Iryna Chervachidze

Email: irynach@bu.edu

EDUCATION highlights:

2019 - 2021 **Boston University** Master of Science, Software engineering, in progress

WORK EXPERIENCE highlights:

Boston University, **Graduate teaching assistant for MET CS 521, Informational Structures with Python, May 2020 – present**

- Holding online TA sessions explaining code and home assignments
- Checking and correcting home assignments
- Providing feedback to students
- Answering student email inquires and providing assistance on course-related topics using Zoom meetings.

Boston University/Harvard University, Graduate Research Assistant for Prof. Pinsky and Prof. Klawansky, Sept 2020 – present

- Design, implementation, and maintenance of requested programs in Python 3 to perform confidence intervals/expected intervals simulations for various data samples and output results in a convenient graphical format
- Participation in weekly research meetings

TECHNICAL SKILLS:

- **Python 3**, including essentials of Numpy, Pandas, and Matplot libraries
- **Java**, including generics, Abstract classes, concurrent threads, exception handling, inner anonymous classes, streams and lambdas, and JDBC
- Essential **SQL/PL** using **Oracle's SQL Developer**
- **Oracle, SQLLite**
- **Database design** principles, including ERD/EERD diagramming, associative relationships, bridging entities, specialization/generalization relationship, normalization process, DDL, DML and stored procedures/triggers
- Basic **MATLAB**
- **HTML/CSS** (in progress)
- **JavaScript** (in progress)
- Conceptual understanding of the following **Machine Learning** algorithms:
    - Linear regression/Ordinary least squares regression
    - Gradient descent optimization
    - Logistic regression

Iryna Chervachidze
Email: irynach@bu.edu

- Neural networks

PROGRAMMING PROJECTS highlights:

Python 3 Project: Listen and Spell Practice Program. The idea of the project is to practice spelling words grouped by a common rule. Implementation includes several classes, assert statements to test instance methods, i/o files with .txt and .json, and user input validation. This program uses a third party library for sound playback easily installed through pip install.

The user is presented with a simple menu of lessons to practice, each 10 words long, and can choose either one. Inside each lesson, the user is presented with a spoken recorded by the narrator, and gets to spell it. At the end of each lesson, the user receives a spelling score (a number of correctly spelled words).

The program can be easily extended to include new custom lessons, collect commonly misspelled words for further practice or download spelling lists. It can also be extended into a mobile app.

Data Structures and Algorithms Project: Finding the Shortest Path in the Graph Using Heuristic Algorithms. Design and implementation of a program that finds the shortest path for the given undirected graph, in Java. (Note: this project does not use Dijkstra's shortest path algorithm). This program uses several data structures and relies heavily on OOP principles. Data structures used: Java *arrays*, *ArrayList*, *Stack*, and a custom-created *adjacency list graph*.

The graph data is provided in a .txt file in a square matrix form, where each column/row name represents a vertex (node) on the graph and the value at the intersection of the row/column represents an edge (distance) between the nodes defined by the row name and column name. The user inputs the start node (such as A or any other available node), and the program recursively finds the shortest path to Z node using two algorithms. It prints out the traversed path, the shortest path (those may differ depending on the dead ends encountered) such as A => B => K => Z, and the total distance travelled for each algorithm. The algorithms are programmed in such a way that if they reach a dead end, they retrace the path back until they find an alternative node to follow.

The program can accommodate 26x26 size matrix input (graph), provided it is supplied in the .txt file. User input validation is included, i.e. the user cannot enter a non-existent starting point. The program can also be easily extended to specify any end vertex, not just Z. Even though this feature is not part of the original project, the program is also capable to “learn” the dead ends in the graph it encounters on the first traversal, so as to avoid them altogether starting with the second traversal, thus making it more efficient.

Iryna Chervachidze

Email: irynach@bu.edu

Java Project: School Supplies Store Simulation. Abstract classes, inheritance, generics, concurrent threads, exception handling, Junit tests, JDBC with SQL Lite. This project simulates activity of a small school supply store. Supply side is represented by the class Shop that controls the shop's inventory. Inventory items are stored in a SQLLite database. The demand side of the simulation is represented by the generic class Customer. Customer's buying preferences may vary from items belonging to a particular class (such as Book or Supplies) to items of certain category (math, science, language arts, history). To introduce some uncertainty into the simulation, customer preferences are chosen at random. To experiment with multiple threads, I created two shops that run concurrently, each in its own thread. Simulation runs for 30 logical days and then prints out the report about the sales, profits and top ten items most in demand for that period. The user has a choice to continue simulation at that point or quit.

Database Design and SQL/PL Project: Town Youth Soccer Club Database. A database designed for a Town Youth Soccer Club. Database contains 16 entities, 5 of which are in generalization/specialization relationship and the rest are in associative relationships. The design contains bridging entities where necessary to model M:N relationships. The database also has a history table that tracks changes in fees for annual participation and uniforms. SQL implementation includes stored procedures to populate tables as well as a trigger that records changes of fees in the history table once the fee amount changes. The project includes an Oracle SQL script that contains DDL to create and populate tables and three sample queries that feature aggregate functions, joins, and subqueries.

Python 3 Research Project (currently in progress): "Expected intervals" random sampling simulation for various sampling sizes based on two empirical data sets. Libraires used: NumPy, Pandas, MatplotLib. This program is intended to aid in the development of empirically derived "expected intervals" for ratios and difference of means of two populations. It contains two parts: a) construction of a dataset containing ratios and differences of means using random sampling without replacement for various sampling sizes b) construction of graphs illustrating the resulting sampling distributions for ratios and differences of means.

Briefly, the program first loads two data sets, such as level of cholesterol in male and female populations, or other comparable data. It then runs a specified number of random sampling of a given sampling size and records all necessary values in Panda's DataFrame (means, ratios of means and differences of means). It then uses this data to display sampling distributions in a graphical format (scatterplots).

OTHER TECHNICAL/MISCELLANEOUS SKILLS:

- Unified Modelling Language
- LucidChart
- PyCharm
- Eclipse

Iryna Chervachidze

Email: irynach@bu.edu

- Jupiter Notebooks
- Visual Studio Code
- Basic version control with Git
- Russian and Ukrainian languages (native proficiency)