```
# Name: Fahmi Omer
# UWO Email: fomer4@uwo.ca
# Assignment 3 - University Rankings
# Program Description: Take input from files and calculate various
attributes about it using the intaken values and write desired information
to a output text file

# Create function to call with the selected country, the name of the
university ranking file, and the name of the capitals file.
def getInformation(selectedCountry, rankingFileName, capitalsFileName):

    # Change the selected country input into the function to upper case
for usage
    selectedCountry = selectedCountry.upper()

    # Open the output.txt file to write to in the program
    outputFile = open("output.txt", "w")

    # Make empty lists to stor the data from the unin ranking file and the
capitals file
    topUniRows = []
    capitalsRows = []

    # Try-except clauses to open the files, put the header in a separate
list it doesnt interfere with the storage list, then appending each line
as a list into the storage list
    try:
        topUnis = open(rankingFileName, 'r', encoding='utf8')
    except IOError:
        outputFile.write("file not found")
        quit()

    topUnisHeaders = topUnis.readline().rstrip().split(",")
    for row in topUnis:
        topUniRows.append(row.rstrip().split(","))

    try:
        capitals = open(capitalsFileName, 'r', encoding='utf8')
    except IOError:
        outputFile.write("file not found")
        quit()

    capitalsHeaders = capitals.readline().rstrip().split(",")
    for row in capitals:
        capitalsRows.append(row.rstrip().split(","))

    # Close files that are done having data taken from
    topUnis.close()
    capitals.close()

    # Forloops to make each string in the list a capital so that it can be
compared correctly
    for university in range(len(topUniRows)):
        for attribute in range(len(topUniRows[university])):
            if type(topUniRows[university][attribute]) is str:
                topUniRows[university][attribute] =
topUniRows[university][attribute].upper(
                )

    for capital in range(len(capitalsRows)):
        for attribute in range(len(capitalsRows[capital])):
            if type(capitalsRows[capital][attribute]) is str:
                capitalsRows[capital][attribute] =
capitalsRows[capital][attribute].upper(
                )

    # 1 Univeristy count, calculate the total number universities and
write it to the file
    outputFile.write("Total number of universities => " +
str(len(topUniRows)))

    # 2 Available countries
    # Make list to hold available countrues
    availableCountries = []
    # Forloop to determine if a country is already in the list or not and
if not, then add it
    for university in topUniRows:
        if university[2] not in availableCountries:
            availableCountries.append(university[2])

    # Write each country to the file along
    outputFile.write("\n\nAvailable Countries => ")
    for country in availableCountries:
        outputFile.write(country)
        if country != availableCountries[-1]:
            outputFile.write(", ")

    # 3 Available continents
    # Find all available continents using the available countries and
write each to the file
    availableContinents = []
    for capital in capitalsRows:
        if capital[0] in availableCountries and capital[5] not in
availableContinents:
```

```
            availableContinents.append(capital[5])

    outputFile.write("\n\nAvailable Continents => ")
    for continent in availableContinents:
        outputFile.write(continent)
        if continent != availableContinents[-1]:
            outputFile.write(", ")

    # 4 The university with top international rank
    # Find all universities in the country and print the first one as it
has the highest international rank
    universitiesInCountry = []

    for university in topUniRows:
        if selectedCountry == university[2]:
            universitiesInCountry.append(university)
    worldTopInCountry = universitiesInCountry[0]

    outputFile.write("\n\nAt international rank => " +
                     worldTopInCountry[0] + " the university name is => "
+ worldTopInCountry[1])

    # 5 The university with top national rank
    # Make a variable to hold the highest ranked national university and
comare it to other national rankings then print the highest
    topUniInCountry = universitiesInCountry[0]
    for university in universitiesInCountry:
        if int(university[3]) < int(topUniInCountry[3]):
            topUniInCountry = university
    outputFile.write("\n\nAt national rank => " +
                     topUniInCountry[3] + " the university name is => " +
topUniInCountry[1])

    # 6 The average score
    # Make a variable to hold the score and add all scores from the
universities in the country, then average it and print it with 2 decimal
places
    totalScore = 0
    for university in universitiesInCountry:
        totalScore += float(university[8])
    averageScore = totalScore/len(universitiesInCountry)
    outputFile.write("\n\nThe average score =>
{:.2f}%".format(averageScore))

    # 7 The Continent Relative Score
    # Store the continent of the selected country in a variable, then find
all the countries in the continent, then find the highest scored country
in this list and calculate the relative score
    universitiesInContinent = []
    countriesInContinent = []
    for capital in capitalsRows:
        if selectedCountry in capital:
            selectedContinent = capital[5]
    for capital in capitalsRows:
        if selectedContinent in capital:
            countriesInContinent.append(capital[0])
    for university in topUniRows:
        if university[2] in countriesInContinent:
            universitiesInContinent.append(university)
    relativeScore =
100*(averageScore/float(universitiesInContinent[0][-1]))

    outputFile.write("\n\nThe relative score to the top university in " +
selectedContinent +
                     " is => ({} / {}) x 100% =
{:.2f}%".format(averageScore, universitiesInContinent[0][-1],
relativeScore))

    # 8 Find the capital city
    # Go through the capitals file and if the selected country's capital
is there, store it in a variable and write it
    for capital in capitalsRows:
        if selectedCountry in capital:
            capitalCity = capital[1]
    outputFile.write("\n\nThe capital is => {}".format(capitalCity))

    # 9 The universities that hold the capital name
    # Using the capital found earlier, go through all universities and see
which has the capital in it's name, then write it to the file with a
number counter
    outputFile.write("\n\nThe universities that contain the capital name
=>")
    uni_number = 1
    for university in topUniRows:
        if capitalCity in university[1]:
            outputFile.write("\n\t#{} {}".format(uni_number,
university[1]))
            uni_number += 1

    # Close file after all has been written to it
    outputFile.close()
```

```
# Name: Fahmi Omer
# UWO Email: fomer4@uwo.ca
# Assignment 2 - Pizza Ordering System
# Program Description: Recieve input from the user in the form of a pizza
order with specifications on size, toppings, and more, and output the
receipt with all information and prices of the order, incorporating the
usage of various techniques such as functions, using a multi-file program,
loops, conditions, lists and conditionals, and formatting.

# Importing functions from the pizzaReceipt file
from pizzaReceipt import *

# Setting presets for code to run
morePizza = True
TOPPINGS = ('ONION', 'TOMATO', 'GREEN PEPPER', 'MUSHROOM', 'OLIVE',
'SPINACH', 'BROCCOLI',
            'PINEAPPLE', 'HOT PEPPER', 'PEPPERONI', 'HAM', 'BACON',
'GROUND BEEF', 'CHICKEN', 'SAUSAGE')
pizzaOrder = []

# Verify that the user wants a pizza
if input("Do you want to order a pizza? ").upper() in ['Q', 'NO']:
    morePizza = False

# While loop thats continues if they want more pizzas
while (morePizza):

    # Ask for size of the pizza and continue until a valid size is entered
    size = input("\nChoose a size: S, M, L, or XL: ").upper()
    while (size not in ['S', 'M', 'L', 'XL']):
        size = input("Choose a size: S, M, L, or XL: ").upper()

    # Setting presets for topping input loop
    moreToppings = True
    pizzaToppings = []

    # While loop that continues until user adds all desired toppigns
through a boolean variable
    while (moreToppings):
        topping = input(
            "Type in one of our toppings to add it to your pizza. To see
the list toppings, enter \"LIST\". When you are done adding toppings,
enter \"X\"\n")

        if (topping.upper() in TOPPINGS):
            pizzaToppings.append(topping.upper())
            print("Added {} to you pizza".format(topping.upper()))
        elif (topping.upper() == "LIST"):
            print(TOPPINGS)
        elif (topping.upper() == "X"):
            moreToppings = False
        else:
            print("Invalid topping")

    # Add size and pizzaToppings into a tuple and add it into pizzaOrder
list
    pizzaOrder.append((size, pizzaToppings))

    # Ask user if they are ordering more pizzas and set boolean value to
continue or not
    if (input("Do you want to continue ordering? ").upper() in ["NO",
"Q"]):
        morePizza = False

# Call receipt function
generateReceipt(pizzaOrder)
```

```
# Create function
def generateReceipt(pizzaOrder):
    # Set variable for total
    total = 0
    # Check if customer ordered a pizza
    if len(pizzaOrder) == 0:
        print("You did not order anything")
    else:
        print("Your Order:")

        # Make forloop run for the amount of pizzas the customer ordered
        for i in range(len(pizzaOrder)):
            # Find what the size of the ordered pizza is and determine the
price of pizza and toppings
            if pizzaOrder[i][0] == "S":
                PIZZACOST = 7.99
                additionalToppingCost = 0.50
            elif pizzaOrder[i][0] == "M":
                PIZZACOST = 9.99
                additionalToppingCost = 0.75
            elif pizzaOrder[i][0] == "L":
                PIZZACOST = 11.99
                additionalToppingCost = 1.00
            elif pizzaOrder[i][0] == "XL":
                PIZZACOST = 13.99
                additionalToppingCost = 1.25

            # Print the pizza number title, the size, as well as the price
for the size, different spacing for XL due to the second character
            if pizzaOrder[i][0] == "XL":
                print("Pizza {}: {}{:>19}".format(
                    i+1, pizzaOrder[i][0], PIZZACOST))
            else:
                print("Pizza {}: {}{:>20}".format(
                    i+1, pizzaOrder[i][0], PIZZACOST))
            # Add pizza cost to total cost
            total += PIZZACOST

            # Print out all of the toppings
            for j in range(len(pizzaOrder[i][1])):
                print("- {}".format(pizzaOrder[i][1][j]))
            # Check to see if there are more than three toppings, if there
are, then print out "Extra Topping", the size and the cost
            if (len(pizzaOrder[i][1]) > 3):
                for k in range(len(pizzaOrder[i][1])-3):
                    if pizzaOrder[i][0] == "XL":
                        print("Extra Topping ({}){:>12.2f}".format(
                            pizzaOrder[i][0], additionalToppingCost))  #
fix number of decimal output
                    else:
                        print("Extra Topping ({}){:>13.2f}".format(
                            pizzaOrder[i][0], additionalToppingCost))  #
fix number of decimal output
                    # Add additional topping costs onto total cost
                    total += additionalToppingCost

        # Calculate the tax on the order and print the tax line
        tax = total*0.13
        print("Tax:{:>26.2f}".format(tax))

        # Add the tax onto the total and print out total value
        total += tax
        print("Total:{:>24.2f}".format(round(total, 2)))
```

```
# Name: Fahmi Omer
# UWO Email: fomer4@uwo.ca
# Assignment 1 - Inflation Rate
# Program Description: Code created with the purpose of calculating
personal inflation rates from previous to desired years using various
input, operations, loops, and functions

#Creating variables to prompt the user for input, take in input, and store
it for later use
personalInterestYear = input("Please enter the year that you want to
calculate the personal interest rate for: ")
expenditureCategories = int(input("\nPlease enter the number of
expenditure categories: "))

#Creating variables to store the expense totals of the previous year and
year of interest
prevYrExpenseTotal = 0
interestExpenseTotal = 0

#Forloop to iterate through all of the expenditure categories, output a
prompt for input, take them in as an integer input add them to the totals
for i in range(expenditureCategories):
    prevYrExpenseTotal += float(input("\nPlease enter expenses for
previous year: "))
    interestExpenseTotal += float(input("\nPlease enter expenses for year
of interest: "))

#Calculate the inflation rate and store it in a variable
inflationRate = ((interestExpenseTotal-prevYrExpenseTotal)
/interestExpenseTotal)*100

#Determining the type of inflation and storing as a string in a variable
if inflationRate < 3:
    inflationType = "Low"

elif inflationRate >= 3 and inflationRate < 5:
    inflationType = "Moderate"

elif inflationRate >= 5 and inflationRate < 10:
    inflationType = "High"

else:
    inflationType = "Hyper"

#Outputting the inflation rate as well as the type of inflation
print("\nPersonal inflation rate for {} is
%.1f".format(personalInterestYear) % (inflationRate) + "%")
```

```
print("\nType of Inflation: {}".format(inflationType))
```