Koç University COMP125
Programming with Python: Midterm 3

Instructor: Beren Semiz

Date: May 9th, 2021

9:40 AM – 11:10 AM

Deadline to submit on Blackboard: 11:15 AM

Instructions

Make sure you read and understand every part of this document

Download & Extract the zip to your Desktop:

Blackboard -> Assessments -> MT3-> MT3.zip

Zip file includes:

housing.py stroke.py Midterm3_Spring2021.pdf

Open the .py files using Spyder.

Do not change the names of the files. Do not change the names of the functions that are given to you. Do not use Turkish characters. You may add new functions if you want to (helper functions).

When you are finished:

Submit a single compressed archive (.zip or .rar file) containing:

housing.py stroke.py

Multiple attempts are allowed, so upload ASAP, clean up later. We will grade only the LAST version submitted to Blackboard.

By submitting this file, you automatically agree to the honor pledge: "You certify that you have completed this exam on your own without any help from anyone else. You understand that the only sources of authorized information in this exam are (i) the course textbook, (ii) the material that is posted at Blackboard for this class, and (iii) any study notes handwritten by yourself. You have not used, accessed or received any information from any other unauthorized source in taking this exam. The effort in the exam belongs completely to you."

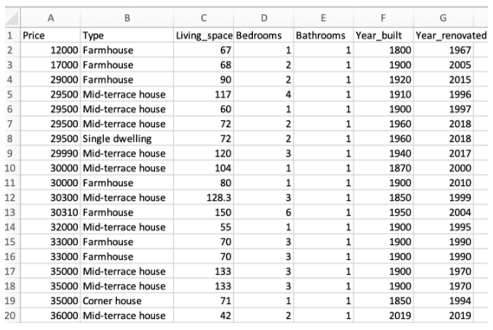
Communication before & during exam:

Zoom session (https://kocun.zoom.us/j/92425986854)
Meeting ID: 924 2598 6854, Passcode: comp125
Students are not allowed to speak or chat via this channel once the exam starts.
Violations may be punished via disciplinary action !!!
Only clarification questions regarding the midterm are allowed.
Installation related or my_code_is_not_working questions will not be answered.

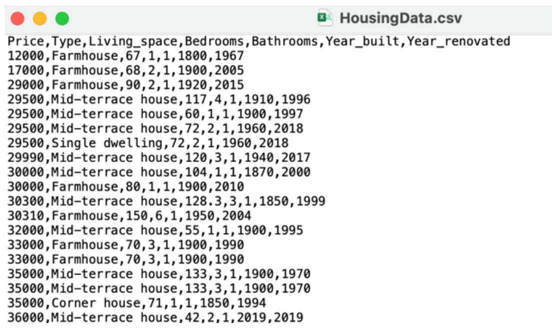
Q1: Housing Prices – 55 pts

Assume that you are the head of a real estate company and you need to build a website for the houses you have in your portfolio. You are given a csv file called HousingData.csv and you need to implement several tasks under housing.py file.

You can check the contents of HousingData.csv either using Excel (if your settings are correct) or using another program (such as TextEdit or NotePad++). Partially shown below:



	A	B	C	D	E	F	G
1	Price	Type	Living_space	Bedrooms	Bathrooms	Year_built	Year_renovated
2	12000	Farmhouse	67	1	1	1800	1967
3	17000	Farmhouse	68	2	1	1900	2005
4	29000	Farmhouse	90	2	1	1920	2015
5	29500	Mid-terrace house	117	4	1	1910	1996
6	29500	Mid-terrace house	60	1	1	1900	1997
7	29500	Mid-terrace house	72	2	1	1960	2018
8	29500	Single dwelling	72	2	1	1960	2018
9	29990	Mid-terrace house	120	3	1	1940	2017
10	30000	Mid-terrace house	104	1	1	1870	2000
11	30000	Farmhouse	80	1	1	1900	2010
12	30300	Mid-terrace house	128.3	3	1	1850	1999
13	30310	Farmhouse	150	6	1	1950	2004
14	32000	Mid-terrace house	55	1	1	1900	1995
15	33000	Farmhouse	70	3	1	1900	1990
16	33000	Farmhouse	70	3	1	1900	1990
17	35000	Mid-terrace house	133	3	1	1900	1970
18	35000	Mid-terrace house	133	3	1	1900	1970
19	35000	Corner house	71	1	1	1850	1994
20	36000	Mid-terrace house	42	2	1	2019	2019



HousingData.csv

```
Price,Type,Living_space,Bedrooms,Bathrooms,Year_built,Year_renovated
12000,Farmhouse,67,1,1,1800,1967
17000,Farmhouse,68,2,1,1900,2005
29000,Farmhouse,90,2,1,1920,2015
29500,Mid-terrace house,117,4,1,1910,1996
29500,Mid-terrace house,60,1,1,1900,1997
29500,Mid-terrace house,72,2,1,1960,2018
29500,Single dwelling,72,2,1,1960,2018
29990,Mid-terrace house,120,3,1,1940,2017
30000,Mid-terrace house,104,1,1,1870,2000
30000,Farmhouse,80,1,1,1900,2010
30300,Mid-terrace house,128.3,3,1,1850,1999
30310,Farmhouse,150,6,1,1950,2004
32000,Mid-terrace house,55,1,1,1900,1995
33000,Farmhouse,70,3,1,1900,1990
33000,Farmhouse,70,3,1,1900,1990
35000,Mid-terrace house,133,3,1,1900,1970
35000,Mid-terrace house,133,3,1,1900,1970
35000,Corner house,71,1,1,1850,1994
36000,Mid-terrace house,42,2,1,2019,2019
```

Part A: (15 pts)

Write a function called read_data(filename) to read the data from the given csv file into a matrix (stored as a list of lists).

- filename is the name of the file that should be read, e.g., "HousingData.csv"
- The return value should be a matrix in list of lists representation such that:
  - Each row in the matrix is one row of the data file
  - Each row has the same column order as the file:
    [Price, Type, Living_space, Bedrooms, Bathrooms, Year_built, Year_renovated]
- Living_space values should be **float**; Type values should be **string**; and the remaining values should be **integers**.
- Your matrix should not include the header you have in the csv file.

Part B: (15 pts)

You would like to provide a filtering functionality on the website you are building. For example, your customers should be able to filter the houses in the system based on some specific features (e.g., minimum number of bedrooms, etc.)

Write a function called filter_data(lst,bed,bath,year) to filter the input lst such that:

- lst is the list of lists you obtained from **Part A**
- bed is number of bedrooms
- bath is number of bathrooms
- year is the year that the building was built
- The return value should contain all rows from lst that have number of bedrooms greater than bed, number of bathrooms greater than bath and the year built greater than year.
  (bed, bath and year not inclusive)

For example, the output for filter_data(lst,2,2,1900) should contain all rows where number of bedrooms is greater than 2, number of bathrooms is greater than 2 and the year built is greater than 1900.

Part C: (10 pts)

You have realized that you need another filtering option on your website, which will group the houses based on the specified min_price and max_price values. Write a function called filter_price(lst, min_price, max_price) such that:

- lst is the list of lists you obtained from **Part A**
- The return value should be the number of houses between this specific range **(min_price and max_price not inclusive).** Your return value should be an **integer**.

For example:

- return value for filter_price(lst, 50000, 150000) should be 337
- return value for filter_price(lst, 40000, 120000) should be 222

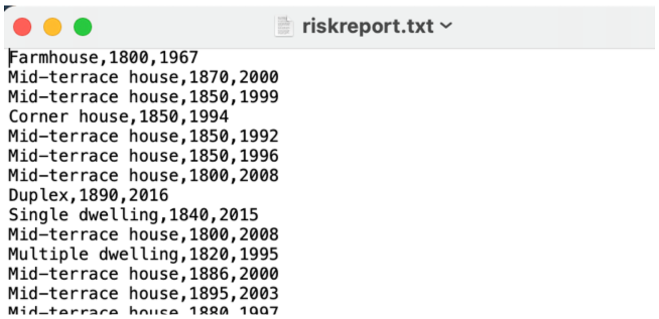
Part D: (15 pts)

Recently you have learned that the houses which were built before a specific year should be checked for safety regulations to be included on the website. That is why you need to provide a risk report for the old buildings you have in your portfolio. Your report should include the type of the building, year that the building was built and year that the building was renovated; so that it can be analyzed by the higher authorities easily.

Write a function called report(lst, year) such that:

- lst is the list of lists you obtained from **Part A**
- year is the year that the building was built
- There won't be a return value. Instead, you will create a text file called riskreport.txt which keeps the type of the building (Type), year that the building was built (Year_built) and year that the building was renovated (Year_renovated) for the buildings which had been built before the given year. **(year not inclusive)**
- The order should be as follows: Type, Year_built, Year_renovated. You should have a single comma between each element.

For example, for report(lst, 1900) the riskreport.txt file should look like this (*partially shown*):


riskreport.txt

```
Farmhouse,1800,1967
Mid-terrace house,1870,2000
Mid-terrace house,1850,1999
Corner house,1850,1994
Mid-terrace house,1850,1992
Mid-terrace house,1850,1996
Mid-terrace house,1800,2008
Duplex,1890,2016
Single dwelling,1840,2015
Mid-terrace house,1800,2008
Multiple dwelling,1820,1995
Mid-terrace house,1886,2000
Mid-terrace house,1895,2003
Mid-terrace house,1880,1997
```

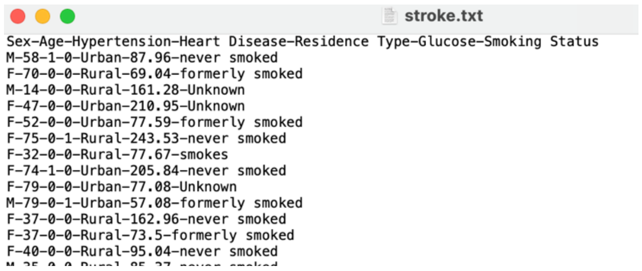
You don't need to submit riskreport.txt file – submitting .py files is sufficient.

Q2: Patient Records – 45 pts

You have been working on a study to predict the stroke risk in a population. That is why, you have been collecting the following demographical and clinical data from many subjects.

Sex-Age-Hypertension-Heart Disease-Residence Type-Glucose-Smoking Status

You are given a txt file called stroke.txt and you need to implement several tasks under stroke.py file. The contents of this txt file is *partially* shown below:



```
Sex-Age-Hypertension-Heart Disease-Residence Type-Glucose-Smoking Status
M-58-1-0-Urban-87.96-never smoked
F-70-0-0-Rural-69.04-formerly smoked
M-14-0-0-Rural-161.28-Unknown
F-47-0-0-Urban-210.95-Unknown
F-52-0-0-Urban-77.59-formerly smoked
F-75-0-1-Rural-243.53-never smoked
F-32-0-0-Rural-77.67-smokes
F-74-1-0-Urban-205.84-never smoked
F-79-0-0-Urban-77.08-Unknown
M-79-0-1-Urban-57.08-formerly smoked
F-37-0-0-Rural-162.96-never smoked
F-37-0-0-Rural-73.5-formerly smoked
F-40-0-0-Rural-95.04-never smoked
```

<u>Part A:</u> (15 pts)

Write a function called read_data(filename) to read the data from the given txt file into a matrix (stored as a list of lists).

- filename is the name of the file that should be read, e.g., "stroke.txt"
- The return value should be a matrix in list of lists representation such that:
    - Each row in the matrix is one row of the data file
    - Each row has the same column order as the file:

  [Sex, Age, Hypertension, Heart Disease, Residence Type, Glucose, Smoking Status]
- Age values should be **integer**; Glucose values should be **float**; and the remaining values should be **strings**.
- Your matrix <u>should not</u> include the header you have in the txt file.

<u>Part B:</u> (15 pts)

You would like to investigate how glucose level differs between female (F) and male (M) subjects having a specific smoking status. Write a function stats_glucose(lst, smoking_stat) such that:

- lst is the list of lists you obtained from **Part A**
- smoking_stat is the smoking status of the subject
- Your task is to calculate the average glucose level for female and male subjects having a specific smoking_stat. Let the average glucose level for the female subjects with smoking_stat be *avg_F* and the average glucose level for the male subjects with smoking_stat be *avg_M*. Your return value should be a list of tuples in the form [('F' , *avg_F*) , ('M' , *avg_M*)]
- *avg_F* and *avg_M* should be **floats**. Do not round them.

For example the return value for stats_glucose(lst, 'Unknown') should be

[('F', 96.60666666666667), ('M', 116.321)]

For example the return value for stats_glucose(lst, 'never smoked') should be

[('F', 143.3812), ('M', 108.80666666666667)]

<u>Part C:</u> (15 pts)

You would like to implement some statistical analyses on your data. Write a function calculate_ratio(lst,age) such that:

- lst is the list of lists you obtained from **Part A**
- age is the age of the subject
- For the subjects who are older than age (not inclusive) calculate the ratio of the number of subjects living in Rural area and number of subjects living in Urban area. Your return value should be a **float.** Do not round your answer. The division order should be as follows: $\frac{\# Rural}{\# Urban}$

For example, the return value for calculate_ratio(lst,40) should be **0.75**, because above age 40 there are 27 people living in rural area and 36 people living in urban area.

For example, the return value for calculate_ratio(lst,65) should be **0.7857142857142857**, because above age 65 there are 11 people living in rural area and 14 people living in urban area.

Submission + Honor Pledge

Solve each question in its own file. **DO NOT CHANGE THE NAMES OF THE FILES. DO NOT CHANGE THE HEADERS OF THE GIVEN FUNCTIONS (FUNCTION NAMES, FUNCTION PARAMETERS). DO NOT USE TURKISH CHARACTERS.**

When you are finished, compress your MT3 folder containing all of your answers:

housing.py stroke.py

The result should be a SINGLE compressed file (file extension: .zip or .rar). Upload this compressed file to Blackboard.