Укрупнённый алгоритм процесса взаимодействия дата-сайентиста с заказчиком

- получены от заказчика ссылки на источники данных;
- проведен анализ структуры данных;
- формализована структура данных;
- определён список инструментов для работы;
- создан код в *Python* для загрузки данных и создания *DataFrame*;
- обработаны данные используя средства *Python/pandas*;
- получен результат обработки;
- результат преобразован в формат отчета, определённый заказчиком;
- заказчику передан отчёт.

Данные для анализа поступают в разных форматах.
Большие по объему наборы данных упакованы в архив.
Загрузка данных возможна по URL-ссылке.

Основные проблемы при выгрузке данных из файлов в DataFrame:

- Первая строка данных интерпретируется как строка заголовков
- Данные считываются в неправильной кодировке
- Файл данных упакован в архив

Функции чтения данных из файлов разных форматов в DataFrame

Функция	Значение
`pd.read_csv(path)`	Чтение файла .csv
`pd.read_table(path)`	Чтение файла .txt
`pd.read_excel(path)`	Чтение файла .xlsx и .xls
`pd.read_json(path)`	Чтение файла .json

Основные параметры функций read_csv() и read_table()

Параметр	Описание
`path`	Первый параметр, адрес файла, можно передавать адрес в интернете
`header = None`	Не использовать информацию из первой строки файла в качестве имён столбцов
`index_col = ИндексСтолбца`	Использовать в качестве индекса строк данные в указанном столбце, например `index_col = 0`
`name = [НаименованияСтолбцов]`	Задать имена столбцов, например `name = ["country", "population"]`
`sep = ";")`	Использовать в качестве разделителя символ `";"`
`encoding = "ТипКодировки"`	Указать тип кодировки считываемого файла, например `encoding = "koi8-r"`

Основные параметры функции read_excel()

Параметр	Значение
`path`	Первый параметр, адрес файла, можно передавать адрес в интернете
`sheet_name`	Ссылка на лист/листы в *Excel*-файле, например `sheetname = [0, 1, "Sheet5"]`
`na_values`	Список значений, которые будут считаться пропусками

Функции выгрузки данных из DataFrame в файлы разных форматов

Функция	Значение
`pd.to_csv(path)`	Выгрузка файла .csv
`pd.to_excel(path)`	Выгрузка файла .xlsx

Чтение данных из архива

Механизм, используемый в функции `read_csv()`, позволяет проводить чтение текстового файла из архива, не распаковывая его.

Ограничение — в ***zip***-архиве должен быть один файл.

```
data = pd.read_csv('students_performance.zip')
```

Запись данных в архив

Механизм, используемый в функции `to_csv()`, позволяет проводить упаковку CSV-файлов, например в ***zip***-архив.

```
compression_opts = dict(method='zip', archive_name='out.csv') #
Определяем параметры архивирования — метод сжатия, имя файл в архиве

data.to_csv('out.zip', index=False, compression=compression_opts) #
Записываем данные в архив
```

Считывание данных по интернет-ссылке

При чтении данных из интернета достаточно выбрать необходимую функцию и вместо пути до файла указать ссылку на файл.

```
data =
pd.read_excel('https://github.com/asaydn/test/raw/master/january.xlsx
')
```

Применение конструкции *with … as …* для чтения/записи файла

Конструкция применяется для гарантии того, что критические функции будут выполнены в любом случае.

```
with open('path/filename') as f: # Открываем файл и связываем его с объектом "f"
    # Работа с файлом...
    # ...не забываем про отступ...
    # ...
# Нет отступа = работа с файлом закончена, файл filename закрыт
```

Используемые модули

Библиотека	Описание
`import json`	Модуль для работы с *JSON*-файлами
`from pprint import pprint`	Модуль для вывода информации в структурированном виде
`import xml.etree.ElementTree as ET`	Модуль для работы со структурами *XML*