Warsztaty modelowania

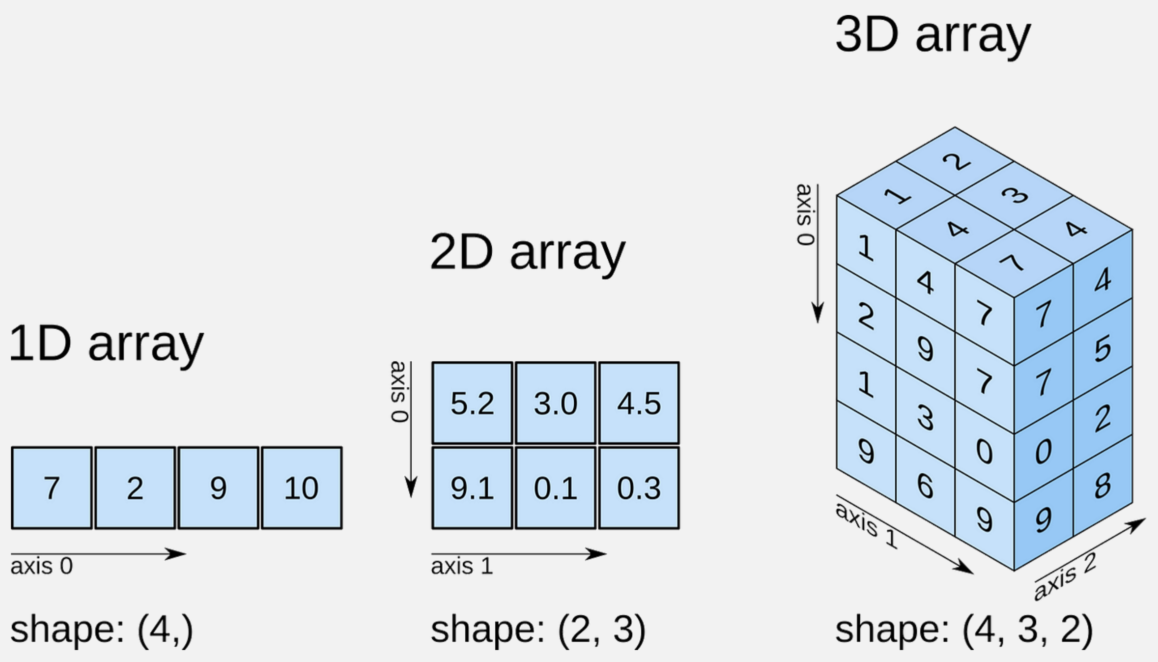
01 – biblioteka pandas

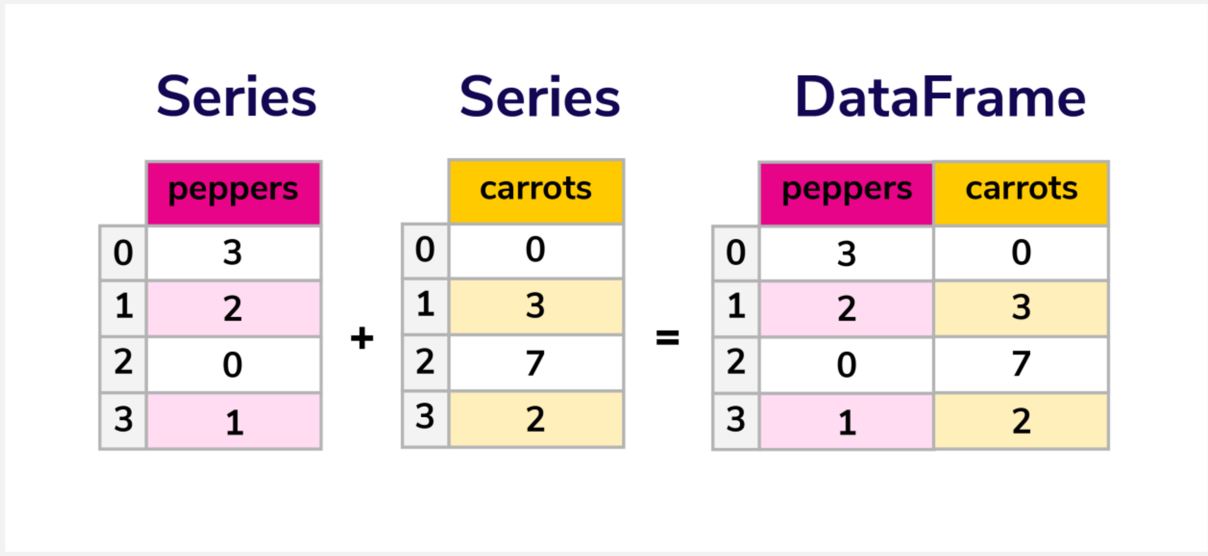
opracowała

Patrycja Naumczyk

O czym będzie?

1. Numpy – czyli bebechy pandas
2. Pandas – struktury danych
   a) Serie vs tabele
   b) Indeksowanie
3. Tworzenie nowych kolumn i przypisywanie wartości
4. Funkcje agregujące
5. Method chaining




Series + Series = DataFrame

	peppers
0	3
1	2
2	0
3	1

+

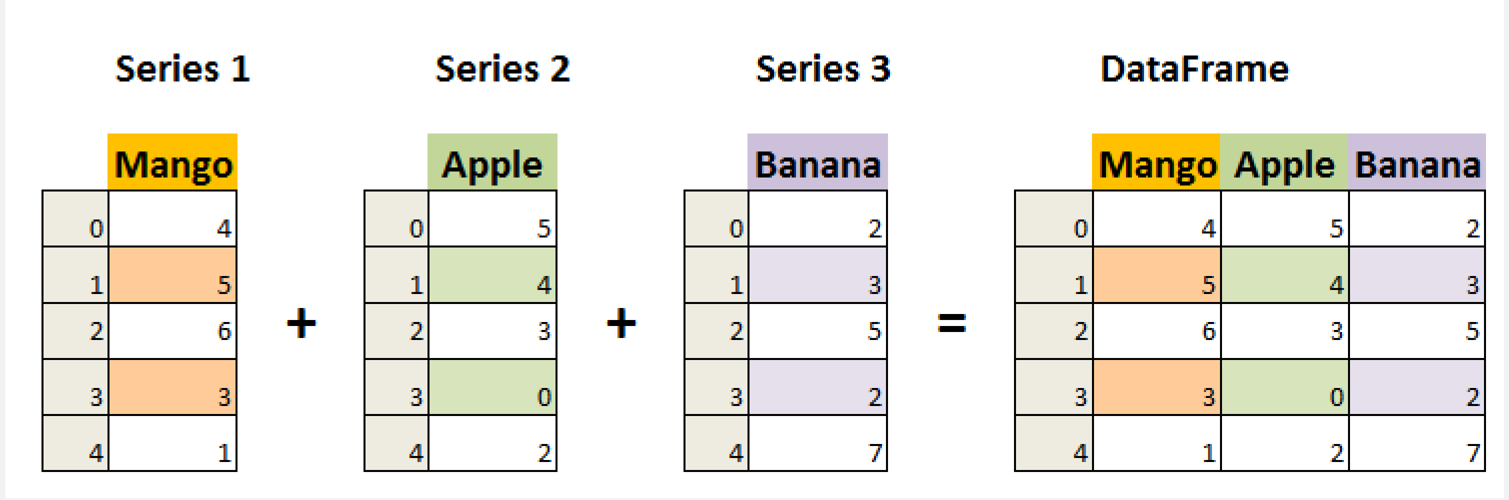
	carrots
0	0
1	3
2	7
3	2

=

	peppers	carrots
0	3	0
1	2	3
2	0	7
3	1	2

Series i dataframe

1. pd.Series()
2. pd.DataFrame()

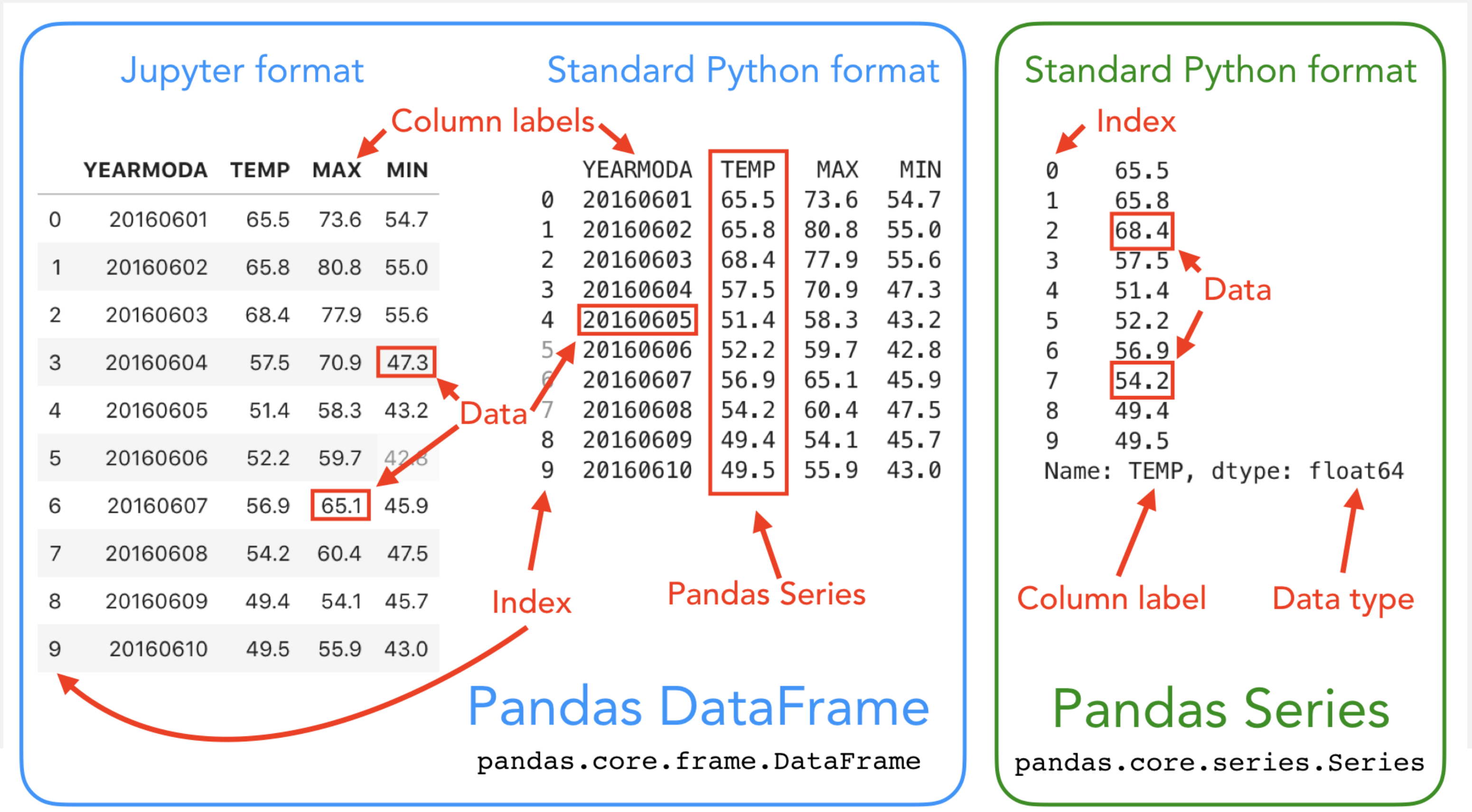


	Mango
0	4
1	5
2	6
3	3
4	1

	Apple
0	5
1	4
2	3
3	0
4	2

	Banana
0	2
1	3
2	5
3	2
4	7

	Mango	Apple	Banana
0	4	5	2
1	5	4	3
2	6	3	5
3	3	0	2
4	1	2	7



	YEARMODA	TEMP	MAX	MIN
0	20160601	65.5	73.6	54.7
1	20160602	65.8	80.8	55.0
2	20160603	68.4	77.9	55.6
3	20160604	57.5	70.9	47.3
4	20160605	51.4	58.3	43.2
5	20160606	52.2	59.7	42.8
6	20160607	56.9	65.1	45.9
7	20160608	54.2	60.4	47.5
8	20160609	49.4	54.1	45.7
9	20160610	49.5	55.9	43.0

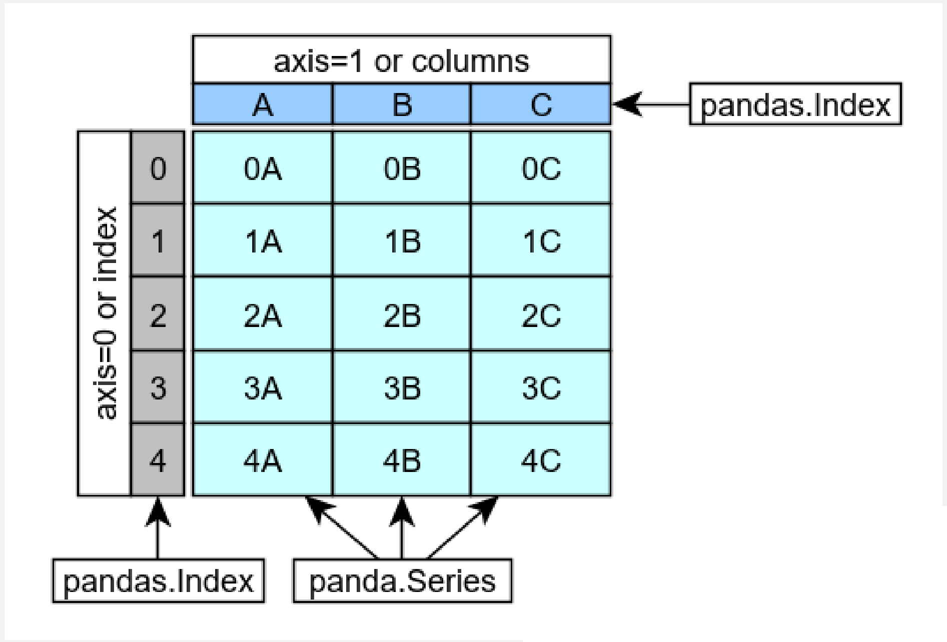
```
   YEARMODA  TEMP   MAX   MIN
0  20160601  65.5  73.6  54.7
1  20160602  65.8  80.8  55.0
2  20160603  68.4  77.9  55.6
3  20160604  57.5  70.9  47.3
4  20160605  51.4  58.3  43.2
5  20160606  52.2  59.7  42.8
6  20160607  56.9  65.1  45.9
7  20160608  54.2  60.4  47.5
8  20160609  49.4  54.1  45.7
9  20160610  49.5  55.9  43.0
```

```
pandas.core.frame.DataFrame
```

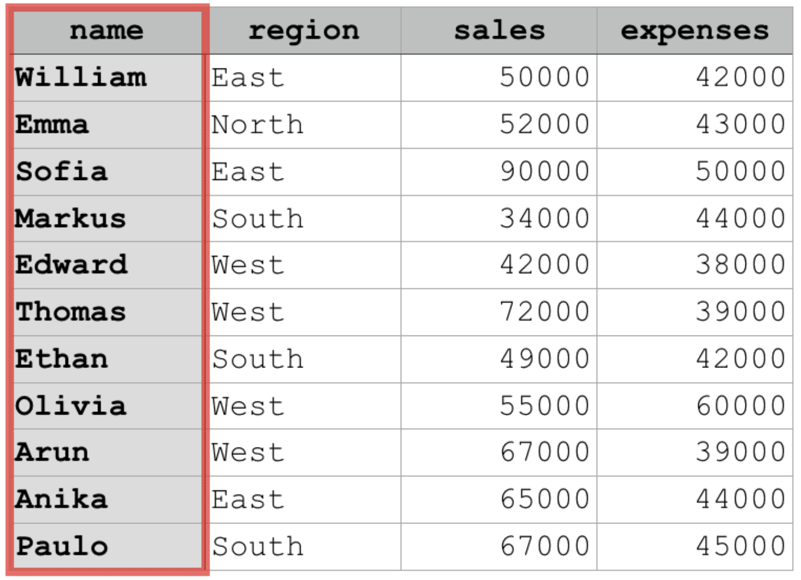
```
0    65.5
1    65.8
2    68.4
3    57.5
4    51.4
5    52.2
6    56.9
7    54.2
8    49.4
9    49.5
Name: TEMP, dtype: float64
```

```
pandas.core.series.Series
```

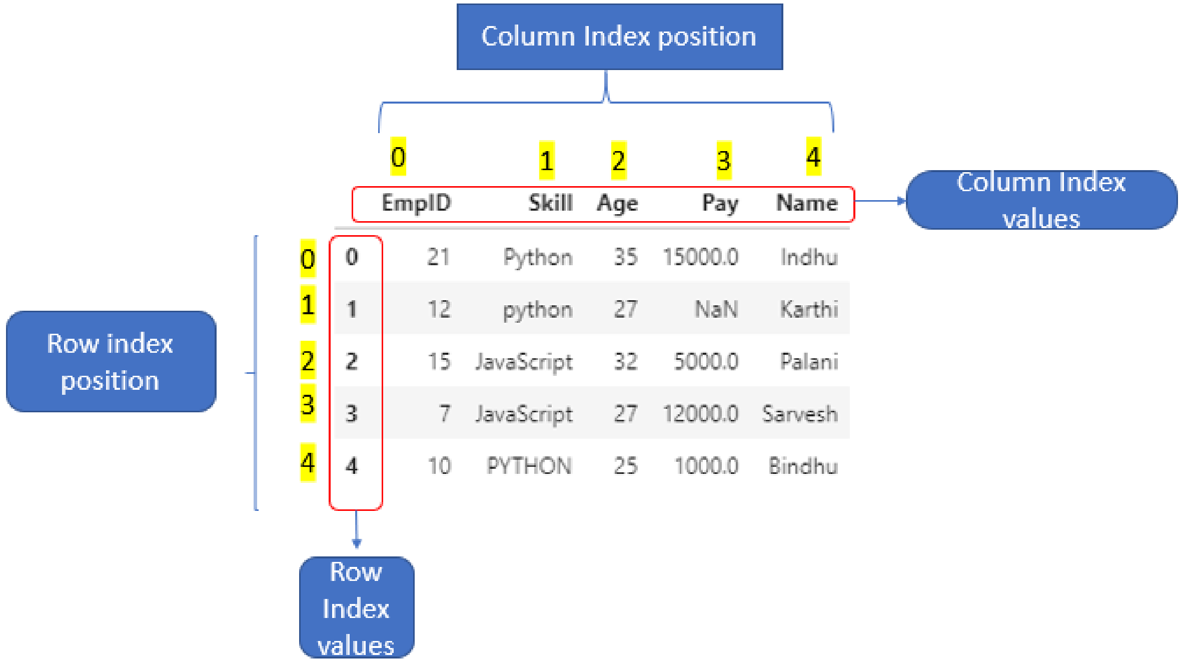
Indeksowanie



An index is like a group of row labels



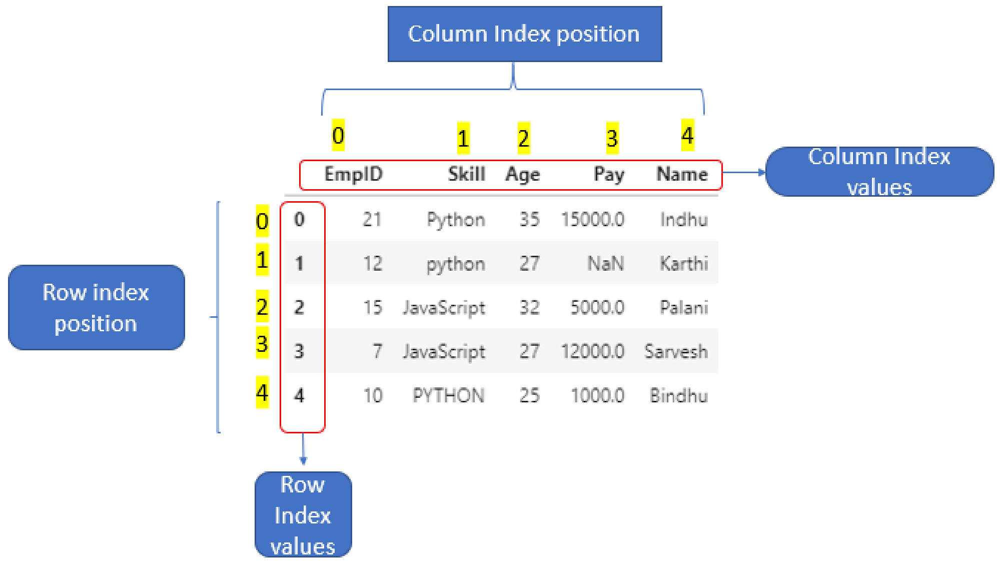
name	region	sales	expenses
William	East	50000	42000
Emma	North	52000	43000
Sofia	East	90000	50000
Markus	South	34000	44000
Edward	West	42000	38000
Thomas	West	72000	39000
Ethan	South	49000	42000
Olivia	West	55000	60000
Arun	West	67000	39000
Anika	East	65000	44000
Paulo	South	67000	45000



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

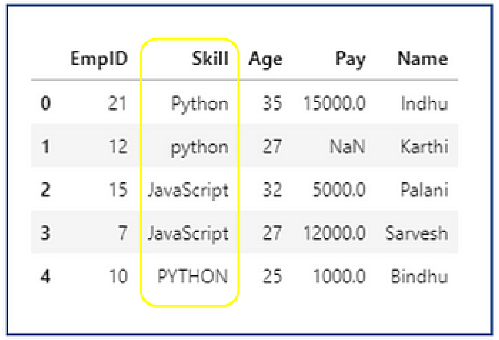
Indeksowanie cd.

1. Ustawianie dowolnego indeksu:
   a) set_index()
2. Czyszczenie indeksu:
   a) reset_index()
3. Odwołanie wprost:
   a) df[ ]



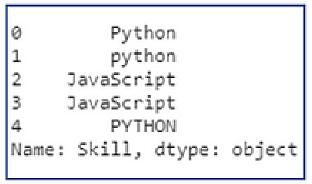
	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df



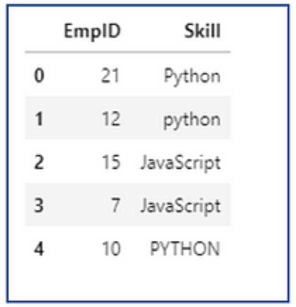
	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df["Skill"]



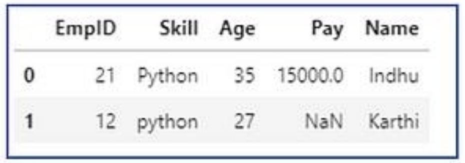
```
0        Python
1        python
2    JavaScript
3    JavaScript
4        PYTHON
Name: Skill, dtype: object
```

df[["EmpID","Skill"]]



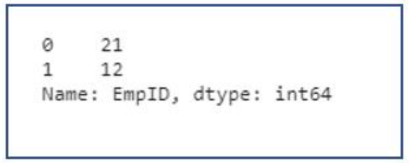
	EmpID	Skill
0	21	Python
1	12	python
2	15	JavaScript
3	7	JavaScript
4	10	PYTHON

df[0:2]



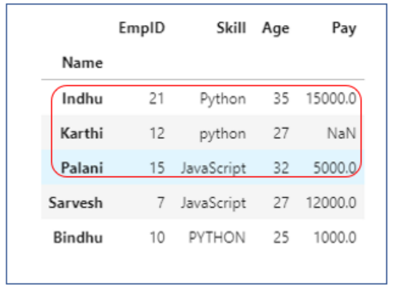
	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi

df[0:2]["EmpID"]



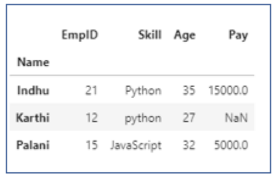
```
0    21
1    12
Name: EmpID, dtype: int64
```

df1



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

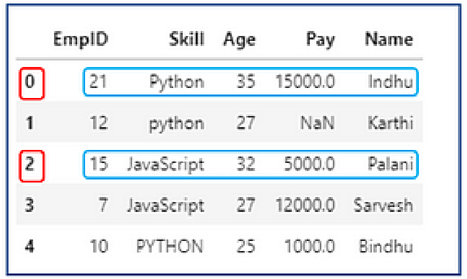
df1["Indhu":"Palani"]



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0

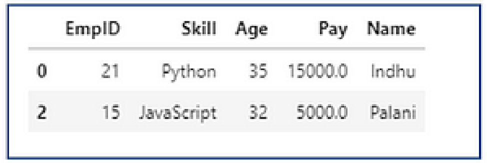
df

Step 2



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df[0:4:2]

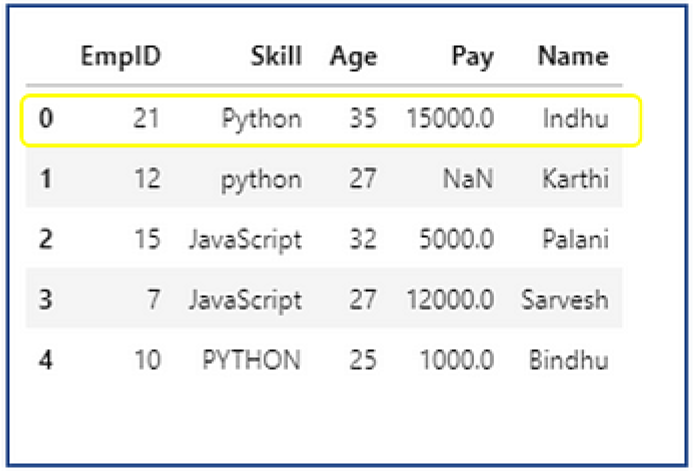


	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
2	15	JavaScript	32	5000.0	Palani

Indeksowanie cd.

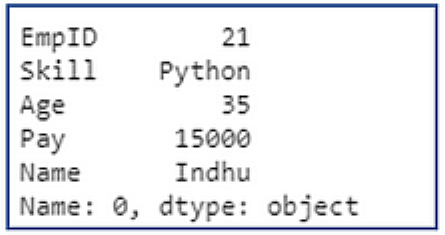
1. Ustawianie dowolnego indeksu:
   a) set_index()
2. Czyszczenie indeksu:
   a) reset_index()
3. Odwołanie wprost:
   a) df[ ]
4. Odwołanie pozycyjne (jak w numpy):
   a) iloc[ ]

df



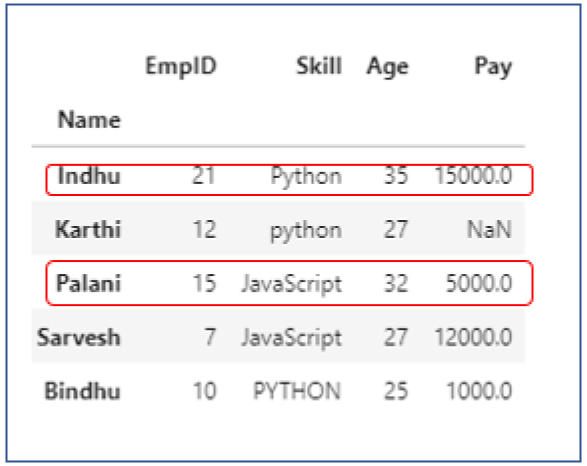
	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df.iloc[0]



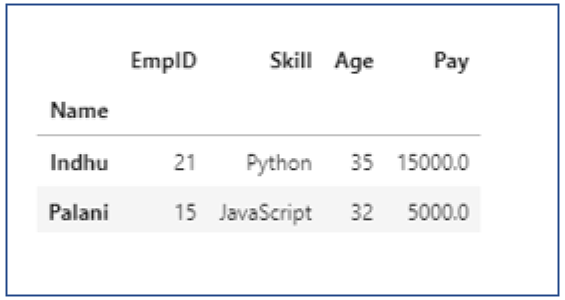
```
EmpID        21
Skill    Python
Age          35
Pay       15000
Name      Indhu
Name: 0, dtype: object
```

df1



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

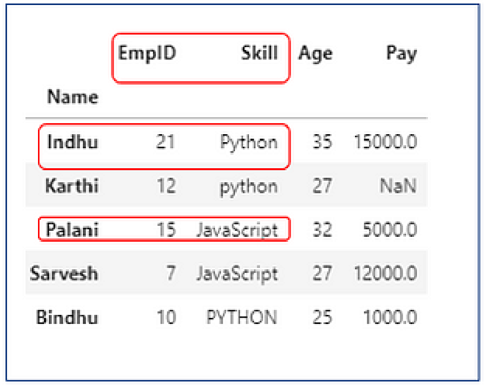
df1.iloc[[0,2]]



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Palani	15	JavaScript	32	5000.0

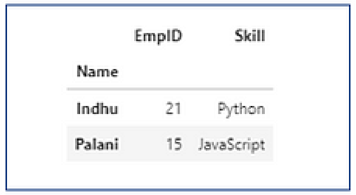
df1

Step 2



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

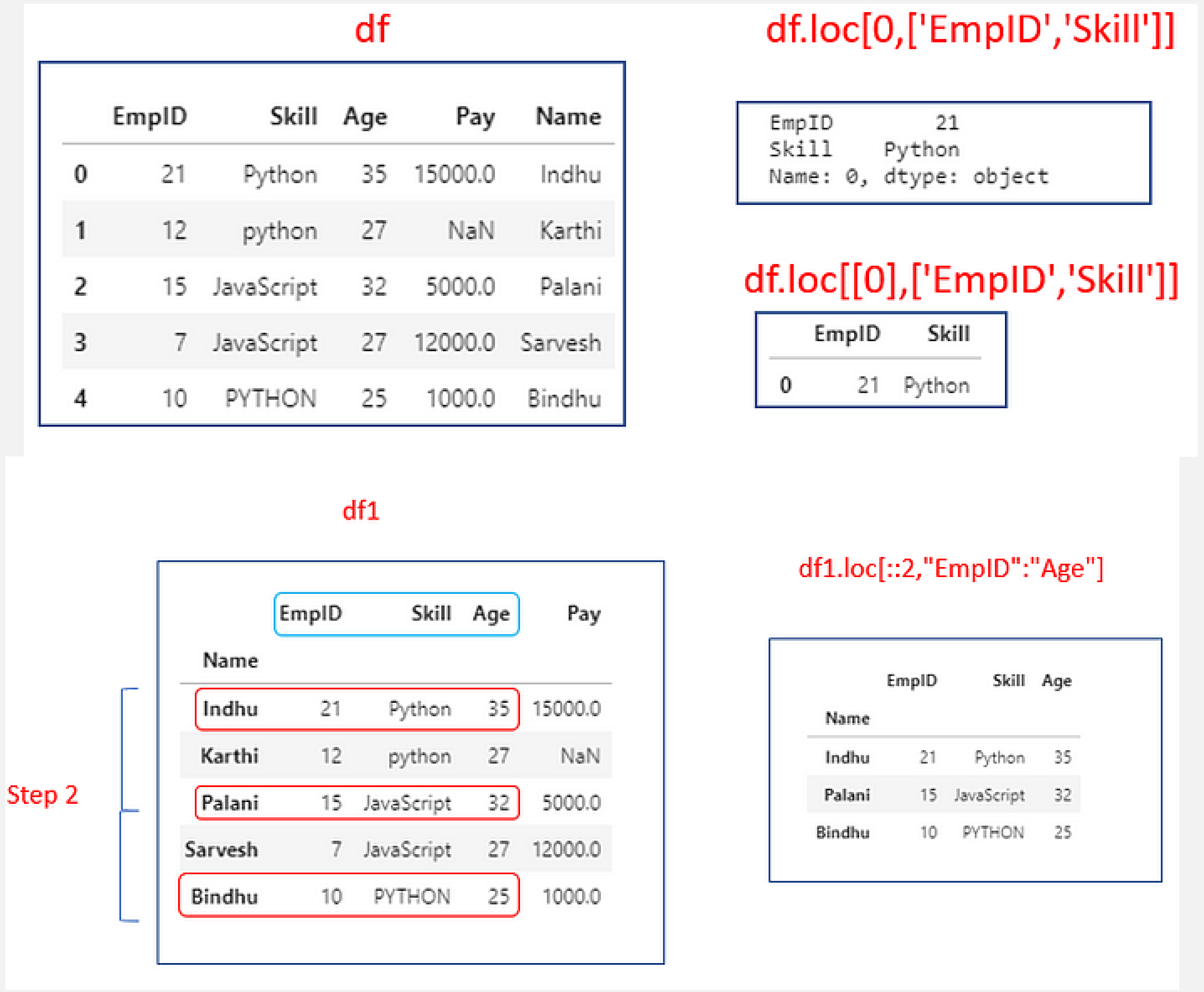
df1.iloc[0:4:2,0:2]



	EmpID	Skill
Name		
Indhu	21	Python
Palani	15	JavaScript

Indeksowanie cd.

1. Ustawianie dowolnego indeksu:
   a) set_index()
2. Czyszczenie indeksu:
   a) reset_index()
3. Odwołanie wprost:
   a) df[ ]
4. Odwołanie pozycyjne (jak w numpy):
   a) iloc[ ]
5. Odwołanie do etykiet:
   a) loc[ ]



df

	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df.loc[0,['EmpID','Skill']]

```
EmpID        21
Skill    Python
Name: 0, dtype: object
```

df.loc[[0],['EmpID','Skill']]

	EmpID	Skill
0	21	Python

df1

	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

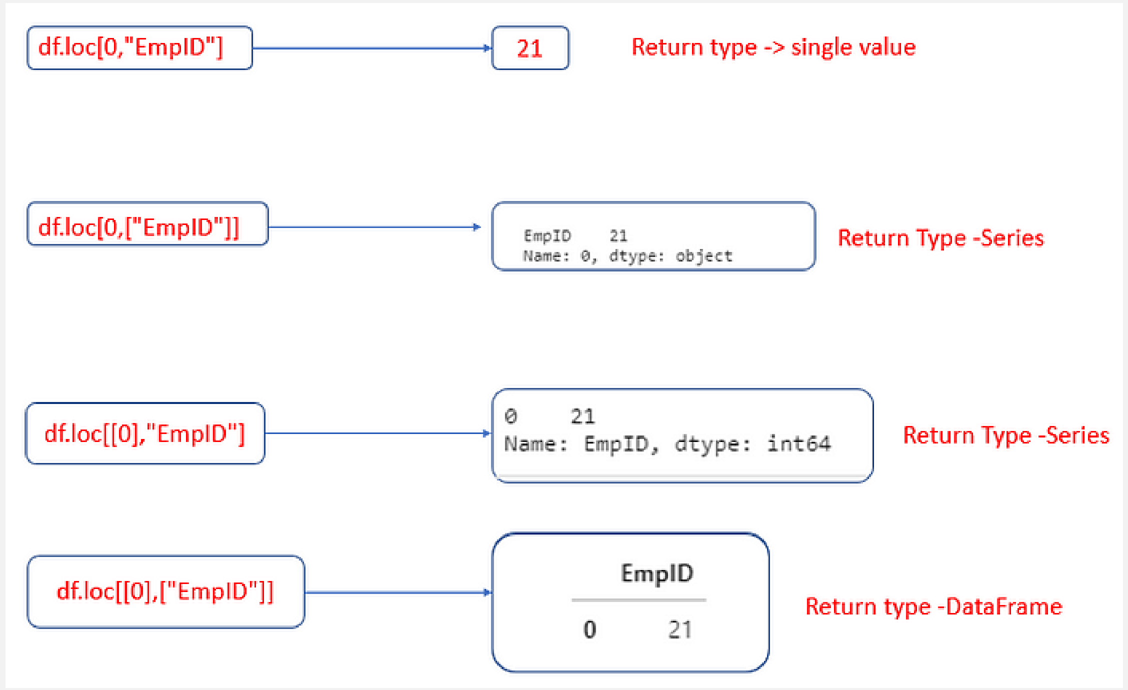
df1.loc[::2,"EmpID":"Age"]

	EmpID	Skill	Age
Name			
Indhu	21	Python	35
Palani	15	JavaScript	32
Bindhu	10	PYTHON	25

Porównanie iloc[ ] i loc[ ]

Input given in iloc	Return Type
1.Both row_index and column_index given as single integer	Single value
2. One input is given as single integer and other input is given as list of integer/integers	Series
3. Both row_index and column_index given as list of integer/integers.	DataFrame

Input given in loc	Return Type
1.Both row_index and column_index given as single label	Single value
2. One input is given as single label and other input is given as list of label/labels	Series
3. Both row_index and column_index given as list of label/labels.	DataFrame



```
df.loc[0,"EmpID"]
```

```
21
```

```
df.loc[0,["EmpID"]]
```

```
EmpID    21
Name: 0, dtype: object
```

```
df.loc[[0],"EmpID"]
```

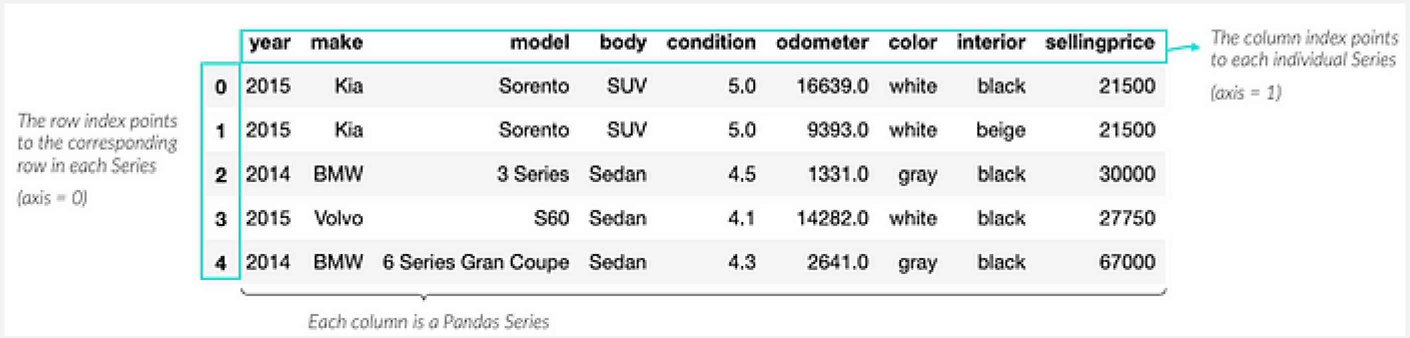
```
0    21
Name: EmpID, dtype: int64
```

```
df.loc[[0],["EmpID"]]
```

	EmpID
0	21

Indeksowanie cd.

1. Ustawianie dowolnego indeksu:
   a) set_index()
2. Czyszczenie indeksu:
   a) reset_index()
3. Odwołanie wprost:
   a) df[ ]
4. Odwołanie pozycyjne (jak w numpy):
   a) iloc[ ]
5. Odwołanie do etykiet:
   a) loc[ ]
6. Maski logiczne

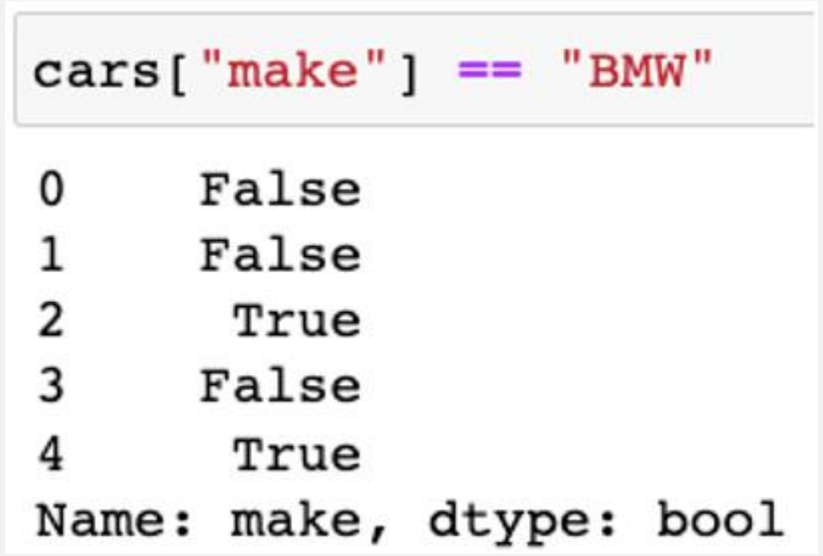


The row index points to the corresponding row in each Series (axis = 0)

The column index points to each individual Series (axis = 1)

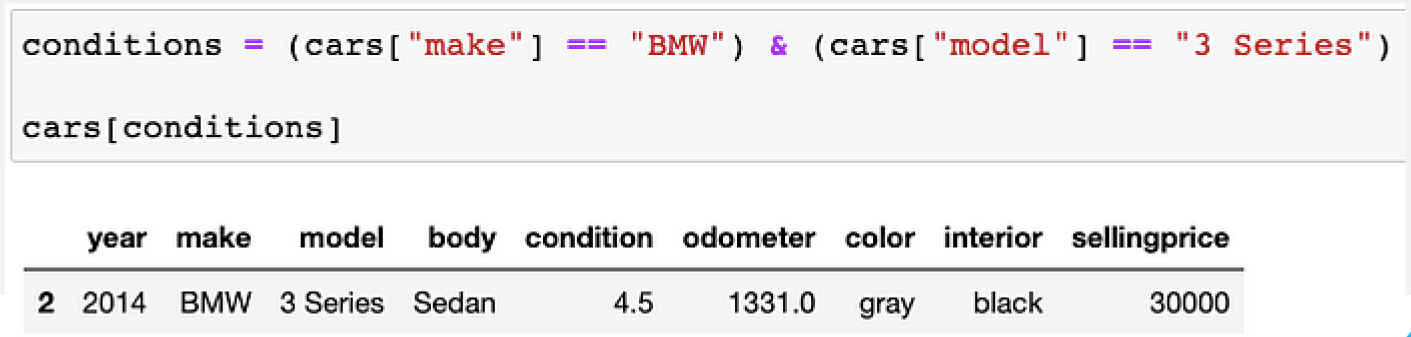
	year	make	model	body	condition	odometer	color	interior	sellingprice
0	2015	Kia	Sorento	SUV	5.0	16639.0	white	black	21500
1	2015	Kia	Sorento	SUV	5.0	9393.0	white	beige	21500
2	2014	BMW	3 Series	Sedan	4.5	1331.0	gray	black	30000
3	2015	Volvo	S60	Sedan	4.1	14282.0	white	black	27750
4	2014	BMW	6 Series Gran Coupe	Sedan	4.3	2641.0	gray	black	67000

Each column is a Pandas Series



```
cars["make"] == "BMW"
```

```
0     False
1     False
2      True
3     False
4      True
Name: make, dtype: bool
```



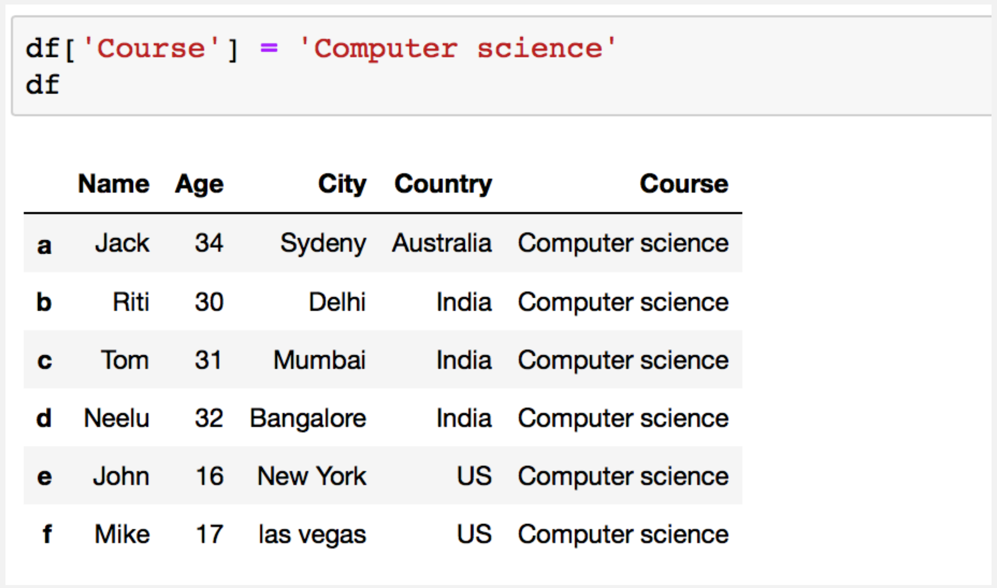
```
conditions = (cars["make"] == "BMW") & (cars["model"] == "3 Series")

cars[conditions]
```

	year	make	model	body	condition	odometer	color	interior	sellingprice
2	2014	BMW	3 Series	Sedan	4.5	1331.0	gray	black	30000

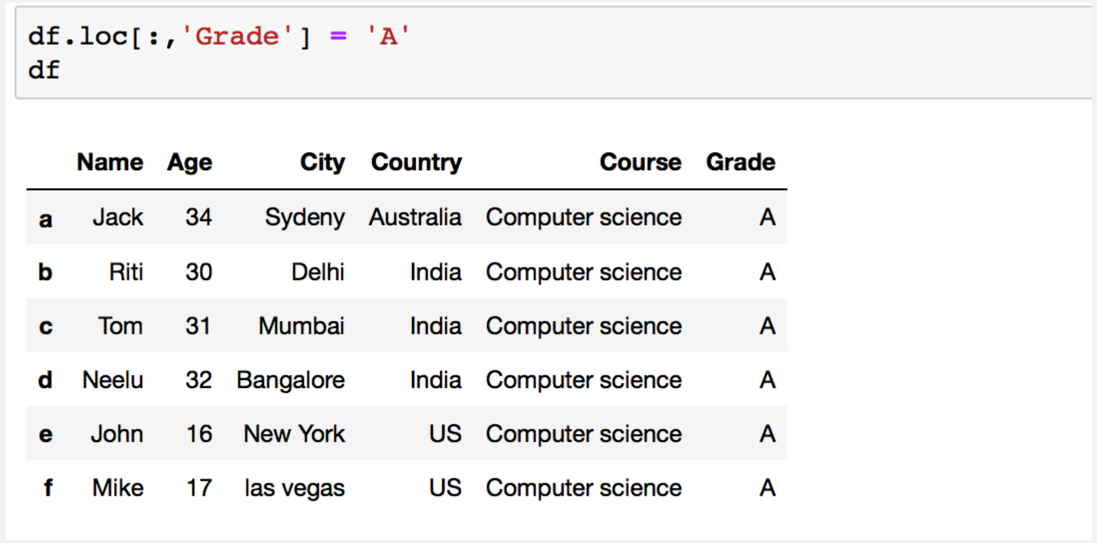
Tworzenie kolumn

1. Przypisanie wartości



```
df['Course'] = 'Computer science'
df
```

	Name	Age	City	Country	Course
a	Jack	34	Sydeny	Australia	Computer science
b	Riti	30	Delhi	India	Computer science
c	Tom	31	Mumbai	India	Computer science
d	Neelu	32	Bangalore	India	Computer science
e	John	16	New York	US	Computer science
f	Mike	17	las vegas	US	Computer science



```
df.loc[:,'Grade'] = 'A'
df
```

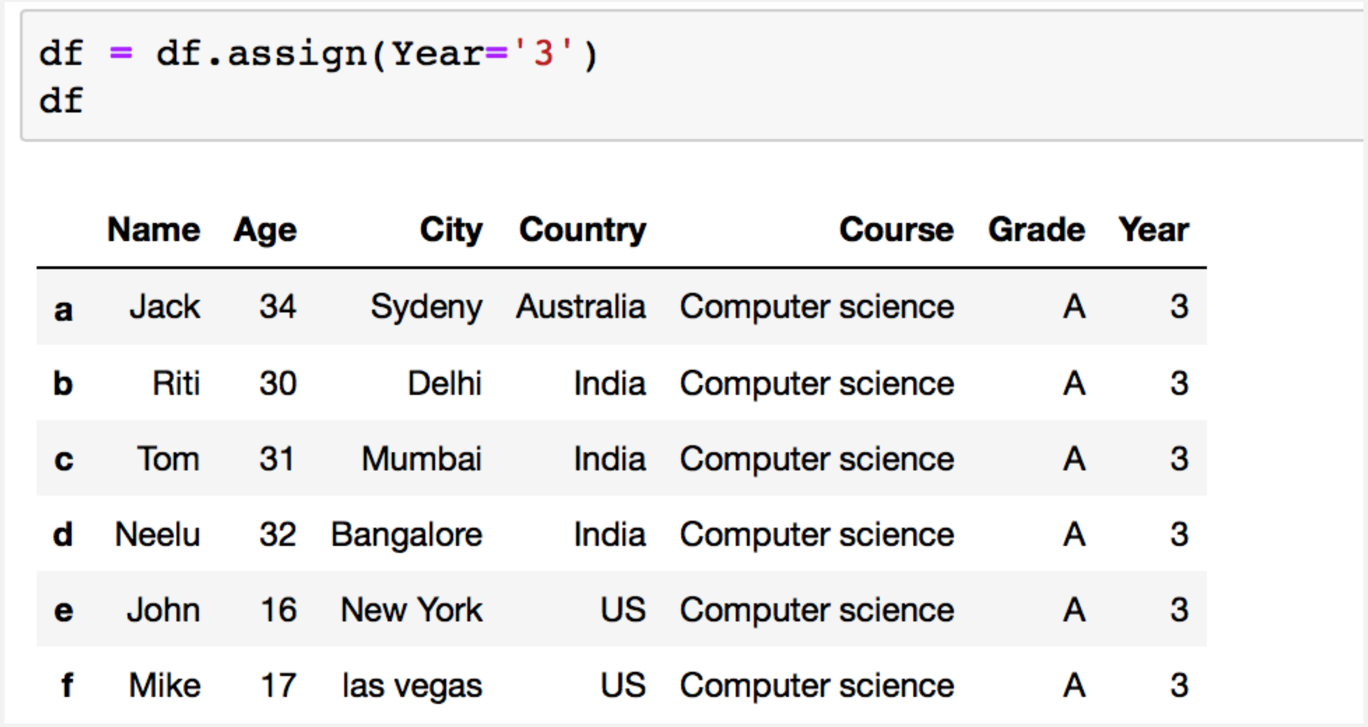
	Name	Age	City	Country	Course	Grade
a	Jack	34	Sydeny	Australia	Computer science	A
b	Riti	30	Delhi	India	Computer science	A
c	Tom	31	Mumbai	India	Computer science	A
d	Neelu	32	Bangalore	India	Computer science	A
e	John	16	New York	US	Computer science	A
f	Mike	17	las vegas	US	Computer science	A

Tworzenie kolumn

1. Przypisanie wartości
2. Operacje arytmetyczne
   a) Operatory matematyczne (+ - * /)
   b) Metody (add() sub() mul() div())

Tworzenie kolumn

1. Przypisanie wartości
2. Operacje arytmetyczne
   a) Operatory matematyczne (+ - * /)
   b) Metody (add() sub() mul() div())
3. Metoda assign()

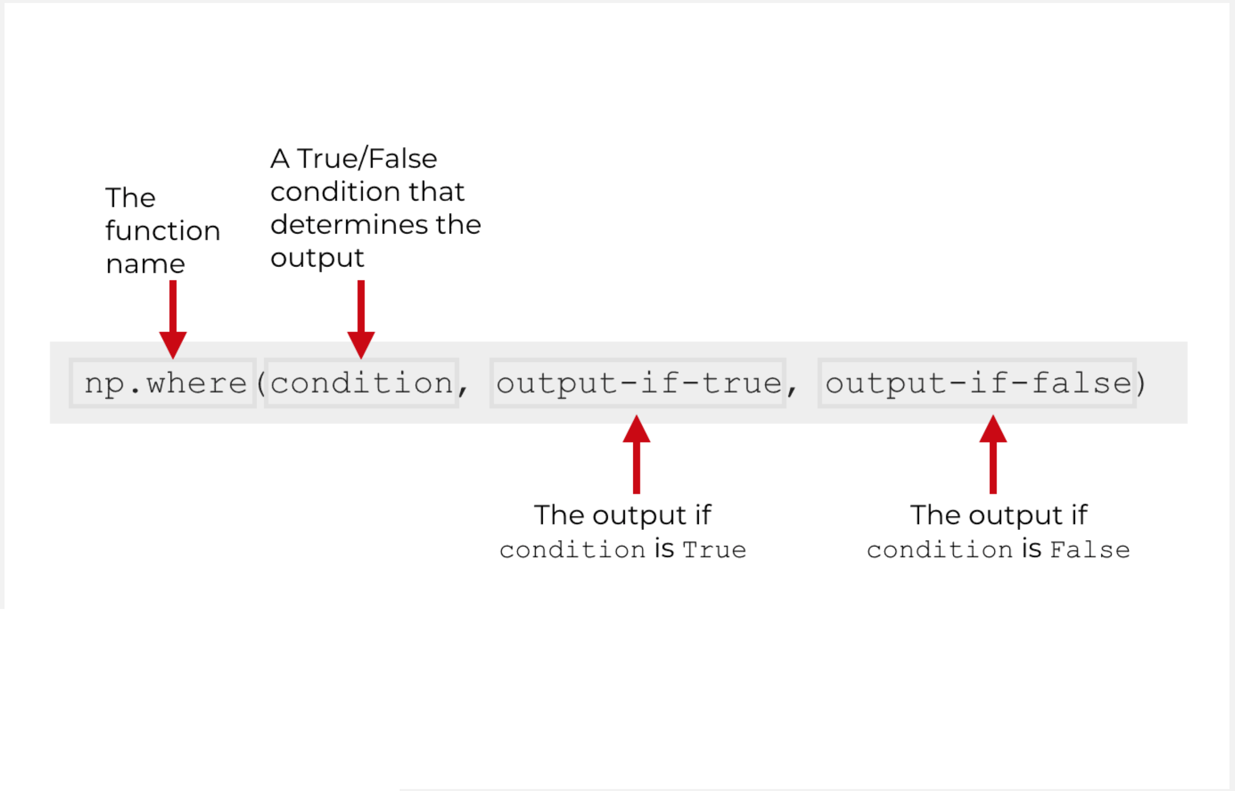


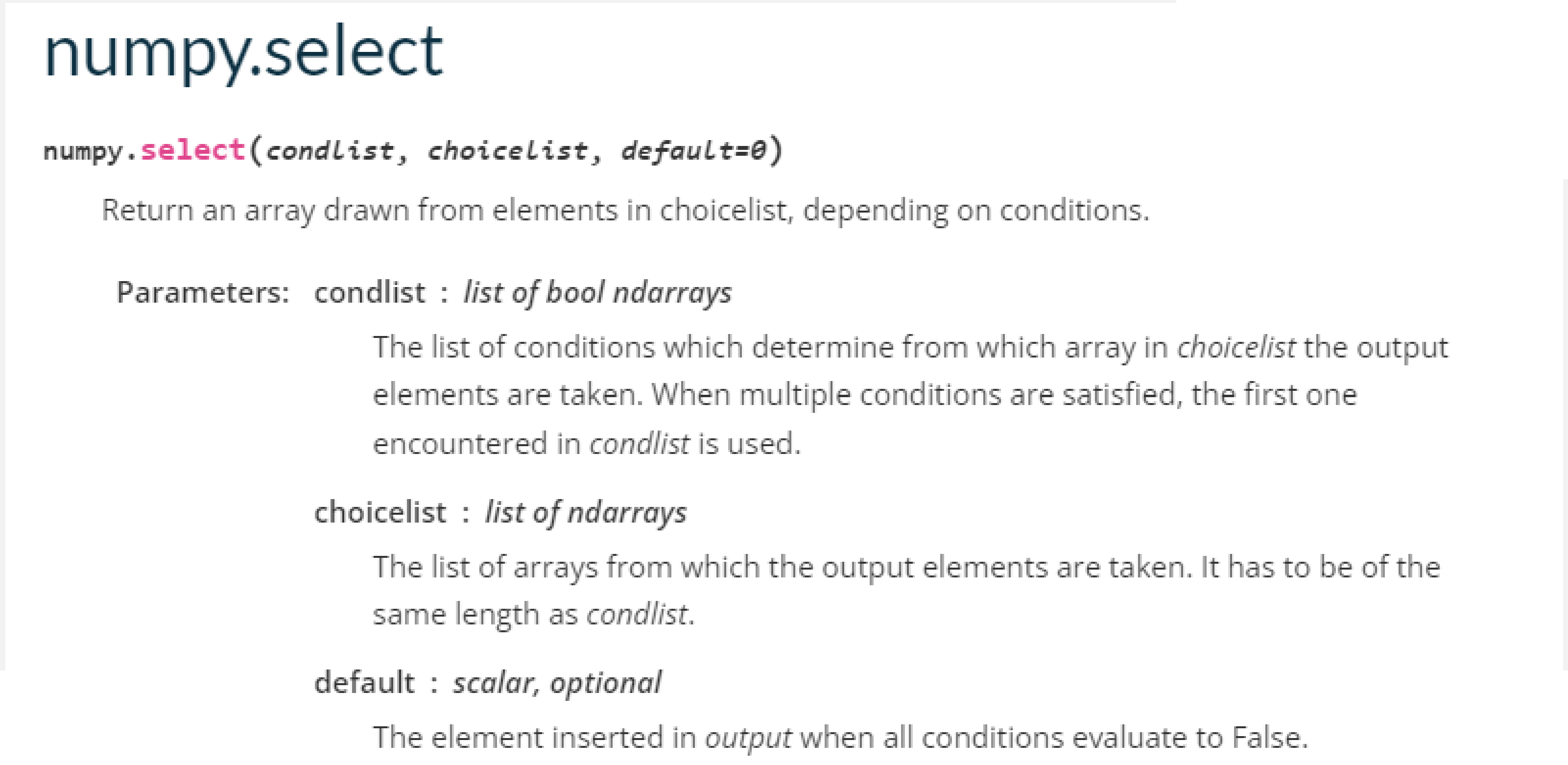
```python
df = df.assign(Year='3')
df
```

	Name	Age	City	Country	Course	Grade	Year
a	Jack	34	Sydeny	Australia	Computer science	A	3
b	Riti	30	Delhi	India	Computer science	A	3
c	Tom	31	Mumbai	India	Computer science	A	3
d	Neelu	32	Bangalore	India	Computer science	A	3
e	John	16	New York	US	Computer science	A	3
f	Mike	17	las vegas	US	Computer science	A	3

Tworzenie kolumn

1. Przypisanie wartości
2. Operacje arytmetyczne
3. Metoda assign()
4. Przypisanie warunkowe:
   a) Funkcja np.where()
   b) Funkcja np.select()


The function name
A True/False condition that determines the output
np.where(condition, output-if-true, output-if-false)
The output if condition is True
The output if condition is False



numpy.select

```
numpy.select(condlist, choicelist, default=0)
```

Return an array drawn from elements in choicelist, depending on conditions.

Parameters: condlist : *list of bool ndarrays*

The list of conditions which determine from which array in *choicelist* the output elements are taken. When multiple conditions are satisfied, the first one encountered in *condlist* is used.

choicelist : *list of ndarrays*

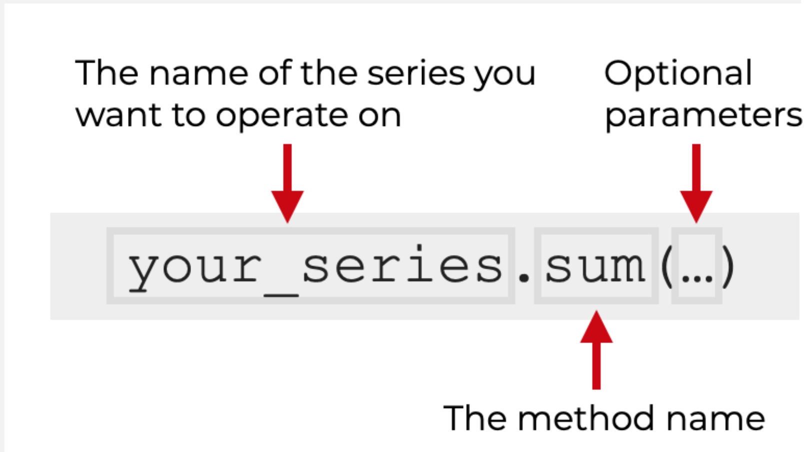
The list of arrays from which the output elements are taken. It has to be of the same length as *condlist*.

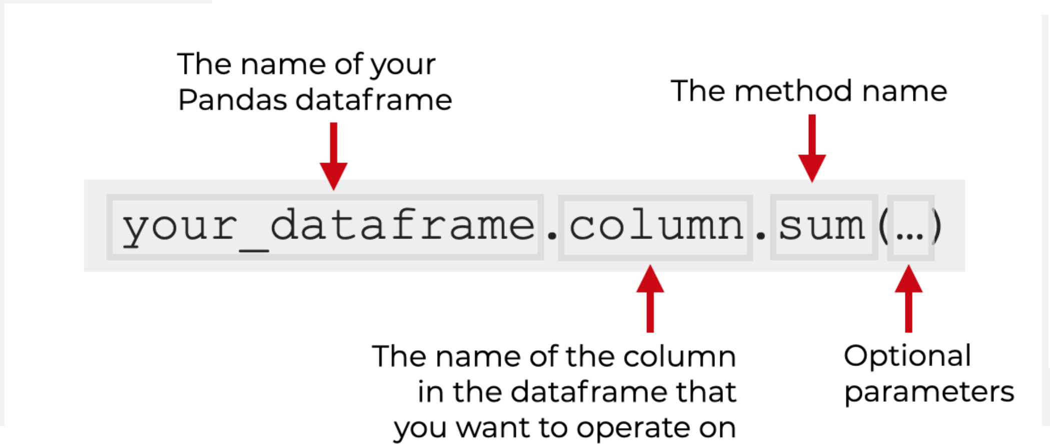
default : *scalar, optional*

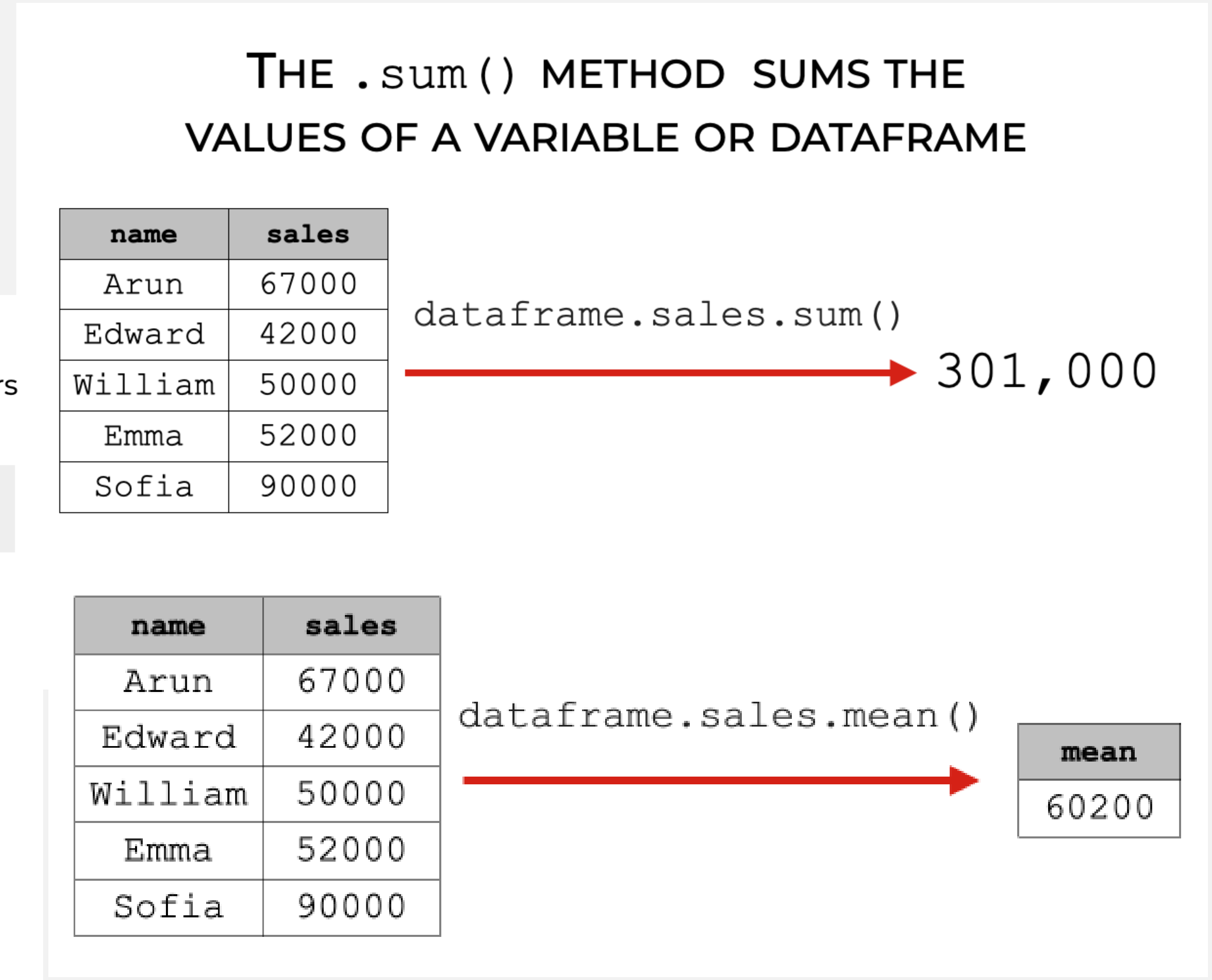
The element inserted in *output* when all conditions evaluate to False.

Funkcje agregujące

1. Operacje matematyczne i statystyczne




The name of your Pandas dataframe
The method name
your_dataframe.column.sum(…)
The name of the column in the dataframe that you want to operate on
Optional parameters


THE .sum() METHOD SUMS THE VALUES OF A VARIABLE OR DATAFRAME

name	sales
Arun	67000
Edward	42000
William	50000
Emma	52000
Sofia	90000

```
dataframe.sales.sum()
```

301,000

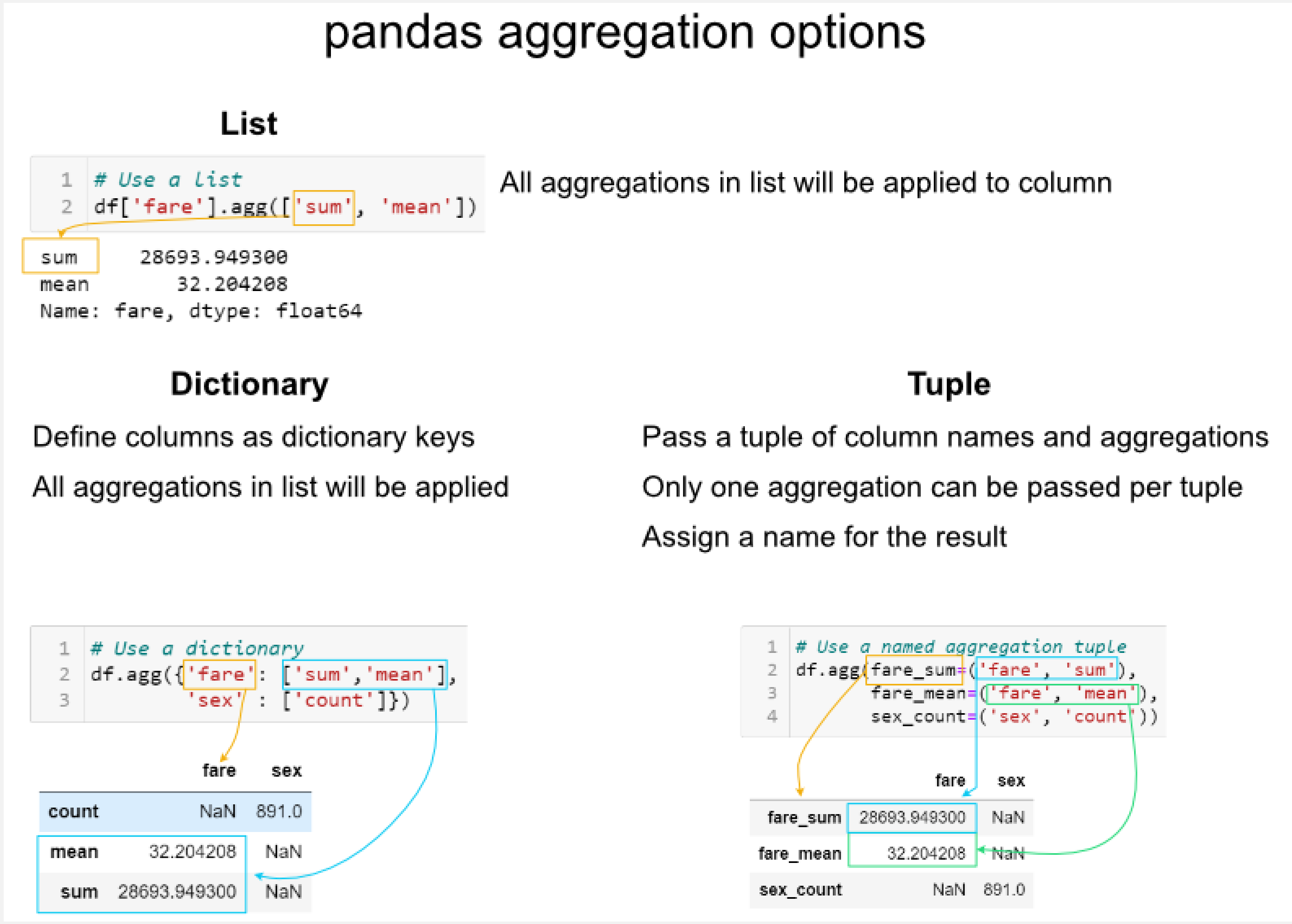
name	sales
Arun	67000
Edward	42000
William	50000
Emma	52000
Sofia	90000

```
dataframe.sales.mean()
```

mean
60200

Funkcje agregujące

1. Operacje matematyczne i statystyczne
2. Metoda agg()



pandas aggregation options

List

```
# Use a list
df['fare'].agg(['sum', 'mean'])
```

All aggregations in list will be applied to column

```
sum     28693.949300
mean       32.204208
Name: fare, dtype: float64
```

Dictionary

Define columns as dictionary keys

All aggregations in list will be applied

```
# Use a dictionary
df.agg({'fare': ['sum','mean'],
        'sex' : ['count']})
```

	fare	sex
count	NaN	891.0
mean	32.204208	NaN
sum	28693.949300	NaN

Tuple

Pass a tuple of column names and aggregations

Only one aggregation can be passed per tuple

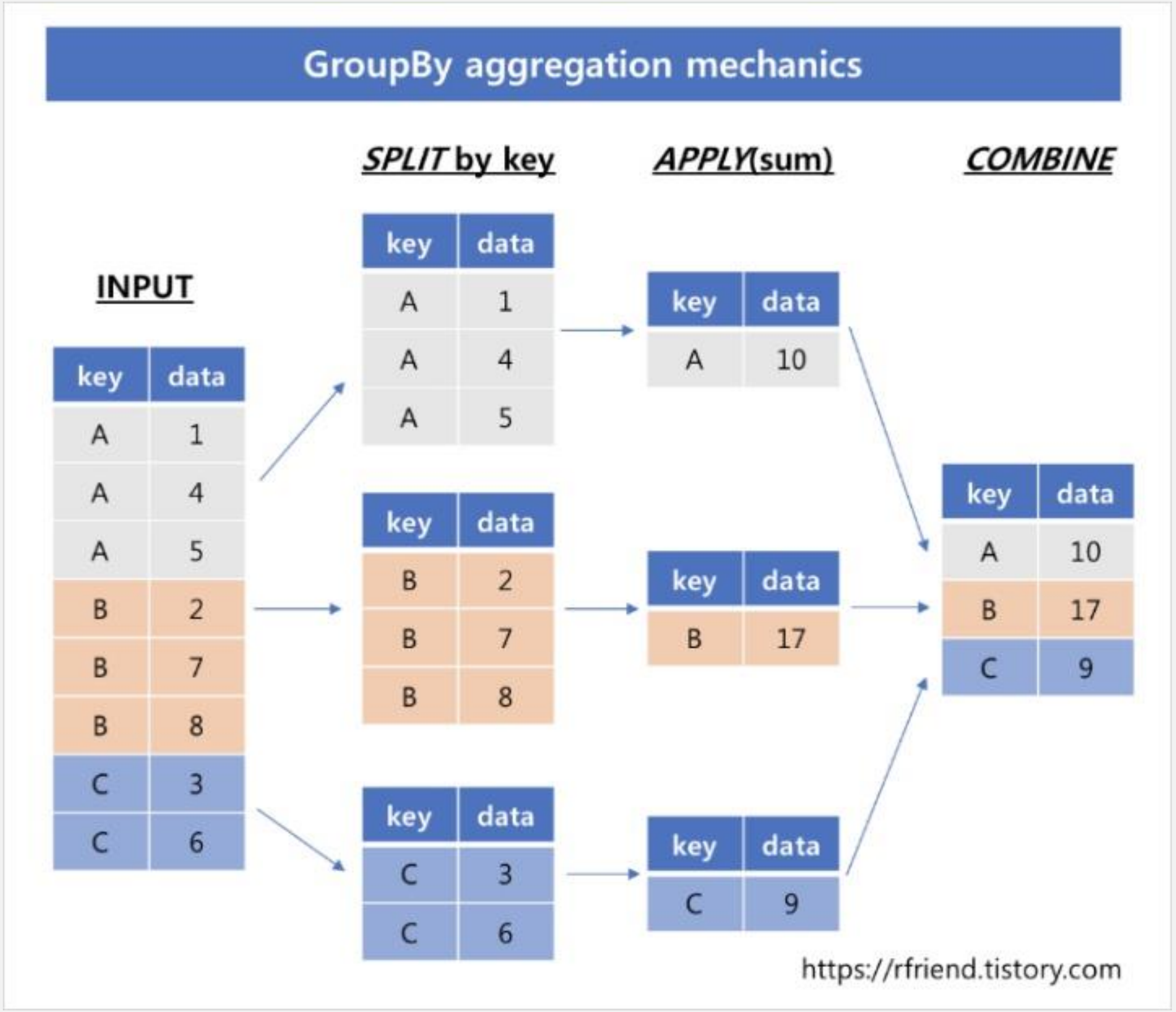
Assign a name for the result

```
# Use a named aggregation tuple
df.agg(fare_sum=('fare', 'sum'),
       fare_mean=('fare', 'mean'),
       sex_count=('sex', 'count'))
```

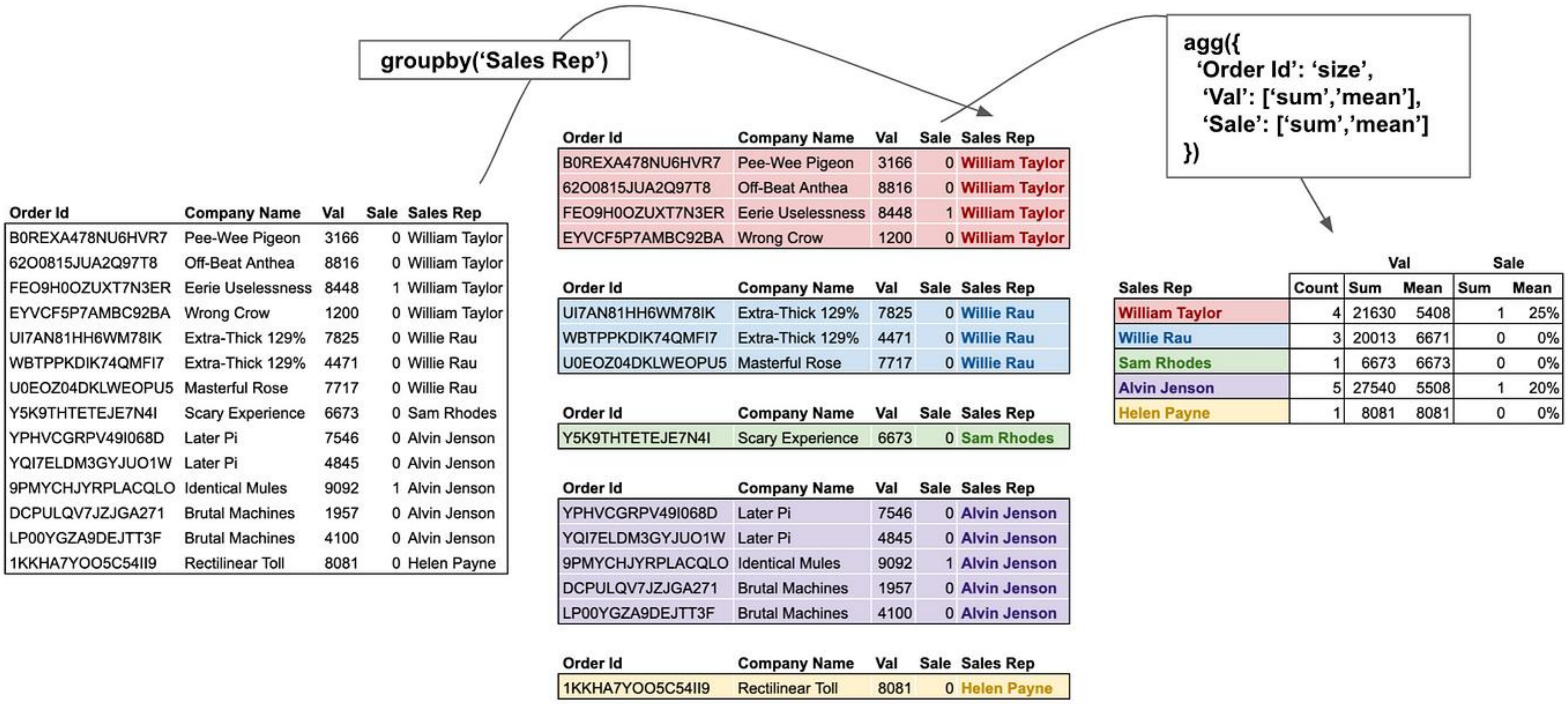
	fare	sex
fare_sum	28693.949300	NaN
fare_mean	32.204208	NaN
sex_count	NaN	891.0

Funkcje agregujące

1. Operacje matematyczne i statystyczne
2. Metoda agg()
3. Metoda groupby()


GroupBy aggregation mechanics
INPUT
SPLIT by key
APPLY(sum)
COMBINE
key | data
A 1, A 4, A 5, B 2, B 7, B 8, C 3, C 6
A 10
B 17
C 9
https://rfriend.tistory.com

Funkcje agregujące



```
groupby(‘Sales Rep’)
```

Order Id	Company Name	Val	Sale	Sales Rep
B0REXA478NU6HVR7	Pee-Wee Pigeon	3166	0	William Taylor
62O0815JUA2Q97T8	Off-Beat Anthea	8816	0	William Taylor
FEO9H0OZUXT7N3ER	Eerie Uselessness	8448	1	William Taylor
EYVCF5P7AMBC92BA	Wrong Crow	1200	0	William Taylor
UI7AN81HH6WM78IK	Extra-Thick 129%	7825	0	Willie Rau
WBTPPKDIK74QMFI7	Extra-Thick 129%	4471	0	Willie Rau
U0EOZ04DKLWEOPU5	Masterful Rose	7717	0	Willie Rau
Y5K9THTETEJE7N4I	Scary Experience	6673	0	Sam Rhodes
YPHVCGRPV49I068D	Later Pi	7546	0	Alvin Jenson
YQI7ELDM3GYJUO1W	Later Pi	4845	0	Alvin Jenson
9PMYCHJYRPLACQLO	Identical Mules	9092	1	Alvin Jenson
DCPULQV7JZJGA271	Brutal Machines	1957	0	Alvin Jenson
LP00YGZA9DEJTT3F	Brutal Machines	4100	0	Alvin Jenson
1KKHA7YOO5C54II9	Rectilinear Toll	8081	0	Helen Payne

Order Id	Company Name	Val	Sale	Sales Rep
B0REXA478NU6HVR7	Pee-Wee Pigeon	3166	0	William Taylor
62O0815JUA2Q97T8	Off-Beat Anthea	8816	0	William Taylor
FEO9H0OZUXT7N3ER	Eerie Uselessness	8448	1	William Taylor
EYVCF5P7AMBC92BA	Wrong Crow	1200	0	William Taylor

Order Id	Company Name	Val	Sale	Sales Rep
UI7AN81HH6WM78IK	Extra-Thick 129%	7825	0	Willie Rau
WBTPPKDIK74QMFI7	Extra-Thick 129%	4471	0	Willie Rau
U0EOZ04DKLWEOPU5	Masterful Rose	7717	0	Willie Rau

Order Id	Company Name	Val	Sale	Sales Rep
Y5K9THTETEJE7N4I	Scary Experience	6673	0	Sam Rhodes

Order Id	Company Name	Val	Sale	Sales Rep
YPHVCGRPV49I068D	Later Pi	7546	0	Alvin Jenson
YQI7ELDM3GYJUO1W	Later Pi	4845	0	Alvin Jenson
9PMYCHJYRPLACQLO	Identical Mules	9092	1	Alvin Jenson
DCPULQV7JZJGA271	Brutal Machines	1957	0	Alvin Jenson
LP00YGZA9DEJTT3F	Brutal Machines	4100	0	Alvin Jenson

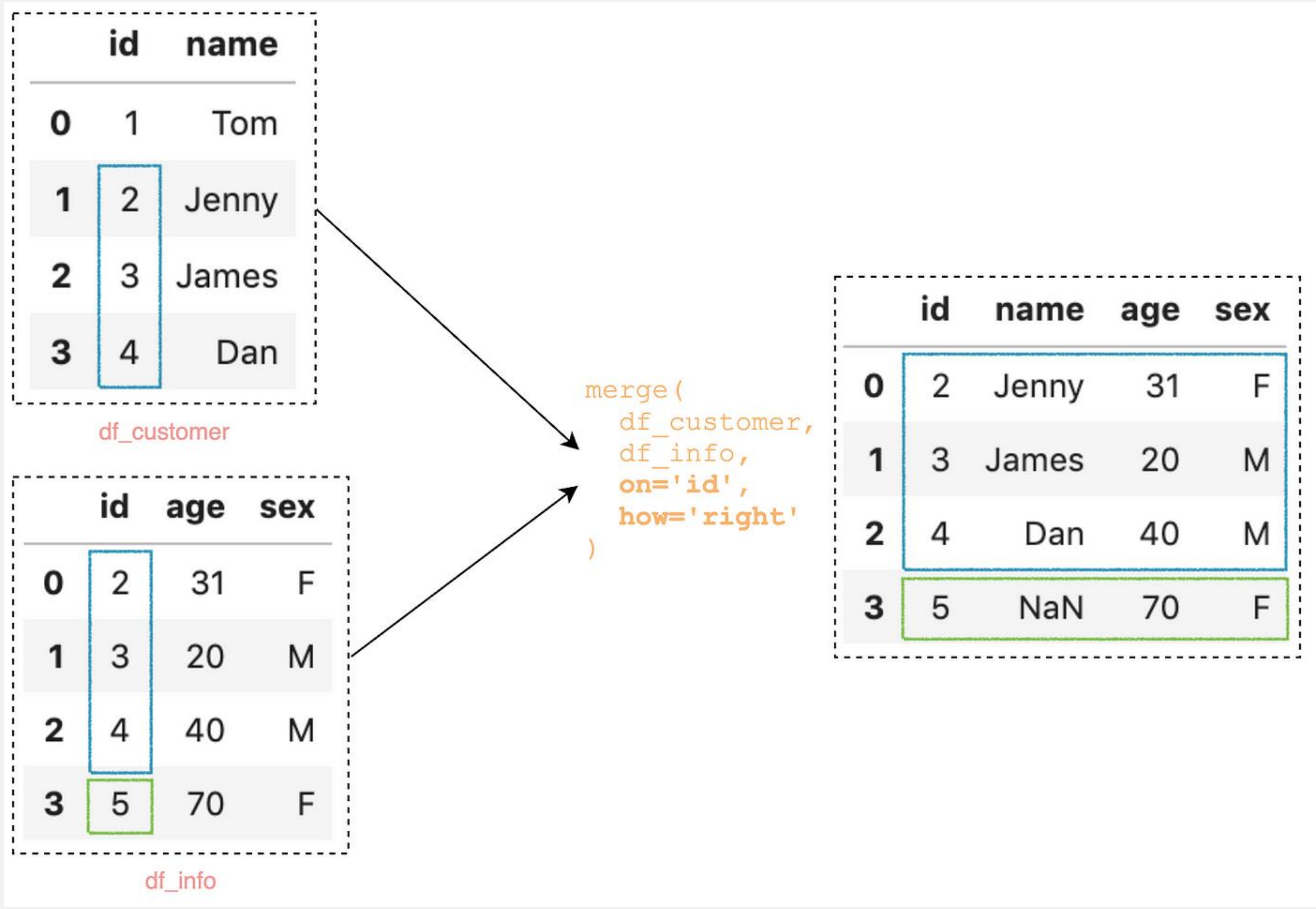
Order Id	Company Name	Val	Sale	Sales Rep
1KKHA7YOO5C54II9	Rectilinear Toll	8081	0	Helen Payne

```
agg({
  ‘Order Id’: ‘size’,
   ‘Val’: [‘sum’,’mean’],
   ‘Sale’: [‘sum’,’mean’]
})
```

		Val		Sale	
Sales Rep	Count	Sum	Mean	Sum	Mean
William Taylor	4	21630	5408	1	25%
Willie Rau	3	20013	6671	0	0%
Sam Rhodes	1	6673	6673	0	0%
Alvin Jenson	5	27540	5508	1	20%
Helen Payne	1	8081	8081	0	0%

Funkcje agregujące

1. Operacje matematyczne i statystyczne
2. Metoda agg()
3. Metoda groupby()
4. Funkcja pd.merge()



df_customer

	id	name
0	1	Tom
1	2	Jenny
2	3	James
3	4	Dan

df_info

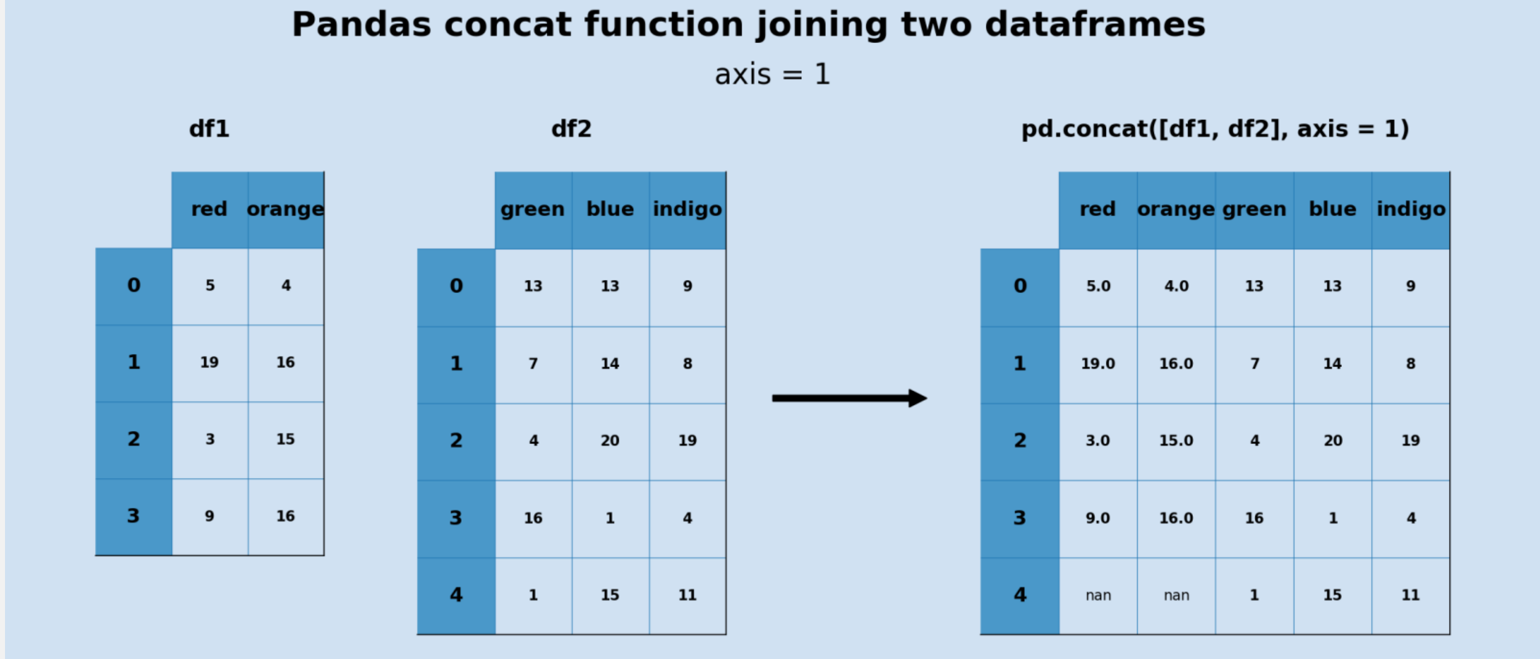
	id	age	sex
0	2	31	F
1	3	20	M
2	4	40	M
3	5	70	F

```
merge(
   df_customer,
   df_info,
   on='id',
   how='right'
)
```

	id	name	age	sex
0	2	Jenny	31	F
1	3	James	20	M
2	4	Dan	40	M
3	5	NaN	70	F

Funkcje agregujące

1. Operacje matematyczne i statystyczne
2. Metoda agg()
3. Metoda groupby()
4. Funkcja pd.merge()
5. Funkcja pd.concat()



Pandas concat function joining two dataframes

axis = 1

df1

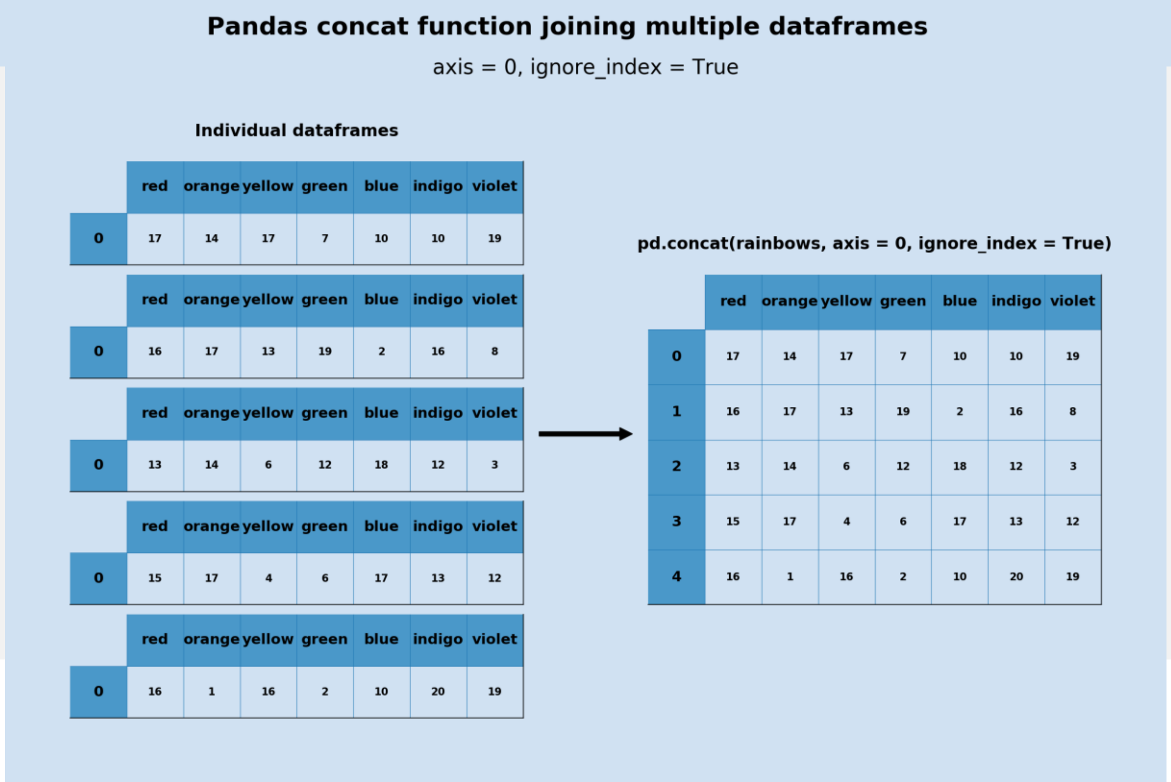
	red	orange
0	5	4
1	19	16
2	3	15
3	9	16

df2

	green	blue	indigo
0	13	13	9
1	7	14	8
2	4	20	19
3	16	1	4
4	1	15	11

pd.concat([df1, df2], axis = 1)

	red	orange	green	blue	indigo
0	5.0	4.0	13	13	9
1	19.0	16.0	7	14	8
2	3.0	15.0	4	20	19
3	9.0	16.0	16	1	4
4	nan	nan	1	15	11



Pandas concat function joining multiple dataframes

axis = 0, ignore_index = True

Individual dataframes

	red	orange	yellow	green	blue	indigo	violet
0	17	14	17	7	10	10	19

	red	orange	yellow	green	blue	indigo	violet
0	16	17	13	19	2	16	8

	red	orange	yellow	green	blue	indigo	violet
0	13	14	6	12	18	12	3

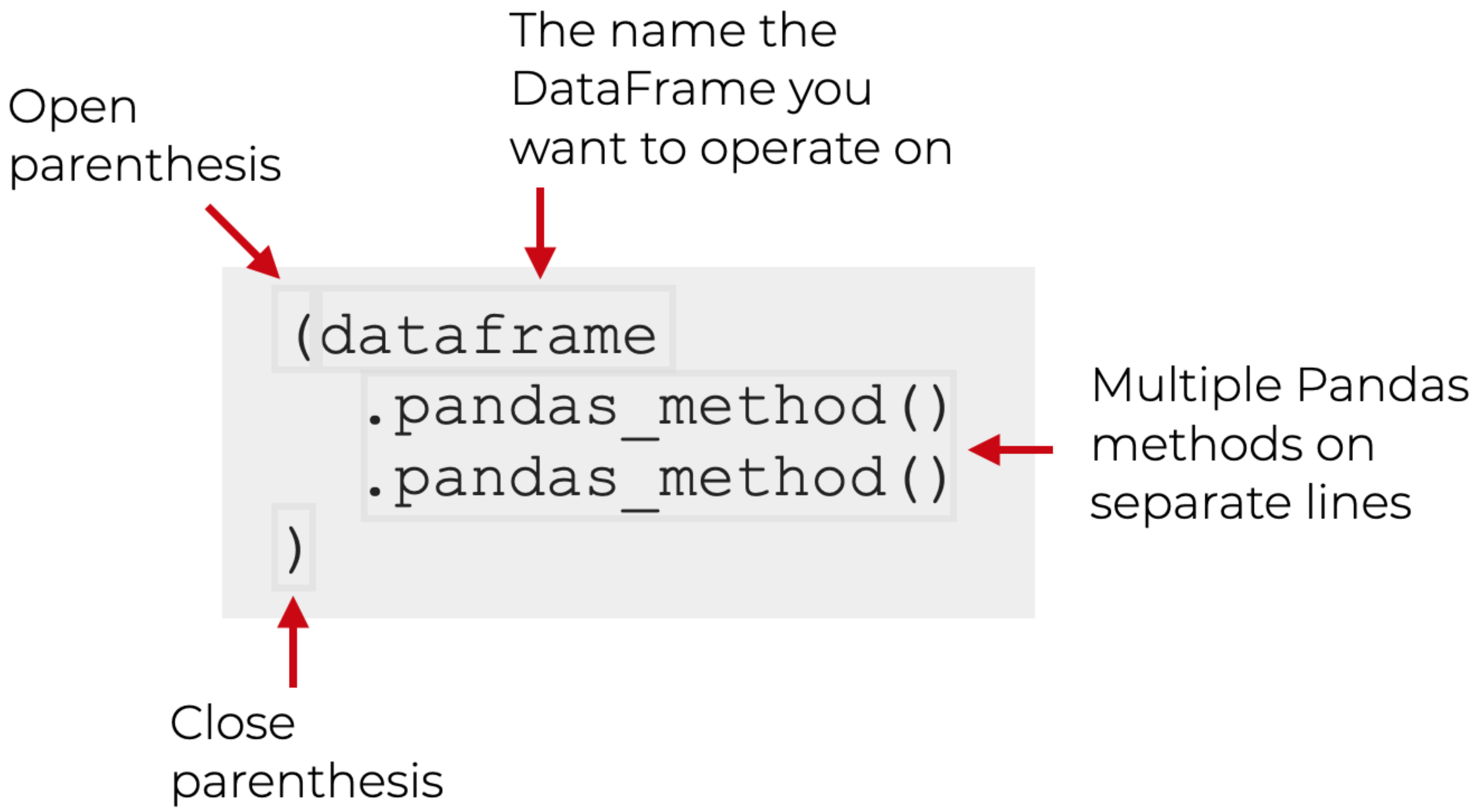
	red	orange	yellow	green	blue	indigo	violet
0	15	17	4	6	17	13	12

	red	orange	yellow	green	blue	indigo	violet
0	16	1	16	2	10	20	19

pd.concat(rainbows, axis = 0, ignore_index = True)

	red	orange	yellow	green	blue	indigo	violet
0	17	14	17	7	10	10	19
1	16	17	13	19	2	16	8
2	13	14	6	12	18	12	3
3	15	17	4	6	17	13	12
4	16	1	16	2	10	20	19

Method chaining


Open parenthesis
The name the DataFrame you want to operate on
Multiple Pandas methods on separate lines
Close parenthesis

```
(dataframe
    .pandas_method()
    .pandas_method()
)
```

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
     )
```