Project work

Introduction

The project work is a more demanding programming task that requires mastering the skills learned during the course. Your task is to program in Python a ***management system for a small car renting company***.

Files

Your system will rely on four text files, which contain the data needed to run the company. You can find models of these files in Moodle for "Files for project". The idea is that while running your program, the content of these files change. If you manage to corrupt the content of your files, you can always start again by downloading the files from Moodle.

`Vehicles.txt`: This file contains the information of the cars that the company owns. The information of one car is given in one row. Rows look like this:

```
AMG-111,Toyota Yaris,65,Air Conditioning,Hybrid,Automatic Transmission
```

The data is given in comma-separated form. First there is the register plate number (`AMG-111`), which identifies the car. Then there is the model of the car (`Toyota Yaris`). The number following the model (`65`) is the daily rate of this car. This information is need when the price of the rent is computed. After that come the properties: `Air Conditioning,Hybrid,Automatic Transmission`. There can be several properties, but always at least one. Different properties are separated by a comma. The newline symbol "`\n`" ends each row in every file.

The content of this file does not change, your program should only read information from it.

`Customers.txt`: This file contains information about the clients who rent cars. The information of one client is given in one line. Different values are separated by a comma. One line looks like this:

```
09/09/1999,Tom,Shark,Tom.Shark@gmail.com
```

First there is the birthday (`09/09/1999`), which serves as the identifier of the customer. This means that there cannot be two customers with exactly same birthday. Of course, this is not "realistic" assumption, but in our "small world" this assumption may hold. Then comes person's first name (`Tom`) and last name (`Shark`). The fourth value is the email address of the customer (`Tom.Shark@gmail.com`).

The content of this file may change: if a new customer rents a car, her/his information is ***appended*** to the file. Customers are ***never*** removed.

`RentedVehicles.txt`: This file contains information about the rented vehicles. One row looks like this:

```
VIG-326,17/08/1978,10/23/2022 12:23
```

First there is the register plate number (`VIG-326`) of the rented car, then the birthday (`17/08/1978`) of the customer who rented the car. The third value is the time (`10/23/2022 12:23`) when the renting started. The three values are separated by a comma.

The content of this file may change in two ways: (a) When a car is rented, a new line is **appended** to the file; (b) When car is returned, the corresponding line is **removed** from the file. The file contains only "open" rents.

`transActions.txt`: This file contains information about "ended" rents. Each transaction is in its own line and one line looks like this:

```
BMC-69,17/04/1997,10/15/2022 14:01,10/19/2022 17:20,5,210.00
```

First there is the register plate number (`BMC-69`) of the car that was rented and the day of the birth (`17/04/1997`) of the customer who rented the car. The third value is the time (`10/15/2022 14:01`) when the rent started followed by the time (`10/19/2022 18:04`) when the car was returned. The fourth value is the duration in days (`5`). Note that each starting day is billed as a rental day. The last value is the price (`210.0`) of the rent using two decimals. When car is returned and the price is computed, corresponding information is appended to this file.

Different operations

The program needs to be menu-based. When the program starts it prints the following menu:

```
You may select one of the following:
1) List available cars
2) Rent a car
3) Return a car
4) Count the money
0) Exit
What is your selection?
```

This menu should be also displayed after every operation until you end the program by selecting 0.

In the following, each operation is described in detail.

1) List available cars: As described above, the file `Vehicles.txt` contains all cars owned by the company. The program should list all cars and their properties in that file, except those that are listed in `RentedVehicles.txt`. Here is a part of the output as an example:

```
The following cars are available:
* Reg. nr: BKV-943, Model: Ford Fiesta, Price per day: 35
Properties: Manual Transmission
* Reg. nr: JMB-535, Model: Ford Fiesta, Price per day: 48
Properties: Air Conditioning, Manual Transmission
* Reg. nr: SKF-124, Model: Ford Focus, Price per day: 52
Properties: Air Conditioning, Manual Transmission
* Reg. nr: MSQ-731, Model: Ford Focus Wagon, Price per day: 52
Properties: Air Conditioning, Manual Transmission
```

2) Rent a car: First the program should ask the register number of the car to be rented. The program should also check that a car with this register number exists and is available for renting. After that ask the birthday of the customer. The program should check that the birthday is given in the right form, that is, DD/MM/YYYY. The program should also check that the date given by the user is sensible. This means, for instance, that DD and MM have acceptable values. It should be also checked that the customer's age is below 100 years and at least 18 years during the year 2022.

There are two choices: (a) the customer already exists in `Customers.txt`. In this case, you need to do nothing concerning the file Customers.txt. (b) If the customer does not exist in `Customers.txt`, the program needs to ask customer's first name, last name, and email address. The program should do some checking of the validity of the email address. For instance, email addresses must contain the symbol "`@`" and a dot "`.`". Append the information about a new customer to the file `Customers.txt`.

After the data of the customer is processed, the program should add a new line to the file `RentedVehicles.txt`, which contains the register number of the car, customer's birthday and a timestamp when the rent started, as explained above. The current time is obtained from datetime module. The program also writes a confirmation to the display:

```
Hello Matti
You rented the car MER-611
```

Here `Matti` is the first name of the customer and `MER-611` is the register number.

3) Return a car: First ask the register plate number of the car to be returned. After that the program should find the right row in `RentedVehicles.txt` to obtain the details of the rent. If the car with this register number does not exist or is not rented, the program should inform the user. After the right data is fetched from `RentedVehicles.txt`, the program should find from the file `Vehicles.txt` the daily price of the rent of this car. Using `datetime` module, the program gets the current date and time. The program should compute the number of started days. Then the price can be computed.

The corresponding line from `RentedVehicles.txt` needs to be removed. After that the programs adds a line to `transActions.txt` according to the example above.

4) Count the money: In this task the program reads the file `transActions.txt` and computes the sum of the earned money appearing in the last column of the rows. The output of the method should write to the display:

```
The total amount of money is XXX.XX euros
```

Here XXX.XX is the sum computed from `nsActions.txt`.

Submission and grading

The program is submitted to ***CodeGrade***, but teachers will grade the work according to the checklist below. From each item it is possible to obtain 0-5 points. This means that it is possible to gather 100 points.

To pass the project work, you need to gather ***at least 50 points***. The grade of the project work is determined by this table:

Points	50-59	60-69	70-79	80-89	90-100
Grade	1	2	3	4	5

The items in the following **checklist** will be evaluated:

1) The menu is printed correctly.
2) The program is divided into functions in a proper manner.
3) The available cars are printed correctly a) no rented cars are displayed, b) all available cars are displayed, c) properties of the cars are displayed correctly.
4) When car is rented, check that the car exists and that it is not rented already.
5) The program checks that the birthday given by the user is correctly formed.
6) The program checks that the customer is not too young or too old.
7) The program checks that the email address of the customer is correctly formed.
8) The data of a new customer is appended to the file correctly - if it is not there already.
9) Information about the rent is appended correctly to the file `RentedVehicles.txt`
10) The program displays the confirmation of the rent to the screen.
11) When car is returned, the program finds the right row in `RentedVehicles.txt`.
12) The program correctly finds the daily cost of the rent in `Vehicles.txt`.
13) The program computes the number of the days and cost of the rent correctly.
14) When a car is returned, the corresponding row should be removed from `RentedVehicles.txt`.
15) When car is returned, a line should be added to `transActions.txt`.
16) Compute the earned money correctly.
17) All opened files are closed.
18) Required data is passed as arguments to the functions.
19) Input check. Display appropriate message if the input is not between 0-4.
20) Program ends when 0 is given in menu.

If you get less than 50 points, your project work is needs to be corrected / completed. You need to resubmit the project work by 5/12. Note that the grades of resubmitted works will be worse than of the actual returns, so it is better to make the program as good as possible in the first place.

Timetable

- 31/10: The submission box opens in ***CodeGrade*** (at latest).
- 14/11: Deadline for submitting the project work.
- 28/11: By this date, you will get the grade. Resubmission opens for works that need improving.
- 5/12: Resubmission closes.