**Санкт-Петербургский государственный университет Факультет прикладной математики и процессов управления**

**Лабораторная работа №2**

**Создание дата-сета**

**Вариант - 4**

|  |  |  |
| --- | --- | --- |
| Студент гр. Б15-ПУ |  | Шайдуров В.Д. |
| Преподаватель |  | Щеголева Н.С. |

Санкт-Петербург

2022 г.

**Оглавление**

[**1. Цель работы 3**](#_heading=h.gjdgxs)

[**2. Задача 3**](#_heading=h.30j0zll)

[**3. Описание программы 5**](#_heading=h.1fob9te)

[**3.1. Структура исполняемых файлов 5**](#_heading=h.3znysh7)

[**3.2. Описание функций 5**](#_heading=h.2et92p0)

[**3.3. Общий ход программы. 8**](#_heading=h.tyjcwt)

[**4. Рекомендации пользователя 9**](#_heading=h.3dy6vkm)

[**5. Рекомендации программиста 9**](#_heading=h.1t3h5sf)

[**6. Вывод 10**](#_heading=h.4d34og8)

[**7. Ссылки: 11**](#_heading=h.2s8eyo1)

# **Цель работы**

Создание дата-сета удовлетворяющего специфическим условиям при генерации данных.

# **Задача**

Необходимо создать базу данных посетителей платной поликлиники.

**A.** *Сгенерировать дата-сет в котором будут следующие наборы свойств:*

1. ФИО - Иванов Иван Иванович

2. Паспортные данные - 1234 123456

3. СНИЛС - 123-456-789 12

4. Симптомы - боль в горле

5. Выбор врача - лор

6. Дата посещения врача- 2020-01-22T08:30+03:00

7. Анализы - мазок на ковид

8. Дата получения анализов - 2020-01-23T09:30+03:00

9. Стоимость анализов - 2000 руб.

10. Карта оплаты - “1234 5678 1234 5678”

**B.** *Дополнительная информация по каждому свойству:*

1. ФИО - свободный вариант генерации данных.

2. Паспортные данные - уникальные значения.

3. СНИЛС - уникальные значения.

4. Симптомы - ”Словарь” по возможным симптомам.

5. Выбор врача - ”Словарь” по возможным специальностям которые могут работать в поликлинике.

6. Дата посещения врача- определенный вариант генерации данных.

7. Анализы - ”Словарь” по возможным анализам.

8. Дата получения анализов - после посещения врача (через сутки).

9. Стоимость анализов - свободный вариант генерации данных.

10. Карта оплаты - возможность вероятность банка (Сбербанк и тд.), через какую платежную систему (Visa и тд.) производится оплата. Оплачивать могут несколько раз с одной карты.

**C.** *Ограничения дата-сета:*

1. Всего строк в дата-сете - минимум 50 000.

2. ФИО - словарь должен состоять только из славянских ФИО.

3. Паспортные данные - только русские, белорусские и казахские паспорта.

4. СНИЛС - уникальный, но привязан к клиенту (ФИО и паспортные данные), которые могут повторяться при повторном посещение. Повторное посещение может быть к врачу минимально через 24 часа после получения анализов.

5. Симптомы - ”Словарь” должен состоять минимум из 5000 симптомов. То есть можем быть комбинация итоговых симптомов (не более 10 штук).

6. Выбор врача - ”Словарь” должен состоять минимум из 50 врачей.

7. Дата посещения врача - В рабочие время и дни недели.

8. Анализы - ”Словарь” должен состоять минимум из 250 анализов.

9. Дата получения анализов - В рабочие время и дни недели.

10. Стоимость анализов - только в рублях.

11. Карта оплаты - минимальное количество повторов - 5 раз.

# **Описание программы**

# **Структура исполняемых файлов**

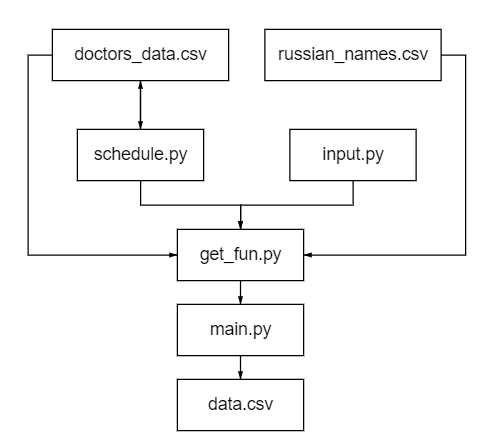


Рисунок 3.1. Карта кода

Файл doctors\_data исходная база данных, использующаяся для генерации собственного дата-сета, в которой содержится список врачей, с: симптоматикой болезней (симптомы заболеваний, с которыми чаще всего сталкивается доктор); анализами, назначаемыми доктором, и ценой за приём. Файл russian\_names также исходная база данных, которая содержит славянские имена и фамилии с указанием пола. Исполняемый файл schedule.py добавляет в doctors\_data личное расписание для каждого доктора на всю неделю. Исполняемый файл input.py открывает пользовательский интерфейс в котором можно задать параметры для генерации данных. Исполняемый файл get\_fun.py вызывает schedule.py и input.py, а также использует russian\_names и doctors\_data для различных функций, помогающих в создании собственного дата-сета data через main.py.

# **Описание функций**

В таблице 3.1 представлено описание функций исполняемого файла schedule.py.

Таблица 3.1

|  |  |  |
| --- | --- | --- |
| Имя функции | Входные  Данные | Описание функции |
| create\_schedule() | - | Каждому доктору из списка докторов представленном в doctors\_data составляется индивидуальное расписание по схеме “5 дней работаю 2 отдыхаю”, где два выходных дня могут стоять в любой день недели, и для которых можно задать процент выпадания. Так же произвольным образом определяется рабочие часы и время отдыха для каждого дня, вероятность которых также можно задавать в программе. |

Описание функций input.py представлено в таблице 3.2.

Таблица 3.2

|  |  |  |
| --- | --- | --- |
| Имя функции | Входные  Данные | Описание функции |
| get\_input() | - | Отображает примитивный интерфейс для пользователя и передаёт введённые значения по нажатию кнопки(“Создать дата-сет”) в get\_fun.py. |

В таблице 3.3 представлено описание 12 функций get\_fun.py.

Таблица 3.3

|  |  |  |
| --- | --- | --- |
| Имя функции | Входные  Данные | Описание функции |
| get\_fio() | - | *Генерация ФИО:* Генерация ФИО для каждого посетителя будет проводиться по следующим правилам, используя список имён и фамилий из russian\_names:  Если выбрано случайным образом **женское** имя, то выбираем фамилию из списка и приписываем к ней “a”, a отчество получаем приписыванием к имени “-овна”/”-евна”/”-ична” если оно оканчивается на {б, в, г, д, ж, з, к, л, м, н, п, р, с, т, ф, х, ц, ч, ш, щ}/{ь, й}/{остальные случаи} соответственно. Если выбрано случайным образом **мужское**, то фамилия остаётся без изменений, а к имени прибавляется “-ович”/”-евич”/”-ич” по аналогичным правилам.  При создании дата-сета можно указать процент посещения мужского и женского пола. |
| get\_passport() | person\_id – номер клиента | *Генерация паспорта:* В самом начале программы создаются массивы с уникальными номерами для паспортов (на каждую страну: Россия, Белоруссия, Казахстан) - отдельно, длины которых равны количеству посетителей, таким образом каждый человек имеет три индивидуальных паспортных номера (на каждую страну), но при генерации выбирается один из них с заранее задаваемой вероятностью через интерфейс. |
| get\_snils() | *Генерация СНИЛС:* Создаётся массив с уникальными значениями равный длине количества посетителей, но единый (не разбиваемый на страны), так-как действует только на территории РФ. |
| get\_card() | *Генерация способа оплаты:* Как и с паспортами, в самом начале программы для каждого посетителя создаётся массив с уникальными 12-ти значными номерами, соответствующие последним 12-ти из 16-ти цифр на карте. Первая обозначает платёжную систему: 0 – наличные (тогда 15 цифр не дописываются), 2 - МИР, 4 – Mastercard, 5 – Visa. Вторя, третья, четвёртая обозначают банк выдавший карту. Таким образом получается 16 цифр, определяющие номер карты. Через интерфейс можно задать вероятности платежной системы и эмитента выдавшего карту. |
| get\_methods() | person\_id – номер клиента  visits – кол-во посещений | Возвращает перемешанный массив с различными способами оплаты, где любые пять посещений больной расплачивается своим любимым способом оплаты, тогда как остальные = visits - 5, платёж может быть произведён уже с другой карты или другим способом. |
| get\_doctor() | - | *Генерация карточки больного:* Возвращает случайного доктора и его индекс из doctors\_data. |
| get\_symptoms() | idx – индекс доктора | Из doctors\_data по индексу доктора берётся соответствующая ему симптоматика из которой определяет случайным образом от 5 до 10 симптомов больного. |
| get\_analysis\_and\_bill() | Из doctors\_data по индексу доктора определяет случайным образом анализы которые необходимо сдать больному, а также их стоимость. |
| get\_visit() | idx – индекс доктора  previous\_visit – дата прошл. визита | *Генерация времени и даты посещения:* Возвращает точную дату и время посещения больного, во временных промежутках, когда доктор работает.  Вызывает в себе get\_date() и get\_time(). |
| get\_date() | week\_day – день недели | Проверяет чтобы в сгенерированную дату доктор работал, по след. схеме [[1]](https://lifehacker.ru/kakoj-den-nedeli/). |
| get\_time() | ot, do – временные промежутки | Определяет в какое время пришёл пациент, основываясь на временных промежутках из расписания доктора. |
| get\_analysis\_ready() | person\_come –  дата посещения  врача | Возвращает, в течении двух дней после приёма, время получения анализов клиента. |

# **Общий ход программы.**

Задаётся число посетителей, для которых по очереди свободным образом генерируется ФИО, паспортные данные, СНИЛС, симптомы, лечащий врач, дата посещения врача, анализы, дата получения анализов и их стоимость, способ оплаты (карты разных платёжных систем и эмитентов выпустивших карты).

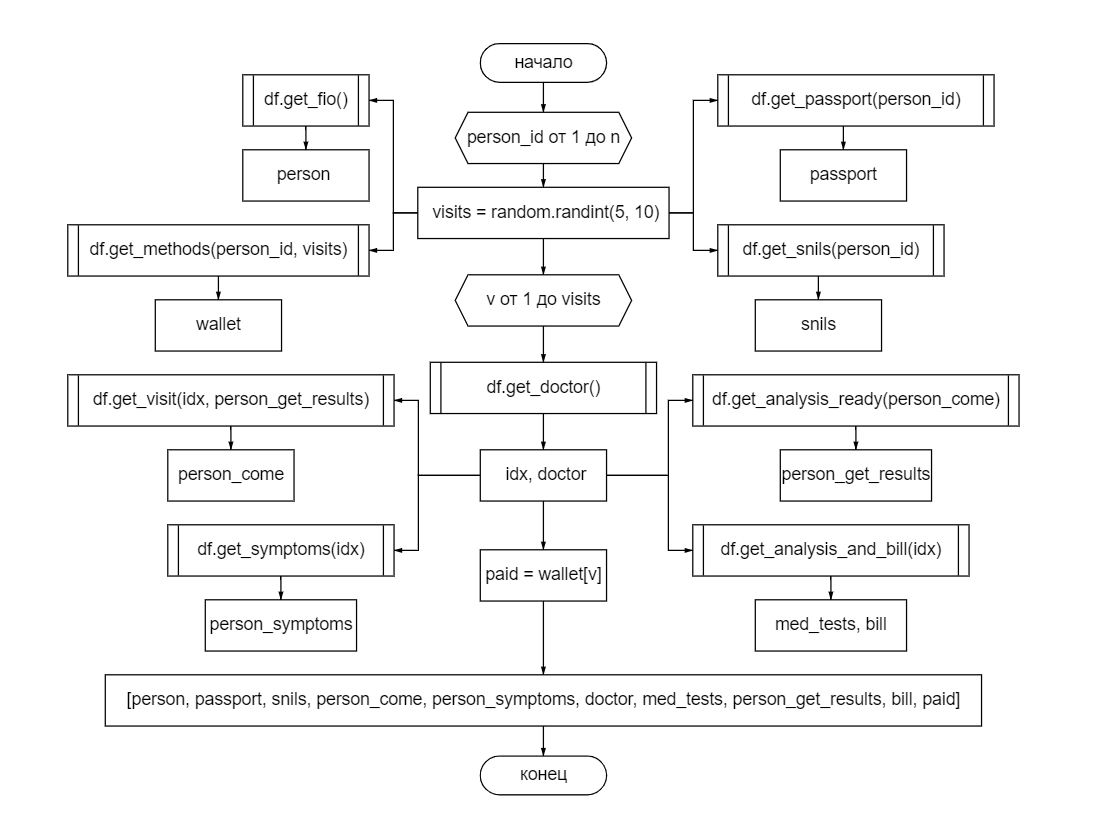


Рисунок 3.2. Блок схема main.py

# **Рекомендации пользователя**

После запуска программы введите нужные параметры, можно оставить по умолчанию. Не бойтесь ввести данные неверно, программа выдаст предупреждение.

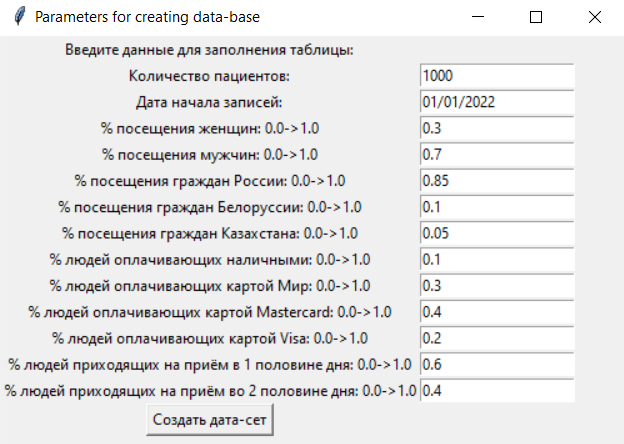
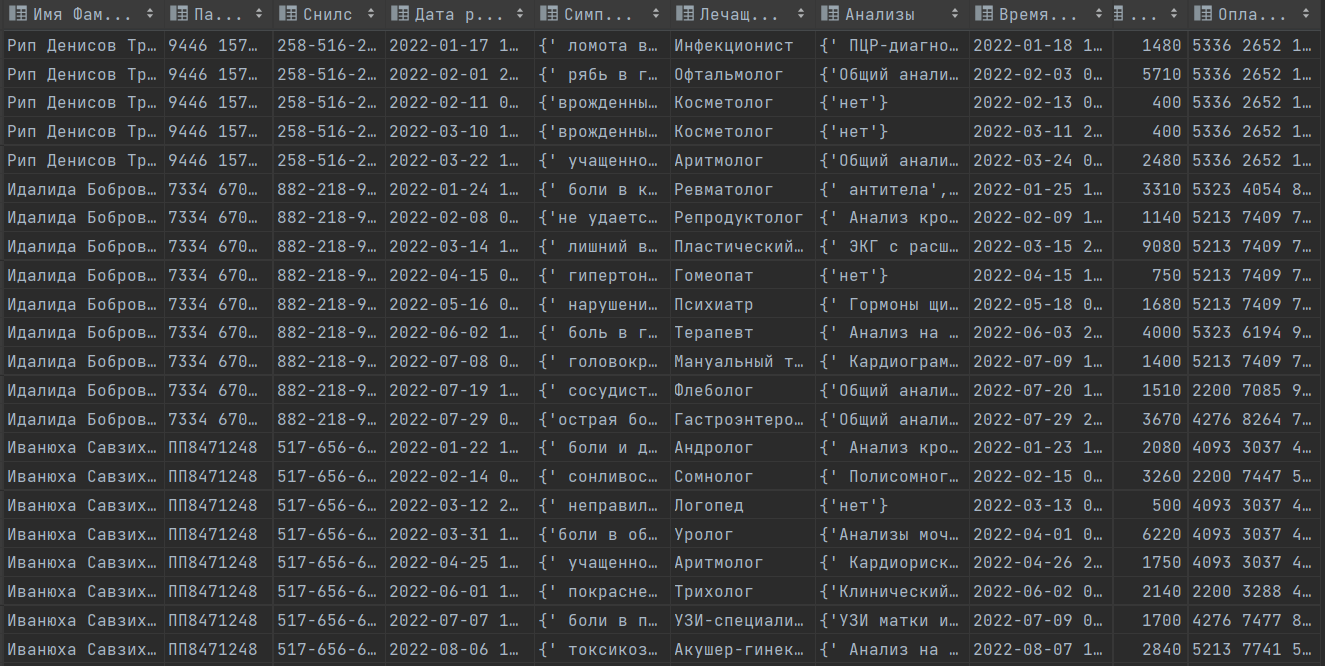


Рисунок 4.1. Блок схема main.py

Затем нажмите на кнопку «Создать дата-сет». В результате сгенерируется дата-сет в data.csv.



Как видно данные получаются упорядоченные, т.е сначала записываются k посещений первого пациента потом второго и так далее, это связано с ограничивающим условием, что человек может прийти на новый приём только через сутки. Чтобы такого не было, достаточно зайти в файловый редактор наподобие Microsoft Excel и отсортировать все строки по времени посещения пациентов.

# **Рекомендации программиста**

Для запуска программы необходима 64-битная операционная система Windows и python не ниже 3.10. Для работы с кодом необходимо IDE PyCharm или другая любая среда разработки для python.

Не забудьте импортировать следующие библиотеки:

1. datetime
2. pandas
3. random
4. numba

Необходимое количество оперативной памяти: 120 МБ (75000 строк).

# **Вывод**

В ходе выполнения данной работы была разработана программа генерирующая синтетические данные удовлетворяющий специфическим условиям.

# **Ссылки:**

[1. Дата-сет](https://disk.yandex.ru/d/zysLn_x1jpkWhw)

[2. git\_hub](https://github.com/Mooncake911/Generation-of-medical-data.git)