Семинар: знакомство с NumPy. Задач

(**Num**eric **Py**thon) (http://www.numpy.org/)

Дополнительные материалы

Для самых любознательных и тех, кто хочет порешать задачи, есть сборни
https://www.machinelearningplus.com/python/101-numpy-exercises-python/
(https://www.machinelearningplus.com/python/101-numpy-exercises-python/)

Для тех, кому нужно руководство по матричным операциям в Numpy:
https://www.programiz.com/python-programming/matrix (https://www.programiz
programming/matrix)

numpy reference: https://numpy.org/doc/stable/reference/index.html
(https://numpy.org/doc/stable/reference/index.html)

In [1]:

```
import numpy as np
```

Задания для самостоятельного решения

Часть 1

1. Развернуть одномерный массив (сделать так, чтобы его элементы шл
2. Найти максимальный нечетный элемент в массиве.
3. Замените все нечетные элементы массива на ваше любимое число.
4. Создайте массив первых n нечетных чисел, записанных в порядке убы
   `n=5` , то ответом будет `array([9, 7, 5, 3, 1])` . *Функции, которые*
   решении: `.arange()`
5. Вычислите самое близкое и самое дальнее числа к данному в рассма
   Например, если на вход поступают массив `array([0, 1, 2, 3, 4])`
   будет `(1, 4)` . *Функции, которые могут пригодиться при решении:*
   `.argmin()`

In [62]:

```
#Задача 1.1
#код:

A = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
print(np.flip(A))
```

```
[8 7 6 5 4 3 2 1 0]
```

В [63]:

```
#Задача 1.2
#код:

array = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
odd_array = array[array % 2 != 0]
max_odd = np.max(odd_array)
print(max_odd)

# A = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
# if (max_A % 2 != 0):
#     print(A.max())
# elif ((max_A - 1) % 2 != 0):
#     print((A - 1).max())
# else:
#     print('There are no odd elements in the array')
```

```
7
```

В [64]:

```
#Задача 1.3
#код:

favorite_number = int(input('Enter your favorite number: ')
array = np.array([1, 2, 3, 4, 5, 6, 7, 8])
array[array % 2 != 0] = favorite_number
print(array)

#N = int(input('Enter your favorite number: '))
#array = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
#new_array = np.where(A % 2 != 0, N, array)
#print(new_array)
```

```
Enter your favorite number: 9
[9 2 9 4 9 6 9 8]
```

В [25]:

```
#Задача 1.4
#код:

n = 5
print(np.arange(2*n-1, 0, -2))

#array = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
#y = [i for i in array if (i % 2)]
#print(list(reversed(sorted(y))) if y else 'Error')
```

```
[9 7 5 3 1]
```

B [53]:

```python
#Задача 1.5
#код:

def find_closest_and_farthest(array, number):
    differences = np.abs(array - number)  # вычисляем разни
    closest_index = np.argmin(differences)  # находим индек
    farthest_index = np.argmax(differences)  # находим инде
    closest_number = array[closest_index]  # получаем самое
    farthest_number = array[farthest_index]  # получаем сам
    return closest_number, farthest_number

array = np.array([1, 2, 3, 4, 5, 6, 7, 8])
number = float(input('Enter number: '))
closest, farthest = find_closest_and_farthest(array, number
print(closest, farthest)
```

```
Enter number: 1.87
2 8
```

Задания для самостоятельного реше

Часть 2. В этом разделе использовать только методы numpy. Не испо

Каждая задача в этом разделе решается одной-двумя строками кода. или найти подходящий метод в numpy

Везде, где встречаются массивы или матрицы, подразумевается, что это

Задача 2.1

Создайте одномерный массив чисел от 0 до 8.

Сделайте это двумя способами: с помощью метода `arange` из numpy и с списка (list) в numpy array

В [23]:

```
#Код:

#Способ 1:

#size = int(input('Enter size (If the size is 3, the one-di
#print(np.arange(size))
array = np.arange(9)
print(array)
# или print(np.arange(9))

#Способ 2:

list = [0, 1, 2, 3, 4, 5, 6, 7, 8]
array2 = np.array(list)
print(array2)
```

```
[0 1 2 3 4 5 6 7 8]
[0 1 2 3 4 5 6 7 8]
```

Тесты:

Выходные данные: `array([0, 1, 2, 3, 4, 5, 6, 7, 8])`

Задача 2.2

Вам дан numpy array под названием `test_array` . Выведите положительн

Подсказка: здесь можно также использовать булевы массивы в качестве б

В [19]:

```
#Код:

test_array = np.array([0, -1,  5, -3,  7, -1,  6, -1,  8, -

#for i in test_array:
#    if (test_array[i] > 0):
#        print(test_array[i])

positive_elements = test_array[test_array > 0]
print(positive_elements)
```

```
[5 7 6 8]
```

Тесты:

Входные данные:
`test_array = np.array([ 0, -1,  5, -3,  7, -1,  6, -1,  8, -1])`
Выходные данные:
`array([5, 7, 6, 8])`

Задача 2.3

Замените отрицательные элементы из массива `test_array` на 0. Подска функция `np.where`

В [52]:

```
#Код:

test_array = np.array([ 0, -1,  5, -3,  7, -1,  6, -1,  8,
array = np.where(test_array > 0, test_array, 0)
print(array)
```

```
[0 0 5 0 7 0 6 0 8 0]
```

Тесты:

Входные данные:
`test_array = np.array([ 0, -1,  5, -3,  7, -1,  6, -1,  8, -1])`
Выходные данные:
`array([0, 0, 5, 0, 7, 0, 6, 0, 8, 0])`

Задача 2.4

Вам дан массив из 10 чисел. С помощью одной функции в numpy найдите элемента в заданном массиве

В [45]:

```
#Код:

array = np.array([6, 8, 4, 2, 11, -3, 7, 22, -20, -30])
max_position = np.argmax(array)
print(max_position)
```

```
7
```

Тесты:

Входные данные:
`array = np.array([6, 8, 4, 2, 11, -3, 7, 22, -20, -30])`
Выходные данные:
`7`

Задача 2.5

Вам дан словарь, в котором подсчитаны частоты женских имен в выборке девочек.

В [60]:

```
names = dict()
names['Катя'] = 8
names["Маша"] = 5
names["Вероника"] = 2
names["Ярослава"] = 1
names["Александра"] = 10

vec = np.array([[8], [5], [2], [1], [10]])
vec / 26
```

Out[60]:

```
array([[0.30769231],
       [0.19230769],
       [0.07692308],
       [0.03846154],
       [0.38461538]])
```

Посчитайте процентные доли имен с помощью нормирования вектора зна

Подсказка: просто возьмите этот вектор значений и поделите его на сумму

Тесты:

Выходные данные:
`array([0.30769231, 0.19230769, 0.07692308, 0.03846154, 0.384615`

Примечание: мы только что произвели операцию **нормирования вектора** элементов 1), а получившиеся элементы можно интерпретировать как **вер** это веростности того, что случайно взятая новорожденная девочка будет и имя)

Задача 2.6

Вам дан массив `arr`. Выведите все его элементы, которые строго больш

Подсказка: здесь можно использовать булевы массивы в качестве булевой

В [52]:

```
#Код:

array = np.array([7, 2, 10, 2, 7, 4, 9, 4, 9, 8])
result = array[(array > 5) & (array < 9)]
print(result)
```

```
[7 7 8]
```

Тесты:

Входные данные:
`arr = np.array([7, 2, 10, 2, 7, 4, 9, 4, 9, 8])`
Выходные данные:

```
array([7, 7, 8])
```

Двумерные массивы

Задача 2.7

Создайте методом из numpy одномерный массив из 12 элементов от 0 до матрицу размером 3x4 (вам пригодится функция `np.reshape`)

B [17]:

```
#Код:

array = np.arange(12).reshape(3, 4)
print(array)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

Задача 2.8

Соедините вертикально и горизонтально две созданные матрицы a и b.

B [8]:

```
a = np.arange(8).reshape(2,4)
b = np.ones(8).reshape(2,4)

vert = np.vstack((a, b)) #вертикально
print(vert)

hor = np.hstack((a, b)) #горизонтально
print(hor)
```

```
[[0. 1. 2. 3.]
 [4. 5. 6. 7.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
[[0. 1. 2. 3. 1. 1. 1. 1.]
 [4. 5. 6. 7. 1. 1. 1. 1.]]
```

Задача 2.9

Сохраните в отдельную переменную второй столбец (столбец с индексом двумерного массива.

Обратите внимение, что при вырезании столбца из матрицы, у вас получи array, а не двумерная матрица с одним столбцом!

В [16]:

```
arr = np.arange(12).reshape(3,4)
print(arr)

arr = np.arange(12).reshape(3,4)
second_column = arr[:, 1]
print(second_column)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[1 5 9]
```

Задача 2.10

Создайте массив из случайных чисел размера 3 на 4 (3 строки и 4 столбца переменную `random_array` . Найдите среднее, медиану, стандартное откл матрицы `random_array` по строкам и по столбцам. По одной строке на ср стандартное отклонение для операций по строкам и по столбцам - всего 6

В [54]:

```
#Код:

random_array = np.random.random((3, 4))

mean_rows = np.mean(random_array, axis=1)
median_rows = np.median(random_array, axis=1)
std_rows = np.std(random_array, axis=1)

mean_columns = np.mean(random_array, axis=0)
median_columns = np.median(random_array, axis=0)
std_columns = np.std(random_array, axis=0)

print("Mean by rows:", mean_rows)
print("Median by rows:", median_rows)
print("Standard deviation by rows:", std_rows)

print("Mean by columns:", mean_columns)
print("Median by columns:", median_columns)
print("Standard deviation by columns:", std_columns)
```

```
Mean by rows: [0.36517122 0.44272072 0.47223484]
Median by rows: [0.27197364 0.4629567  0.48072071]
Standard deviation by rows: [0.27983268 0.25585405 0.33074471]
Mean by columns: [0.40264729 0.43044709 0.50795903 0.36578228]
Median by columns: [0.3419704  0.39136374 0.53454966 0.20197689
Standard deviation by columns: [0.30030458 0.2937282  0.2702875
8]
```

Задача 2.11

Найдите средние значения по строкам у созданной в задании 2.10 матриц полученные средния значения из соответствующих строк начальной матри Проверьте, что после вычитания средних среднее по каждой строке стало

В [56]:

```
#Код:

mean_subtracted = random_array - mean_rows[:, np.newaxis]
mean_subtracted_mean_rows = np.mean(mean_subtracted, axis=1

print("Mean after subtracting mean rows:", mean_subtracted_
```

```
Mean after subtracting mean rows: [0.00000000e+00 2.77555756e-17
6e-17]
```

Задача 2.12

Пусть мы имеем 2 массива (заданы ниже в коде). Требуется создать масси
результат поэлементно True, если четные элементы 1го массива больше 2
элементы 2го больше элементов первого. В противном случае возвращать

Реализовать задачу с помощью numpy

Пример массивов и результата:

a = [7, 3, 5, 9, 2, 5, 4, 8, 9, 0]

b = [2, 8, 6, 7, 1, 3, 2, 9, 5, 1]

res = [True, True, False, False, True, False, True, True, True, True]

Вы можете сделать задачу с помощью слайсинга или комбинации функций

В [6]:

```
#Код:

a = np.array([7, 3, 5, 9, 2, 5, 4, 8, 9, 0])
b = np.array([2, 8, 6, 7, 1, 3, 2, 9, 5, 1])

res = np.where((a % 2 == 0) & (a > b), True, np.where((b %
print(res)
```

```
[False False False False  True False  True  True False  True]
```

Задача 2.13

Создайте массив размера 10 со значениями от 0 до 1 (интервал между зна
включая концы. См np.linspace + слайсинг

В [57]:

```
#Код:

array = np.linspace(0, 1, 12)[1:-1]
print(array)
```

```
[0.09090909 0.18181818 0.27272727 0.36363636 0.45454545 0.54545
 0.63636364 0.72727273 0.81818182 0.90909091]
```

Задача 2.14

Создайте случайный вектор размера 10, и замените в нем максимальное :

В [58]:

```
#Код:

vec = np.random.rand(10)
max_value = np.max(vec)
max_index = np.argmax(vec)
vec[max_index] = 0

print(vec)
```

```
[0.26385122 0.         0.76093037 0.32311292 0.14835792 0.62043
 0.45235718 0.34333733 0.48010545 0.01246289]
```