Выполнил студент БВТ2002 Митрофанов Алексей

О задании

Задание состоит из двух разделов, посвященных работе с табличными данными с помощью библиотеки pandas и визуализации с помощью matplotlib. В каждом разделе вам предлагается выполнить несколько заданий.

Задание направлено на освоение jupyter notebook (будет использоваться в дальнейших заданиях), библиотекам pandas и matplotlib.

0. Введение

Сейчас мы находимся в jupyter-ноутбуке (или ipython-ноутбуке). Это удобная среда для написания кода, проведения экспериментов, изучения данных, построения визуализаций и других нужд, не связанных с написаем production-кода.

Ноутбук состоит из ячеек, каждая из которых может быть либо ячейкой с кодом, либо ячейкой с текстом размеченным и неразмеченным. Текст поддерживает markdown-разметку и формулы в Latex.

Для работы с содержимым ячейки используется *режим редактирования* (*Edit mode*, включается нажатием клавиши **Enter** после выбора ячейки), а для навигации между ячейками искользуется *командный режим* (*Command mode*, включается нажатием клавиши **Esc**). Тип ячейки можно задать в командном режиме либо с помощью горячих клавиш (**y** to code, **m** to markdown, **r** to edit raw text), либо в меню *Cell -> Cell type*.

После заполнения ячейки нужно нажать *Shift + Enter*, эта команда обработает содержимое ячейки: проинтерпретирует код или сверстает размеченный текст.

In [ ]:
```
# ячейка с кодом, при выполнении которой появится output
2 + 2
```

Out[ ]: 4

А это ___ячейка с текстом___.

Ячейка с неразмеченым текстом.

Попробуйте создать свои ячейки, написать какой-нибудь код и текст какой-нибудь формулой.

In [ ]:
```
# your code
```

Здесь находится ~~не~~большая заметка о используемом языке разметки Markdown. Он позволяет:

0. Составлять упорядоченные списки

1. # Делать

заголовки

разного уровня

3. Выделять *текст* ~~при~~ **необходимости**
4. Добавлять ссылки

- Составлять неупорядоченные списки

Делать вставки с помощью LaTex:

$$\begin{cases} x = 16 \sin^3(t) \\ y = 13 \cos(t) - 5 \cos(2t) - 2 \cos(3t) - \cos(4t) \\ t \in [0, 2\pi] \end{cases}$$

1. Табличные данные и Pandas

Pandas — удобная библиотека для работы с табличными данными в Python, если данных не слишком много и они помещаются в оперативную память вашего компьютера. Несмотря на неэффективность реализации и некоторые проблемы, библиотека стала стандартом в анализе данных. С этой библиотекой мы сейчас и познакомимся.

Основной объект в pandas это DataFrame, представляющий собой таблицу с именованными колонками различных типов, индексом (может быть многоуровневым). DataFrame можно создавать, считывая таблицу из файла или задавая вручную из других объектов.

В этой части потребуется выполнить несколько небольших заданий. Можно пойти двумя путями: сначала изучить материалы, а потом приступить к заданиям, или же разбираться "по ходу". Выбирайте сами.

Материалы:

1. Pandas за 10 минут из официального руководства
2. Документация (стоит обращаться, если не понятно, как вызывать конкретный метод)
3. Примеры использования функционала

Многие из заданий можно выполнить несколькими способами. Не существуют единственно верного, но попробуйте максимально задействовать арсенал pandas и ориентируйтесь на простоту и понятность вашего кода. Мы не будем подсказывать, что нужно использовать для решения конкретной задачи, попробуйте находить необходимый функционал сами (название метода чаще всего очевидно). В помощь вам документация, поиск и stackoverflow.

In [ ]:
```
%matplotlib inline
import pandas as pd
```

Данные можно скачать отсюда.

1. Откройте файл с таблицей (не забудьте про её формат). Выведите последние 10 строк.

Посмотрите на данные и скажите, что они из себя представляют, сколько в таблице строк, какие столбцы?

In [ ]:
```
data = pd.read_csv('./data.csv')

data.head(5) # первые 5 строк
```

Out[ ]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98

In [ ]:
```
data.tail(10) # последние 10 строк
```

Out[ ]:

	order_id	quantity	item_name	choice_description	item_price
4612	1831	1	Carnitas Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	$9.25
4613	1831	1	Chips	NaN	$2.15
4614	1831	1	Bottled Water	NaN	$1.50
4615	1832	1	Chicken Soft Tacos	[Fresh Tomato Salsa, [Rice, Cheese, Sour Cream]]	$8.75
4616	1832	1	Chips and Guacamole	NaN	$4.45
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	$11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	$11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$11.25
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	$8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$8.75

In [ ]:
```
# вывод размерности датафрейма
data.shape
```

Out[ ]:
```
(4622, 5)
```

- Данные представляют собой список заказов с перечнем заказанных позиций, таблица имеет следующие столбцы: номер заказа, количество, название позиции, описание выбора клиента, цена
- Таблица содержит 4622 строки, 5 столбцов

2. [0.25 баллов] Ответьте на вопросы:

1. Сколько заказов попало в выборку? **1834**
2. Сколько уникальных категорий товара было куплено? (item_name) **50**

In [ ]:
```
data['order_id'].unique().shape
```

Out[ ]:
```
(1834,)
```

In [ ]:
```
data['item_name'].unique().shape
```

Out[ ]:
```
(50,)
```

3. [0.25 баллов] Есть ли в данных пропуски? В каких колонках?

In [ ]:
```
data.isna().any()
```

Out[ ]:
```
order_id              False
quantity              False
item_name             False
choice_description     True
item_price            False
dtype: bool
```

- В данных есть пропуски, они встречаются только в столбце `choice_description`

Заполните пропуски пустой строкой для строковых колонок и нулём для числовых.

In [ ]:
```
data.dtypes
```

Out[ ]:
```
order_id               int64
quantity               int64
item_name             object
choice_description    object
item_price            object
dtype: object
```

In [ ]:
```
# проверка значения и типа во 2 строке столбца `choice_description`
print(data['choice_description'][2])
print(type(data['choice_description'][2]))
```

```
[Apple]
<class 'str'>
```

- Столбец `choice_description` имеет тип object, однако в уже заполненых ячейках лежат данные типа str, поэтому **заполним пустые ячейки пустыми строками**

In [ ]:
```
data = data.fillna('')
```

4. [0.5 баллов] Посмотрите внимательнее на колонку с ценой товара. Какого она типа? Создайте новую колонку так, чтобы в ней цена была числом.

Для этого попробуйте применить функцию-преобразование к каждой строке вашей таблицы (для этого есть соответствующая функция).

In [ ]:
```
# проверка значения и типа ячейки во 2 строке столбца `item_price`
print(data['item_price'][2])
print(type(data['item_price'][2]))
```

```
$3.39
<class 'str'>
```

- Цена представлена в датафрейме типом str (строкой)
- Приведем значения цены к числовому типу с фиксированной запятой

In [ ]:
```
def str_to_decimal(x: str):
    from decimal import Decimal
    # убираем пробелы и знак '$', преобразуем в Decimal
    return Decimal(x.strip('$').strip())
```

In [ ]:
```
data['converted_price'] = data['item_price'].apply(str_to_decimal)
```

In [ ]:
```
# проверка значения и типа ячейки во 2 строке столбца `converted_price`
print(type(data['converted_price'][2]))
```

```
<class 'decimal.Decimal'>
```

Какая средняя/минимальная/максимальная цена у товара?

In [ ]:
```
# создаем новую колонку для хранения цены за штуку. значения в ней получа
data['price_for_one'] = data.apply(lambda x: x['converted_price'] / x['qua
```

In [ ]:
```
data['price_for_one'].mean()
```

Out[ ]:
```
7.084424491562094
```

In [ ]:
```
data['price_for_one'].min()
```

Out[ ]:
```
Decimal('1.09')
```

In [ ]:
```
data['price_for_one'].max()
```

Out[ ]:
```
Decimal('11.89')
```

- Средняя цена: **7.08**
- Минимальная цена: **1.09**
- Максимальная цена: **11.89**

Удалите старую колонку с ценой.

In [ ]:
```
data.pop('item_price')
```

Out[ ]:
```
0        $2.39
1        $3.39
2        $3.39
3        $2.39
4       $16.98
         ...
4617    $11.75
4618    $11.75
4619    $11.25
4620     $8.75
4621     $8.75
Name: item_price, Length: 4622, dtype: object
```

5. [0.25 баллов] Какие 5 товаров были самыми дешёвыми и самыми дорогими? (по choice_description)

Для этого будет удобно избавиться от дубликатов и отсортировать товары. Не забудьте про количество товара.

Самые **дешевые** товары:

In [ ]:
```
data[['choice_description', 'converted_price','price_for_one', 'quantity']
```

Out[ ]:

	choice_description	converted_price	price_for_one	quantity
28	[Dr. Pepper]	1.09	1.09	1
228	[Mountain Dew]	1.09	1.09	1
34		1.09	1.09	1
352	[Coca Cola]	2.18	1.09	2
200	[Diet Coke]	1.09	1.09	1

Самые **дорогие** товары:

In [ ]:
```
data[['choice_description', 'converted_price','price_for_one', 'quantity']
```

Out[ ]:

	choice_description	converted_price	price_for_one	quantity
4239	[Tomatillo Green Chili Salsa, [Black Beans, Ch...	11.89	11.89	1
2442	[Tomatillo Green Chili Salsa, [Rice, Fajita Ve...	11.89	11.89	1
3350	[Fresh Tomato Salsa, [Cheese, Guacamole, Lettu...	11.89	11.89	1
1505	[Fresh Tomato Salsa, [Rice, Pinto Beans, Chees...	11.89	11.89	1
1132	[Fresh Tomato Salsa, [Rice, Black Beans, Chees...	11.89	11.89	1

6. [0.5 баллов] Сколько раз клиенты покупали больше 1 Chicken Bowl (item_name)?

In [ ]:
```
# фильтруем датафрейм по наличию Chicken Bowl в `item_name`
is_chicken_bowl = data['item_name'] == 'Chicken Bowl'
filtered_data = data[['order_id', 'item_name','quantity']][is_chicken_bowl

# группируем по номеру заказа, суммируем количество
#
# (Иногда Chicken Bowl заказывают с разным choice_description, поэтому в
# есть несколько вхождений Chicken Bowl с одним и тем же номером заказа,
# с разным количеством.
# Поэтому будем считать не количество вхождений Chicken Bowl или максимал
# quantity, а будем считать сумму quantity для каждого номера заказа)

quantity_sum = filtered_data.groupby('order_id')['quantity'].sum()

more_than_one = quantity_sum > 1
quantity_sum[more_than_one].shape
```

Out[ ]:
```
(114,)
```

Получилось 114 вхождений

- Ответ: 114 раз клиенты покупали больше одного Chicken Bowl

7. [0.5 баллов] Какой средний чек у заказа? Сколько в среднем товаров покупают?

Если необходимо провести вычисления в терминах заказов, то будет удобно сгруппировать строки по заказам и посчитать необходимые статистики.

In [ ]:
```
# сгрупируем вхождения по номеру заказа, просуммируем цены позиций, найде
data.groupby('order_id')['converted_price'].sum().mean()
```

Out[ ]:
```
18.811428571428575
```

- Средний чек у заказа: **18.81**

In [ ]:
```
# замечание: можно посчитать сколько УНИКАЛЬНЫХ товаров за заказ покупают
# а можно посчитать общее число количества товаров за заказ

### подсчет количества УНИКАЛЬНЫХ товаров за заказ (не учитывая choice_de
# отбрасываем дублирующиеся позиции, группируем по заказу, подсчитываем к
# вхождений, рассчитываем среднее
data.drop_duplicates(subset=['order_id', 'item_name']).groupby('order_id')
```

Out[ ]:
```
quantity              2.34024
item_name             2.34024
choice_description    2.34024
converted_price       2.34024
price_for_one         2.34024
dtype: float64
```

In [ ]:
```
### подсчет общего числа купленных товаро за заказ ###
# все тоже самое, но без отбрасывания дубликатов
data.groupby('order_id').count().mean()
```

Out[ ]:
```
quantity              2.520174
item_name             2.520174
choice_description    2.520174
converted_price       2.520174
price_for_one         2.520174
dtype: float64
```

- Среднее количество *уникальных* товаров за заказ: **2.34**
- Среднее количество *всех* товаров за заказ: **2.52**

8. [0.25 баллов] Сколько заказов содержали ровно 1 товар?

In [ ]:
```
item_count = data.groupby('order_id')['item_name'].count()
exactly_one = item_count == 1
item_count[exactly_one].shape
```

Out[ ]:
```
(128,)
```

- 128 заказов содержали ровно 1 товар

9. [0.25 баллов] Какая самая популярная категория товара?

In [ ]:
```
# your code
data.groupby('item_name').count().sort_values(by = 'order_id', ascending=I
```

Out[ ]:

item_name	order_id	quantity	choice_description	converted_price	price_for_one
Chicken Bowl	726	726	726	726	726
Chicken Burrito	553	553	553	553	553
Chips and Guacamole	479	479	479	479	479
Steak Burrito	368	368	368	368	368
Canned Soft Drink	301	301	301	301	301

- **Chicken Bowl** -- самая популярная категория товаров

13. [0.75 баллов] Создайте новый DateFrame из матрицы, созданной ниже. Назовите колонки index, column1, column2 и сделайте первую колонку индексом.

In [ ]:
```
import numpy as np
df = np.random.rand(10, 3)
df

df = pd.DataFrame(data = df, columns=['index', 'col1', 'col2'])
df = df.set_index('index')
df
```

Out[ ]:

index	col1	col2
0.919831	0.486365	0.239615
0.975470	0.408335	0.446012
0.504736	0.641840	0.917223
0.958952	0.451109	0.955871
0.177391	0.697084	0.356080
0.114203	0.870182	0.960869
0.113381	0.477500	0.954255
0.607316	0.647218	0.491001
0.742483	0.448178	0.868391
0.186867	0.122353	0.593265

Сохраните DataFrame на диск в формате csv без индексов и названий столбцов.

In [ ]:
```
df.to_csv('./rand_matrix.csv', header=False, index=False)
```

In [ ]:
```
with open('./rand_matrix.csv') as f:
    print(f.read())
```

```
0.4863647370448123,0.239615006805179
0.4083351960143423,0.446011600308593
0.6418403150635593,0.9172225054255714
0.45110929562431956,0.9558714618441014
0.6970839245897686,0.35608023716416826
0.8701819678661633,0.9608694334191897
0.47750043168737033,0.9542545650783083
0.6472176750030866,0.4910012525321388
0.4481783622538278,0.8683905589863673
0.1223526115189032,0.5932651994097472
```

2. Визуализации и matplotlib

При работе с данными часто неудобно делать какие-то выводы, если смотреть на таблицу и числа в частности, поэтому важно уметь визуализировать данные. В этом разделе мы этим и займёмся.

У matplotlib, конечно, же есть документация с большим количеством примеров, но для начала достаточно знать про несколько основных типов графиков:

- plot — обычный поточечный график, которым можно изображать кривые или отдельные точки;
- hist — гистограмма, показывающая распределение некоторое величины;
- scatter — график, показывающий взаимосвязь двух величин;
- bar — столбцовый график, показывающий взаимосвязь количественной величины от категориальной.

В этом задании вы попробуете построить каждый из них. Не менее важно усвоить базовые принципы визуализаций:

- на графиках должны быть подписаны оси;
- у визуализации должно быть название;
- если изображено несколько графиков, то необходима поясняющая легенда;
- все линии на графиках должны быть чётко видны (нет похожих цветов или цветов, сливающихся с фоном);
- если отображена величина, имеющая очевидный диапазон значений (например, проценты могут быть от 0 до 100), то желательно масштабировать ось на весь диапазон значений (исключением является случай, когда вам необходимо показать малое отличие, которое незаметно в таких масштабах).

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib
```

На самом деле мы уже импортировали matplotlib внутри %pylab inline в начале задания.

Работать мы будем с той же выборкой покупкок. Добавим новую колонку с датой покупки.

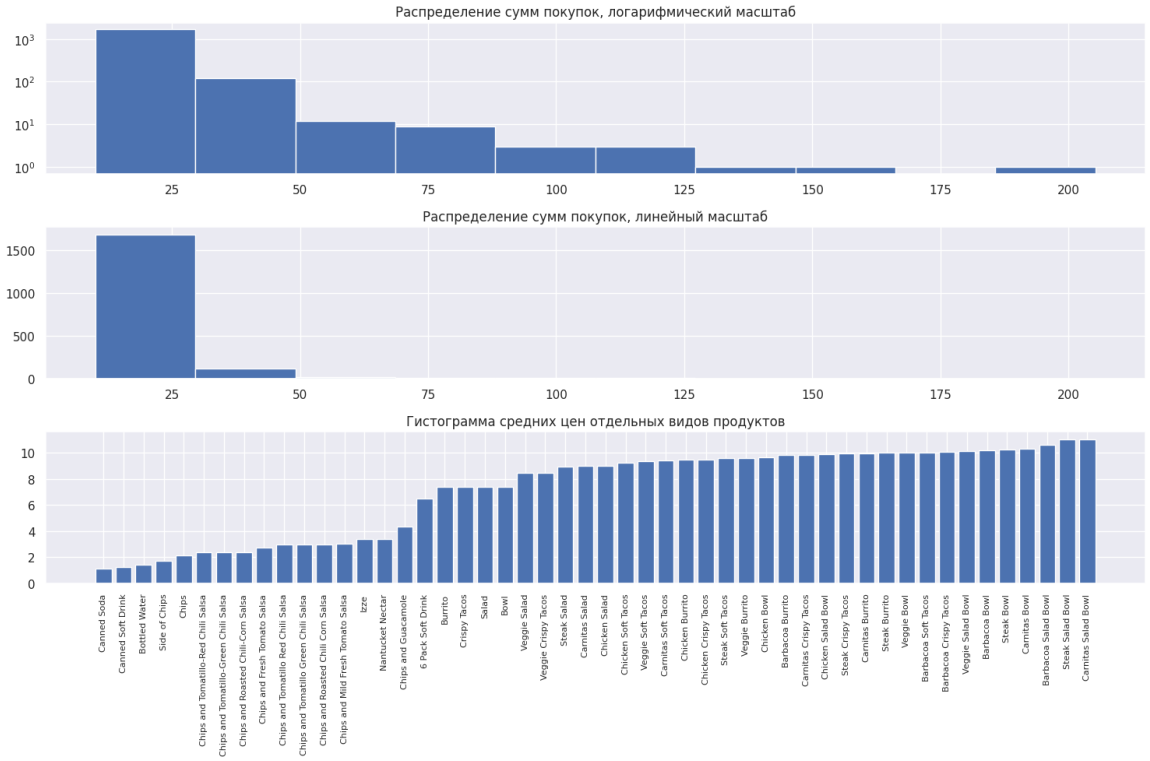
In [ ]:
```
import datetime
import random

start = datetime.datetime(2018, 1, 1)
end = datetime.datetime(2018, 1, 31)
delta_seconds = int((end - start).total_seconds())



dates = pd.DataFrame(index=data.order_id.unique())
dates['date'] = [
    (start + datetime.timedelta(seconds=random.randint(0, delta_seconds)))
    for _ in range(data.order_id.nunique())]

# если DataFrame с покупками из прошлого заказа называется не df, заменит
data['date'] = data.order_id.map(dates['date'])
```

1. [1 балл] Постройте гистограмму распределения сумм покупок и гистограмму средних цен отдельных видов продуктов item_name.

Изображайте на двух соседних графиках. Для этого может быть полезен subplot.

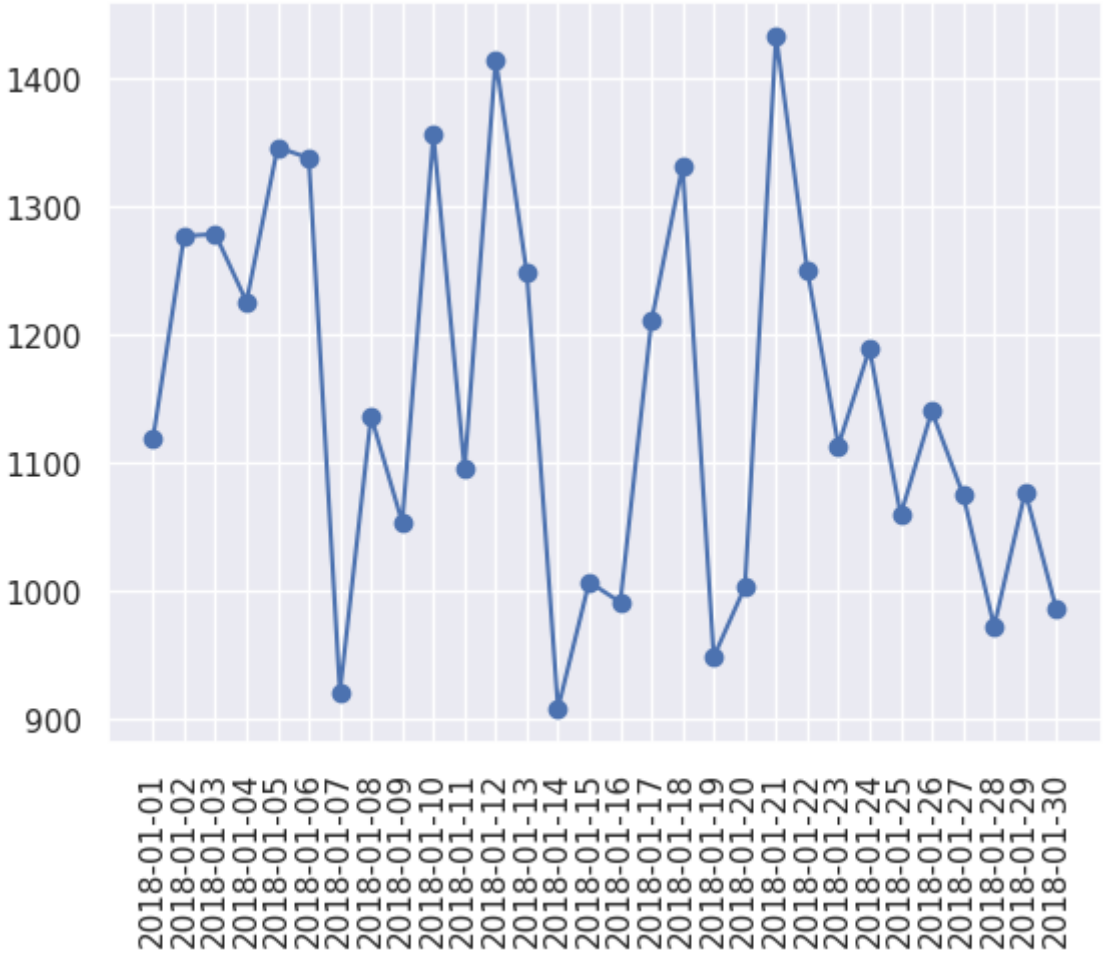
In [ ]:
```
import seaborn as sns
sns.set()
```

In [ ]:
```
# суммарные стоимости заказов
sums = data.groupby('order_id')['converted_price'].sum()

# создаем фигуру с тремя визуализациями, расположенными вертикально
fig, (ax1log, ax1lin, ax2) = plt.subplots(3)

# настройка логарифмического масштаба оси y
ax1log.set_yscale('log')
# построение гистограммы
sums.hist(ax=ax1log)
# установка названия визуализации
ax1log.set_title('Распределение сумм покупок, логарифмический масштаб')

# настройка линейного масштаба оси y
ax1lin.set_yscale('linear')
sums.hist(ax=ax1lin)
ax1lin.set_title('Распределение сумм покупок, линейный масштаб')

# средние цены отдельных видов продуктов
mean_prices = data.groupby('item_name')['price_for_one'].mean().sort_value
labels = mean_prices.index
values = mean_prices.values

ax2.bar(labels, values)
# настройка подписей по оси x
ax2.set_xticks(labels= ax2.get_xticklabels(), ticks= ax2.get_xticks(), ro
# установка размера шрифта
plt.xticks(fontsize = 8)
ax2.set_title('Гистограмма средних цен отдельных видов продуктов')

fig.set_size_inches(15, 10)
fig.tight_layout()
plt.show()
```



2. [1 балл] Постройте график зависимости суммы покупок от дней.

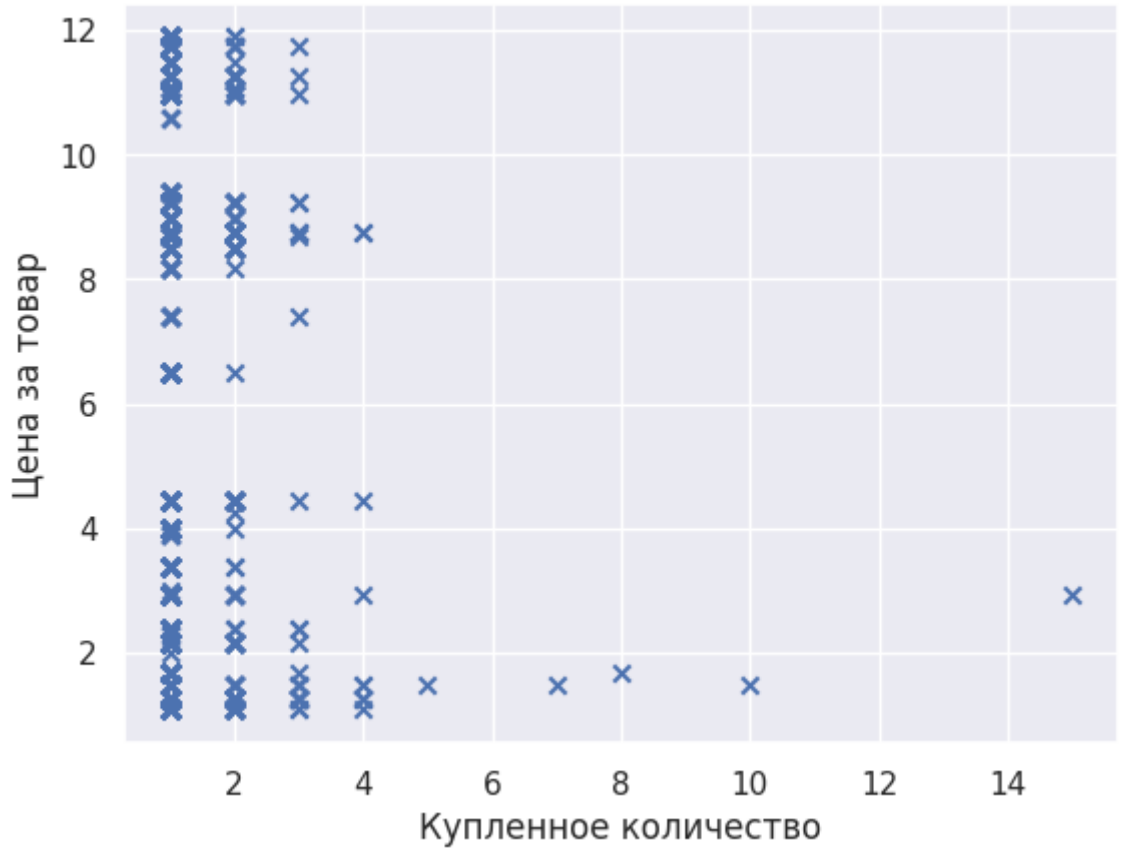
In [ ]:
```
date_sums = data.groupby('date')['converted_price'].sum()
dates = date_sums.index
sums = date_sums.values

plt.plot_date(x = dates, y = sums, xdate=True, linestyle='solid')
plt.xticks(rotation = 90)
pass
```



3. [1 балл] Постройте график зависимости денег за товар от купленного количества (scatter plot).

In [ ]:
```
# your code

prices = data['price_for_one']
quantity = data['quantity']

fig, ax = plt.subplots()

ax.scatter(x= quantity, y=prices, marker='x')
ax.set_ylabel('Цена за товар')
ax.set_xlabel('Купленное количество')
```

Out[ ]:
```
Text(0.5, 0, 'Купленное количество')
```



Сохраните график в формате pdf (так он останется векторизованным).

In [ ]:
```
fig.savefig('scatter.pdf', format='pdf')
```

Кстати, существует надстройка над matplotlib под названием seaborn. Иногда удобнее и красивее делать визуализации через неё.