Практическая работа № 6.1
Библиотека Numpy для работы с данными

Задание 6.1. Библиотека NumPy.

6.1.1. Библиотека ***NumPy***. Импорт библиотеки.

6.1.2. Массивы. Структура.

6.1.3. Создание массива. Функции ***array( )*** - создание массива и ***values( )*** - преобразование датафрейма в массив (пример 6.1.2).

6.1.4. Функции создания массива заданного вида.

6.1.5. Индексирование массивов NumPy. Индекс и срез.

Массив NumPy

Это мощный многомерный массив в форме строк и колонок. С помощью библиотеки можно создавать массивы NumPy из вложенного списка Python и получать доступ к его элементам.

Массив NumPy — это не то же самое, что и класс array.array из Стандартной библиотеки Python, который работает только с одномерными массивами.

Одномерный массив NumPy.

```
import numpy as np

a = np.array([1,2,3])
print(a)
```

Результатом кода выше будет [1 2 3].

Многомерные массивы.

```
import numpy as np

a = np.array([[1,2,3],[4,5,6]])
print(a)
```

Результат — [[1 2 3] [4 5 6]].

Атрибуты массива NumPy

ndarray.ndim

Возвращает количество измерений массива.

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print(a.ndim)
```

Вывод кода сверху будет 2, поскольку «a» — это 2-мерный массив.

ndarray.shape

Возвращает кортеж размера массива, то есть (n,m), где n — это количество строк, а m — количество колонок.

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print(a.shape)
```

Вывод кода — (2,3), то есть 2 строки и 3 колонки.

ndarray.size

Возвращает общее количество элементов в массиве.

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print(a.size)
```

Вывод — 6, потому что 2 x 3.

ndarray.dtype

Возвращает объект, описывающий тип элементов в массиве.

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print(a.dtype)
```

Вывод — «int32», поскольку это 32-битное целое число.Можно явно определить тип данных массива NumPy.

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]], dtype = float)
print(a.dtype)
```

Этот код вернет float64, потому что это 64-битное число с плавающей точкой.

ndarray.itemsize

Возвращает размер каждого элемента в массиве в байтах.

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print(a.itemsize)
```
Вывод — 4, потому что 32/8.

ndarray.data

Возвращает буфер с актуальными элементами массива. Это альтернативный способ получения доступа к элементам через их индексы.

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print(a.data)
```
Этот код вернет список элементов.

ndarray.sum()

Функция вернет сумму все элементов ndarray.

```
import numpy as np
a = np.random.random((2,3))
print(a)
print(a.sum())
```

Сгенерированная в этом примере матрица — [[0.46541517 0.66668157 0.36277909] [0.7115755 0.57306008 0.64267163]], следовательно код вернет 3.422183052180838. Поскольку используется генератор случайных чисел, ваш результат будет отличаться.

ndarray.min()

Функция вернет элемент с минимальным значением из ndarray.

```
import numpy as np
a = np.random.random((2,3))
print(a.min())
```
Сгенерированная в этом примере матрица — [[0.46541517 0.66668157 0.36277909] [0.7115755 0.57306008 0.64267163]], следовательно код вернет 0.36277909. Поскольку используется генератор случайных чисел, ваш результат будет отличаться.

ndarray.max()

Функция вернет элемент с максимальным значением из ndarray.

```
import numpy as np
a = np.random.random((2,3))
print(a.min())
```

Сгенерированная в этом примере матрица — [[0.46541517 0.66668157 0.36277909] [0.7115755 0.57306008 0.64267163]], следовательно код вернет 0.7115755. Поскольку используется генератор случайных чисел, ваш результат будет отличаться

Функции NumPy

type(numpy.ndarray)Это функция Python, используемая, чтобы вернуть тип переданного параметра. В случае с массивом numpy, она вернет numpy.ndarray.

```
import numpy as np

a = np.array([[1,2,3],[4,5,6]])
print(type(a))
```

Код выше вернет numpy.ndarray.

numpy.zeroes()

numpy.zeros((rows, columns), dtype)

Эта функция создаст массив numpy с заданным количеством измерений, где каждый элемент будет равняться 0. Если dtype не указан, по умолчанию будет использоваться dtype.

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

Код вернет массив numpy 3×3, где каждый элемент равен 0.

numpy.ones(

numpy.ones((rows,columns), dtype)

Эта функция создаст массив numpy с заданным количеством измерений, где каждый элемент будет равняться 1. Если dtype не указан, по умолчанию будет использоваться dtype.

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

Код вернет массив numpy 3 x 3, где каждый элемент равен 1.

numpy.empty()

numpy.empty((rows,columns))

Эта функция создаст массив, содержимое которого будет случайным — оно зависит от состояния памяти.

```
import numpy as np
np.empty((3,3))
```

```
print(a)
```
Код вернет массив numpy 3×3, где каждый элемент будет случайным.

numpy.arrange()
numpy.arrange(start, stop, step)
Эта функция используется для создания массива numpy, элементы которого лежат в диапазоне значений от start до stop с разницей равной step.

```
import numpy as np
a=np.arange(5,25,4)
print(a)
Вывод этого кода — [5 9 13 17 21]
```

numpy.linspace()
numpy.linspace(start, stop, num_of_elements)
Эта функция создаст массив numpy, элементы которого лежат в диапазоне значений между start до stop, а num_of_elements — это размер массива. Тип по умолчанию — float64.

```
import numpy as np
a=np.linspace(5,25,5)
print(a)
Вывод — [5 10 15 20 25].
```

numpy.logspace()
numpy.logspace(start, stop, num_of_elements)
Эта функция используется для создания массива numpy, элементы которого лежат в диапазоне значений от start до stop, а num_of_elements — это размер массива. Тип по умолчанию — float64. Все элементы находятся в пределах логарифмической шкалы, то есть представляют собой логарифмы соответствующих элементов.

```
import numpy as np
a = np.logspace(5,25,5)
print(a)
Вывод — [1.e+05 1.e+10 1.e+15 1.e+20 1.e+25].
```

numpy.sin()
numpy.sin(numpy.ndarray)
Этот код вернет синус параметра.

```
import numpy as np
a = np.logspace(5,25,2)
print(np.sin(a))
```

Вывод кода сверху равен [0.0357488 -0.3052578]. Также есть cos(), tan() и так далее.

numpy.reshape()
numpy.reshape(dimensions)

Эта функция используется для изменения количества измерений массива numpy. От количества аргументов в reshape зависит, сколько измерений будет в массиве numpy.

```
import numpy as np
a = np.arange(9).reshape(3,3)
print(a)
```

Вывод этого года — 2-мерный массив 3×3.

numpy.random.random()
numpy.random.random((rows, column))

Эта функция возвращает массив с заданным количеством измерений, где каждый элемент генерируется случайным образом.

```
a = np.random.random((2,2))
```

Этот код вернет ndarray 2×2.

numpy.exp()
numpy.exp(numpy.ndarray)

Функция вернет ndarray с экспоненциальной величиной каждого элемента.

```
b = np.exp([10])
```

Значением кода выше будет 22025.4657948.

numpy.sqrt()
numpy.sqrt(numpy.ndarray)

Эта функция вернет ndarray с квадратным корнем каждого элемента.

```
b = np.sqrt([16])
```

Этот код вернет значение 4.

Базовые операции NumPy

```
a = np.array([5, 10, 15, 20, 25])
b = np.array([0, 1, 2, 3])
```

Этот код вернет разницу двух массивов

```
c = a - b
```

Этот код вернет массив, где каждое значение возведено в квадрат

```
b**2
```

Этот код вернет значение в соответствии с заданным выражением

```
10 * np.sin(a)
```

Этот код вернет True для каждого элемента, чье значение удовлетворяет условие

```
a < 15
```

Базовые операции с массивом NumPy

```
a = np.array([[1,1], [0,1]])
b = np.array([[2,0],[3,4]])
```

Этот код вернет произведение элементов обоих массивов

```
 a * b
```

Этот код вернет матричное произведение обоих массивов

```
 a @ b
```

или

```
 a.dot(b)
```

Пример 6.1.1

Создание массивов

```
import numpy as np
arr1 = np.array([1,2,3,4])
print('arr1 =',arr1)
arr2 = np.array([5,6,7,8], dtype = float)
print('arr2 =',arr2)
arr2.dtype

arr3 = np.array([[1,2,3],[4,5,6],[7,8,9]])
np.shape(arr3)    # размерность
```

Пример 6.1.2

Преобразовать датафрейм или его часть, содержащую количественные показатели, в объект ndarray

```
import numpy as np
import pandas as pd

df = pd.DataFrame()
x = df.values
type(x)
```

x, полученный таким образом из `DataFrame` практически представляет собой матрицу – массив с двумя индексами (номер строки и номер колонки, соответственно).

Используя функцию values, массив можно получить из **Series**:

```
import numpy as np
import pandas as pd
my_series = pd.Series([5, 6, 7, 8, 9, 10])
my_series.values
```

Пример 6.1.3

Массив заданного вида

np.**empty**(**5**) # одномерный массив из пяти элементов, память для которого выделена, но не инициализирована

np.**zeros**((**10**, **7**)) # массив размером 10x7, заполненный нулями

np.**ones**((**3**,**3**,**3**)) # массив размером 3x3x3, заполненный единицами

np.**eye**(**3**) # единичная матрица (элементы главной диагонали равны 1, остальные — 0) размера 3x3

np.**full**((**3**, **5**), **3.14**) # массив 3x5 заполненный числом 3.14

np.**arange**(**0**, **21**, **7**) # одномерный массив, заполненный числами в диапазоне от 0 до 20 с шагом 7

np.**linspace**(**0**, **1**, **5**) # массив из пяти чисел, равномерно распределённых в интервале между 0 и 1 включительно

np.**random**.randint(**0**, **10**, (**3**, **3**)) # массив размера 3x3, заполненный случайными числами из диапазона от 0 до 9 (включительно)

Пример 6.1.4

Индексация и срез

Основные правила:

- индекс **первого** элемента массива равен 0;
- для обращения к элементу массива по индексу необходимо указать **имя** переменной, в которой хранится массив, и **индекс** в квадратных скобках;
- допускается использование **отрицательных** индексов;

```
my_array = np.array([x for x in range(10)])
my_array[5]
my_array[-1]
my_array[3:6]
my_array[1:8:3]
```

```
my_array = np.array([[1,2,3,4], [10,11,12,13], [45,46,47,48]])
my_array[1][2]
```

```
my_array[1,2]
```

Функции с массивами

Функция	**Описание**
abs	Абсолютное значение целых, вещественных или комплексных элементов массива
sqrt	Квадратный корень каждого элемента массива
exp	Экспонента (e^x) каждого элемента массива
log, log10, log2, log1p	Натуральный (по основанию e), десятичный, двоичный логарифм и функция log(1+x) соответственно
modf	Дробные и целые части массива в виде отдельных массивов
isnan	Массив логических (булевых) значений, показывающий, какие из элементов исходного массива являются NaN (не числами)
cos, sin, tan	Обычные тригонометрические функции
arccos, arcsin, arctan	Обратные тригонометрические функции

Создадим массив students

ID (номер в журнале)	**Рост, см**	**Масса тела, кг**	**Средний балл**
1	135	34	4
2	160	43	5
3	163	40	4.3
4	147	44	5
5	138	41	4.7

Вычисление **среднего**

```
mean = np.mean(students[:,-1])
```

Вычисление **медианы**

Многое в нахождении медианы зависит от того, **четное или нечетное количество наблюдений.**

Если n нечетное. Тут все просто: в середине окажется одно центральное значение. $k = \frac{n+1}{2} = 3$ медиана равна x_3=34 (следующий урок)

Если n четное. В середине оказывается два наблюдения. В таком случае мы находим среднее арифметическое этих двух значений.

k между n/2 и n/2+1

Узнаем, как учится средний ученик, с помощью **функции median:**

```
median = np.median(students[:,-1])
```

Вычисление **коэффициента корреляции** столбцов матрицы:

```
corr = np.corrcoef(students[:,1], students[:,2])
```