"ACTIVITY 3 | GROUP 4 | GCIS 123 | PROF. D. COVACEVIC"

"By Timofey Lola, Hamad Alameri, Khalifa Bin Sulaiman"

"Each memner has contributed, Tim mainly has worked on a general concept of the code and functions, Khalifa mainly solved the mathematical part, Hamad has worked mainly on code visual design"

"This code is designed to work with CSV files. It loads the data from a file, then allows to calculate minimums, maximums and averages of numeric columns, then cleans and prepares the data for visualization."

"All the stages are designed for user interaction and include option menus and description for the user to follow the process"

import csv #first we need to import the CSV module to read our file.

def load\_data(path): #this function loads our data so we can work on it

try:

with open(path, mode="r", encoding="utf-8") as file: #mode=r means reading is allowed, encoding utf-8 was suggested by Python program so the file can be read proper;y

text=csv.reader(file)

list\_text=list(text)

data={}

for column in list\_text[0]:

data[column] = []

for line in range(1,len(list\_text)):

for el in range(len(list\_text[line])):

data[list\_text[0][el]].append(list\_text[line][el]) #.append is used to add a single item at the end of the list which exists

return data

except Exception:

return Exception

def min\_value(column): #here we analyze our minimums

minimal=1000 #this value is changeable, and is used to showcase an ultimate minimal which can't be reached due to Grades.csv file specifics

for value in column:

if value.isnumeric()==True: #.isnumeric function is used to check whether our string is entierly numerical

if int(value)<minimal:

minimal=int(value)

return minimal

def max\_value(column): #here we analyze our maximums

maxim=-1000 #this value is changeable, and is used to showcase an ultimate maxim which can't be reached due to Grades.csv file specifics

for value in column:

if value.isnumeric()==True:

if int(value)>maxim:

maxim=int(value)

return maxim

def avg\_value(column): #here we analyze our averages

total=0

count=0

for value in column:

if value.isnumeric():

total+=int(value)

count+=1

if count>0:

return int(total/count)

else:

return 0

def clean\_and\_prepare\_data(data): #this function restructures our data so we can see our minimums, maximums or averages

while True:

column=input("Enter column name: ")

if column in data.keys(): #we use .keys to return our object of viewing

try:

int(data[column][0])

value=int(input("choose an option by number: 1. min 2. max 3. average: "))

if value==1:

val=min\_value(data[column])

elif value==2:

val=max\_value(data[column])

elif value==3:

val=avg\_value(data[column])

else:

print("wrong choice")

continue

for number in range(len(data[column])):

if data[column][number]=='':

data[column][number]=str(val)

return data[column], column

except:

print("enter a numerical column") #we make sure that the chosen column consists of numbers, for ex. column Last Name wouldn't work

else:

print("doesn't exist")

def sort\_value(data, status): #we use a separate function to regulate the order in which user would like the data to be sorted in

for i in range(1,len(data)):

j=i-1

key=int(data[i])

while j>=0 and ((key<int(data[j]) and status==True)or(key>int(data[j]) and status==False)):

data[j+1]=int(data[j])

j=j-1

data[j+1]=key

return data

def analyze\_data(data): #here we use this function to carry data analysis in a desired order

choice=int(input("choose an option by number: 1. ascending 2. descending: "))

if choice==1:

sort\_value(data, status=True)

elif choice==2:

sort\_value(data, status=False)

else:

print("wrong choice")

return data

def visualize\_data(data, column):

print("column:", column)

print("legend: each ‘\*’ represents 5 units")

for value in data:

print("\*"\*min(int(value)//5,20))

print("Visualisation completed!\nThank you and good bye!")

if \_\_name\_\_ == "\_\_main\_\_":

data=load\_data(path="grades.csv")

full\_data, column=clean\_and\_prepare\_data(data)

print(full\_data)

data\_stats=analyze\_data(data=full\_data)

print(data\_stats)

visualize\_data(data\_stats, column)