Lab 3: Functions, File Processing, and Arrays

Name: Shaheer Ziya University Number: 3035946760

Exercise 1: Recursive Function to Evaluate a Finite Sum

AIM:

An approximation to the function $x/(1-x)^2$ for $|x| < 1$ is given by the finite sum:

$$\frac{x}{(1-x)^2} = \sum_{i=1}^{n} ix^i$$

where n is a finite large number. Write a Python program that implements the recursive function `fsum(x, n)` to compute the above finite sum. Your program should also contain the code that displays this sum to 8 decimal places for x = 0.1, 0.2, 0.3, 0.4 and n = 2, 5, 10, 50, 100 by using this function.

ALGORITHM:

Notice that $\frac{x}{1-x^2} = \sum_{i=1}^{x} ix^i$ can be re-written as $nx + \sum_{i=1}^{n-1} ix^{n-1}$

This can be implemented using a recursive function where we set the base case to be n = 1 for which we simply return x, otherwise we return nx + fsum(x, n-1).

PROGRAM:

```
# Recurrence Relation Fucntion Approximation
# Created by Shaheer Ziya


# import math


def fsum(x, n):
  '''Aprroximate the fucntion x/(1+x)^2 using a recurrence relation'''
  # Base Case
  if n == 1:
    return 1 * x
  # Recursive Call
  else:
    return (n * pow(x, n)) + fsum(x, n-1)
```

```
def main():
  for x in [0.1, 0.2, 0.3, 0.4]:
    for n in [2, 5, 10, 50, 100]:
      print(f"The function x/(1-x)^2 evaluated at {x} with {n} steps is {fsum(x, n):.8f}")
    print()


main()
```

OUTPUT:

```
➜  PHYS2160 git:(main) ✗ /usr/local/bin/python3 "/Users/matthewsummons/Documents/GitHub/PHYS2160/Lab 3/1.py"
The function x/(1-x)^2 evaluated at 0.1 with 2 steps is 0.12000000
The function x/(1-x)^2 evaluated at 0.1 with 5 steps is 0.12345000
The function x/(1-x)^2 evaluated at 0.1 with 10 steps is 0.12345679
The function x/(1-x)^2 evaluated at 0.1 with 50 steps is 0.12345679
The function x/(1-x)^2 evaluated at 0.1 with 100 steps is 0.12345679

The function x/(1-x)^2 evaluated at 0.2 with 2 steps is 0.28000000
The function x/(1-x)^2 evaluated at 0.2 with 5 steps is 0.31200000
The function x/(1-x)^2 evaluated at 0.2 with 10 steps is 0.31249971
The function x/(1-x)^2 evaluated at 0.2 with 50 steps is 0.31250000
The function x/(1-x)^2 evaluated at 0.2 with 100 steps is 0.31250000

The function x/(1-x)^2 evaluated at 0.3 with 2 steps is 0.48000000
The function x/(1-x)^2 evaluated at 0.3 with 5 steps is 0.60555000
The function x/(1-x)^2 evaluated at 0.3 with 10 steps is 0.61221598
The function x/(1-x)^2 evaluated at 0.3 with 50 steps is 0.61224490
The function x/(1-x)^2 evaluated at 0.3 with 100 steps is 0.61224490

The function x/(1-x)^2 evaluated at 0.4 with 2 steps is 0.72000000
The function x/(1-x)^2 evaluated at 0.4 with 5 steps is 1.06560000
The function x/(1-x)^2 evaluated at 0.4 with 10 steps is 1.11029555
The function x/(1-x)^2 evaluated at 0.4 with 50 steps is 1.11111111
The function x/(1-x)^2 evaluated at 0.4 with 100 steps is 1.11111111
```

Exercise 2: Manipulating the Data from a Text File

AIM:

A text file called `HKFM.txt` contains the data of first marriages registered in Hong Kong from 1995 to 2020 (Source: https://www.censtatd.gov.hk/tc/scode160.html by Census and Statistics Department, HKSAR) which are delimited by tab as follows:

```
[Number of first marriages registered in HK by sex and age group]
Sex/Age group (years)    1995 2000 2005 2010 2015 2020
Male
16-19     289  232  220  175  155  31
20-24     4331 3076 3512 3737 3036 851
                      ⋮
>= 50     386  493  1690 1068 1014 581
Female
16-19     1213 966  935  683  511  98
20-24     10066     6613 7972 8286 5805 1545
                      ⋮
>= 50     140  93   156  227  320  319
```

Write a Python program that reads the data from this file, find the total number and dominant age group of first marriages registered in Hong Kong by sex and year, and finally print a table of the results with the following format on the screen:

```
   Sex  Year  Total Number  Dominant Age Group
-----------------------------------------------
  Male  1995         34080               25-29
  Male  2000         26176               25-29
                       ⋮
Female  1995         34232               25-29
Female  2000         26605               25-29
                       ⋮
```

ALGORITHM:

Initialize two dictionaries to store the data for each of the years for the males and females.

Initialize two lists to store the years in which the data is collected and age groups for the population.

Read the file, iterating over the lines containing the male and female data separately.

In each line, separate the words delimited by a space/tab.

Each item in this new list corresponds to a column of the data

Read the age groups from the lines and store it in the initialized list [1st Column].

Go over the columns and add it to the corresponding year in the data dictionary.

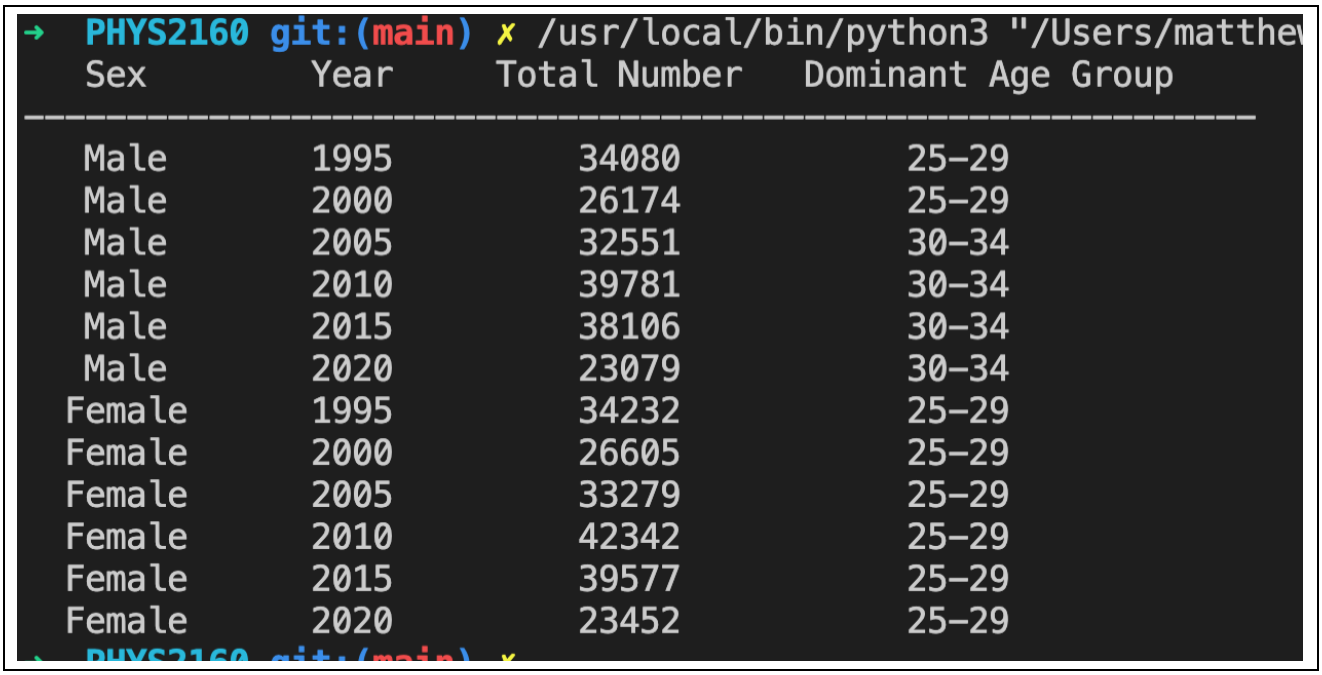
Print the data in the desired format where the total number is simply the sum of data in a given year and the most dominant age group can be found by mapping the index of the mode of the data in a year to it's age group

PROGRAM:

```
# 2.py
# Data Analysis
# Created by Shaheer Ziya

filePath = r"Lab 3/HKFM.txt"

maleYrData, femaleYrData = {}, {}
Ages = []

def YearMode(data):
  '''Find the mode of the number of marriages in a year and return its corresponding Age Group'''
  # Find the index of the mode value and match it with it's corresponding Age Group
  return Ages[data.index(max(data))]

# Init dicts with years as keys
years = ["1995", "2000", "2005", "2010", "2015", "2020"]
for year in years:
  maleYrData[year] = []
  femaleYrData[year] = []
```

```
with open(filePath, 'r') as f:
  for (lineNum, line) in enumerate(f):
    # Skip header lines
    if lineNum in (0, 1, 2): continue
    # Male Data
    elif lineNum < 11:
      # Separate words in line
      lineList = line.split(" ")
      # Obtain Age Groups (Need only be done once)
      Ages += lineList[0],

      # Obtain data for each of the years
      for idx, year in enumerate(years):
        maleYrData[year] += int(lineList[idx+1]),

    # Go over the data for the females
    elif lineNum > 11:
      lineList = line.split(" ")
      # Obtain data for each of the years
      for idx, year in enumerate(years):
        femaleYrData[year] += int(lineList[idx+1]),



# Print the organized data
print(f"{'Sex':^10} {'Year':^10} {'Total Number':^15} {'Dominant Age Group':^15}")
print("-"*60)

# Print the statistics for the males
for year in years:
  print(f"{'Male':^10} {year:^10} {sum(maleYrData[year]):^15} {YearMode(maleYrData[year]):^15}")

# Print the statistics for the females
for year in years:
  print(f"{'Female':^10} {year:^10} {sum(femaleYrData[year]):^15} {YearMode(femaleYrData[year]):^15}")
```

OUTPUT:


→ PHYS2160 git:(main) ✗ /usr/local/bin/python3 "/Users/matthe
Sex Year Total Number Dominant Age Group
Male 1995 34080 25-29
Male 2000 26174 25-29
Male 2005 32551 30-34
Male 2010 39781 30-34
Male 2015 38106 30-34
Male 2020 23079 30-34
Female 1995 34232 25-29
Female 2000 26605 25-29
Female 2005 33279 25-29
Female 2010 42342 25-29
Female 2015 39577 25-29
Female 2020 23452 25-29

Exercise 3: Printing a Histogram to a Text File

AIM:

A text file called `HKPop2020data.txt` contains the data of the mid-year population in Hong Kong for 2020 (Source: Hong Kong Digest of Statistics 2021 Edition by Census and Statistics Department Hong Kong Special Administrative Region) which are delimited by space as follows:

```
[Mid-year population in Hong Kong by age group and sex for 2020]
[Age Group] [Male] [Female]
0-4 137400 127400
5-9 154600 144800
10-14 154400 150700
            ⋮
80-84 79900 94000
>=85 79100 138700
```

Write a Python program to show the statistics of the data in this file by printing a histogram in the following format to a text file `HKPop2020hist.txt`:

```
Mid-year Population in Hong Kong by Age Group and Sex for 2020
(in nearest ten thousands)
  0-4 | ##############&&&&&&&&&&&&&& (14/14)
  5-9 | ###############&&&&&&&&&&&&&& (15/14)
10-14 | ###############&&&&&&&&&&&&&& (15/15)
                      ⋮
 >=85 | ########%%%%%%%%%%%%%% (8/14)
#-Male; %-Female
```

ALGORITHM:

Load data from the file, skipping the header lines.

The first column is the data for the males where the second column is the data for the females (Zero-Indexed)

Reformat the data into the desired types.

Write the desired lines onto the file rounding where necessary

PROGRAM:

```
# Print Histogram to File
# Created by Shaheer Ziya

filePath = r"Lab 3/HKPop2020data.txt"



data = []
with open(filePath, 'r') as f:
  for lineNum, line in enumerate(f):
    # Skip the first 2 lines
    if lineNum in (0, 1): continue

    data += [line.split(" ")]

for line in data:
  line[1] = int(line[1])
  line[2] = int(line[2])

with open(r'HKPop2020hist.txt', 'w') as f:
  f.write("Mid-year Population in Hong Kong by Age Group and Sex for 2020\n")
  f.write("(in nearest ten thousands)\n")
  for line in data:
    f.write(f"{line[0]:^5} | {('#' * round(line[1]/1e4)) + ('&' * round(line[2]/1e4)):<65}
({round(line[1]/1e4)}/{round(line[2]/1e4)})\n")
```

OUTPUT:

HKPop2020hist.txt

```
Mid-year Population in Hong Kong by Age Group and Sex for 2020
(in nearest ten thousands)
 0-4   | ############&&&&&&&&&&&&                                          (14/13)
 5-9   | ############&&&&&&&&&&&&&                                         (15/14)
10-14  | ############&&&&&&&&&&&&&                                         (15/15)
15-19  | ############&&&&&&&&&&&&                                          (14/13)
20-24  | ##################&&&&&&&&&&&&&&&&&&                              (19/19)
25-29  | ####################&&&&&&&&&&&&&&&&&&&&&&&&                      (22/26)
30-34  | ####################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&            (23/32)
35-39  | ######################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&    (24/37)
40-44  | ####################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&            (23/34)
45-49  | ######################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&        (24/34)
50-54  | ####################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&            (24/32)
55-59  | ############################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& (30/34)
60-64  | ############################&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&    (29/30)
65-69  | ####################&&&&&&&&&&&&&&&&&&&&&&                        (22/23)
70-74  | ################&&&&&&&&&&&&&&&&&&                                (17/17)
75-79  | #########&&&&&&&&&&                                              (10/10)
80-84  | ########&&&&&&&&&                                                (8/9)
>=85   | ########&&&&&&&&&&&&&&                                           (8/14)
```

Exercise 4: Evaluating a Test with Arrays

AIM:

A test consisting of 20 multiple-choice questions with 5 possible choices (A, B, C, D, and E) is conducted for a group of 5 students. Write a Python program to evaluate the answers of these students using the following algorithm:

(a) Read the string of the correct answers to the questions from the user and store the answers into an array of characters.

(b) Read the string of the answers of a student from the user and store the answers into an array of characters.

(c) Construct a Boolean array to indicate whether the answer of the student to each question is correct.

(d) Use the array in (c) to count the number of correct answers and then print the results.

(e) Repeat steps (b) to (d) for each student.

You can assume that all the inputs are in the required format. Here are the sample input and output of this program:

```
Enter the correct answers to the MC questions:
EEDAECAEEEBCADDBCEEB
Enter the answers of Student-1:
CEACBBDBDBCBCEEADABB
Number of correct answers: 2
Answers to the following questions are correct:
2 20
Enter the answers of Student-2:
EEDAECAEEEBCADDBCEEB
Number of correct answers: 20
Answers to the following questions are correct: ALL
⋮
```

ALGORITHM:

Obtain the list of all solutions and store the result in an array of characters (C-string)

Iterate over the 5 students, asking for the answers of each and storing their response in a array of characters (C-String)

The Boolean array is implicitly created during the comparison of each character

For each student go over their answers, character by character comparing it with the solution list and adding the questions they got right into a list

Print out their results at the end of the iteration for the student.

PROGRAM:

```
# Test Checking Bot
# Created by Shaheer Ziya

# String = Array of Characters

def main():
  correctAns = input("Enter the correct answers to the MC questions: ")

  # Iterate over 5 students
  for student in range(1, 5+1):
    # Obtain student answers
    stdAns = input(f"Enter the answers of Student-{student}: ")
    correctQuestions = []
    # Iterate over every answer and compare with solutions
    for sol, stdAns, Q in zip(correctAns, stdAns, range(1, 20+1)):
      # If the answer is correct, append the question number to the list
      if stdAns == sol:
        correctQuestions.append(Q)
    # For each student, print how many questions they got correct & the list of correct questions
    print(f"Number of correct answers: {len(correctQuestions)}")
    print("Answers to the following questions are correct:")
    if len(correctQuestions) == 20: print("ALL")
    else:
      for i in correctQuestions:
        print(i, end=" ")
      print()
main()
```

OUTPUT:

```
→  PHYS2160 git:(main) ✗ /usr/local/bin/python3 "/Users/matthewsummor
y"
Enter the correct answers to the MC questions: EEDAECAEEEBCADDBCEEB
Enter the answers of Student-1: CEACBBDBDBCBCEEADABB
Number of correct answers: 2
Answers to the following questions are correct:
2 20
Enter the answers of Student-2: EEDAECAEEEBCADDBCEEB
Number of correct answers: 20
Answers to the following questions are correct:
ALL
Enter the answers of Student-3:
```