

```
'''
```

1.2.1

Создать массив 5x2. Создать массив 5x2. Вывести Все значения массива, значение элемента с индексом (3,1) и Второй столбец. Индексация начинается с нуля.

```
'''
```

```
import numpy as np
x = np.array([[1,2],[3,4],[5,6],[7,8],[9,10]])
print(x)
print(x[3][1])
print(x[1])
```

Program output:

```
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]]
8
[3 4]
```

```
'''
```

1.2.2

Пример. Выполнить следующее:

1. Создать Вектор (одномерный массив) размера 10, заполненный нулями.
2. Создать Вектор размера 10, заполненный единицами.
3. Создать Вектор размера 10, заполненный заданным числом.
4. Создать Вектор со значениями от 10 до 19.

```
'''
```

```
import numpy as np
a = np.zeros(10)
b = np.ones(10)
c = np.full(10, 5)
d = np.arange(10,20)
print(f"{a}\n{b}\n{c}\n{d}")
```

Program output:

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
[5 5 5 5 5 5 5 5 5 5]
[10 11 12 13 14 15 16 17 18 19]
```

```
'''
```

1.2.3

Создать массив 10x10 со случайными значениями, найти минимум, максимум и среднее значение

```
'''
```

```
import numpy as np
```

```
Z = np.random.random((10,10))
```

```
Zmin, Zmax, Zmean = Z.min(), Z.max(), Z.mean()
```

```
print(Zmin, Zmax, Zmean)
```

Program output:

0.0007425851441517084 0.9985341796241479 0.5006945895491264

```
'''
```

1.2.4

Задасть матрицу размерности 5 на 5 и поменять 2 строки в матрице местами.

```
'''
```

```
import numpy as np
A = np.arange(25).reshape(5,5)
A[[0,1]] = A[[1,0]]
print(A)
```

Program output:

```
[[ 5  6  7  8  9]
 [ 0  1  2  3  4]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

```
'''
```

1.2.5

Выяснить результат следующих Выражений:

```
0 * np.nan
```

```
np.nan == np.nan
```

```
np.inf > np.nan
```

```
np.nan - np.nan
```

```
0.3 == 3 * 0.1
```

```
'''
```

```
import numpy as np
```

```
print(0 * np.nan)
```

```
print(np.nan == np.nan)
```

```
print(np.inf > np.nan)
```

```
print(np.nan - np.nan)
```

```
print(0.3 == 3 * 0.1)
```

Program output:

```
nan
```

```
False
```

```
False
```

```
nan
```

```
False
```

```
'''
```

1.2.6

Отсортировать массив

```
'''
```

```
import numpy as np
```

```
arr = np.array([2,1,5,3,7,4,6,8])
```

```
print(np.sort(arr))
```

Program output:

```
[1 2 3 4 5 6 7 8]
```

```
'''
1.3.1
Создать 8x8 матрицу и заполнить её в шахматном порядке нулями и единицами.
'''
```

```
import numpy as np
arr = np.zeros(64).reshape(8,8)
for r in range(8):
    for c in range(8):
        if (r+c) % 2 == 0:
            arr[r, c] = 1
print(arr)
```

Program output:

```
[[1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]]
```

```
'''
```

1.3.2

Создать 5x5 матрицу со значениями в строках от 0 до 4. Для создания необходимо использовать функцию arrange

```
'''
```

```
import numpy as np
arr = []
for i in range(5):
    arr.extend(np.arange(5))
a = np.array(arr).reshape(5,5)
print(a)
```

Program output:

```
[[0 1 2 3 4]
 [0 1 2 3 4]
 [0 1 2 3 4]
 [0 1 2 3 4]
 [0 1 2 3 4]]
```



```
'''
```

1.3.3

Создать массив 3x3x3 со случайными значениями

```
'''
```

```
import numpy as np
a = np.random.random((3, 3, 3))
print(a)
```

Program output:

```
[[[0.67844967 0.19530666 0.36666781]
   [0.5831768  0.08759185 0.91725454]
   [0.28627246 0.98670272 0.99840637]]
```

```
 [[0.25140324 0.92705941 0.17145808]
   [0.49817993 0.89650224 0.98042779]
   [0.03680233 0.9665293  0.20760867]]
```

```
 [[0.49883315 0.83733125 0.51024697]
   [0.00739799 0.23384291 0.7517169 ]
   [0.02740469 0.5543066  0.85290223]]]
```

|||

```
import numpy as np
w = np.random.randint(3, 12)
h = np.random.randint(3, 12)

a = np.zeros(h*w).reshape(h,w)
for i in range(0, w):
    a[0,i] = 1
    a[h-1,i] = 1
for i in range(0, h):
    a[i,0] = 1
    a[i,w-1] = 1
print(a)
```

Program output:

[illegible]

```
'''
```

1.3.5

Создайте массив и отсортируйте его по убыванию.

```
'''
```

```
import numpy as np
```

```
a = np.random.rand(10)
```

```
print(np.sort(a)[::-1])
```

Program output:

```
[0.94399954 0.92383593 0.89199155 0.88294501 0.80066394 0.75385272
 0.63842449 0.51091346 0.24298048 0.14768308]
```

```
'''
```

1.3.6

Создайте матрицу, Выведите ее форму, размер и размерность

```
'''
```

```
import numpy as np
a = np.random.random((5,5))
print(a)
print(f"Форма: {a.shape}")
print(f"Размер: {a.size}")
print(f"Размерность: {a.ndim}")
```

Program output:

```
[[0.01512618 0.99574816 0.57421718 0.26238292 0.2565942 ]
 [0.70604477 0.91335171 0.23174879 0.22457481 0.25597798]
 [0.42090209 0.2842053  0.07086029 0.36204984 0.87616821]
 [0.15782271 0.8060692  0.76621607 0.60147931 0.05616722]
 [0.09249518 0.45758547 0.25759623 0.27877316 0.20562013]]
```

Форма: (5, 5)

Размер: 25

Размерность: 2

```
'''
```

2.2.1

Создать Series из списка Python, словаря Python, и массива Numpy (установить буквенные метки для последнего).

```
'''
```

```
import pandas as pd
import numpy as np
lst = [1,2,3,4,5]
d = {'a':1, 'b':2, 'c':3}
ndarr = np.array([1,2,3,4,5])
s1 = pd.Series(lst)
s2 = pd.Series(d)
s3 = pd.Series(ndarr, ['a', 'b', 'c', 'd', 'e'])
print(s1)
print(s2)
print(s3)
```

Program output:

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
a    1
b    2
c    3
dtype: int64
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

```
'''
```

2.2.2

Дано gBa Series. Напечатать их первые элементы и Все элементы после третьего (Во Втором фрейме).

```
'''
```

```
import numpy as np
import pandas as pd
s1 = pd.Series([1, 2, 3, 4, 5], ['a', 'b', 'c', 'd', 'e'])
s2 = pd.Series([5, 4, 3, 2, 1])
print(s1['a'])
print(s2[0])
print(s2[3:])
```

Program output:

```
1
5
3    2
4    1
dtype: int64
```

```
'''
```

2.2.3

Создайте новый фрейм данных.

```
'''
```

```
import pandas as pd
import numpy as np
dataframe = pd.DataFrame ( )
dataframe['Имя'] = ['Джеки Джексон', 'СтуВен стуВенсон']
dataframe['Возраст'] = [38, 25]
dataframe['Водитель'] = [True, False]
print(dataframe)
```

Program output:

	Имя	Возраст	Водитель
0	Джеки Джексон	38	True
1	СтуВен стуВенсон	25	False

```
'''
```

2.2.4

Загрузите фрейм данных по ссылке:

https://web.archive.org/web/20201120015841id_/https://raw.githubusercontent.com/chrisalbon/simulated_datasets/master/titanic.csv

```
'''
```

```
import numpy as np
```

```
import pandas as pd
```

```
url =
```

```
"https://web.archive.org/web/20201120015841id_/https://raw.githubusercontent.com/chrisalbon/simulated_datasets/master/titanic.csv"
```

```
dataframe = pd.read_csv(url)
```

```
print(dataframe.head(5))
```

Program output:

	Name	PClass	...	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	...	1	1
1	Allison, Miss Helen Loraine	1st	...	0	1
2	Allison, Mr Hudson Joshua Creighton	1st	...	0	0
3	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	1st	...	0	1
4	Allison, Master Hudson Trevor	1st	...	1	0

[5 rows x 6 columns]


```
...
```

2.2.5

Проанализировать характеристики фрейма данных

```
...
```

```
import numpy as np
import pandas as pd
url =
"https://web.archive.org/web/20201120015841id_/https://raw.githubusercontent.com/chrisalbon/simulated_datasets/master/titanic.csv"

dataframe = pd.read_csv(url)
print(f"{dataframe.head(2)}\n")
print(f"{dataframe.tail(3)}\n")
print(f"{dataframe.shape}\n")
print(dataframe.describe(include='all'))
```

Program output:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1

	Name	PClass	Age	Sex	Survived	SexCode
1310	Zenni, Mr Philip	3rd	22.0	male	0	0
1311	Lievens, Mr Rene	3rd	24.0	male	0	0
1312	Zimmerman, Leo	3rd	29.0	male	0	0

(1313, 6)

	Name	PClass	Age	Sex	Survived	SexCode
count	1313	1313	756.000000	1313	1313.000000	1313.000000
unique	1310	4	NaN	2	NaN	NaN
top	Connolly, Miss Kate	3rd	NaN	male	NaN	NaN
freq	2	711	NaN	851	NaN	NaN
mean	NaN	NaN	30.397989	NaN	0.342727	0.351866
std	NaN	NaN	14.259049	NaN	0.474802	0.477734
min	NaN	NaN	0.170000	NaN	0.000000	0.000000
25%	NaN	NaN	21.000000	NaN	0.000000	0.000000

50%	NaN	NaN	28.000000	NaN	0.000000	0.000000
75%	NaN	NaN	39.000000	NaN	1.000000	1.000000
max	NaN	NaN	71.000000	NaN	1.000000	1.000000

```
'''
```

2.2.6

Выберите индивидуальные данные или срезы фрейма данных

```
'''
```

```
import numpy as np
```

```
import pandas as pd
```

```
url =
```

```
"https://web.archive.org/web/20201120015841id_/https://raw.githubusercontent.com/chrisalbon/simulated_datasets/master/titanic.csv"
```

```
"
```

```
dataframe = pd.read_csv(url)
```

```
print(dataframe.iloc[1:4])
```

Program output:

	Name	PClass	...	Survived	SexCode
1	Allison, Miss Helen Loraine	1st	...	0	1
2	Allison, Mr Hudson Joshua Creighton	1st	...	0	0
3	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	1st	...	0	1

[3 rows x 6 columns]

```
'''
```

2.2.7

Требуется отобрать строки фрейма данных на основе некоторого условия. Необходимо сформировать новый фрейм данных из пассажиров первого класса.

```
'''
```

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

```
url =
```

```
"https://web.archive.org/web/20201120015841id_/https://raw.githubusercontent.com/chrisalbon/simulated_datasets/master/titanic.csv"
```

```
dataframe = pd.read_csv(url)
```

```
print(dataframe[dataframe['PClass'] == '1st'].head(2))
```

Program output:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1

```
'''
```

2.3.1

Найди евклидово расстояние между двумя Series (точками) a и b, не используя Встроенную формулу.

```
'''
```

```
import pandas as pd
import numpy as np
s1 = pd.Series([1, 2, 3, 4, 5])
s2 = pd.Series([5, 4, 3, 2, 1])

print(sum((s1 - s2)**2)**.5)
```

Program output:

6.324555320336759

```
'''
```

2.3.2

Найдите В Интернете ссылку на любой csv файл и сформируйте из него фрейм данных (например, коллекцию фреймов данных можно найти здесь: <https://github.com/akmand/datasets>).

```
'''
```

```
import numpy as np
import pandas as pd
url = "https://raw.githubusercontent.com/akmand/datasets/refs/heads/main/airlines.csv"
dataframe = pd.read_csv(url)
print(dataframe.head(5))
```

Program output:

	Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay
0	CO	269	SFO	IAH	3	15	205	1
1	US	1558	PHX	CLT	3	15	222	1
2	AA	2400	LAX	DFW	3	20	165	1
3	AA	2466	SFO	DFW	3	20	195	1
4	AS	108	ANC	SEA	3	30	202	0

```
...
```

2.3.3

Проделайте с получившемся из предыдущего задания фреймом данных те же действия, что и в примерах 2.2.5–2.2.7.

```
...
```

```
import numpy as np
import pandas as pd
url = "https://raw.githubusercontent.com/akmand/datasets/refs/heads/main/airlines.csv"
dataframe = pd.read_csv(url)
print(dataframe.head(2))
print(dataframe.tail(3))
print(dataframe.shape)
print(dataframe.describe())
print(dataframe.iloc[1:4])
print(dataframe[dataframe['Airline'] == 'AA'].head(2))
```

Program output:

	Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay
0	CO	269	SFO	IAH	3	15	205	1
1	US	1558	PHX	CLT	3	15	222	1

	Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay
539380	FL	609	SFO	MKE	5	1439	255	0
539381	UA	78	HNL	SFO	5	1439	313	1
539382	US	1442	LAX	PHL	5	1439	301	1

(539383, 8)

	Flight	DayOfWeek	...	Length	Delay
count	539383.000000	539383.000000	...	539383.000000	539383.000000
mean	2427.928630	3.929668	...	132.202007	0.445442
std	2067.429837	1.914664	...	70.117016	0.497015
min	1.000000	1.000000	...	0.000000	0.000000
25%	712.000000	2.000000	...	81.000000	0.000000
50%	1809.000000	4.000000	...	115.000000	0.000000
75%	3745.000000	5.000000	...	162.000000	1.000000
max	7814.000000	7.000000	...	655.000000	1.000000

[8 rows x 5 columns]

Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay
---------	--------	-------------	-----------	-----------	------	--------	-------

1	US	1558	PHX	CLT	3	15	222	1
2	AA	2400	LAX	DFW	3	20	165	1
3	AA	2466	SFO	DFW	3	20	195	1
	Airline	Flight	AirportFrom	AirportTo	DayOfWeek	Time	Length	Delay
2	AA	2400	LAX	DFW	3	20	165	1
3	AA	2466	SFO	DFW	3	20	195	1


```
'''
```

3.2.1

Прошкалируйте числовой признак в диапазон между двумя значениями.

```
'''
```

```
import numpy as np
from sklearn import preprocessing
feature = np.array([[ -500.5], [-100.1], [0], [100.1], [900.9]])
minmax_scale = preprocessing.MinMaxScaler(feature_range = (0, 1))
scaled_feature = minmax_scale.fit_transform(feature)
print(scaled_feature)
```

Program output:

```
[[0.        ]
 [0.28571429]
 [0.35714286]
 [0.42857143]
 [1.        ]]
```

```
'''
```

3.2.2

Преобразуйте признак, чтобы он имел среднее значение 0 и стандартное отклонение 1.

```
'''
```

```
import numpy as np
from sklearn import preprocessing
x = np.array([[ -1000.1], [-200.21], [500.5], [600.6], [9000.9]])
scaler = preprocessing.StandardScaler()
standardized = scaler.fit_transform(x)
print(standardized)
print(f"Среднее: {round(standardized.mean())}")
print(f"Стандартное отклонение: {standardized.std()}")
```

Program output:

```
[[-0.76058191]
 [-0.54177399]
 [-0.35009651]
 [-0.3227144 ]
 [ 1.97516681]]
```

Среднее: 0

Стандартное отклонение: 1.0

```

'''
3.2.3
Дан фрейм данных
dfTest = pd.DataFrame({'A':[14.00,90.20,90.95,96.27,91.21],
                        'B':[103.02,107.26,110.35,114.23,114.68],
                        'C':['big','small','big','small','small']})
'''

```

Необходимо масштабировать его числовые столбцы

```

'''
from sklearn import preprocessing
import pandas as pd
scaler = preprocessing.MinMaxScaler()
dfTest = pd.DataFrame({'A':[14.00,90.20,90.95,96.27,91.21],
                        'B':[103.02,107.26,110.35,114.23,114.68],
                        'C':['big','small','big','small','small']})

dfTest[['A', 'B']] = scaler.fit_transform(dfTest[['A', 'B']])
print(dfTest)
'''

```

Program output:

	A	B	C
0	0.000000	0.000000	big
1	0.926219	0.363636	small
2	0.935335	0.628645	big
3	1.000000	0.961407	small
4	0.938495	1.000000	small

```
'''
```

3.3.2

Загрузить фрейм данных по ссылке: <https://raw.githubusercontent.com/akmand/datasets/master/iris.csv>. Необходимо Выполнить нормализацию первого числового признака (sepal_length_cm) с использованием минимаксного преобразования, а Второго (sepal_width_cm) с задействованием z-масштабирования.

```
'''
```

```
import pandas as pd
from sklearn import preprocessing
url = "https://raw.githubusercontent.com/akmand/datasets/master/iris.csv"
dataframe = pd.read_csv(url)

minmax = preprocessing.MinMaxScaler()
dataframe[['sepal_length_cm']] = minmax.fit_transform(dataframe[['sepal_length_cm']])
std = preprocessing.StandardScaler()
dataframe[['sepal_width_cm']] = std.fit_transform(dataframe[['sepal_width_cm']])
print(dataframe.head(5))
```

Program output:

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	species
0	0.222222	1.032057	1.4	0.2	setosa
1	0.166667	-0.124958	1.4	0.2	setosa
2	0.111111	0.337848	1.3	0.2	setosa
3	0.083333	0.106445	1.5	0.2	setosa
4	0.194444	1.263460	1.4	0.2	setosa