SHENKAR — שנקר

ENGINEERING. DESIGN. ART. הנדסה. עיצוב. אמנות.

Final Project - Scientific computing & Algorithms in Python

Lecturer: Or Perets

Overview

This is your final project in advanced Python course!

After deep understanding of Data Structures (Stack, Queue, Linked-List, Hash-Table & Graphs), Data Science and little bit of Machine Learning, you are ready to build your main project.

All tools are allowed here! New libraries, internet, friends :)

Goals

1. Check your learning progress in scientific computing topics.
2. Use your abilities to build end-to-end project with python.
3. Deep understanding of Data Analysis with Python using data science libraries, such as: numpy, matplotlib, pandas, sklearn.
4. **The most important: Have Fun!**

Requirements

I. Group size can be single, pair or triple

- Recommendation: do it alone.
- **The project evaluation will be different for each group size.**

II. Report

Your group should write a report about your progress and results.
The report should be written in **English** and should contains **screenshots** of your results (as described in **Part D**).

GOOD LUCK!

Part A - Data Structures and Algorithms

Q1

Write the class “SortedStack”.

“SortedStack” represent a **generic Stack** data structure which keeps the stack sorted.

The class should contain the following methods:

function	description
is_empty(self)	Return True if the queue is empty, else False.
size(self)	Return the size of data structure.
push(self, x)	Add x to data structure in the relevant place.
pop(self)	Return the first item in data structure.
top(self)	Return the first item **without** pop it.

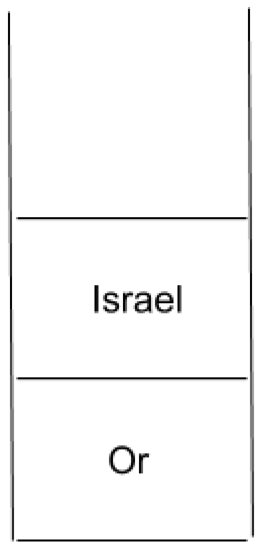
Guidelines

1. The stack should include **Students** details.
2. Each **Student** has id, first name, last name, age, gender, average.
3. Validations:
    a. ID must have 9 digits.
    b. Age can not be a negative number.
    c. Gender is binary variable: 0 - Female, 1 - Male.
    d. Average range is 0 to 100.
4. The stack always sorted by Student`s average from top to bottom.

Code Example

```
stack = SortedStack()
stack.push(Student(305884721, “Israel”, “Israeli”, 31, 1, 91.8))
stack.push(Student(304993082, “Or”, “Perets”, 29, 1, 89.5))

# Current Stack
```

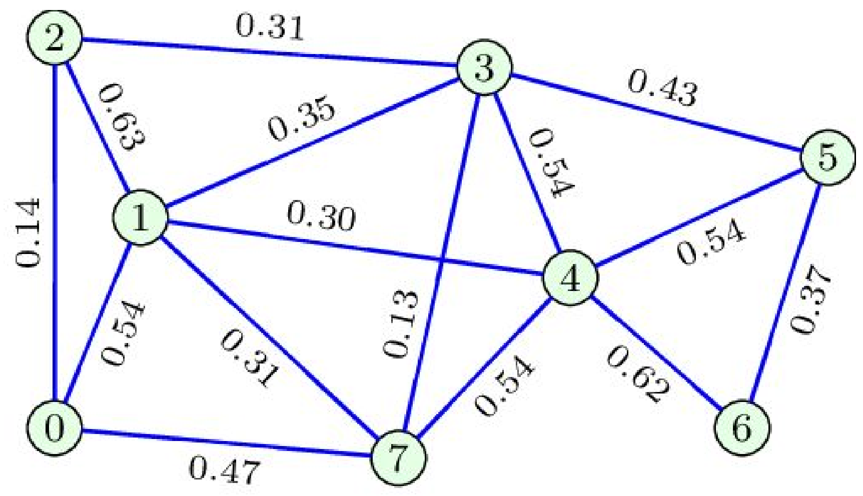


```
print(stack.is_empty())   # False
print(stack.size())  # 2
print(stack.pop())
# ID: 305884721
# First Name: Israel
# Last Name: Israeli
# Age : 31
# Gender : Male
# Average: 91.8
```

Q2

In class (and EX3), we implemented the class Graph which represent an undirected and unweighted Graph.

In this exercise, you need to implement the class “WeightedGraph” which represent an **undirected and weighted Graph.**



Let ***G = (V, E)*** be a weighted and undirected Graph, where:

1. V - Set of vertices.
2. E - Set of edges.

Each edge includes 3 items: source_vertex, dest_vertex, weight.

You need to find the **best data structure** to support the above problem and implement it.

The class should contain the following methods:

function	description
get_vertices(self)	Return a list of graph vertices.
get_edges(self)	Return a list of graph edges. [**Remember:** (source, target, weight)]
add_vertex(self, vertex, connections)	Add vertex to graph structure. Connection is list of dictionaries includes all vertex connection + weights.
delete_vertex(self, vertex)	Delete the vertex from graph.
BFS(self, source, target)	Return all paths from source vertex to target vertex.
get_shortest_path(self, source, target)	Return the shortest path from source to target (remember the weights!).
serialize(self)	Return the serialized version of your graph.
deserialize(dict)	Gets as input a dict with graph values and return WeightedGraph object.

In addition, this question should include 2 more methods:

function	description
save_graph_to_json(graph, file_name)	Gets a WeightedGraph object and save it to json file called file_name +" .json".
load_graph_from_json(file_name)	Gets a file_name and return a WeightedGraph object with given data.

Attached to the project a json file called “weighted_graph.json”.
Use your methods and read this file, build an object of WeightedGraph and print to the screen each function from above table.

You can choose your own parameters for the above methods.

Part B - Data Analysis

You have 2 tables for this part: **apps, user_reviews.**

The tables attached in “csv” files and includes data of Apps in GooglePlay store and user reviews about those apps. **(All right reserved to Kaggle).**

Apps - Table Keys

App - Application name
Category - Category the app belongs to
Rating - Overall user rating of the app (as when scraped)
Reviews -Number of user reviews for the app (as when scraped)
Size - Size of the app (as when scraped)
Installs - Number of user downloads/installs for the app (as when scraped)
Type - Paid or Free
Price - the Price of the app (as when scraped)
Content Rating - Age group the app is targeted at - Children / Mature 21+ / Adult
Genres - An app can belong to multiple genres (apart from its main category). For eg, a musical family game will belong to Music, Game, Family genres.
Last Updated - Date when the app was last updated on Play Store (as when scraped)
Current Ver - Current version of the app available on Play Store (as when scraped)
Android Ver - Min required Android version (as when scraped)

<u>User Reviews- Table Keys</u>

App - Name of app
Translated_Review - User review (Preprocessed and translated to English)
Sentiment - Positive/Negative/Neutral (Preprocessed)
Sentiment_Polarity - Sentiment polarity score
Sentiment_Subjectivity - Sentiment subjectivity score

<u>Q1 - Intro</u>

1. Print the <u>number</u> of rows in apps table.
2. Print the <u>name</u> of each column in apps table.
3. Repeat (1), (2) on user_reviews table.
4. Write conclusions:
    a. what is the **entity relationship** between the models?
    Attached to the report a simple ERD (Entities Relations Diagram) which describe the connections between those tables.
    b. What is the **Pk** and **Fk** (if exist) of each table?

<u>Q2 - First Exploring</u>

1. How many apps categories exist?
    a. Print your answer with the following statement:
    print("Total count of apps categories: {}".format(apps_categories_count))
2. What is the heaviest app? (compare by size)
3. What is the app with most installations?
4. What is the app that most updated?
5. What is the most popular app genre?

Q3 - Analysis

1. For each app genre, print the **number of apps** which are relevant to this genre.
2. Print a list of the free apps (without charge).
3. What is the most popular: Free app or Paid app?
    a. Proof your answer with specific numbers.
4. Write a function called “get_app_details_by_letter(letter)” which gets as input a letter and returns a DataFrame that contains all apps which starts with “letter”.

Q4 - Advanced Analysis

1. How many apps classified as Positive sentiment?
2. Repeat (1) with Neutral and Negative.
3. What is the sentiment of the app with the most rating?
4. What is the average polarity of free apps?
5. Write a function called “get_average_polarity(app_name)” which gets as input an app name and if the app exist in our dataset, the function returns the average of sentiment polarity. Otherwise, return an error message.
6. Write a function called “get_sentiment(app_name)” which gets as input an app name and if the app exist in our dataset, the function returns in the app_reviews are most Positive, Negative or Neutral.
    a. Hint: Use section (5).
    b. Hint: -1 < polarity < 1 (Negative to Positive).

Q5 - Advanced Analysis + Visualization

Do not forget: Title, labels, ticks, etc.

1. Display a bar diagram with count of total items by Sentiment (Positive, Negative, Neutral).
2. For each app genre, calculate the average rating and display it in a bar diagram.
3. What is the polarity difference between Free apps and Paid apps? You can describe your answer by diagram.
4. For each app category, calculate the following measures:
    a. Average of rating.
    b. STD of rating.
    c. Mode of rating.
    d. Median of rating.

   Explain in your own words, what is the most stable category? Why?
5. Based on section (4), do we have app category that it values are normal distributor?

   (You can assume $0 < \varepsilon < 0.5$)

Part C - Introduction to Machine Learning & Business Intelligence

On this part, you need to build K-Nearest-Neighbors Classifier for our dataset from part (B).

Your dataset is **apps** and **user_reviews.**

As we know, KNN is **supervised** algorithm and each supervised algorithm must have data X and labels Y.

Columns in X:

1. App
2. Category
3. Rating
4. Size
5. Type
6. Price
7. Genre
8. Sentiment_Polarity
9. Sentiment_Subjectivity

Column in Y:

1. Sentiment

Your assignment is to build KNN Classifier for this problem and find the **optimal K.**

K value can be 1 up to 20.

Print your answer with the following statement:

```
print(“Best K: {}, Best Accuracy: {}”.format(optimal_k, acc))
```

Display your results with Graphical User Interface which describes clearly the connection between the selected K and the Accuracy.

<u>Question</u>

Why we do not want to use K=data-length?

Answer this question in your report.

Part D - Report

The report of the final assignment should be descriptive, informative but with the same time short and easy to understand.

The report should be **maximum** 6 pages.

Requirements:

Title

Authors - Your names

Abstract - A short paragraph (10-14 lines) which describe your project: What did you explore, What is your findings, What this project includes.

Method - At least 2 pages that describe your work flow: What did you implement, Which algorithms did you use, How did you analyze the dataset, etc.

Conclusions - At least 2 pages describes your conclusions from this project. You can add mathematical equations, visualizations, etc.