pandas

- Pandas - библиотека для обработки и анализа данных. Предназначена для данных разной природы - матричных, панельных данных, временных рядов. Претендует на звание самого мощного и гибкого средства для анализа данных с открытым исходным кодом.

```
import pandas as pd
```

В пандас есть две структуры данных:

- Series: одномерный массив с именованными индексами (чаще всего, данные одного типа)
- DataFrame: двухмерный массив, имеет табличную структуру, легко изменяется по размерам, может содержать в себе данные разных типов

Оба типа можно создавать вручную с помощью функций из самой библиотеки:

- pandas.Series(data=None, index=None, dtype=None)
- pandas.DataFrame(data=None, index=None, columns=None, dtype=None)
- **data** - данные, которые надо записать в структуру
- **index** - индексы строк
- **columns** - названия столбцов
- **dtype** - тип данных

Кроме data, остальные параметры опциональны

Series

	Марка машины
0	Toyota
1	Honda
2	Ford
3	Chevrolet
4	Volkswagen

DataFrame

	марка машины	год выпуска	цена
0	Toyota	2022	25120.60
1	Honda	2021	22536.90
2	Ford	2020	15305.10
3	Chevrolet	2022	21733.50
4	Volkswagen	2021	45345.88

Создание Series (одномерного массива)

	Марка машины
0	Toyota
1	Honda
2	Ford
3	Chevrolet
4	Volkswagen

```
car_prices = pd.Series({
    'Toyota': 25000,
    'Honda': 22000,
    'Ford': 23000,
    'Chevrolet': 21000,
    'Volkswagen': 27000
})

car_prices2 = pd.Series({
    1:'Toyota',
    2:'Honda',
    3:'Ford',
    4:'Chevrolet',
    5:'Volkswagen',
})


# Выводим таблицу Series
print(car_prices, end='\n\n')
print(car_prices2)
```

Создание DataFrame (Таблицы)

	марка машины	год выпуска	цена
0	Toyota	2022	25 000
1	Honda	2021	22 000
2	Ford	2020	15 000
3	Chevrolet	2022	21 000
4	Volkswagen	2021	45 000

> **1 Вариант**

```
# Создаем исходную таблицу DataFrame
data = {
    'марка машины': ['Toyota', 'Honda', 'Ford', 'Chevrolet', 'Volkswagen'],
    'год выпуска': [2022, 2021, 2020, 2022, 2021],
    'цена': [25120.60, 22536.90, 15305.10, 21733.50, 45345.88]
}

df = pd.DataFrame(data)

display(df)

df
```

Типы данных

- object - строковый тип данных или смешанный текст/число
- int64 - целочисленный тип, занимает 64 бита
- float64 - тип с плавающей точкой (вещественные числа), занимает 64 бита
- bool - логический тип данных для хранения значений True/False
- datetime64 - тип для хранения дат и времени
- timedelta64 - разница между датами
- category - тип для категориальных данных (ограниченный набор значений)

```
display(df)
print(df.dtypes)
```

2 Вариант

```
columns = ['марка машины', 'год выпуска', 'цена']

data = [['Toyota',         '2022',        '25000'],
        ['Honda',          '2021',        '22000'],
        ['Ford',           '2020',        '23000'],
        ['Chevrolet',      '2022',        '21000'],
        ['Volkswagen',     '2021',        '27000']]

df2 = pd.DataFrame(data, columns=columns)

display(df2)
```

Зададим индексы

```
columns = ['марка машины', 'год выпуска', 'цена']

data = [['Toyota', '2022', '25000'],
        ['Honda', '2021', '22000'],
        ['Ford', '2020', '23000'],
        ['Chevrolet', '2022', '21000'],
        ['Volkswagen', '2021', '27000']]

index = [100, 200, 300, 400, 500] # должны быть уникальными !!!

df2 = pd.DataFrame(data=data, index=index, columns=columns, dtype=object)

# Выводим таблицу
display(df2)
print(df2.dtypes)
```

3 Вариант

```
text = '''
      марка машины, год выпуска, цена
      Toyota, 2022, 35 000
      Honda, 1987, 1 500
      Ford, 2005, 6 000
      Volkswagen, 2023, 40000
'''.strip()

with open('car.csv', 'w') as f:
  f.write(text)


df2 = pd.read_csv('/content/car.csv')

display(df2)
```

`delimiter=','` (разделитель)

```
text = '''
      марка машины; год выпуска; цена
      Toyota; 2022; 35 000
      Honda; 1987; 1 500
      Ford; 2005; 6 000
      Volkswagen; 2023; 40000
'''.strip()

with open('car.csv', 'w') as f:
  f.write(text)


df2 = pd.read_csv('car.csv')
df3 = pd.read_csv('car.csv', delimiter=';')

display(df2)
print(end='\n\n')
display(df3)
```

`encoding` (кодировка)

```
text = '''
      марка машины; год выпуска; цена
      Toyota; 2022; 35 000
      Honda; 1987; 1 500
      Ford; 2005; 6 000
      Volkswagen; 2023; 40000
'''.strip()

with open('car.csv', 'w', encoding='cp1251') as f:
  f.write(text)


df4 = pd.read_csv('car.csv', delimiter=';', encoding='cp1251')

display(df4)
```

Записать используя методы Pandas

```
df.to_csv('car2.csv') # Записываем в csv
```

При чтение файла мы получаем лишнию индексацию.

```
df_car2 = pd.read_csv('car2.csv') # Читаем csv
df_car2
```

```
df_car2 = pd.read_csv('car2.csv', index_col=0) # Указываем колонку и индексами
df_car2
```

Можем сразу записать файлы исключая индексы

```
df.to_csv('car2.csv', index=False) # Записываем в csv
df_car2 = pd.read_csv('car2.csv') # Читаем csv
df_car2
```

```
pd.read_
```

```
df.to_
```

Загрузка таблицы

База вин = https://drive.google.com/file/d/1I0IrgrlR-Phv1sQZHOfr1ld866yMTahE/view

```
!pip install --upgrade gdown
```

1 способ загрузки таблицы

```
import gdown

path_download = 'https://drive.google.com/uc?id=1I0IrgrlR-Phv1sQZHOfr1ld866yMTahE'

gdown.download(path_download, None, quiet=False)
```

```
%cat wine_base.csv
```

Выходные данные были обрезаны до нескольких последних строк (5000).

```
2186,Italy,"Aromas recall coffee, toast and weedy underbrush. The lean palate offers licorice, espresso, black pepper and dried fruit flavors al
2187,Italy,"Berry, dried sage, licorice and menthol aromas lead the nose. The warm, one-dimensional palate offers toasted oak, coconut, spirit-s
2188,Italy,"Aromas of oak, toast and coconut carry through to the palate along with spirit-soaked plum, fig and anisette. It doesn't have enough
2189,France,"Black currant fruits dust the aromas of this wine. The palate shows a stalky character, with dusty tannins and a dry core. It's old
2190,US,"This unoaked, Dijon-clone wine is lively and lush, with tongue-coating fruit. Melon, banana and papaya flavors steer it in a tropical d
2191,Italy,"Spanish broom, jasmine, peach and apple aromas lead the nose of this charming white that hails from an estate located just outside o
2192,US,"Soft and earthy, this wine's herbal, funky nature suggests Carneros as much as Sonoma Coast, with black tea and crunchy dried cranberry
2193,US,"Here is a well-focused wine with appealing toastiness as soon as it's opened. Fruit flavors are still tight, but unfold into tangy rasp
2194,Spain,"Stripped-back cherry and raspberry flavors take on a jammy character with airing. This is tight, bright and mildly acidic on the pal
2195,US,"Tempting buttery flavors and a lively texture combine well in this medium-bodied wine. It smells a bit like butterscotch, and this note
2196,Germany,"Profoundly floral, this aromatic white balances ebullient violet and lavender perfume against crisp grapefruit and gooseberry flav
2197,US,"This rich, almost sweet wine owes a lot of its exuberance to the flavors that oak barrels impart—toasted, smoky, seemingly sweet but no
2198,France,"This four-variety blend shows that crisp, refreshing white wines can emerge from the Southern Rhône. Hints of pineapple, pear and m
2199,Spain,"Mild aromas of cherry and nectarine come across simple and sweet. Following the nose's lead, this is mellow on the palate, with rasp
2200,Germany,"Ripe melon and grapefruit flavors are plush and pristine in this forward off-dry Riesling. Zesty tangerine acidity brightens the m
2201,France,"This is attractive in a bright, fruit-driven style. Bright berry fruit is tinged with apricot and cocoa on the midpalate, then fini
2202,US,"A light but flavorful and dry bubbly, this has a pale pink color and frothy effervescence. Aromas like red cherries and peaches lead to
2203,Spain,"Full, intense blackberry aromas skirt complexity but score points via power and grit. Rubbery tannins and plenty of oak give this we
2204,France,"This medium-bodied blend of Grenache (70%), Syrah (25%) and Mourvèdre (5%) is aged in large oak vats. Leather and spice notes accen
2205,Portugal,"Although it's young, this wine has seen some time in oak, which has given it a ripe fruity character balanced by toast. It's rich
2206,Italy,"This Syrah opens with aromas of ripe dark-skinned fruit, ground black pepper and a whiff of cedar that all carry over to the dense,
2207,Spain,"Light pops of cherry and raspberry aromas emerge from a low-volume, feminine bouquet. This Trepat is typically fresh, with a lean bo
2208,US,"With a vivid golden color, bold and buttery aromas, and rich, almost nutty flavors, this is a full-bore rendering. It is full bodied an
2209,Chile,"A fruity, forward nose of melon, citrus and guava aromas comes with a hint of match stick. Vivacious and lively on the tongue, this
2210,Italy,"Aromas of dark berry, violet, vanilla and a whiff of toast lead the nose on this bright, sleek red. Made from 70% Sangiovese and 30%
2211,US,"Aromatic and drinking at its peak, this remains fresh, tart and tangy, with palate-penetrating boysenberry and blueberry fruit. Think L
2212,Italy,"Aromas of mocha, espresso, coconut, ripe black-skinned fruit and new wood lead the nose on this hearty blend of 40% Cabernet Sauvign
2213,Portugal,"A new venture to the Douro by Tejo-based Lagoalva, this is juicy, fruity and soft. While it misses the minerality of many Douro w
2214,US,"Delicate and dry, this offers lightly spicy strawberry fruit and supporting acids. It's pretty and pale, light but not dilute, and woul
2215,Italy,"Fruity and easygoing, this blend of 60% Sangiovese and 40% Syrah doles out crushed black cherry, blackberry, spiced blueberry, white
2216,Italy,"Made entirely with Sangiovese, this offers scents of blue flowers, ripe berry and mocha. The simple, forward palate delivers fleshy
2217,US,"Young vine flavors of bright raspberry fruit shine through, backed with juicy, tongue-tickling acids. This is not a complex wine, but f
2218,US,"Soft and forward, this has a plush mouthfeel, a mix of banana marshmallows and lemon meringue pie. It's quite tasty and easy-drinking,
2219,Italy,"Sangiovese gets a spicy lift from 5% Alicante on this bright, easy drinking wine. It offers scents of blue flowers, plum, blackberry
2220,US,"This 100% Zin comes from one of Paso Robles' most famous vineyards. The briary blackberry and mulberry flavors are very ripe, veering i
2221,France,"The top wine from Tendil & Lombardi, it's warm, ripe, richly textured and fruity. It has a soft character, creamy and full in the m
2222,US,"From old Lodi vines, V. Sattui's annual heartland Zin is minty and herbal, high-toned and tannic, able to find a nice equilibrium betwe
2223,Portugal,"A full-bodied, ripe and juicy wine that is developing well. It has weight, soft and warm tannins and already delicious fruit. Bla
2224,Italy,"Elegantly structured, it starts with aromas of cut grass, stone fruit and pineapple alongside whiffs of mineral and Alpine herbs. Th
2225,Germany,"Dry, bold and intensely mineral, this powerful Riesling is lifted by whiffs of lemongrass and mint, along with a shining streak of
2226,US,"The 2012 has more stuffing than the previous vintage, offering round, pretty fruit flavors of hard cherry candy. There's a streak of re
2227,US,"Fields Family aims for balance from 60-year-old vines, aiming to produce refined Zins with subtle power. They've achieved that here, of
2228,Portugal,"Grande in all senses, this wine comes in a powerfully heavy bottle. The wine itself is very structured with ripe and dense tannin
2229,France,"The aromas hint at the wood aging of this rich wine, while the palate also brings out the rich and structured fruits. Dense and wit
2230,US,"Pushes all the right Chardonnay buttons, with creamy tropical mango, papaya, pineapple and lime flavors. It also folds in buttered toas
2231,US,"The Attaché brings appealing and lively cherry fruit, framed with citrusy acidity. The power in this wine is all forward and fruit-driv
2232,US,"Classic Santa Lucia Pinot Noir, big, bold and delicious. What it lacks in delicacy it more than makes up for in ripe, jammy fruit. Rasp
2233,Italy,"It opens with a delicate floral fragrance of white spring flowers accented by a whiff of apple and Alpine herbs. The steely, linear
2234,Italy,"It opens with intense aromas of tropical fruit, green apples and a note of Mediterranean herbs. The juicy palate delivers white peac
2235,US,"Bone dry and tart, this is made without oak, allowing the grape's natural acidity and flavors to star. There's a minerality to the lemo
2236,Italy,"This blend of 50% Merlot, 40% Cabernet Sauvignon and 10% Cabernet Franc opens with aromas of cherries, berries, blue flower, sun-bak
2237,France,"Big, ripe and juicy, this solidly based wine also has great fruitiness. Black fruits, some bitter chocolate and licorice point to r
2238,France,"It shows great depth of flavor, with delicious fresh berries, black fruits, balanced acidity and fragrance. The tannins are present
2239,France,"From a small parcel in the Lamothe-Vincent vineyard, this rich, structured wine shows a strong relationship with terroir and minera
2240,Portugal,"Petit Verdot seems right at home in the southern Alentejo. This wine is fully ripe, powerful and genuinely fruity. Very juicy and
2241,Portugal,"This wine shows some of the natural tannic structure of the Douro while really bringing out plenty of fruit. It is rich, full of
```

```
df = pd.read_csv('/content/wine_base.csv')
df.head(2)
```

2 способ загрузки таблицы

```
path_download = "https://drive.google.com/uc?id=1I0IrgrlR-Phv1sQZHOfr1ld866yMTahE"

df2 = pd.read_csv(path_download, index_col=0  )
df2
```

Функции вида `pd.read_формат()` и `pd.to_формат()` считывают и записывают данные соответственно.

Считайте данные в формате **csv** (***comma separated value/значения, разделённые запятыми***) функцией `pd.read_csv()`.

Список самых важных аргументов:

- **filepath_or_buffer** - текстовая строка с названием (адресом) файла;
- **sep** - символ, которым отделены элементы датафрейма в файле (по умолчанию `","`);
- **header** - номер строки, в которой в файле указаны названия столбцов (`None`, если нет);
- **names** - список с названиями столбцов;
- **index_col** - столбец, из которого надо взять названия индексов
- **usecols** - список столбцов для чтения из файла

```
df = pd.read_csv('wine_base.csv', index_col=0)
df
```

Переименование названия столбцов (parameter = names)

```
new_names = ['страна', 'описание', 'наименование', 'баллы', 'цена', 'провинция', 'регион_1', 'регион_2', 'сорт', 'вин

df_rus = pd.read_csv('wine_base.csv', names=new_names, index_col=0, header=1)
df_rus.head()
```

Выбираем столбцы для загрузки (parameter = usecols)

```
data3 = pd.read_csv('wine_base.csv', usecols=['country', 'points', 'price'])
data3.head()
```

```
data3
```

Задать кол-во столбцов и строк при выводе

```
MaxDF = pd.read_csv('/content/sample_data/mnist_test.csv')
MaxDF
```

```
MaxDF[:100]
```

> Расширяем таблицу

```
pd.set_option("display.max_columns", 300)
pd.set_option("display.max_rows", 500)
```

```
MaxDF = pd.read_csv('/content/sample_data/mnist_test.csv')
MaxDF
```

```
MaxDF[:500]
```

loc / iloc

```
data3.head()
```

```
data3.loc[1, 'price']
```

```
110.0
```

```
data3.iloc[1, 2]
```

```
110.0
```

```
data3.loc[1:3, ['points', 'price']]
```

```
data3.iloc[1:3, 1:]
```

Удаление

В Pandas есть несколько способов удалить данные из DataFrame:

Удалить строки:

- data.drop(index) - удалить строки с указанными индексами
- data.drop(labels, axis=0) - удалить строки по значениям в выбранном столбце

Удалить столбцы:

- data.drop(columns) - удалить столбцы с указанными именами
- data.drop(labels, axis=1) - удалить столбцы по значениям в выбранной строке

Удалить duplicates:

- data.drop_duplicates() - удалить дубликаты строк

Удалить NaN:

- data.dropna() - удалить строки/столбцы с NaN значениями

```
df_rus.head(2)
```

```
# Удалим 2 столбца
df_rus.drop(columns=['описание', 'наименование'], inplace=True)
```

```
df_rus.head()
```

```
# Удалим 2 строку
df_rus.drop(2, inplace=True)

df_rus.head()
```

```
df_rus.tail()
```

```
df_rus[-3:].index
```

```
Int64Index([150927, 150928, 150929], dtype='int64')
```

```
df_rus = df_rus.drop(df_rus[-3:].index)
df_rus.tail()
```

Вставить данные

Добавить строку:

- `df.loc[index] = [values]` - вставить строку по указанному индексу
- `df.append(other_df)` - конкатенировать два DataFrame

Добавить столбец:

- `df[column_name] = [values]` - вставить столбец по имени
- `df.insert(loc, column, values)` - вставить столбец по индексу

Обновить ячейки:

- `df.loc[row, column]` = value - присвоить значение по строке/столбцу
- `df.fillna(values)` - заполнить пропущенные ячейки

Добавление столбцов и строк

```
# Создание списка с названием столбцов
columns = ['A', 'H', 'T', 'P']

# Создание списка со значениями ячеек
values = [[1, 2, 3, 4],
          [5, 6, 7, 8],
          [9, 10, 11, 12],
          [13, 14, 15, 16]]

# Создание таблицы из подготовленных данных и вывод на экран
df = pd.DataFrame(values, columns=columns)
df
```

Добавляем столбец

```
df['B'] = [1,1,1,1]
df
```

```
df.loc[:, 'B'] = [2, 2, 2, 2]
df
```

Добавляем строку к таблице выше

```
df.loc[4] = [0,0,0,0,0]
df
```

Обновить значения в таблице

```
df.iloc[0, 1] = 200
df
```

```
df.loc[1, 'P'] = 99
df
```

Удалить столбцы

```
df = df.drop(columns=['B'])
df
```

Создание копии

```
data_backup = df.copy()
```

Конкатенация 2 таблиц

```
# Создание списка с названием столбцов
columns = ['t', 'G', 'C']

# Создание списка со значениями ячеек
values = [[10, 20, 30],
          [50, 60, 70],
          [90, 100, 110],
          [130, 140, 150],
          [170, 180, 190]]

# Создание таблицы из подготовленных данных и вывод на экран
df_2 = pd.DataFrame(values, columns=columns)
display(df_2)
df_2.shape
```

```
NEW_table = pd.concat([df, df_2], axis=1)
NEW_table
```

Сортировка

- **sort_values(axis=0)** сортирует по строкам (индексам оси 0**) >> default=0**
- **sort_values(axis=1)** сортирует по столбцам (индексам оси 1)

По столбцу

```
df = df.sort_values('H', ascending=True) # ascending=False -> По убыванию
df
```

Сбросить индекс

```
df.index = [0,1,2,3,4]
df
```

По строке

```
df = df.sort_values(by=4, axis=1, ascending=True) # by=4, axis=1 это строки, ascending=True по возрастанию
df
```

```
df = df.sort_values(by='P', axis=0, ascending=True) # by=4, axis=1 это столбцы, ascending=True по возрастанию
df
```

```
df = df.reset_index(drop = True)
df
```

Перестановка по названию столбцов

```
NEW_table = NEW_table[['C','H','A','t', 'G', 'P', 'T']]
NEW_table
```

Сохранить

> Сохранить локально

```
df.to_csv('new_salen_train.csv', index=False)
# df = pd.read_csv('new_salen_train.csv')
```

> Сохранить на Google диск

```
df.to_csv('/content/drive/My Drive/Базы/hh_parsed.csv', sep=';', encoding='cp1251')
```

Анализ данных

```
new_names = ['страна', 'описание', 'наименование', 'баллы', 'цена', 'провинция', 'регион_1', 'регион_2', 'сорт', 'вин

df_rus = pd.read_csv('wine_base.csv', names=new_names, index_col=0, header=0)
df_rus.head()
```

Метод . `shape` возвращает размер датафрейма: количество строк и количество столбцов

```
df_rus.shape
```

```
(150930, 10)
```

Метод .size возвращает количество элементов в таблице:

```
df_rus.size
```

```
1509300
```

Иногда данные в некоторых ячейках таблицы пропущены. Это случается по разным причинам: возможно, данных по определенному элементу просто не было, а возможно, человек, который заполнял таблицу, случайно пропустил ячейку.

Метод `.count()` возвращает количество *непустых* записей в каждом столбце:

> Доля пропусков

```
print(df_rus.shape, end='\n\n')
display(df_rus.count())
print()
display(df_rus.isna().sum())
```

```
pd.DataFrame({
    'count': df_rus.count(),
    'isna': df_rus.isna().sum(),
    'percent': round((df_rus.isna().sum() / df_rus.shape[0]) * 100, 3)
})
```

```
df_rus.head()
```

```
df_rus.isna().head()
```

График data.count

Data: df_rus.head(100)

Показать код

Используя метод .head(), вы можете применить метод .count() только к первым ста строкам таблицы:

```
df_rus.head(100).count() # 0 - 100
```

```
страна          100
описание        100
наименование     83
баллы           100
цена             96
провинция       100
регион_1         92
регион_2         43
сорт            100
винодельня      100
dtype: int64
```

Заполнение пустых значений

В отдельном взятом стобце

```
df_rus.head(1)
```

```
df_rus['цена'].fillna('Бесплатно !', inplace=True)
```

```
df_rus['наименование'].fillna('Портвейн', inplace=True)
```

```
df_rus.count()
```

```
страна          150925
описание        150930
наименование    150930
баллы           150930
цена            150930
провинция       150925
регион_1        125870
регион_2         60953
сорт            150930
винодельня      150930
dtype: int64
```

Во всей таблице

```
df_rus = df_rus.fillna('Ошибочка')
```

```
df_rus.count()
```

```
страна          150930
описание        150930
наименование    150930
баллы           150930
цена            150930
провинция       150930
регион_1        150930
регион_2        150930
сорт            150930
винодельня      150930
dtype: int64
```

Индексация

```
df_rus[444:600].count() # 444 - 600
```

```
страна          156
описание        156
наименование    156
баллы           156
цена            156
провинция       156
регион_1        156
регион_2        156
сорт            156
винодельня      156
dtype: int64
```

```
df_rus[80:100:5]
```

```
df_rus[["баллы"]]
```

```
df_rus[["баллы", "цена"]][:10:2]
```

Изменить тип данных

```
maska = df_rus['цена'] > 1000
df_rus[maska]
```

```
df_rus.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150930 entries, 0 to 150929
Data columns (total 10 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   страна        150930 non-null  object
 1   описание      150930 non-null  object
 2   наименование  150930 non-null  object
 3   баллы         150930 non-null  int64 
 4   цена          150930 non-null  int64 
 5   провинция     150930 non-null  object
 6   регион_1      150930 non-null  object
 7   регион_2      150930 non-null  object
 8   сорт          150930 non-null  object
 9   винодельня    150930 non-null  object
dtypes: int64(2), object(8)
memory usage: 12.7+ MB
```

Замена значений

```
df_rus['цена'] = df_rus['цена'].replace('Бесплатно !', 0)
```

```
df_rus["цена"] = df_rus["цена"].astype("int64")
df_rus.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150929 entries, 1 to 150929
Data columns (total 10 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   страна        150924 non-null  object
 1   описание      150929 non-null  object
 2   наименование  150929 non-null  object
 3   баллы         150929 non-null  int64 
 4   цена          150929 non-null  int64 
 5   провинция     150924 non-null  object
 6   регион_1      125869 non-null  object
 7   регион_2      60952 non-null   object
 8   сорт          150929 non-null  object
 9   винодельня    150929 non-null  object
dtypes: int64(2), object(8)
memory usage: 12.7+ MB
```

```
df_rus.loc[maska, 'наименование']
```

```
10651    Ried Loibenberg Smaragd
13318        Roger Rose Vineyard
26296             Clos du Mesnil
34920                   Портвейн
34922                   Портвейн
34927                   Портвейн
34939                   Портвейн
34942                   Портвейн
51886             Clos du Mesnil
83536             Clos du Mesnil
Name: наименование, dtype: object
```

```
maska = df_rus['цена'] > 1000
df_rus[maska]
```

.loc // .iloc

```
df_rus.loc[:10, ['наименование', 'цена']]
```

```
df_rus.iloc[:10, 2:4]
```

```
df_rus.loc[maska]
df_rus.loc[df_rus.цена > 1000]
```

*Посмотрим на скорость выполнения кода с помощью функций **Pandas** и циклов **Python**...* 🕓

⁉️ Зачем `.loc`, если есть `df[]` ⁉️

Делаем 3 000 000 строк

Показать код

```
(3000000, 9)
```

```
# определение функции для преобразования housing_median_age
def transform_housing_median_age(data):
    # создание пустого списка для хранения новых значений
    new_values = []
    # проход по каждому значению в столбце housing_median_age
    for value in data.housing_median_age:
        # проверка условий и присвоение нового значения в соответствии с ними
        if value <= 15:
            new_value = 0
        elif 15 < value <= 30:
            new_value = 1
        else:
            new_value = 2
        # добавление нового значения в список
        new_values.append(new_value)
    # замена столбца housing_median_age на список новых значений
    data.housing_median_age = new_values
    # возврат преобразованного датафрейма
    return data
```

`%%time`, предоставляет данные о времени, затраченном на выполнение кода в этой ячейке. Эти данные включают в себя несколько значимых параметров:

1. `CPU times`: Этот блок показывает общее время, затраченное процессором на выполнение кода. В вашем примере это 1.57 секунды пользовательского времени и 102 миллисекунд системного времени. Пользовательское время отражает время, которое процессор затратил на выполнение команд, созданных вашим кодом, в то время как системное время представляет собой время, затраченное процессором на выполнение системных операций, связанных с вашим кодом.

2. `total` : Этот параметр представляет собой общее время, затраченное на выполнение кода. Он объединяет пользовательское и системное время.
3. `Wall time` : Этот параметр отображает реальное время, которое потребовалось для выполнения кода от начала до конца. Это может быть больше, чем сумма пользовательского и системного времени, так как оно учитывает внешние факторы, такие как задержки ввода-вывода или загрузка процессора другими задачами.

```
%%time
df = transform_housing_median_age(df)
```

```
CPU times: user 1.57 s, sys: 102 ms, total: 1.67 s
Wall time: 1.68 s
```

```
%%time
df[df.housing_median_age <= 15] = 0
df[(df.housing_median_age > 15)&(df.housing_median_age <= 30)] = 1
df[df.housing_median_age > 30] = 2
```

```
CPU times: user 141 ms, sys: 9.32 ms, total: 150 ms
Wall time: 152 ms
```

```
%%time
df.loc[df.housing_median_age <= 15, 'housing_median_age'] = 0
df.loc[(df.housing_median_age > 15)&(df.housing_median_age <= 30), 'housing_median_age'] = 1
df.loc[df.housing_median_age > 30, 'housing_median_age'] = 2
```

```
CPU times: user 42.9 ms, sys: 0 ns, total: 42.9 ms
Wall time: 38.6 ms
```

Способ	Время
Python	1680
.loc	38.6
Pandas	152

```
df.loc[df.housing_median_age <= 15, 'housing_median_age'] = 0
```

График

Показать код

```
df_rus.loc[maska]
df_rus.loc[df_rus.цена > 1000, ['наименование', 'баллы', 'цена']]
```

```
df_rus.loc[df_rus.наименование=='Портвейн', 'баллы'] = 100
df_rus.loc[df_rus.цена > 1000, ['наименование', 'баллы', 'цена']]
```

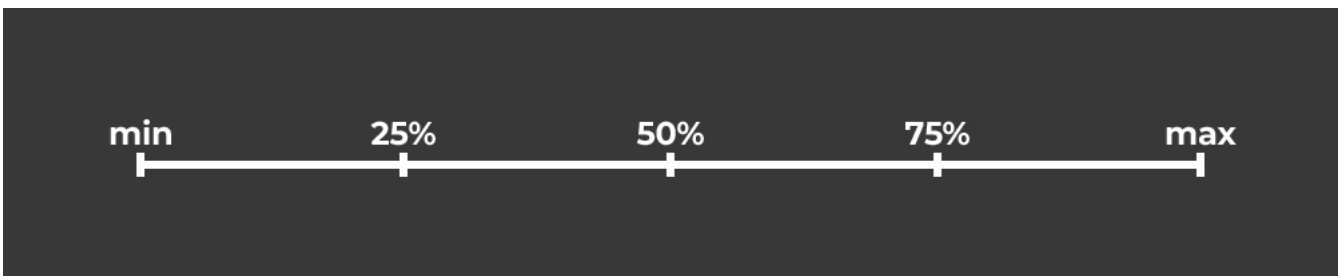
```
df_rus.loc[((df_rus.баллы == 100) & (df_rus.страна =='France')), ['баллы','страна', 'цена']]
```

Описательные статистики

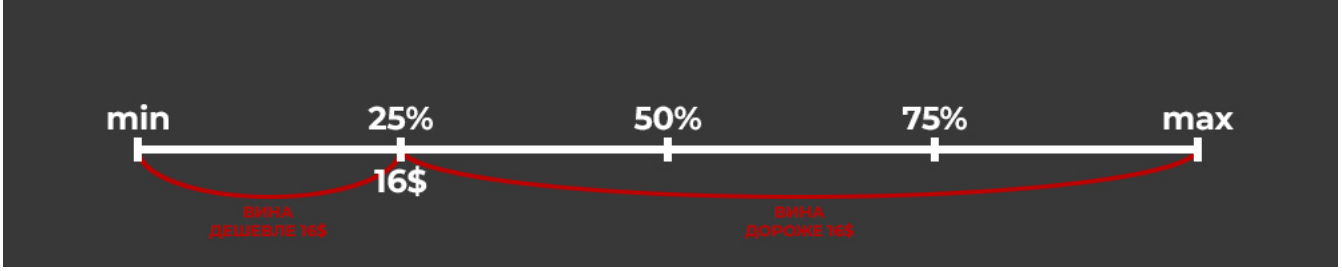
Теперь изучите методы для более подробного изучения данных датафрейма:

Метод `.describe()` возвращает таблицу с описательными статистиками числовых столбцов. Рассмотрите каждый пункт::

- **count** - количество заполненных элементов
- **mean** - среднее значение по столбцу
- **std** - среднее квадратичное отклонение
- **min** - минимальное значение
- **процентили** - тут будет легче рассмотреть на примере. Представим шкалу на которой расположены все цены вин из таблицы. Обозначим на этой шкале 25-ю процентиль, 50-ю процентиль и 75-ю процентиль:



На этой шкале можно расположить все числовые данные, например цены. 25-ой процентилью будет 16$. Это значит, что 25% вин дешевле, а остальные 75% дороже.



75-ая проццентиль будет равна 40$. Соответственно, 75% вин имеют стоимость ниже 40$, а остальные 25% - выше.

```
df_rus.shape
```

```
(150929, 10)
```

```
df_rus.iloc[0]
```

```
страна                                                    Spain
описание        Ripe aromas of fig, blackberry and cassis are ...
наименование                 Carodorum Selección Especial Reserva
баллы                                                        96
цена                                                        110
провинция                                       Northern Spain
регион_1                                                  Toro
регион_2                                              Ошибочка
сорт                                           Tinta de Toro
```