**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Генерация датасета»**

**Вариант – 3**

**Студент гр. 22Б16-пу \_\_\_\_\_\_\_\_\_\_\_\_\_ Шарабарин М.С.**

**Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

# Оглавление

[Оглавление 2](#_Toc1834409691)

[Цель работы 3](#_Toc1239184037)

[Описание задачи (формализация задачи) 3](#_Toc1261516898)

[Теоретическая часть 4](#_Toc1247091128)

[Ограничения: 5](#_Toc1693265965)

[Основные шаги программы 6](#_Toc1751887202)

[Описание программы 9](#_Toc2107399932)

[Рекомендации для пользователя 12](#_Toc956650621)

[Рекомендации для программиста 13](#_Toc1380207196)

[Исходный код программы 14](#_Toc2113287483)

[Контрольный пример 14](#_Toc1558326528)

[Вывод 15](#_Toc1849527496)

[Источники 15](#_Toc529728386)

# Цель работы

Целью лабораторной работы является разработка скрипта генерации расписаня поездов rzd. Датасет должен включать личные данные пассажиров, информацию о поездах, вагонах и местах, а также данные о платежных картах для оплаты билетов.

# Описание задачи (формализация задачи)

Задача состоит в создании датасета для железнодорожных билетов со следующими требованиями:

1. **ФИО**: Славянские имена и фамилии.
2. **Паспортные данные**: Русские паспортные данные с уникальными значениями.
3. **Откуда и куда**: Города в РФ, с различием отправной и конечной точек.
4. **Дата отъезда и приезда**: Случайные даты в пределах года.
5. **Рейс**: Номера поездов в заданных диапазонах.
6. **Выбор вагона и места**: Учет классов вагонов и типов мест.
7. **Стоимость**: В зависимости от длины маршрута и типа вагона.
8. **Карта оплаты**: Генерация карт с возможностью многократного использования с повторением не больше пяти раз и возможностью настраивать вероятность к какому банку и платежной системе принадлежит карта.

# Теоретическая часть

Для создания датасета использованы несколько программных модулей:

* **Faker** (from faker import Faker):
  + Используется для генерации фиктивных данных, таких как имена и фамилии. Это полезно для создания тестовых данных, которые не содержат личной информации.
* **requests** (import requests):
  + Используется для выполнения HTTP-запросов к API Яндекс.Расписаний, что позволяет получать актуальные данные о поездах и станциях.
* **json**:
  + Используется для работы с форматированием и сериализацией JSON-ответов от API, а также для сохранения полученных данных в файл.
* **pandas** (import pandas as pd):
  + Используется для работы с данными в виде таблиц. В частности, для создания и обработки DataFrame, что позволяет удобно манипулировать данными (например, объединять их и сохранять в CSV-файлы).
* **itertools.combinations**:
  + Хотя в данном коде не видно конкретного использования этого модуля, он обычно применяется для создания комбинаций, что может быть полезно при работе с разными наборами данных.
* **concurrent.futures**:
  + Этот модуль позволяет выполнять несколько задач одновременно с использованием потоков, что может ускорить процесс получения данных от API.
* **csv**:
  + Используется для работы с CSV-файлами (чтение и запись), что важно для сохранения созданного датасета.
* **collections.defaultdict**:
  + Используется для создания словарей, которые по умолчанию создают новые записи для отсутствующих ключей, упрощая подсчёты и статистику.
* **numpy** (import numpy as np):
  + Используется для математических операций и обработки массивов, что полезно при обработке данных.
* **os**:
* В основном используется для работы с файловой системой, например, для смены директории.

# Ограничения:

* Количество строк в датасете ограничивается вводом пользователя.
* ФИО пассажиров только славянские.
* Уникальность паспортных данных и их принадлежность к русским.
* Разные точки отправки и прибытия только по территории РФ.
* Рейсы уникальные из определенных номеров, которые бывают до полного возвращения на вокзал.
* Вагоны и места генерируются по заданным типам.
* Веса банков и платежных систем определяются пользователем и должны в сумме давать число больше нуля.
* Логика выбора карт оплаты с ограничением на 5 повторов.

# Основные шаги программы

1. **Настройка параметров**: Функция configure\_params запрашивает у пользователя значения вероятностей для банков и платежных систем, настраивая их распределение.
2. **Валидация номеров карт**:
   1. luhn\_checksum вычисляет контрольную сумму номера карты по алгоритму Луна.
   2. is\_luhn\_valid проверяет номер карты на валидность, используя этот алгоритм.
   3. generate\_luhn\_valid\_card\_number создает валидный номер карты с заданным префиксом и длиной.
3. **Генерация номеров кредитных карт**:
   1. generate\_credit\_card создает номер карты, выбирая банк и платежную систему на основе предварительно заданных вероятностей.
   2. generate\_cards генерирует уникальные номера кредитных карт, учитывая максимальное количество повторений.
4. **Расчеты, связанные с поездками**:
   1. calculate\_distance\_from\_time вычисляет расстояние, основываясь на времени в пути.
   2. generate\_wagon\_seat генерирует случайные номера вагонов и мест для поездов.
   3. ticket\_price рассчитывает стоимость билета на основе времени в пути и класса вагона.
5. **Работа с данными о станциях**:
   1. get\_all\_station\_codes запрашивает коды станций из API Яндекса и сохраняет данные в JSON.
   2. timetable\_between\_stations получает расписание поездов между станциями, включая все необходимые детали.
6. **Обработка данных**:
   1. combine\_dataframes\_n\_to\_1 объединяет два DataFrame, сопоставляя строки.
   2. passport\_number генерирует случайный номер паспорта.
   3. generate\_dataset создает набор данных, включая имена, номера паспортов и кредитных карт, а также информацию о поездах, вызывая другие функции для этого.
7. **Данные собираются и записываются в файл dataset.csv.**

**Блок схема программы**

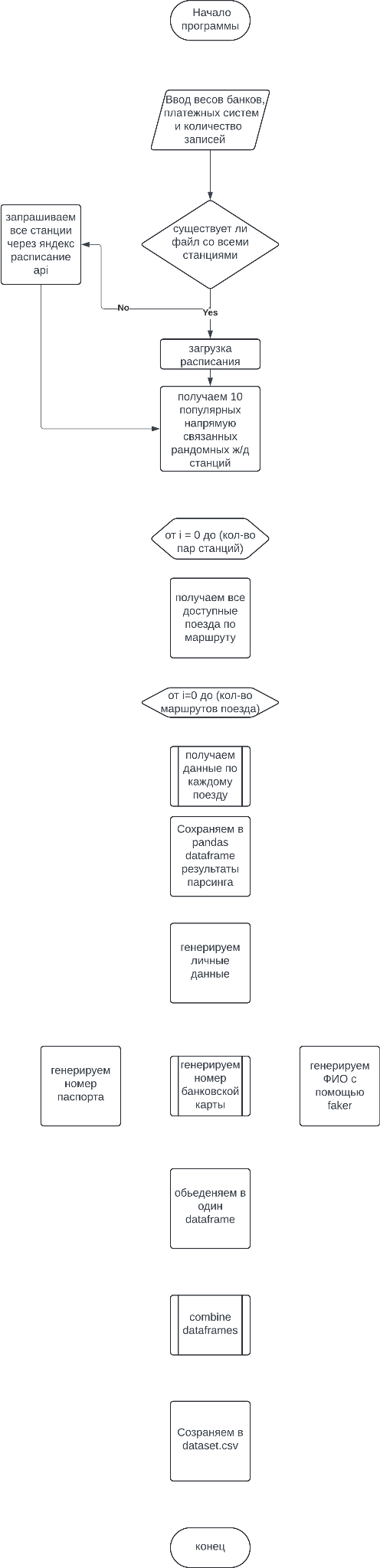


Рис 1. Блок-схема основной программы

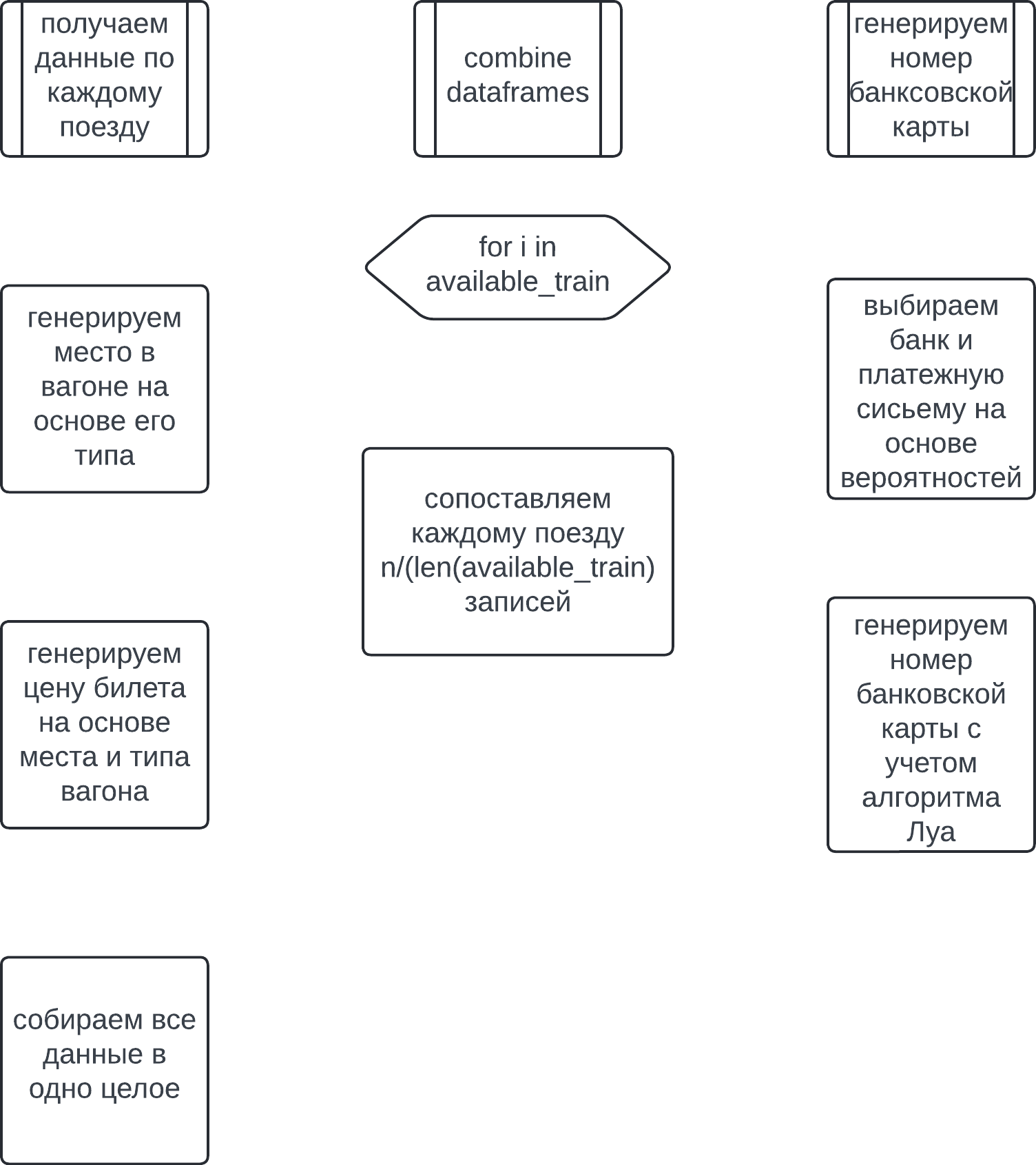


Рис 2. Блок-схема подпрограмм

# Описание программы

Программная реализация написана на языке Python 3.11.9 с использованием следующих библиотек: faker, random, math, datetime, и csv. Программа организована через модульную структуру с акцентом на генерацию данных для железнодорожных билетов. В процессе разработки программы использовались 11 функций, каждая из которых имеет чётко определённое назначение и 4 основных модуля:

1. **configure\_params(system\_distribution, bank\_name)**: Функция предназначена для настройки вероятностей распределения банков и платежных систем, интерактивно запрашивая у пользователя их значения.

* **luhn\_checksum(card\_number)**: Вычисляет контрольную сумму по алгоритму Луна для проверки валидности номера кредитной карты.
* **is\_luhn\_valid(card\_number)**: Проверяет, является ли номер карты валидным с использованием алгоритма Луна.
* **generate\_luhn\_valid\_card\_number(prefix, length)**: Генерирует валидный номер кредитной карты с заданным префиксом и длиной с использованием алгоритма Луна.
* **generate\_credit\_card(bank\_prefixes, bank\_probabilities, system\_prefixes, system\_probabilities, length=16)**: Генерирует номер кредитной карты, выбирая банк и платежную систему на основе заданного распределения вероятностей.
* **generate\_cards(bank\_distribution, system\_distribution, num\_cards, max\_repeats=5)**: Генерирует указанное количество уникальных номеров кредитных карт с учетом максимальных повторений.
* **calculate\_distance\_from\_time(duration\_seconds)**: Рассчитывает расстояние в километрах на основе времени в пути, считая, что средняя скорость поезда составляет 60 км/ч.
* **generate\_wagon\_seat(duration\_seconds)**: Генерирует случайные номера вагонов и мест для поездов, основываясь на времени в пути.
* **ticket\_price(duration\_seconds)**: Рассчитывает стоимость билета на основе времени в пути и случайно выбранного класса вагона.
* **get\_all\_station\_codes(all\_station\_codes\_url, all\_station\_code\_json)**: Запрашивает все доступные коды станций из API Яндекса и сохраняет данные в JSON файл.
* **timetable\_between\_stations(params, timetable\_between\_stations\_url)**: Получает информацию о расписании поездов между станциями, включая номер поезда, стоимость билета, место и т.д.
* **combine\_dataframes\_n\_to\_1(df1, df2, n)**: Объединяет два DataFrame, так чтобы каждая n-я строка из df1 была сопоставлена с одной строкой из df2.
* **passport\_number()**: Генерирует случайный номер паспорта.
* **generate\_dataset(data\_size)**: Создает набор данных, который включает имя, номер паспорта и номер кредитной карты, а также данные о поездах, вызывая другие функции для генерации необходимой информации.

# Рекомендации для пользователя

1. **Настройка параметров:**
   * При вводе вероятностей для банков и платежных систем рекомендуется внимательно следить за суммой вводимых значений. Она должна составлять 100% для корректного распределения.
2. **Проверка номеров карт:**
   * Убедитесь, что вводимые номера карт соответствуют формату, прежде чем проверять их валидность. Несоответствующий формат может привести к неправильной валидации.
   * Используйте функцию генерации валидных номеров карт для тестирования и разработки, чтобы избежать использования реальных данных.
3. **Генерация кредитных карт:**
   * Если вам необходимо сгенерировать много карт, укажите разумное значение для максимального количества повторений, чтобы избежать избыточного числа дубликатов.
4. **Работа с поездками:**
   * При расчетах расстояния и стоимости билетов учитывайте, что точность зависит от корректных вводимых данных (время в пути, класс вагона).
   * Проверяйте наличие актуального расписания поездов, так как оно может меняться.
5. **Обработка данных:**
   * Перед сочетанием DataFrame убедитесь, что столбцы для объединения имеют совместимые типы данных для избежания ошибок.
   * При запросе данных через API (например, коды станций) учитывайте возможные сетевые проблемы и обрабатывайте ошибки, чтобы избежать сбоев в работе приложения.

# Рекомендации для программиста

1. **Кодовая структура и читаемость:**
   * Реализуйте структурированный стиль кода с четким разделением функций на группы использования – например, функции для работы с картами, функциями для генерирования данных, обработки поездок и работы с API.
   * Используйте документацию и комментарии для каждой функции, чтобы другие разработчики могли быстро понять предполагаемую логику работы.
2. **Валидация и обработка ошибок:**
   * Добавьте дополнительные проверки входных данных, чтобы убедиться, что пользователи вводят корректные значения (например, числовые данные для вероятностей и длины карт).
   * Обязательно обрабатывайте возможные исключения при взаимодействии с API и при выполнении операций с данными (например, при сетевых запросах или работе с файлами).
3. **Параметризация и конфигурация:**
   * Рассмотрите возможность использования конфигурационных файлов (например, формата JSON или YAML) для хранения вероятностей банков и платежных систем. Это упростит изменение параметров без необходимости изменения кода.
4. **Тестирование:**
   * Напишите юнит-тесты для критичных функций (например, валидности карт и генерации данных) для повышения надежности приложения.
   * Тестируйте функции на предмет обработки крайних и некорректных значений для повышения устойчивости кода.
5. **Оптимизация производительности:**
   * Для функций, которые могут обрабатывать большие объемы данных (например, объединение DataFrame), проанализируйте и оптимизируйте алгоритмы для повышения скорости выполнения.
   * Используйте подходы, снижующие сложность алгоритмов, такие как выбор более быстрых структур данных, когда это возможно.

# Исходный код программы

**<https://colab.research.google.com/drive/1rsYQU9-_6MBwro3S1v2XemC0t2w9DRxS#scrollTo=TLELH3XyEta6>**

# Контрольный пример

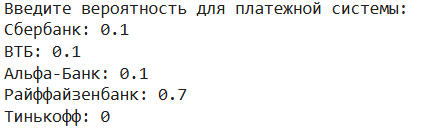


Рис 3. пример ввода весов платежных систем

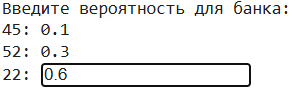


Рис 4. пример ввода весов банков



Рис 5. пример ввода количества билетов



Рис 6. пример датасета билетов

# Вывод

В данном проекте осуществлена генерация большого объема данных для симуляции работы транспортной системы, в частности — для создания данных о поездах и пассажирах. Основная задача заключалась в создании набора данных, который включает информацию о путешествиях на поездах, включая номера поездов, станции отправления и назначения, классы вагонов, а также сгенерированные номера паспортов и банковских карт.

# Источники

* Яндекс.Расписания API: <https://yandex.ru/dev/rasp/doc/ru/>
* Faker библиотека: <https://faker.readthedocs.io/en/master/>
* Pandas библиотека: <https://pandas.pydata.org/>
* Numpy библиотека: https://numpy.org/
* CSV модуль: <https://docs.python.org/3/library/csv.html>
* Concurrent Futures модуль: <https://docs.python.org/3/library/concurrent.futures.html>