**Словари (дикты)**

Ассоциативный тип данных – в нём каждый элемент является парой ключ-значение. Для создания необходимо указать элементы внутри фигурных скобок – {}  
Синтаксис элементов в словаре – {key: value}

salaries = {'Ivan': 30000} # key 'Ivan' is associated with value 30000

Чтобы узнать, что ассоциировано с ключом 'Ivan', необходимо проиндексироваться по нему:

salaries['Ivan']

30000

Ключами словаря могут быть строки и числа, а значениями почти что угодно – числа, строки, списки, даже другие словари!

**Задавание нового элемента в словаре**

salaries['Anna'] = 50000

salaries

{'Ivan': 30000, 'Anna': 50000}

**Итерирование по словарю**

* по ключам
* for name in salaries: # salaries.keys() is analogous
* print(name)
* Ivan

Anna

* по значениям
* for salary in salaries.values():
* print(salary)
* 30000

50000

**Считывание csv**

pd.read\_csv('path\_to\_your.csv') # read\_excel for reading excel files

Считывает csv файл, который лежит по указанному в скобках пути. На Windows пути к файлам содержат символ \, который является специальным символом в строках во многих языках программирования, включая Python. Поэтому необходимо сделать следующее – либо удвоить все \ в строке, содержащей путь, либо поставить r перед строкой:

* путь на Windows – C:\user\docs\Letter.txt
* строка, содержащая путь – 'C:\user\docs\Letter.txt'
* также допустимые варианты –'C:\\user\\docs\\Letter.txt' или r'C:\user\docs\Letter.txt'

На сервере мы работаем с Unix путями, например /home/user/letter.txt. С ними таких проблем не возникает – достаточно поместить путь в кавычки, чтобы всё было хорошо.

**Дополнительные аргументы функции read\_csv**

Аргументы (или параметры) – это настройки, которые мы можем задать для функции.

* **encoding** – параметр в read\_csv, отвечает за кодировку текста, которая может быть различной. Самая распространённая – utf-8. Пример указания кодировки:

pd.read\_csv('path\_to\_your.csv', encoding='Windows-1251') # now you are reading file encoded with Windows-1251

* **sep**– разделитель между ячейками в строке (по умолчанию ,)

pd.read\_csv('path\_to\_your.csv', encoding='Windows-1251', sep=';') # now you additionally specified that fields are separated with ;

* **parse\_dates** – указывает, стоит ли воспринимать даты как даты (по умолчанию они воспринимаются пандасом как строки).  
  Параметр с датами может принимать несколько значений:
  + True – пытается перевести в дату первую колонку
  + список колонок – пытается перевести в дату указанные в списке колонки

# And create\_data, payment\_data columns will be treated as data

pd.read\_csv('path\_to\_your.csv', encoding='Windows-1251', sep=';', parse\_dates=['create\_data', 'payment\_data'])

Документация

**Запись в файл**

Датафрэйм можно записать в различный формат. Пожалуй, самый распространённый – csv. Для записи в этот формат необходимо применить к датафрэйму метод to\_csv и передать в него путь, по которому вы хотите создать файл.

df.to\_csv('my.csv')

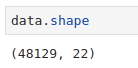
**Дополнительные параметры**

* index – записать индекс датафрэйма в csv как первую колонку
* sep – используемый при записи разделитель колонок

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_csv.html)

**Размеры датафрэйма**

shape – атрибут, хранящий данные о размерах таблицы. Возвращает кортеж (для простоты воспринимайте его как неизменяемый список). В случае датафрэйма кортеж содержит два значения – число строк и число колонок в нём.



Например, в данном случае размер таблицы: 48129 строк и 22 колонки.

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.shape.html)

**Типы колонок**

Чтобы узнать типы колонок в вашем датафрэйме, воспользуйтесь атрибутом dtypes – он возвращает серию с описанием типа каждой колонки. Типы более-менее совпадают с типами в python, однако есть и различия:

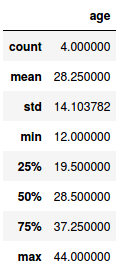
* здесь у типов присутствует описание размера (числа битов)
* все сложные типы (не числа или логические значения) отображаются как object

Информация о типе важна для дальнейшей работы с датафрэймом (например, чтобы не произвести сложение строк, думая, что это числа).

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dtypes.html)

## Метод describe

describe – удобный метод для вывода описания числовых колонок в датафрэйме:



describe выводит информацию о числе строк, среднем значении, стандартном отклонении, минимуме, максимуме и значениях 25-го, 50-го и 75-го квартилей. Он действует только на числах, так как большинство этих параметров неочевидно определяются для других типов данных (например, строк).

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html)

## Переименование колонок

В идеале названия колонок осмысленны, актуальны, не содержат пробелов и на английском языке. Конечно, для каких-то задач, они могут быть и с пробелами, и на другом языке. В любом случае, если вы хотите их переименовать, для этого есть метод rename. Пример переименования колонки x в name, а колонки y в salary:

# Rename columns

df = df.rename(columns={'x': 'name', 'y': 'salary'})

А вот пример переименования лэйблов строк из 0 в Ivanov и из 1 в Vasilev:

# Rename index (row names)

df = df.rename(index={0: 'Ivanov', 1: 'Vasilev'})

Один из способов переименования – передать словарь, в котором ключами являются старые названия, а значениями – новые.

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rename.html)

**Обращение к колонкам**

В пандасе существует множество способов обратиться к колонке датафрэйма. Самый удобный:

df.column\_name

* df – датафрэйм
* column\_name – название колонки

Чтобы это работало, название колонки должно состоять из одного компонента (например, слова), и не должно совпадать с названием методов датафрэйма (имя колонки count не сработает). Для языковой однородности – ещё и на английском, но это не является обязательным.

Но что делать, если название колонки состоит из двух слов? В таком случае необходимо либо переименовать колонку, либо использовать другой способ доступа:

df['column name']

Работает для всех случаев кроме тех, когда в названии присутствуют одинарные кавычки. Тогда либо используйте вокруг названия двойные, либо поставьте \ перед кавычками внутри. Лучше называть колонки без кавычек.

Для получения нескольких колонок передайте внутрь квадратных скобок список с именами желаемых колонок:

df[['column1', 'column2', 'column3']]

[Документация](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html)

**Применение вычислительных методов**

Существует набор методов, доступных для колонок датафрэймов. Например, есть колонка money в датафрэйме, содержащая полученные объёмы денег. Применив метод sum, можно посчитать их сумму.

df.money.sum()

69120

Ещё несколько примеров:

* product – перемножение
* std – среднеквадратичное отклонение
* var – дисперсия

[Больше информации](https://pandas.pydata.org/pandas-docs/stable/user_guide/computation.html#method-summary)

**Цепочка методов (method chaining)**

Приём для объединения нескольких действий в одно. Большинство методов датафрэймов возвращают вам результат, который довольно часто тоже является датафрэймом. Следовательно, от него тоже можно вызвать метод.

Стандартная запись:

df = df.query('income >= 1000')

df = df.groupby(['title', 'status'], as\_index=False).agg({'income': 'sum', 'id': 'count'}) # groupby is usually immediately followed by agg

df = df.sort\_values(['title', 'status'])

Сокращённая запись:

df = df.query('income >= 1000').groupby(['title', 'status'], as\_index=False).agg({'income': 'sum', 'id': 'count'}).sort\_values(['title', 'status'])

Как можно заметить, эта запись довольно длинная и не очень удобная для чтения. Обычно, такая цепочка оформляется в блок, где каждый метод идёт на своей строке. Есть два варианта оформления, какой из них выбрать – вопрос предпочтения и конвенции в вашей организации:

# \ after each nonfinal line to demarcate line continuation for python

df = df.query('income >= 1000') \

.groupby(['title', 'status'], as\_index=False) \

.agg({'income': 'sum', 'id': 'count'}) \

.sort\_values(['title', 'status'])

# Parentheses around the whole expression for the same purpose as backslash in previous example

df = (df.query('income >= 1000')

.groupby(['title', 'status'], as\_index=False)

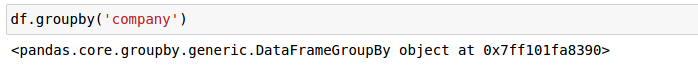
.agg({'income': 'sum', 'id': 'count'})

.sort\_values(['title', 'status']))

[Больше информации](https://towardsdatascience.com/the-unreasonable-effectiveness-of-method-chaining-in-pandas-15c2109e3c69)

## Группировка

Часто используемый приём для вычисления чего-либо по данным. Осуществляется с помощью метода groupby – группирует данные в датафрэйме по указанным колонкам:



Применение одного метода groupby не даёт видимого эффекта, хотя на самом деле все строки были объединены в группы по значению в колонке company: с одним значением в одну группу, с другим – в другую. groupby обычно используется не сам по себе, а в связке с agg или другим методом. Можно использовать несколько колонок для группировки, передав их в виде списка.

### Дополнительные параметры

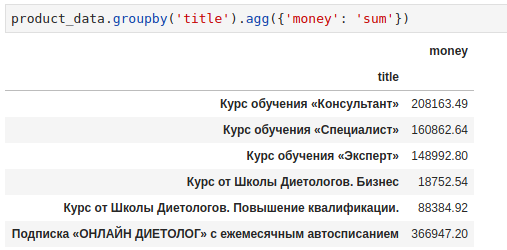
as\_index – принимает True или False для обозначения того, нужно ли использовать переданные для группировки колонки в качестве индекса, по умолчанию True

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html)

## Агрегация

agg – функция для агрегирования данных, применяется после группировки методом groupby

В agg указывается, на каких колонках какие действия необходимо произвести:



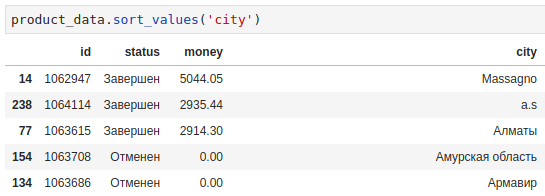
Существуют разные способы передать в agg что и как вы хотите агрегировать. Самый простой и полный – использовать словарь, в котором ключами являются названия колонок, а значениями – применяемые к ним функции. Чтобы применить несколько функций, используйте список функций. Можно передать как сами функции (sum), так и обозначающие их строки ('sum').

В результате агрегации из массива значений (колонка) получается одно значение на каждую агрегирующую функцию.

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.agg.html)

**Сортировка значений**

Для такой сортировки используется метод sort\_values, принимающий колонку или список колонок, по которым будет идти сортировка (обратите внимание: заглавные буквы считаются меньше обычныx):



**Дополнительные параметры**

ascending – принимает логическое значение, которое показывает, сортировать ли колонку по возрастанию.

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sort_values.html)

**value\_counts**

Метод, который считает, сколько раз встречается каждое уникальное значение переменной. Например, имеется следующий набор данных:



Посчитать, сколько раз встречается каждое имя (name), можно с помощью следующей команды:

df['name'].value\_counts()

Результат возвращается в формате pd.Series (серии) :

Persik 2

Tolya 1

Barsik 1

Name: name, dtype: int64

Также метод value\_counts принимает на вход несколько **параметров**:

* normalize – показать относительные частоты уникальных значений (по умолчанию равен False).
* dropna – не включать количество NaN (по умолчанию равен True)
* bins – сгруппировать количественную переменную (например, разбить возраст на возрастные группы); для использования данного параметра необходимо указать, на сколько групп разбить переменную

Несколько примеров:

1) Получаем частоту встречаемости (напр. Persik – в 40% наблюдений),  также не удаляем из результата NaN:

df['name'].value\_counts(normalize=True, dropna=False)

Persik 0.4

Tolya 0.2

Barsik 0.2

NaN 0.2

Name: name, dtype: float64

2) Разбиваем year на 2 промежутка:

df['year'].value\_counts(bins=2)

(2017.5, 2020.0] 3

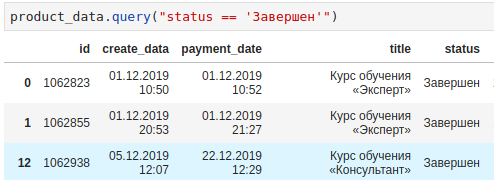
(2014.994, 2017.5] 2

Name: year, dtype: int64

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.value_counts.html)

## Запросы

В пандасе есть возможность фильтровать данные, используя SQL-like синтаксис. Для этого применяется метод query, принимающий строку с запросом. Внутри него можно использовать названия колонок (если они без пробелов). При использовании строк внутри запроса экранируйте кавычки с помощью \ или используйте другую пару.



В query также можно передать сразу несколько условий. Условия, которые должны выполняться одновременно, соединяются с помощью and или &:

product\_data.query("title == 'Курс обучения «Эксперт»' and status == 'Завершен'")

Когда должно удовлетворяться одно из условий – or или |:

product\_data.query("title == 'Курс обучения «Эксперт»' or status == 'Завершен'")

[Документация](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.query.html)

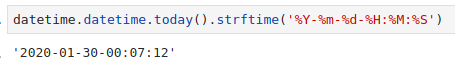
**Время**

Для работы с датой и временем можно использовать модуль datetime. Для получения данных о времени в момент вызова функции используйте функцию today в одноимённом подмодуле:

import datetime # by convention imports are placed in the head of file and separated with 2 blank lines from other code

date = datetime.datetime.today()

Само по себе это даст вам специальный тип даты. Чтобы перевести его в строку сделайте следующее:



Метод strftime форматирует дату в соответствии с переданным в него форматом:

* % – обозначает, что далее следует часть даты
* Y – год четырьмя знаками
* m – месяц двумя знаками
* d – день
* H – час
* M – минуты
* S – секунды

Можно использовать только часть фрагментов даты, разделители между ними – на ваше усмотрение (в примере это - и :). Несколько примеров:

from datetime import datetime

# current date and time

now = datetime.now()

print(f'Full time format of now is {now}')

Full time format of now is 2020-06-01 17:54:40.010540

# Year

year = now.strftime("%Y")

print("year:", year)

year: 2020

# Month

month = now.strftime("%m")

print("month:", month)

month: 06

# Day

day = now.strftime("%d")

print("day:", day)

day: 01

# Time

time = now.strftime("%H:%M:%S")

print("time:", time)

time: 17:54:40

# Date and time

date\_time = now.strftime("%m/%d/%Y, %H:%M:%S")

print("date and time:",date\_time)

date and time: 06/04/2020, 17:54:40

[Документация](https://docs.python.org/3/library/datetime.html)