



2.4 Индексы в Pandas

MultilIndex

```
import pandas as pd
import numpy as np

index = [
    ('city_1', 2010, 1),
    ('city_1', 2010, 2),

    ('city_1', 2020, 1),
    ('city_1', 2020, 2),

    ('city_2', 2010, 1),
    ('city_2', 2010, 2),

    ('city_2', 2020, 1),
    ('city_2', 2020, 2),

    ('city_3', 2010, 1),
    ('city_3', 2010, 2),

    ('city_3', 2020, 1),
    ('city_3', 2020, 2),
]

population = [
    101, 1010, 201, 2010, 102, 1020, 202, 2020, 103, 1030, 203, 2030,
]
```

```

pop = pd.Series(population, index=index)
print(pop)
# ⇒ (city_1, 2010, 1)    101
#   (city_1, 2010, 2)   1010
#   (city_1, 2020, 1)    201
#   (city_1, 2020, 2)   2010
#   (city_2, 2010, 1)    102
#   (city_2, 2010, 2)   1020
#   (city_2, 2020, 2)    202
#   (city_2, 2020, 2)   2020
#   (city_3, 2010, 1)    103
#   (city_3, 2010, 2)   1030
#   (city_3, 2020, 2)    203
#   (city_3, 2020, 2)   2030
# dtype: int64
index = pd.MultiIndex.from_tuples(index)

```

```

pop = pop.reindex(index)
print(pop)
# ⇒ city_1 2010 1    101
#           2    1010
#           2020 1    201
#           2    2010
# city_2 2010 1    102
#           2    1020
#           2020 2    202
#           2    2020
# city_3 2010 1    103
#           2    1030
#           2020 2    203
#           2    2030
# dtype: int64

```

```

print(pop[:, 2010])
# ⇒ city_1 1    101
#           2    1010

```

```

# city_2 1 102
#      2 1020
# city_3 1 103
#      2 1030
# dtype: int64

print(pop[:, :, 2])
# ⇒ city_1 2010 1010
#      2020 2010
# city_2 2010 1020
#      2020 2020
# city_3 2010 1030
#      2020 2030

pop_df = pop.unstack()

print(pop_df)
# ⇒      1  2
# city_1 2010 101 1010
#      2020 201 2010
# city_2 2010 102 1020
#      2020 202 2020
# city_3 2010 103 1030
#      2020 203 2030

print(pop_df.stack())
# ⇒ city_1 2010 1 101
#      2 1010
#      2020 1 201
#      2 2010
# city_2 2010 1 102
#      2 1020
#      2020 1 202
#      2 2020
# city_3 2010 1 103
#      2 1030

```

```
#      2020 1   203
#           2  2030
#  dtype: int64
```

Детализация

```
import pandas as pd

index = [
    ('city_1', 2010, 1),
    ('city_1', 2010, 2),

    ('city_1', 2020, 1),
    ('city_1', 2020, 2),

    ('city_2', 2010, 1),
    ('city_2', 2010, 2),

    ('city_2', 2020, 1),
    ('city_2', 2020, 2),

    ('city_3', 2010, 1),
    ('city_3', 2010, 2),

    ('city_3', 2020, 1),
    ('city_3', 2020, 2),
]

population = [
    101, 1010, 201, 2010, 102, 1020, 202, 2020, 103, 1030, 203, 2030,
]

pop = pd.Series(population, index=index)
print(pop)
# => (city_1, 2010, 1)    101
```

```
# (city_1, 2010, 2) 1010
# (city_1, 2020, 1) 201
# (city_1, 2020, 2) 2010
# (city_2, 2010, 1) 102
# (city_2, 2010, 2) 1020
# (city_2, 2020, 2) 202
# (city_2, 2020, 2) 2020
# (city_3, 2010, 1) 103
# (city_3, 2010, 2) 1030
# (city_3, 2020, 2) 203
# (city_3, 2020, 2) 2030
# dtype: int64
```

```
index = pd.MultiIndex.from_tuples(index)
```

```
pop_df = pd.DataFrame(
    {
        'total': pop,
        'something': list(range(10, 22)),
    }
)
```

```
print(pop_df)
```

```
# ⇒           total something
# (city_1, 2010, 1) 101      10
# (city_1, 2010, 2) 1010     11
# (city_1, 2020, 1) 201      12
# (city_1, 2020, 2) 2010     13
# (city_2, 2010, 1) 102      14
# (city_2, 2010, 2) 1020     15
# (city_2, 2020, 2) 202      16
# (city_2, 2020, 2) 2020     17
# (city_3, 2010, 1) 103      18
# (city_3, 2010, 2) 1030     19
# (city_3, 2020, 2) 203      20
```

```
# (city_3, 2020, 2) 2030      21

print(pop_df['something'])
# ⇒ (city_1, 2010, 1) 10
# (city_1, 2010, 2) 11
# (city_1, 2020, 1) 12
# (city_1, 2020, 2) 13
# (city_2, 2010, 1) 14
# (city_2, 2010, 2) 15
# (city_2, 2020, 2) 16
# (city_2, 2020, 2) 17
# (city_3, 2010, 1) 18
# (city_3, 2010, 2) 19
# (city_3, 2020, 2) 20
# (city_3, 2020, 2) 21
# Name: something, dtype: int64
```

Способы доступа к индексам

```
import pandas as pd

index = [
    ('city_1', 1),
    ('city_1', 2),

    ('city_1', 1),
    ('city_1', 2),

    ('city_2', 1),
    ('city_2', 2),

    ('city_2', 1),
    ('city_2', 2),

    ('city_3', 1),
```

```

('city_3', 2),

('city_3', 1),
('city_3', 2),
]

population = [
    101, 1010, 201, 2010, 102, 1020, 202, 2020, 103, 1030, 203, 2030,
]

index = pd.MultiIndex.from_tuples(index, names=['city', 'num'])
pop = pd.Series(population, index=index)

print(pop)
# ⇒ (city_1, 1)    101
#   (city_1, 2)   1010
#   (city_1, 1)    201
#   (city_1, 2)   2010
#   (city_2, 1)    102
#   (city_2, 2)   1020
#   (city_2, 2)    202
#   (city_2, 2)   2020
#   (city_3, 1)    103
#   (city_3, 2)   1030
#   (city_3, 2)    203
#   (city_3, 2)   2030
# dtype: int64

pop_df = pd.DataFrame(
    {
        'total': pop,
        'something': list(range(10, 22)),
    }
)

pop_df_1 = pop_df.xs('city_1', level='city')['something']

```

```
print(pop_df_1)
# ⇒ num
#    1   10
#    2   11
#    1   12
#    2   13
# Name: something, dtype: int64
```

Как можно создавать MultiIndex

1. Список массивов

Они задают значение индекса на каждом уровне

```
import pandas as pd

i1 = pd.MultiIndex.from_arrays([
    ['a', 'a', 'b', 'b'],
    [1, 2, 1, 2],
])

print(i1)
# ⇒ MultiIndex([('a', 1),
#              ('a', 2),
#              ('b', 1),
#              ('b', 2)],
#              )
```

2. Список кортежей

Они задают значения индекса в каждой точке

```
import pandas as pd

i2 = pd.MultiIndex.from_tuples([
```



```

    ('a', 1),
    ('a', 2),
    ('b', 1),
    ('b', 2),
])

print(i2)
# ⇒ MultiIndex([('a', 1),
#               ('a', 2),
#               ('b', 1),
#               ('b', 2)],
#               )

```

3. Декартово произведение

Произведение обычных индексов

```

import pandas as pd

i3 = pd.MultiIndex.from_product([
    ['a', 'b'],
    [1, 2],
])

print(i3)
# ⇒ MultiIndex([('a', 1),
#               ('a', 2),
#               ('b', 1),
#               ('b', 2)],
#               )

```

4. Описание внутреннего представления

levels - список списков

codes -

```

import pandas as pd

i4 = pd.MultiIndex(
    levels=[
        ['a', 'b'],
        [1, 2],
    ],
    codes=[
        [0, 0, 1, 1], # a a b b
        [0, 1, 0, 1], # 1 2 1 2
    ]
)
print(i4)
# ⇒ MultiIndex([('a', 1),
#               ('a', 2),
#               ('b', 1),
#               ('b', 2)],
#              )

```

Уровням можно задавать названия

```

import pandas as pd

data = {
    ('city_1', 2010): 100,
    ('city_1', 2020): 200,
    ('city_2', 2010): 1001,
    ('city_2', 2020): 2001,
}

s = pd.Series(data)
print(s)
# ⇒ city_1 2010    100
#         2020    200
# city_2 2010   1001

```

```
#      2020  2001
# dtype: int64

s.index.names = ['city', 'year']
print(s)
# ⇒ city  year
#  sity_1 2010   100
#      2020   200
#  sity_2 2010  1001
#      2020  2001
# dtype: int64
```

Создание DataFrame с MultiIndex для строк и столбцов

```
import pandas as pd
import numpy as np

index = pd.MultiIndex.from_product(
    [
        ['city_1', 'city_2'],
        [2010, 2020],
    ],
    names=['city', 'year'],
)

columns = pd.MultiIndex.from_product(
    [
        ['person_1', 'person_2', 'person_3'],
        ['job_1', 'job_2'],
    ],
    names=['worker', 'job'],
)
```

```

rng = np.random.default_rng(1)
data = rng.random((4, 6))

data_df = pd.DataFrame(data, index=index, columns=columns)

print(data_df)
# # => worker    person_1    person_2    person_3
#   job      job_1  job_2  job_1  job_2  job_1  job_2
#   city  year
#   city_1 2010  0.511822  0.950464  0.144160  0.948649  0.311831  0.423326
#         2020  0.827703  0.409199  0.549594  0.027559  0.753513  0.538143
#   city_2 2010  0.329732  0.788429  0.303195  0.453498  0.134042  0.403113
#         2020  0.203455  0.262313  0.750365  0.280409  0.485191  0.980737

```

Задание для самостоятельной работы

- Из получившихся данных выбрать данные по:
 2020 году (для всех столбцов)
 job_1 (для всех строк)
 city_1 и job_2

Индексация и срезы по MultiIndex

```

import pandas as pd

data = {
    ('city_1', 2010): 100,
    ('city_1', 2020): 200,
    ('city_2', 2010): 1001,
    ('city_2', 2020): 2001,
}

s = pd.Series(data)
print(s)

```

```

# ⇒ sity_1 2010 100
#          2020 200
# sity_2 2010 1001
#          2020 2001
# dtype: int64

s.index.names = ['city', 'year']
print(s)
# ⇒ city  year
# sity_1 2010 100
#          2020 200
# sity_2 2010 1001
#          2020 2001
# dtype: int64

print(s['city_1', 2010])
# ⇒ 100

print(s['city_1'])
# ⇒ year
# 2010 100
# 2020 200

```

loc, iloc

```

import pandas as pd

data = {
    ('city_1', 2010): 100,
    ('city_1', 2020): 200,
    ('city_2', 2010): 1001,
    ('city_2', 2020): 2001,
    ('city_3', 2010): 10001,
    ('city_3', 2020): 20001
}

```

```
s = pd.Series(data)
s.index.names = ['city', 'year']
```

```
print(s.loc['city_1':'city_2'])
# ⇒ city  year
#  city_1 2010   100
#         2020   200
#  city_2 2010  1001
#         2020  2001
```

```
print(s[:, 2010])
# ⇒ city
#  city_1   100
#  city_2  1001
#  city_3  1000
```

```
print(s[s > 2000])
# ⇒ city  year
#  city_2 2020   2001
#  city_3 2010  10001
#         2020  20001
```

```
print(s[['city_1', 'city_3']])
# ⇒ city  year
#  city_1 2010   100
#         2020   200
#  city_3 2010  10001
#         2020  20001
```

Задание для самостоятельной работы

- Взять за основу DataFrame со следующей структурой

```
import pandas as pd
import numpy as np
```

```

index = pd.MultiIndex.from_product(
    [
        ['city_1', 'city_2'],
        [2010, 2020],
    ],
    names=['city', 'year'],
)

columns = pd.MultiIndex.from_product(
    [
        ['person_1', 'person_2', 'person_3'],
        ['job_1', 'job_2'],
    ],
    names=['worker', 'job'],
)

```

Выполнить запрос на получения следующих данных:

- Все данные по person_1 и person_3
- Все данные по первому городу и первым двум person-ам (с использованием срезов)

Приведите пример (самостоятельно) с использованием pd.IndexSlice

Перегруппировка MultiIndex

```

import pandas as pd
import numpy as np

rng = np.random.default_rng(1)

index = pd.MultiIndex.from_product(
    [
        ['a', 'c', 'b'],
        [1, 2],
    ],

```

```

    ]
)

data = pd.Series(rng.random(6), index=index)
data.index.names = ['char', 'int']

print(data)
# ⇒ char int
#   a   1   0.511822
#      2   0.950464
#   c   1   0.144160
#      2   0.948649
#   b   1   0.311831
#      2   0.423326
#   dtype: float64

# Возникает ошибка тк индекс не по порядку а с b
print(data['a':'b'])
# ⇒ pandas.errors.UnsortedIndexError: 'Key length (1)
#   was greater than MultiIndex lexsort depth (0)'

# Необходимо отсортировать индексы
data = data.sort_index()

print(data)
# ⇒ char int
#   a   1   0.511822
#      2   0.950464
#   b   1   0.311831
#      2   0.423326
#   c   1   0.144160
#      2   0.948649
#   dtype: float64

print(data['a':'b'])
# ⇒ char int

```



```
#    a    1    0.511822
#      2    0.950464
#    b    1    0.311831
#      2    0.423326
#    dtype: float64
```

Пример

```
import pandas as pd
import numpy as np

index = [
    ('city_1', 2010, 1),
    ('city_1', 2010, 2),

    ('city_1', 2020, 1),
    ('city_1', 2020, 2),

    ('city_2', 2010, 1),
    ('city_2', 2010, 2),

    ('city_2', 2020, 1),
    ('city_2', 2020, 2),

    ('city_3', 2010, 1),
    ('city_3', 2010, 2),

    ('city_3', 2020, 1),
    ('city_3', 2020, 2),
]

population = [
    101, 1010, 201, 2010, 102, 1020, 202, 2020, 103, 1030, 203, 2030,
]
```

```

i = pd.MultiIndex.from_tuples(index)

pop = pop.reindex(i)
print(pop)
# ⇒ city_1 2010 1 101
#           2 1010
#           2020 1 201
#           2 2010
# city_2 2010 1 102
#           2 1020
#           2020 1 202
#           2 2020
# city_3 2010 1 103
#           2 1030
#           2020 1 203
#           2 2030

# Перевод в DataFrame
print(pop.unstack())
# ⇒           1  2
# city_1 2010 101 1010
#           2020 201 2010
# city_2 2010 102 1020
#           2020 202 2020
# city_3 2010 103 1030
#           2020 203 2030

# Перегруппировка последовательностей
print(pop.unstack(level=0))
# ⇒      city_1 city_2 city_3
# 2010 1 101 102 103
#       2 1010 1020 1030
# 2020 1 201 202 203
#       2 2010 2020 2030

print(pop.unstack(level=1))

```

```
# ⇒      2010 2020
# city_1 1 101 201
#      2 1010 2010
# city_2 1 102 202
#      2 1020 2020
# city_3 1 103 203
#      2 1030 2030

print(pop.unstack(level=2))
# ⇒      1  2
# city_1 2010 101 1010
#      2020 201 2010
# city_2 2010 102 1020
#      2020 202 2020
# city_3 2010 103 1030
#      2020 203 2030
```

Конкатенация

NumPy

```
import numpy as np

x = [1, 2, 3]
y = [4, 5, 6]
z = [7, 8, 9]

print(np.concatenate([x, y, z]))
# ⇒ [1 2 3 4 5 6 7 8 9]

x = [[1, 2, 3]]
y = [[4, 5, 6]]
z = [[7, 8, 9]]

print(np.concatenate([x, y, z]))
```

```
# ⇒ [[1 2 3]
#    [4 5 6]
#    [7 8 9]]

# Выбор оси для конкатенации
print(np.concatenate([x, y, z], axis=1))
# ⇒ [[1 2 3 4 5 6 7 8 9]]

print(np.concatenate([x, y, z], axis=0))
# ⇒ [[1 2 3]
#    [4 5 6]
#    [7 8 9]]
```

Pandas

```
import pandas as pd

ser1 = pd.Series(['a', 'b', 'c'], index=[1, 2, 3])
ser2 = pd.Series(['d', 'e', 'f'], index=[4, 5, 6])

print(pd.concat([ser1, ser2]))
# ⇒ 1    a
#    2    b
#    3    c
#    4    d
#    5    e
#    6    f
#    dtype: object

# Дублирование
ser1 = pd.Series(['a', 'b', 'c'], index=[1, 2, 3])
ser2 = pd.Series(['d', 'e', 'f'], index=[1, 2, 6])

print(pd.concat([ser1, ser2]))
# ⇒ 1    a
```

```

# 2 b
# 3 c
# 1 d
# 2 e
# 6 f
# dtype: object

# Требуем уникальность индексов
print(pd.concat([ser1, ser2], verify_integrity=False))
# ⇒ 1 a
# 2 b
# 3 c
# 1 d
# 2 e
# 6 f
# dtype: object

# Пропуск дубликатов
print(pd.concat([ser1, ser2], ignore_index=True))
# ⇒ 0 a
# 1 b
# 2 c
# 3 d
# 4 e
# 5 f
# dtype: object

# Ключ создания MultiIndex
print(pd.concat([ser1, ser2], keys=['x', 'y']))
# ⇒ x 1 a
# 2 b
# 3 c
# y 1 d
# 2 e
# 6 f
# dtype: object

```

```
ser1 = pd.Series(['a', 'b', 'c'], index=[1, 2, 3])
ser2 = pd.Series(['d', 'e', 'f'], index=[4, 5, 6])
```

```
# join
print(pd.concat([ser1, ser2], join='outer'))
# ⇒ 1  a
#  2  b
#  3  c
#  1  d
#  2  e
#  6  f
#  dtype: object
```

```
print(pd.concat([ser1, ser2], join='inner'))
# ⇒ 1  a
#  2  b
#  3  c
#  1  d
#  2  e
#  6  f
#  dtype: object
```

Задание для самостоятельной работы

- Привести пример использования inner и outer (join) для Series (на данных примера)