#########################################################################

# CT60A0203 Introduction to Programming - Online teaching

# Name: Bianca-Ioana Stefanescu

# Student number: 001512335

# Email: Ioana.Stefanescu@student.lut.fi

# Date: 13.11-2023

# By submitting this work, I certify that

#

# I wrote all the code myself and have not shared it with anyone else

# Although I used the internet to search for solutions (e.g., Stack Exchange, GeeksforGeeks, and Programiz), I wrote the code independently.

# I asked ChatGPT and used the answer as a reference, but I wrote the code by myself.

#

#########################################################################

#import module(s) and class(es)

from datetime import datetime

import random

#\_1.0\_add\_student

def add\_student():

#ask the first and last name

while True:

print("Names should contain only letters and start with capital letters.")

the\_first\_name = input("Enter the first name of the student:\n")

the\_last\_name = input("Enter the last name of the student:\n")

#check if the names are only letters and start with capitals

if the\_first\_name.isalpha() and the\_last\_name.isalpha() and the\_first\_name[0].isupper() and the\_last\_name[0].isupper():

#major selection

print("Select student's major:")

print("\t\tCE: Computational Engineering")

print("\t\tEE: Electrical Engineering")

print("\t\tET: Energy Technology")

print("\t\tME: Mechanical Engineering")

print("\t\tSE: Software Engineering")

while True:

#ask the major

the\_choice = input("What is your selection?\n")

#check if the major is correct

if the\_choice in ["CE","EE","ET","ME","SE"]:

print("Student added successfully!\n")

#generate ID

the\_used\_IDs = []

with open('students.txt','r') as the\_file:

for the\_line in the\_file:

the\_number = int(the\_line.strip().split(",")[0])

the\_used\_IDs.append(the\_number)

while True:

the\_ID=random.randint(10000,99999)

if the\_ID not in the\_used\_IDs:

break

#generate email

the\_email = (f"{the\_first\_name.lower()}.{the\_last\_name.lower()}@lut.fi")

#add data to the file

with open('students.txt', 'a') as the\_file:

the\_file.write(f"{the\_ID},{the\_first\_name},{the\_last\_name},{datetime.now().year},{the\_choice},{the\_email}\n")

break

#in case the major is not right

else:

print("Invalid major selection. Try again!")

continue

#in case the name is not right

else:

continue

#\_2.0\_search student

def search\_student():

while True:

#ask input for search

the\_search = input("Give at least 3 characters of the student's first or last name:\n")

#check if the search is valid

if len(the\_search) < 3 or not the\_search.strip():

continue

#open and read the file

with open('students.txt', 'r') as the\_file:

#take a counter to count the matches found

k = 0

print("Matching students:")

#split the content at ","

for the\_line in the\_file:

the\_id = the\_line.strip().split(",")[0]

the\_first\_name = the\_line.strip().split(",")[1]

the\_last\_name = the\_line.strip().split(",")[2]

#check if there is a match

if the\_search.lower() in the\_first\_name.lower() or the\_search.lower() in the\_last\_name.lower():

#count the match

k = k + 1

print(f"ID: {the\_id}, First name: {the\_first\_name}, Last name: {the\_last\_name}")

#in case there is no match

if k == 0:

break

break

#\_3.0\_search course

def search\_course():

while True:

#ask input for search

the\_search = input("Give at least 3 characters of the name of the course or the teacher:\n")

#check if the search is valid

if len(the\_search) < 3 or not the\_search.strip():

print("Invalid input. Try again!")

continue

#open and read the file

with open('courses.txt','r') as the\_file:

#take a counter to count the matches found

k = 0

#split the content at ","

for the\_line in the\_file:

the\_code = the\_line.strip().split(",")[0]

the\_name = the\_line.strip().split(",")[1]

the\_teachers = the\_line.strip().split(",",3)[3]

the\_list = the\_teachers.split(",")

#check if there is a match

if the\_search.lower() in the\_teachers.lower() or the\_search.lower() in the\_name.lower():

print(f"ID: {the\_code}, Name: {the\_name}, Teacher(s): {the\_teachers}")

#count the match

k = k + 1

#in case there is no match

if k == 0:

break

#\_4.0\_add course completion

def add\_course\_completion():

while True:

#ask the course ID

the\_course\_ID = input("Give the course ID:\n")

#open and read the file

with open('courses.txt', 'r') as the\_file:

#split the content at ","

for the\_line in the\_file:

the\_code = the\_line.strip().split(",")[0]

#check if the course exists

if the\_course\_ID == the\_code:

while True:

#ask the student ID

the\_student\_ID = input("Give the student ID:\n")

#open and read the file

with open('students.txt', 'r') as the\_other\_file:

#split the content at ","

for the\_other\_line in the\_other\_file:

the\_ID = the\_other\_line.strip().split(",")[0]

#check if the student exists

if the\_student\_ID == the\_ID:

while True:

#ask the grade

the\_grade = int(input("Give the grade:\n"))

#check if the grade is valid

if 1 <= the\_grade <= 5:

#open and read file

with open('passed.txt', 'r+') as the\_other\_other\_file:

#go through the lines

the\_other\_other\_lines = the\_other\_other\_file.readlines()

#go through the elements

for the\_element, the\_other\_other\_line in enumerate(the\_other\_other\_lines):

#split the content at ","

the\_the\_cID, the\_the\_sID, the\_the\_date, the\_the\_grade = the\_other\_other\_line.strip().split(",")

the\_the\_grade = int(the\_the\_grade)

#check if the student has passed the course

if the\_course\_ID == the\_the\_cID and the\_student\_ID == the\_the\_sID:

#in case the grade is not bigger

if the\_the\_grade < the\_grade:

#ask date

the\_date = check\_date\_validity()

the\_existing\_date = datetime.strptime(the\_the\_date, "%d/%m/%Y")

the\_new\_date = datetime.strptime(the\_date, "%d/%m/%Y")

the\_difference = (the\_new\_date - the\_existing\_date).days

#in case the date is not within 30 days

if int(the\_difference) > 30:

print("Input date is older than 30 days. Contact \"opinto\".")

return

#in case the date is within 30 days

else:

#update data

the\_other\_other\_lines[the\_element] = f"{the\_course\_ID},{the\_student\_ID},{the\_date},{the\_grade}\n"

#open file and enable writing

with open('passed.txt', 'w') as the\_write\_file:

#write new data

the\_write\_file.writelines(the\_other\_other\_lines)

print("Input date is valid.\nRecord added!")

return

else:

print(f"Student has passed the course earlier with {the\_the\_grade}")

return

#in case the student has not passed the course

elif the\_element == len(the\_other\_other\_lines) - 1:

the\_date = check\_date\_validity()

the\_difference = check\_date\_distance(the\_date)

if int(the\_difference) > 30:

print("Input date is older than 30 days. Contact \"opinto\".")

return

else:

the\_other\_other\_file.write(f"{the\_course\_ID},{the\_student\_ID},{the\_date},{the\_grade}\n")

print("Input date is valid.\nRecord added!")

return

#in case the grade is not correct

else:

print("Grade is not a correct grade.")

break

break

break

#\_4.1\_add course completion - check the format of the date

def check\_date\_format(the\_datestr, the\_format = "%d/%m/%Y"):

#check if the format is ok

try:

datetime.strptime(the\_datestr, the\_format)

return True

#in case the format is not ok

except ValueError:

return False

#\_4.2\_add course completion - check the validity of the date

def check\_date\_validity():

while True:

#ask date

the\_datestr = input("Enter a date (DD/MM/YYYY):\n")

#check format

if check\_date\_format(the\_datestr):

#convert to datetime object

the\_date = datetime.strptime(the\_datestr, "%d/%m/%Y")

# check if the date is not later than today

if the\_date <= datetime.now():

return the\_datestr

#in case the date is later than today

else:

print("Input date is later than today. Try again!")

#in case the format is not good or the date is not valid

else:

print("Invalid date format. Use DD/MM/YYYY. Try again!")

#\_4.3\_add course completion - check the distance of the date

def check\_date\_distance(the\_date):

the\_present = datetime.now()

the\_diff = abs(the\_date - the\_present)

the\_days = the\_diff.days

return the\_days

#\_5.0\_create and show record

def create\_and\_show\_record(the\_s\_id):

#open and read files

with open('students.txt', 'r') as the\_s\_file, open('courses.txt', 'r') as the\_c\_file, open('passed.txt', 'r') as the\_p\_file:

#personal information part

the\_data = None

#split the content at ","

for the\_s\_line in the\_s\_file:

the\_s\_line\_data = the\_s\_line.strip().split(",")

#extract information

if the\_s\_line\_data[0] == the\_s\_id:

the\_data = {

'Student ID' : the\_s\_line\_data[0],

'Name' : f"{the\_s\_line\_data[2]}, {the\_s\_line\_data[1]}",

'Starting year' : the\_s\_line\_data[3],

'Major' : the\_s\_line\_data[4],

'Email' : the\_s\_line\_data[5]

}

break

#passed courses part

the\_passed\_courses = []

#take a counter to count the passed courses

the\_courses\_nr = 0

#split the content at ","

for the\_p\_line in the\_p\_file:

the\_p\_line\_data = the\_p\_line.strip().split(",")

#extract information

if the\_p\_line\_data[1] == the\_s\_id:

the\_passed\_courses.append({

'Course ID' : the\_p\_line\_data[0],

'Date' : the\_p\_line\_data[2],

'Grade' : the\_p\_line\_data[3]

})

the\_courses\_nr = the\_courses\_nr + 1

#display personal information part

if the\_data:

print(f"\nStudent ID: {the\_data['Student ID']}")

print(f"Name: {the\_data['Name']}")

print(f"Starting year: {the\_data['Starting year']}")

print(f"Major: {the\_data['Major']}")

print(f"Email: {the\_data['Email']}")

#split content at ","

the\_c\_data = [the\_the\_line.strip().split(",") for the\_the\_line in the\_c\_file]

# take counter to count the credits

the\_total\_credits = 0

# take counter to count the points

the\_grade\_points = 0

print("\nPassed courses:\n")

#extract information

for the\_passed\_course in the\_passed\_courses:

for the\_c\_line\_data in the\_c\_data:

if str(the\_c\_line\_data[0]) == the\_passed\_course['Course ID']:

print(f"Course ID: {the\_c\_line\_data[0]}, Name: {the\_c\_line\_data[1]}, Credits: {the\_c\_line\_data[2]}")

print(f"Date: {the\_passed\_course['Date']}, Teacher(s): {the\_c\_line\_data[3]}, grade: {the\_passed\_course['Grade']}\n")

# count credits

the\_total\_credits += int(the\_c\_line\_data[2])

# count points

the\_grade\_points += int(the\_passed\_course['Grade'])

#check if there are passed courses

if the\_courses\_nr > 0:

the\_average\_grade = round(the\_grade\_points / the\_courses\_nr, 1)

print(f"Total credits: {the\_total\_credits}, average grade: {the\_average\_grade}")

#in case there are no passed courses

else:

print("No passed courses")

#\_5.1\_search and displey record

def search\_and\_display\_record():

while True:

search\_student()

the\_s\_id = input("Enter the student ID:\n")

create\_and\_show\_record(the\_s\_id)

break

#\_X.X\_create the menu

def main():

try:

#display options

print("You may select one of the following:\n")

print("\t\t1) Add student\n")

print("\t\t2) Search student\n")

print("\t\t3) Search course\n")

print("\t\t4) Add course completion\n")

print("\t\t5) Show student's record\n")

print("\t\t0) Exit\n")

#ask for input

the\_choice = input("What is your selection?\n")

#assign functions

if the\_choice == '1':

add\_student()

elif the\_choice == '2':

search\_student()

elif the\_choice == '3':

search\_course()

elif the\_choice == '4':

add\_course\_completion()

elif the\_choice == '5':

search\_and\_display\_record()

elif the\_choice == '0':

exit(0)

#in case the input is not valid

else:

print("Invalid input.")

except ValueError:

print("Invalid input.")

#\_X.0\_call the main function

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

main()