1 Введение

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением паматью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

Стандартная библиотека включает большой набор полезных переносимых функций, начиная от функционала для работы с текстом и заканчивая средствами для написания сетевых приложений. Дополнительные возможности, такие как математическое моделирование, работа с оборудованием, написание веб-приложений или разработка игр, могут реализовываться посредством обширного количества сторонних библиотек, а также интеграцией библиотек, написанных на Си или C++, при этом и сам интерпретатор Python может интегрироваться в проекты, написанные на этих языках. Существует и специализированный репозиторий программного обеспечения, написанного на Python, — PyPI. Данный репозиторий предоставляет средства для простой установки пакетов в операционную систему и стал стандартом де-факто для Python. По состоянию на 2019 год в нём содержалось более 175 тысяч пакетов.

Python стал одним из самых популярных языков, он используется в анализе данных, машинном обучении, DevOps и веб-разработке, а также в других сферах, включая разработку игр. За счёт читабельности, простого синтаксиса и отсутствия необходимости в компиляции язык хорошо подходит для обучения программированию, позволяя концентрироваться на изучении алгоритмов, концептов и парадигм. Отладка же и экспериментирование в значительной степени облегчаются тем фактом, что язык является интерпретируемым. Применяется язык многими крупными компаниями, такими как Google или Facebook. По состоянию на октябрь 2021 года Python занимает первое место в рейтинге TIOBE популярности языков программирования с показателем 11,27%. «Языком года» по версии TIOBE Python объявлялся в 2007, 2010, 2018 и 2020 годах.

1.1 Основные редакторы кода

В рамках данной работы предлагается работать с Python в трёх редакторах кода: VS Code, Colab и Pycharm.

1.1.1 Visual Studio Code

Visual Studio Code (VS Code) — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом.

Данный редактор поддерживает подсветку синтаксиса и отладку Python (при установленном плагине Python).

Для скачивания библиотек необходимо написать в терминале:

pip install название_библиотеки

Скачать редактор можно по ссылке: `https://code.visualstudio.com`

1.1.2 Colab

Colab – это бесплатная интерактивная облачная среда для работы с кодом от Google. В основе лежит блокнот Jupyter для работы на Python, только с базой на Google Диске, а не на компьютере. Главная особенностью являются бесплатные мощные графические процессоры GPU и TPU, благодаря которым можно заниматься не только базовой аналитикой данных, но и более сложными исследованиями в области машинного обучения.

Еще одно достоинство Colab — интеграция с GitHub. Он открывает доступ к любому хранилищу, если ему предоставить профиль на сервисе.

Для скачивания библиотек необходимо написать в блокноте:

%pip install название_библиотеки

Воспользоваться редакторов можно по ссылке: `https://colab.research.google.com` (необходимо иметь Google-аккаунт).

1.1.3 Pycharm

Pycharm – это среда разработки от компании JetBrains, которая специализируется на создании продуктов для программистов, в том числе всяких IDE *(*Integrated Development Environment). Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django. Pycharm совместим с Windows, macOS, Linux; он также имеет несколько вариантов лицензий, которые отличаются функциональностью, стоимостью и условиями использования. *Pycharm Community Edition* – бесплатная версия данного продукта.

Для скачивания библиотек необходимо написать в терминале:

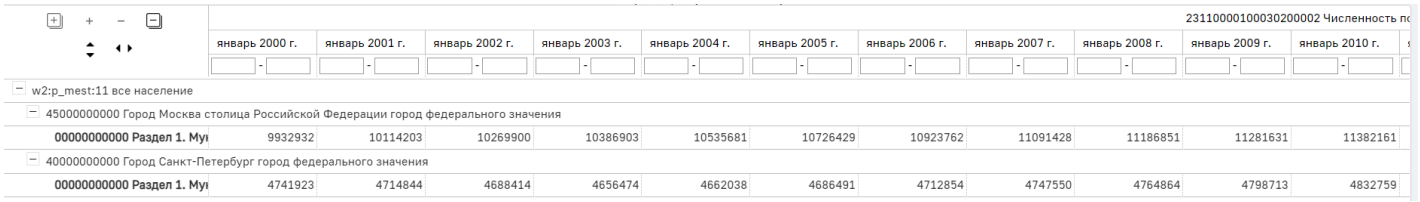
pip install название_библиотеки

Скачать редактор можно по ссылке: `https://www.jetbrains.com/ru-ru/pycharm/download/#section=windows`

2 Пример использования

Рассмотрим пример для обработки данных по численности постоянного населения Москвы и Санкт-Петербурга за период 2000-2021 годов.

Наши данные выглядят следующим образом:



	январь 2000 г.	январь 2001 г.	январь 2002 г.	январь 2003 г.	январь 2004 г.	январь 2005 г.	январь 2006 г.	январь 2007 г.	январь 2008 г.	январь 2009 г.	январь 2010 г.
w2:p_mest:11 все население											
45000000000 Город Москва столица Российской Федерации город федерального значения											
00000000000 Раздел 1. Мун	9932932	10114203	10269900	10386903	10535681	10726429	10923762	11091428	11186851	11281631	11382161
40000000000 Город Санкт-Петербург город федерального значения											
00000000000 Раздел 1. Мун	4741923	4714844	4688414	4656474	4662038	4686491	4712854	4747550	4764864	4798713	4832759

Всего Росстат выделяет нам 6 строчек, из которых нужные нам – 4ая и 6ая.

Открываем Pycharm (Visual Studio Code) и создаем .py-файл. Перво-наперво подключаем нужные нам библиотеки и модули при помощи функции *import*:

```
import pandas as pd
import numpy as np
import scipy.stats as stat
import matplotlib.pyplot as plt
```

Pandas нужен для чтения .xlsx-файла и дальнешей работы с данными; Numpy – для работы с массивами и вычислениями основным статистик; модуль stats из библиотеки Scipy позволяет вычислять некоторые иными статистики, которых нет в Numpy; и, наконец, модуль pyplot для рисования графиков. Приписка *... as ...* в каждой строчке упрощает обращение к библиотекам – теперь нет необходимости писать её длинное название, достаточно лишь написать сокращение, которое мы сами можем выбрать.

Далее следует скачать таблицу и поместить её в одну папку с .py-файлом. Для простоты я переименовал её в "moscow_spb.xlsx". Таким образом, воспользуемся функцией из Pandas для чтения .xlsx-файла:

```
df = pd.read_excel('moscow_spb.xlsx')
```

Теперь мы можем посмотреть, что внутри переменной df.

```
print(df)
```

И мы получим:

```
                                    Unnamed: 0  ... 23110000100030200002 Численность постоянного населения на 1 января.21
0                                          NaN  ...                                    январь 2021 г.
1                                  все население  ...                                               NaN
2  Город Москва столица Российской Федерации горо...  ...                                               NaN
3  Раздел 1. Муниципальные образования субъектов ...  ...                                          12655050
4  Город Санкт-Петербург город федерального значения  ...                                               NaN
5  Раздел 1. Муниципальные образования субъектов ...  ...                                           5384342

[6 rows x 24 columns]
```

Да, он не вывел нам всю таблицу, но можно увидеть, что сейчас в датафрейме 6 строк и 24 колонки. Изначально мы уже определили, что нам нужно 2 строчки, а период с 2000 по 2021 составляет 22 значения. Соответственно, нам необходимо "почистить"эти данные.

Со строчками мы определились выше – 4ая и 6ая, а с колонками всё иначе. Первые две колонки содержат ненужные индексы, а их названия слишком громоздки. Программно это выглядит следующим образом:

```
main_df = pd.DataFrame()
main_df = main_df.append(df.iloc[-3])
main_df = main_df.append(df.iloc[-1])

del main_df["Unnamed: 0"]
del main_df["Unnamed: 1"]

main_df = main_df.reset_index(drop=True)
main_df.columns = np.array([year for year in range(2000, 2022)])
```

Сначала мы создаем пустой датафрейм, куда мы положим нужные нам столбцы и строки. Затем мы добавляем нужную строку при помощи метода .append. То, что находится в скобках после .append – это то, что мы добавляем в main_df. Метод .iloc позволяет обращаться к строкам датафрейма по индексу. Этот индекс начинается с нуля, то есть, df.iloc[0] выдаст нам первую строчку, df.iloc[1] – вторую и т.д.. Однако массивы в Python позволяют принимать отрицательные значения, что равносильно "проходу по массиву"в обратную сторону. Это значит, что df.iloc[-1] выдаст последнюю строку, а df.iloc[-3] - третью с конца. Итак, строки добавлены.

Далее мы определили, что первые две колонки содержат незначащие индексы и подписи, поэтому при помощи функции del эти колонки последовательно удаляются. Так как у них не было названия, к ним можно обратиться как "Unnamed: 0"и "Unnamed: 1"соответственно.

Если мы посмотрим на наш датафрейм сейчас, то увидим громоздкие названия столбцов и неверые индексы у строк. К названиям столбцов можно обратиться при помощи метода .columns, а присвоение чего-либо соизмеримого их просто переименует. Так показано, что мы добавляем массив (array), в котором последовательно указаны все года (все целые значения), принадлежащие полуинтервалу $[2000, 2022)$. У любого датафрейма также есть возможность сбросить по умолчанию нумерацию строк – .reset_index(drop=True).

"Очищенные"данные выглядят следующим образом:

```
        2000        2001        2002  ...        2019        2020        2021
0  9932932.0  10114203.0  10269900.0  ...  12615279.0  12678079.0  12655050.0
1  4741923.0   4714844.0   4688414.0  ...   5383890.0   5398064.0   5384342.0

[2 rows x 22 columns]
```

Первый город – Москва, второй – Санкт-Петербург. То есть, наша задача сводится к тому, чтобы пройти датафрейм построчно, описать данные и нанести их на график.

Метод .iterrows() выдает нам 2 значения – индекс строки и саму строку. Таким образом, запустив цикл, мы можем "пройтись"по всем строкам. То есть,

```
for i, row in main_df.iterrows():
```

На каждой из двух итераций в переменную row будет записан массив с 22мя значениями. Для начала поделим все значения в row на 1e6, чтобы было удобнее воспринимать значения:

```
row = row / 1e6
```

Теперь можно вычислить следующие величины:

- min - минимум;
- max - максимум;
- mean - среднее;
- median - медиана;
- sd (std) - стандартное отклонение;
- interquartile range (iqr) - интерквартильный размах;
- range - размах;
- skewness (skew) - коэффициент асимметрии;
- kurtosis - эксцесс.

```
print('min: ', np.min(row), main_df.columns[np.argmin(row)])
print('max: ', np.max(row), main_df.columns[np.argmax(row)])
print('mean: ', np.mean(row))
print('median: ', np.median(row))
print('sd: ', np.std(row, ddof=1))
print('interquartile range: ', stat.iqr(row))
print('range: ', np.max(row) - np.min(row))
print('skewness: ', stat.skew(row))
print('kurtosis: ', stat.kurtosis(row))
```

Наконец, построим графики.

Чтобы для каждого города отобразить отдельно 2 графика, можно воспользоваться следующим блоком кода:

```
plt.figure()
plt.xlabel("years")
plt.ylabel("mln people")
plt.plot(x, y, label=f'{cities[i]}', marker='o')
plt.axhline(y=np.mean(y), linestyle='--', color='red', label=f'{cities[i]}_mean')
plt.fill_between(x, y, alpha=0.5)
plt.grid(True)
plt.legend()
plt.savefig('moscow_spb.png')
plt.show()
```

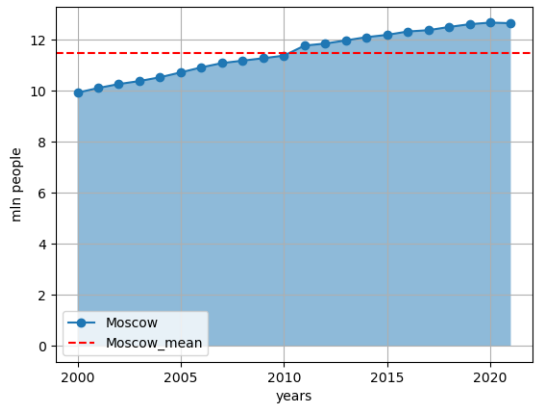
Итак,

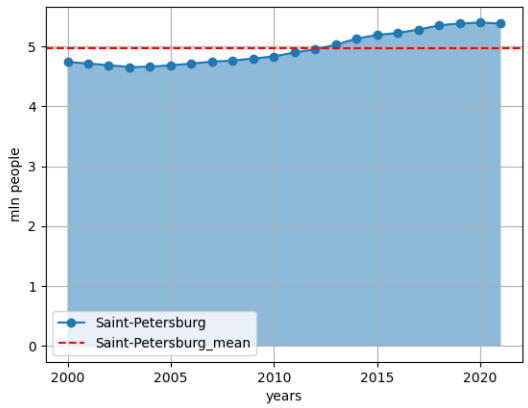
1. plt.figure() – создает пространство для графиков;
2. plt.xlabel("years") – обозначает имя у оси асбцисс;
3. plt.ylabel("mln people") – обозначает имя у оси ординат;
4. plt.plot(x, y, label=..., marker=...) – наносит обычный график на пространство для графиков, подписывая его значением из *label*, и маркируя каждую точку (каждое значение) фигурой из *marker*. x и y:

```
x = main_df.columns
y = row
```

   где x – это годы, а y – это число людей;
5. plt.axhline(y=np.mean(y)) – строит горизонтальную прямую на уровне y=np.mean(y). Для выделения этой прямой из общего графика были изменены стиль (*linestyle*) и цвет (*color*) линии;
6. plt.fill_between(x, y, alpha=0.5) – закрашивает область под графиком с прозрачностью *alpha*;
7. plt.grid() – создает прозрачную сетку (опционально);
8. plt.legend() – создает легенду графика. Сюда автоматически попадают все элементы, у которых прописан параметр *label*;
9. plt.savefig() – сохраняет полученный график в файл;
10. plt.show() – показывает полученный график без сохранения.

Запустив программу, в консоле мы увидим описательную статистику, а также в отдельном окне последовательно появятся два графика:

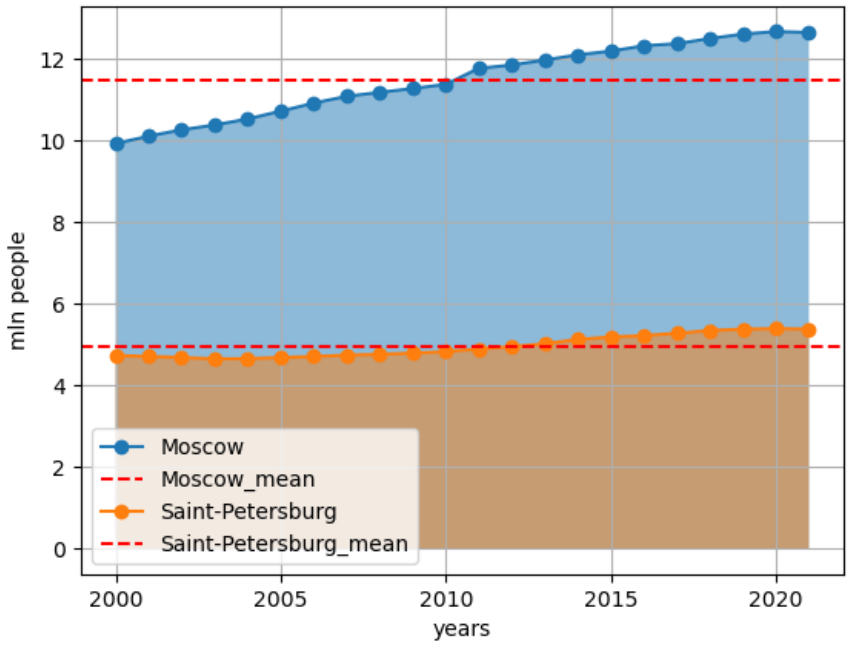




```
Moscow
min:  9.932932 2000
max:  12.678079 2020
mean:  11.496194136363636
median:  11.5794625
sd:  0.9010235074165748
interquartile range:  1.5212312499999996
range:  2.745147000000001
skewness:  -0.24241100795725168
kurtosis:  -1.2845735761574169

Saint-Petersburg
min:  4.656474 2003
max:  5.398064 2020
mean:  4.965300863636363
median:  4.8660515
sd:  0.2754972989837858
interquartile range:  0.49557625000000005
range:  0.7415899999999995
skewness:  0.425938810141327
kurtosis:  -1.4248108915554498
```

Чтобы построить два графика в одном пространстве, неодходимо определить первые 3 и последние 2 строчки из блока кода выше до и после цикла "for"соответственно. Тогда получим:



Полный код можно найти здесь.