pie purchase

    print("Great! We'll have that " + pie\_list[choice\_index] + " 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("------------------------------------------------------------------------")

# Count instances of each pie

print("You purchased: ")

# Loop through the full pie list

for pie\_index in range(len(pie\_list)):

    pie\_count = str(pie\_purchases[pie\_index])

    pie\_name = str(pie\_list[pie\_index])

    # Gather the count of each pie in the pie list and print them alongside the pies

    print(pie\_count + " " + pie\_name)

**Day 2, Activity 5 – To OPEN file and READ the file in Gitbash**

# Store the file path associated with the file (note the backslash may be OS specific)

file = 'file directory and name of file.txt' (in this example it was a text file)

# 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)

**Day 2, Activity 6 – Import random and string Module**

# Import the random and string Module

import random

import string

# Use the string module's custom method: ".ascii\_letters"

print(string.ascii\_letters)

# Use the random module's custom method randint

for x in range(10):

    print(random.randint(1, 10))

**Day 2, Activity 7 - skipped no file!!**

**Day2, Activity 8 – reading csv fie**

# First we'll import the os module

# This will allow us to create file paths across operating systems

import os

# Module for reading CSV files

import csv

csvpath = os.path.join('..', 'Resources', 'contacts.csv')

# # Method 1: Plain Reading of CSV files

# 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)

**Day2, Activity 9 – Comic Books.csv**

# Modules

import os

import csv

# Prompt user for title lookup

title = 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 Comic Book

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] == title:

            print(row[0] + " was published by " + row[8] + " in " + row[9])

            # Set variable to confirm we have found the Comic Book

            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!")

**\*\* Note - open and create files accordingly!!**

**Put in correct path in the coding!**

A screenshot of a computer

Description automatically generated**,**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Day2, Activity 10 – Creating CSV file adding contents in the file using python script**

# Dependencies

import os

import csv

# 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:

    # Initialise 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'])

**Day2, Activity 11 – Zip / Zipper**

The zip() function combines these three lists into a list of tuples called roster.

import csv

import os

# Three Lists

indexes = [1, 2, 3, 4, 5, 6, 7]

employees = ["Michael", "Dwight", "Meredith", "Kelly", "Malini", "Sanjay"]

department = ["Boss", "Sales", "Sales", "HR", "Finance", "Finance"]

# Zip all three lists together into tuples

roster = list(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", newline='') 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)

\*\* ‘W’ is for Write

**Day2, Activity 11 – Work Absenteeism**

import os

import csv

work\_csv = os.path.join("..", "Resources", "Absenteeism\_at\_work.csv")

# Lists to store data

reason\_for\_absence = []

distance\_from\_work = []

service\_time = []

age = []

hit\_target = []

absenteeism\_time\_in\_hours = []

percentage\_of\_day\_missed  = []

# with open(work\_csv, encoding='utf-8') as csvfile:

with open(work\_csv) as csvfile:

    csvreader = csv.reader(csvfile, delimiter=",")

    for row in csvreader:

        # Add reason for absence

        reason\_for\_absence.append(row[1])

        # Add distance from work

        distance\_from\_work.append(row[6])

        # Add service time

        service\_time.append(row[7])

        # Add age

        age.append(row[8])

        # Determine percentage of day missed, given an eight hour work day

        percent = round(int(float(row[20])) / 8 \* 100, 2)

        percentage\_of\_day\_missed.append(str(percent) + "%")

        # Add absenteeism time in hours

        absenteeism\_time\_in\_hours.append(row[20])

# Zip lists together

cleaned\_csv = list(zip(reason\_for\_absence, distance\_from\_work, service\_time, age, percentage\_of\_day\_missed, absenteeism\_time\_in\_hours))

# Set variable for output file

output\_file = os.path.join("absenteeism\_final.csv")

#  Open the output file

with open(output\_file, "w", newline='') as datafile:

    writer = csv.writer(datafile)

    # Write the header row

    writer.writerow(["Reason for Absence", "Distance from Work", "Service Time", "Age", "Percentage of Day Missed", "Absenteeism Time in Hours"])

    # Write in zipped rows

    writer.writerows(cleaned\_csv)

**Day 2, Activity 13 – Functions**

# 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("Malini Sintre")

# -------------#

# 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)

**Day 3, Activity 1 – Cereal**

import os

import csv

cereal\_csv = os.path.join("..", "Resources", "cereal.csv")

# Open and read csv

with open(cereal\_csv) as csv\_file:

    csv\_reader = csv.reader(csv\_file, delimiter=",")

    # Read the header row first (skip this part if there is no header)

    csv\_header = next(csv\_file)

    print(f"Header: {csv\_header}")

    # Read through each row of data after the header

    for row in csv\_reader:

        # Convert row to float and compare to grams of fibre

        if float(row[7]) >= 5:

            print(row)

**Day 3, Activity 1 – Cereal (Bonus) – without header**

import os

import csv

cereal\_csv = os.path.join("..", "Resources", "cereal\_bonus.csv")

with open(cereal\_csv) as csvfile:

    csv\_reader = csv.reader(csvfile, delimiter=",")

    # @NOTE: This time, we do not use `next(csv\_reader)` because there is no header for this file

    # Read through each row of data after the header

    for row in csv\_reader:

        # Convert row to float and compare to grams of fibre

        if float(row[7]) >= 5:

            print(row)

**Day 3, Activity 2 – Dictionaries (example: Actors)**

# Unlike lists, dictionaries store information in pairs

# ---------------------------------------------------------------

# Create a dictionary to hold the actor's names.

actors = {}

# Create a dictionary using the built-in function.

actors = dict()

# A dictionary of an actor.

actors = {"name": "Chris Hemsworth"}

print(f'{actors["name"]}')

# Add an actor to the dictionary with the key "name"

# and the value "Denzel Washington".

actors["name"] = "Denzel Washington"

# Print the actors dictionary.

print(actors)

# Print only the actor.

print(f'{actors["name"]}')

# A list of actors

actors\_list = [

    "Chris Hemsworth",

    "Angelina Jolie",

    "Kristen Stewart",

    "Denzel Washington"]

# Overwrite the value, "Denzel Washington", with the list of actors.

actors["name"] = actors\_list

# Print the first actor

print(f'{actors["name"][0]}')

# ---------------------------------------------------------------

# A dictionary can contain multiple pairs of information

actress = {

    "name": "Angelina Jolie",

    "genre": "Action",

    "nationality": "United States"

}

# ---------------------------------------------------------------

# A dictionary can contain multiple types of information

another\_actor = {

    "name": "Sylvester Stallone",

    "age": 62,

    "married": True,

    "best movies": [

        "Rocky",

        "Rocky 2",

        "Rocky 3"]}

print(f'{another\_actor["name"]} was in {another\_actor["best movies"][0]}')

# ---------------------------------------------------------------

# A dictionary can even contain another dictionary

film = {

    "title": "Interstellar",

    "revenues": {

        "United States": 360,

        "China": 250,

        "United Kingdom": 73

    }

}

print(f'{film["title"]} made {film["revenues"]["United States"]}'" million dollars in the US.")

# ---------------------------------------------------------------

**Day 3, Activity 3 – Dictionaries**

# Dictionary full of info

my\_info = {"name": "Rex",

           "occupation": "dog",

           "age": 21,

           "hobbies": ["barking", "eating", "sleeping", "loving my owner"],

           "wake-up": {"Mon": 5, "Friday": 5, "Saturday": 10, "Sunday": 9}}

# Print out results are stored in the dictionary

print(f'Hello I am {my\_info["name"]} and I am a {my\_info["occupation"]}')

print(f'I have {len(my\_info["hobbies"])} hobbies!')

print(f'On the weekend I get up at {my\_info["wake-up"]["Saturday"]}')

**Day 3, Activity 4 – Compressions (capital letter etc…)**

*https://www.tutorialspoint.com/python/string\_title.htm*

# Loop through each letter in the string

# and append to a list

letters = []

for letter in fish:

    letters.append(letter)

print(letters)

# List comprehensions provide concise syntax for creating lists

letters = [letter for letter in fish]

print(letters)

# We can manipulate each element as we go

capital\_letters = []

for letter in fish:

    capital\_letters.append(letter.upper())

print(capital\_letters)

# List Comprehension for the above

capital\_letters = [letter.upper() for letter in fish]

print(capital\_letters)

# We can also add conditional logic (if statements) to a list comprehension

jan\_temperatures = [30, 29, 33, 26, 41]

hot\_days = []

for temperature in jan\_temperatures:

    if temperature > 32:

        hot\_days.append(temperature)

print(hot\_days)

# List Comprehension with conditional

hot\_days = [temperature for temperature in jan\_temperatures if temperature > 32]

print(hot\_days)

**Day 3, Activity 5 – Compressions (Titlecased 🡪 Lowercase , Uppercase)**

https://www.tutorialspoint.com/python/string\_title.htm

# initialize list of names for user input

names = []

# Note that the i is unused and could be replaced by an \_

for i in range(2):

    name = input("Please enter the name of someone you know. ")

    names.append(name)

# Use a list comprehension to create a list of lowercased names

lowercased = [name.lower() for name in names]

# Use a list comprehension to create a list of titlecased names from the

# lowercased names. Reference: https://www.tutorialspoint.com/python/string\_title.htm

titlecased = [name.title() for name in lowercased]

# Print the invitations

invitations = [f"Dear {name}, please come to the wedding this Saturday!" for name in titlecased]

for invitation in invitations:

    print(invitation)

**Day 3, Activity 6 – Functions**

# Basic Definition

def name(parameters):

    # code goes here

    return

# Simple Function with no parameters

def show():

    print(f"Hi!")

# Use parentheses to run the code in a function

show()

# Simple function with one parameter

def show(message):

    print(message)

# Think of the parameter `message` as a variable

# You assign the string "Hello, World!" when you call the function

# This is like saying `message = "Hello, World!"`

show("Hello, World!")

# Functions can have more than one parameter

def make\_quesadilla(protein, topping):

    quesadilla = f"Here is a {protein} quesadilla with {topping}"

    print(quesadilla)

# Supply the arguments (values) when calling the function

make\_quesadilla("beef", "guacamole")

make\_quesadilla("chicken", "salsa")

# @NOTE: Order is important when supplying arguments!

make\_quesadilla("sour cream", "beef")

# We can also specify default values for parameters

def make\_quesadilla(protein, topping="sour cream"):

    quesadilla = f"Here is a {protein} quesadilla with {topping}"

    print(quesadilla)

# Make a quesadilla using the default topping

make\_quesadilla("chicken")

# Make a quesadilla with a new topping

make\_quesadilla("beef", "guacamole")

# Functions can return a value

def square(number):

    return number \* number

# You can save the value that is returned

squared = square(2)

print(squared)

# You can also just print the return value of a function

print(square(2))

print(square(3))

**Day 3, Activity 7 – Functions**

# Write a function that returns the arithmetic average for a list of numbers

def average(numbers):

    length = len(numbers)

    total = 0.0

    for number in numbers:

        total += number

    return total / length

# Test your function with the following:

print(average([1, 5, 9]))

print(average(range(11)))

**Day 3, Activity 8 – Functions (Time-off)**

import os

import csv

# Path to collect data from the Resources folder

pto\_hours\_csv = os.path.join('..', 'Resources', 'pto\_hours.csv')

# Define the function and have it accept the 'employee\_data' as its sole parameter

def print\_percentages(employee\_data):

    # For readability, it can help to assign your values to variables with descriptive names

    # CSV headers: Employee ID, PTO Hours (Allowed), PTO Hours (Taken), Sick Hours (Allowed),

    #  Sick Hours (Taken), Voting Time (Allowed), Voting Time (Taken)

    employee\_id = str(employee\_data[0])

    pto\_hours\_allowed = int(employee\_data[1])

    pto\_hours\_taken = int(employee\_data[2])

    sick\_hours\_allowed = int(employee\_data[3])

    sick\_hours\_taken = int(employee\_data[4])

    voting\_time\_allowed = int(employee\_data[5])

    voting\_time\_taken = int(employee\_data[6])

    # Total number of allowed time off days can be found by adding each category together

    total\_allowed = pto\_hours\_allowed + sick\_hours\_allowed + voting\_time\_allowed

    # Total number of taken time off days can be found by adding each category together

    total\_taken = pto\_hours\_taken + sick\_hours\_taken + voting\_time\_taken

    # Calculating the rate of use for PTO can be calculated by doing the following

    pto\_rate = (pto\_hours\_taken / pto\_hours\_allowed) \* 100

    # Note that some employees do not have sick time, so we will need to account for potential 0 sick time.

    # Sick hour usage rate can be calculated by doing the following

    if sick\_hours\_allowed == 0:

        sick\_hours\_rate = 0

    else:

        sick\_hours\_rate = (sick\_hours\_taken / sick\_hours\_allowed) \* 100

    # Calculate voting time usage rate by doing the following

    voting\_time\_usage\_rate = (voting\_time\_taken / voting\_time\_allowed) \* 100

    # Calculate the overall time off usage

    overall\_usage\_rate = (total\_taken / total\_allowed) \* 100

    # If the overall time off usage is greater than 50%, note Adequate Time Off

    # Otherwise it is "Encourage employee to take time off".

    if overall\_usage\_rate > 50:

        message = "Adequate Time Off"

    else:

        message = "Encourage employee to take time off"

    # Print out the employee's id and their time off usage rates.

    print(f"Stats for {employee\_id}")

    print(f"PTO Usage Rate: {str(pto\_rate)}")

    print(f"Sick Time Usage Rate: {str(sick\_hours\_rate)}")

    print(f"Voting Time Usage Rate: {str(voting\_time\_usage\_rate)}")

    print(f"Overall Time Off Usage: {str(overall\_usage\_rate)}")

    print(f"{message}")

# Read in the CSV file

with open(pto\_hours\_csv, 'r') as csvfile:

    # Split the data on commas

    csvreader = csv.reader(csvfile, delimiter=',')

    header = next(csvreader)

    # Prompt the user for what employee\_id they would like to search for

    employee\_id\_to\_check = input("Which employee\_id do you want to look for? ")

    # Loop through the data

    for row in csvreader:

        # If the employee\_id is in a row is equal to that which the user input, run the 'print\_percentages()' function

        if employee\_id\_to\_check == row[0]:

            print\_percentages(row)