Homework 3

Michael Brodskiy
Professor: M. Fanaei

March 25, 2023

Listing 1: Problem 1

```
"""
 * ============================================================================
 *
 *        Filename:  HW3Prob1Brodskiy.py
 *        Assignment: Homework 3 Problem 1
 *        Title: Pig Latin Converter
 *
 *     Description:  Translates English to pig latin
 *
 *         Version:  1.0
 *         Created:  03/19/2023
 *        Revision:  N/A
 *          Python:  Python 3.9.2
 *
 *          Author:  M. Brodskiy
 *
 * ============================================================================
"""

def pigLatinConv(tokens):

    vowels = ['a', 'e', 'i', 'o', 'u']

    print("In pig latin, this is:")
    for i in tokens:
        if i[0] in vowels:
            print(i + "ay",end=" ")
        else:
            print(i[1:] + i[0] + "ay",end=" ")

pigLatinConv(input("Please Enter an English Phrase: ").split())
```

Listing 2: Problem 2

```
"""
 * ============================================================================
 *
 *        Filename:  HW3Prob2Brodskiy.py
 *        Assignment: Homework 3 Problem 2
 *        Title: 5-to-3 Letter Converter
 *
 *    Description:  Converts a 5 Letter Word to each 3 Letter Subcomponent
 *
 *         Version:  1.0
 *         Created:  03/19/2023
 *        Revision:  N/A
 *          Python:  Python 3.9.2
 *
 *          Author:  M. Brodskiy
 *
 * ============================================================================
"""
import itertools

def fiveToThree(word):
    allCombos = []
    for i in itertools.combinations(word, 3):
        if (i[0] + i[1] + i[2]) not in allCombos:
            allCombos.append(i[0] + i[1] + i[2])
    return allCombos
```

Listing 3: Problem 3

```
"""
 * ============================================================================
 *
 *        Filename:  HW3Prob3Brodskiy.py
 *        Assignment: Homework 3 Problem 3
 *        Title: String Calculator
 *
 *     Description:  Performs single-digit arithmetic from strings
 *
 *         Version:  1.0
 *         Created:  03/21/2023
 *        Revision:  N/A
 *          Python:  Python 3.9.2
 *
 *          Author:  M. Brodskiy
 *
 * ============================================================================
"""

def stringCalc(phrase):

    tokens = phrase.lower().split()
    numbers = {"zero":0, "one":1, "two":2, "three":3, "four":4, "
       five":5, "six":6, "seven":7, "eight":8, "nine":9}
    num1 = numbers[tokens[0]]
    num2 = numbers[tokens[-1]]

    if len(tokens) == 3:
        if tokens[1] == "plus":
            return num1 + num2
        elif tokens[1] == "times":
            return num1 * num2
        elif tokens[1] == "minus":
            return num1 - num2
    elif len(tokens == 4):
        if tokens[1] + " " + tokens[2] == "divided by":
            return num1 / num2
        elif tokens[1] + " " + tokens[2] == "raised to":
            return num1 ** num2

print(stringCalc(input("Enter calculator phrase: ")))
```

Listing 4: Problem 4

```
"""
 * ============================================================
 *
 *       Filename:  HW3Prob4Brodskiy.py
 *       Assignment: Homework 3 Problem 4
 *       Title: String metric to imperial converter
 *
 *    Description:  Performs metric to imperial conversion
 *                  from string phrases
 *
 *        Version:  1.0
 *        Created:  03/21/2023
 *       Revision:  N/A
 *         Python:  Python 3.9.2
 *
 *         Author:  M. Brodskiy
 *
 * ============================================================
"""

def stringCalc(phrase):

    modifiers = {"kilo":.001, "hecto":.01, "deka":.1, "":1, "deci":10, "centi":100, "milli":1000}
    units = ["meter", "meters", "gram", "grams", "liter", "liters", "foot", "feet", "pound", "pounds", "gallon", "gallons"]
    factors = [.3048, .3048, 453.59, 453.59, 4.546, 4.546, 3.28, 3.28, .0022, .0022, .22, .22]
    tokens = phrase.lower().replace("?","").split()
    initialBaseUnit = ""
    initialUnit = ""
    initialQuant = 0
    initialModifier = ""
    finalBaseUnit = ""
    finalUnit = ""
    finalModifier = ""
    unitListLength = len(units)

    for i in tokens:
        for j in modifiers.keys():
            for k in range(unitListLength):
                if (i == (j + units[k]) and finalUnit == ""):
                    finalBaseUnit = units[k]
```

```
41                      finalUnit = i
42                      finalModifier = j
43                  elif (i == (j + units[k])):
44                      initialBaseUnit = units[k]
45                      initialUnit = i
46                      initialModifier = j
47
48      if (abs(units.index(finalBaseUnit) - units.index(
            initialBaseUnit)) % 5 == 0):
49          initialQuant = 1.0
50      elif (abs(units.index(finalBaseUnit) - units.index(
            initialBaseUnit)) % 6 == 0):
51          initialQuant = float(tokens[tokens.index(initialUnit) -
                1])
52      else:
53          print("The conversion from", initialUnit, "to", finalUnit,
                "is invalid. Exiting...")
54          return "Error!"
55
56      return f"There are {modifiers[finalModifier] * initialQuant *
            factors[units.index(finalBaseUnit)] / modifiers[
            initialModifier]:.4f} {finalUnit} in {initialQuant} {
            initialUnit}"
57
58  print("Enter a conversion phrase below.")
59  print("Prefixes from milli- to kilo- are valid.")
60  print(stringCalc(input("Available units include meters/feet, grams
        /pounds, and liters/gallons:  ")))
```

Listing 5: Problem 5

```
"""
 * ============================================================================
 *
 *        Filename:  HW3Prob5Brodskiy.py
 *        Assignment: Homework 3 Problem 5
 *        Title: Array Manipulations
 *
 *     Description:  Performs various array arithmetic
 *
 *         Version:  1.0
 *         Created:  03/21/2023
 *        Revision:  N/A
 *          Python:  Python 3.9.2
 *
 *          Author:  M. Brodskiy
 *
 * ============================================================================
"""

import numpy as np

A = np.arange(2.0, 19.0, 2).reshape(3, 3)
B = np.arange(9.0, 0.0, -1).reshape(3, 3)

# Part A
print("Element-wise Multiplication:", (A ** 2) * B, sep="\n")
print("Matrix-wise Multiplication:", np.matmul((A ** 2), B), sep="\n")

# Part B
A[B % 3 == 0] = np.sqrt(A[B % 3 == 0])
B[A % 4 == 0] = -B[A % 4 == 0]
print("Element-wise Multiplication:", np.linalg.inv(A) * np.linalg.inv(B), sep="\n")
print("Matrix-wise Multiplication:", np.matmul(np.linalg.inv(A), np.linalg.inv(B)), sep="\n")
```

Listing 6: Problem 6

```
"""
 * ============================================================================
 *
 *        Filename:  HW3Prob6Brodskiy.py
 *        Assignment: Homework 3 Problem 6
 *        Title: Array Diagonal Printer
 *
 *     Description:  Performs various array arithmetic
 *
 *         Version:  1.0
 *         Created:  03/21/2023
 *        Revision:  N/A
 *          Python:  Python 3.9.2
 *
 *          Author:  M. Brodskiy
 *
 * ============================================================================
"""

import numpy as np

def getDiagonals(mat):
    diagonals = [reversed(mat).diagonal(i) for i in range(-mat.
        shape[0]+1,mat.shape[1], -1)]
    diagonals.extend(mat.diagonal(i) for i in range(mat.shape
        [1]-1,-mat.shape[0],-1))
    return diagonals

m = np.random.randint(1, 9)
n = np.random.randint(1, 9)

mat = np.linspace(1, 100, m * n, dtype=int).reshape(m, n)

print("Original Matrix:\n", mat)
print("Diagonals:\n")
diag = getDiagonals(mat)
for i in diag:
    for j in i:
        print(j, end=' ')
    print('\n')
```

Listing 7: Problem 7

```
"""
 * ============================================================================
 *
 *        Filename:  HW3Prob7Brodskiy.py
 *        Assignment: Homework 3 Problem 7
 *        Title: Median and Mode Finder
 *
 *     Description:  Uses numpy to find median and mode on an array
 *
 *         Version:  1.0
 *         Created:  03/21/2023
 *        Revision:  N/A
 *          Python:  Python 3.9.2
 *
 *          Author:  M. Brodskiy
 *
 * ============================================================================
"""

import numpy as np

def findMode(mat):

    freq_list = {}
    mostFreq = 0

    for i in mat:
        for j in i:
            if j not in freq_list:
                freq_list[j] = 1
            else:
                freq_list[j] += 1

    for i in freq_list:
        if freq_list[i] > mostFreq:
            mostFreq = freq_list[i]

    for i in freq_list.keys():
        if freq_list[i] == mostFreq:
            return i

def findMed(mat, axis = 0):

```

```
    medians = []
    count = 0

    if axis == 0:
        for i in mat:
            i.sort()
            width = len(i)
            if width % 2 == 0:
                medians.append((i[(width // 2) - 1] + i[(width //
                    2)]) / 2)
                count += 1
            else:
                medians.append(i[width // 2])
                count += 1
    else:
        mat = mat.T
        for i in mat:
            i.sort()
            width = len(i)
            if width % 2 == 0:
                medians.append((i[(width // 2) - 1]  + i[(width //
                    2)]) / 2)
                count += 1
            else:
                medians.append(i[width // 2])
                count += 1

    return medians

arr1 = np.array([[3, 4, 2, 2], [7, 2, 3, 5], [6, 6, 6, 2]])
arr2 = np.array([[4, 5, 2], [3, 9, 9], [6, 8, 6], [0, 1, 1], [9,
   2, 9]])
arr3 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [1, 1, 1]])

print(arr1)
print("Median (rows):", findMed(arr1, 0))
print("Median (cols):", findMed(arr1, 1))
print("Mode:", findMode(arr1))
print(arr2)
print("Median (rows):", findMed(arr2, 0))
print("Median (cols):", findMed(arr2, 1))
print("Mode:", findMode(arr2))
print(arr3)
print("Median (rows):", findMed(arr3, 0))
print("Median (cols):", findMed(arr3, 1))
```

```
86 print("Mode:", findMode(arr3))
```