# Домашнее задание - Библиотека numpy

Подпишите этот блокнот: укажите ФИО и номер группы. Выполненное задание загрузите в свой репозиторий «**Python_Introduction_2022**» в github

Перед выполнением этого домашнего задания полезно прочитать про возможности и основные методы библиотеки numpy:
http://acm.mipt.ru/twiki/bin/view/Cintro/PythonNumpy

In [1]:
```
import numpy as np
```

1. Создайте равномерное разбиение интервала от -1.3 до 2.5 на 64 отрезка.

In [18]:
```
np.linspace(-1.3,2.5,64)
```

Out[18]:
```
array([-1.3       , -1.23968254, -1.17936508, -1.11904762, -1.05873016,
       -0.9984127 , -0.93809524, -0.87777778, -0.81746032, -0.75714286,
       -0.6968254 , -0.63650794, -0.57619048, -0.51587302, -0.45555556,
       -0.3952381 , -0.33492063, -0.27460317, -0.21428571, -0.15396825,
       -0.09365079, -0.03333333,  0.02698413,  0.08730159,  0.14761905,
        0.20793651,  0.26825397,  0.32857143,  0.38888889,  0.44920635,
        0.50952381,  0.56984127,  0.63015873,  0.69047619,  0.75079365,
        0.81111111,  0.87142857,  0.93174603,  0.99206349,  1.05238095,
        1.11269841,  1.17301587,  1.23333333,  1.29365079,  1.35396825,
        1.41428571,  1.47460317,  1.53492063,  1.5952381 ,  1.65555556,
        1.71587302,  1.77619048,  1.83650794,  1.8968254 ,  1.95714286,
        2.01746032,  2.07777778,  2.13809524,  2.1984127 ,  2.25873016,
        2.31904762,  2.37936508,  2.43968254,  2.5       ])
```

1. Сгенерируйте numpy массив длины $3n$, заполненный циклически числами 1, 2, 3, 1, 2, 3, 1....

In [9]:
```
n = 2
a = np.arange(1,4)
np.tile(a,(n,))
```

Out[9]:
```
array([1, 2, 3, 1, 2, 3])
```

1. Создайте массив первых 10 нечетных целых чисел.

In [154…
```
a = np.arange(1,20)[::2]
print(a, len(a))
b = [i*2+1 for i in range(10)]
print(b)
```

```
[ 1  3  5  7  9 11 13 15 17 19] 10
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

1. Создайте массив нулей размера 10 x 10, а затем создайте в нём "рамку" из единиц по краям.

In [35]:
```
a = np.zeros((10,10))
print(a)
a[0]=1
a[9]=1
a[:, 0]=1
a[:, 9]=1
print(a)
```

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

1. Создайте массив 8 x 8 с шахматной доской из нулей и единиц.

In [40]:
```
a = np.tile(np.array([[0,1],[1,0]]),(4,4))
print(a)
```

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

1. Создайте функцию, которая создает $n \times n$ матрицу с $(i, j)$-элементом, равным $i + j$.

In [60]:
```
def matrix(n = 5):
    a = np.arange(n)
    A = np.tile(a,(n,1))
    return A+A.T
matrix(4)
```

Out[60]:
```
array([[0, 1, 2, 3],
       [1, 2, 3, 4],
       [2, 3, 4, 5],
       [3, 4, 5, 6]])
```

1. Примените функцию dot для перемножения вектор на вектор, матрицу на вектор и матрицу на матрицу.

In [63]:
```
vector = np.arange(4)
matrix = np.arange(16).reshape(4,4)
print(vector.dot(vector), matrix.dot(vector), matrix.dot(matrix),sep='\n')
```

```
14
[14 38 62 86]
[[ 56  62  68  74]
 [152 174 196 218]
 [248 286 324 362]
 [344 398 452 506]]
```

1. Вычислите cos(x) и sin(x) на интервале [0,1] с шагом $0.05$, а затем объедините оба массива чисел как строки в один массив.

In [87]:
```
x = np.append(np.arange(0,1,0.05),1)
print(x)
cos = np.cos(x)
sin = np.sin(x)
np.vstack((cos,sin))
```

```
[0.   0.05 0.1  0.15 0.2  0.25 0.3  0.35 0.4  0.45 0.5  0.55 0.6  0.65
 0.7  0.75 0.8  0.85 0.9  0.95 1.  ]
```

Out[87]:
```
array([[1.        , 0.99875026, 0.99500417, 0.98877108, 0.98006658,
        0.96891242, 0.95533649, 0.93937271, 0.92106099, 0.9004471 ,
        0.87758256, 0.85252452, 0.82533561, 0.7960838 , 0.76484219,
        0.73168887, 0.69670671, 0.65998315, 0.62160997, 0.58168309,
        0.54030231],
       [0.        , 0.04997917, 0.09983342, 0.14943813, 0.19866933,
        0.24740396, 0.29552021, 0.34289781, 0.38941834, 0.43496553,
        0.47942554, 0.52268723, 0.56464247, 0.60518641, 0.64421769,
        0.68163876, 0.71735609, 0.75128041, 0.78332691, 0.8134155 ,
        0.84147098]])
```

1. Создайте матрицу $A$ размера $3 \times 5$ из случайных чисел с равномерным распределением на отрезке $[-1, 3]$ (используйте np.random.rand)

In [99]:
```
import random as rd
A = np.random.rand(3,5)
B = np.array([rd.choice([-1,3]) for i in range(15)]).reshape(3,5)
A*=B
print(A)
```

```
[[ 2.74559379  0.97732243  1.55411094 -0.93653196  2.63970495]
 [ 0.45362795  0.48041351  2.56680844 -0.10032011 -0.33588077]
 [-0.74468784 -0.58897786 -0.16213856  0.17494683 -0.45431561]]
```

1. Найдите сумму всех элементов, сумму внутри строк, сумму внутри столбцов, а также среднее значение, дисперсию и стандартное отклонение чисел для каждой строки матрицы $A$. (Подобно тому, как sorted имеет необязательный аргумент key=, многие функции Numpy имеют необязательный аргумент axis=)

In [106…
```
print(A.sum())
print(A.sum(axis = 1))
print(A.sum(axis = 0))
for i in range(3):
    print('{}: {}, {}, {}'.format(i,A[i].mean(), A[i].var(), A[i].std()))
```

```
8.269676125867162
[ 6.98020015  3.06464902 -1.77517304]
[ 2.45453389  0.86875808  3.95878082 -0.86190524  1.84950857]
0: 1.396040029982859, 1.8018401449887675, 1.3423263928675349
1: 0.612929803050835, 1.053909263156175, 1.0266008295127056
2: -0.355034607860261, 0.1069010029012741, 0.3269571881780153
```

1. Отнимите от каждого элемента матрицы $A$ среднее по строке и поделите на стандартное отклонение строки.

In [132…
```
mean = np.vstack((list(np.tile(np.array([A[i].mean()]),5) for i in range(3))))
std = np.vstack((list(np.tile(np.array([A[i].std()]),5) for i in range(3))))
Achanged = (A-mean)/std
print(Achanged)
```

```
[[ 1.00538421 -0.31193427  0.11775893 -1.73770851  0.92649964]
 [-1.19175613 -0.71551647  0.58997342  1.62095056 -0.30365137]
 [-0.1551741  -0.12908259  1.90325059 -0.69476849 -0.92422541]]
```

1. Отсортируйте матрицу $A$ по 3-ому столбцу, т.е. поменяйте местами строки матрицы так, чтобы 3-й столбец оказался отсортированным. Используйте для этого slicing + argsort + indexing.

In [133…
```
index=np.argsort(A[:,2])
Asorted=A[index]
print(Asorted)
```

```
[[ 1.00538421 -0.31193427  0.11775893 -1.73770851  0.92649964]
 [-1.19175613 -0.71551647  0.58997342  1.62095056 -0.30365137]
 [-0.1551741  -0.12908259  1.90325059 -0.69476849 -0.92422541]]
```

1. Посчитайте две матрицы: $B = A^T A$ и $C = AA^T$. Что вы можете о них сказать? Какого они размера? В чём их сходство и отличие?

In [140…
```
B = A.T.dot(A)
C = A.dot(A.T)
print(B,C,sep='\n')
print('Sizes:\nB: {}, C:{}'.format(np.shape(B), np.shape(C)))
print('Determinants:\nB: {}, C:{}'.format(np.linalg.det(B), np.linalg.det(C)))
print('Traces:\nB: {}, C:{}'.format(np.trace(B), np.trace(C)))
```

```
[[ 2.45515909  0.55913763 -0.88004666 -3.57103239  1.43678233]
 [ 0.55913763  0.62592912 -0.70454526 -0.52808348  0.04756198]
 [-0.88004666 -0.70454526  3.98429861 -0.57063149 -1.8290752 ]
 [-3.57103239 -0.52808348 -0.57063149  6.12981483 -1.46006747]
 [ 1.43678233  0.04756198 -1.8290752  -1.46006747  1.80479835]]
[[ 5.         -4.00357652  0.45939105]
 [-4.00357652  5.          0.55461459]
 [ 0.45939105  0.55461459  5.        ]]
Sizes:
B: (5, 5), C:(3, 3)
Determinants:
B: 1.626400129023234e-30, C:40.223585676558585
Traces:
B: 15.000000000000002, C:15.000000000000002
```

## Отличие

- Имеют разные размеры из-за сущности умножения матриц
- Имеют разные определители

## Сходство

- Квадратные
- Симметричны относительно главной диагонали
- Имеют одинаковый след
- Имеют одинаковый ранг

1. Найдите сумму диагональных элементов матриц $B$ и $C$.

In [142…
```
print('Traces:\nB: {}, C:{}'.format(np.trace(B), np.trace(C)))
print('Sum = ', np.trace(B)+np.trace(C))
```

```
Traces:
B: 15.000000000000002, C:15.000000000000002
Sum =  30.000000000000004
```

1. Посчитайте детерминант и ранг матриц $A$, $B$ и $C$, посмотрите на спектр (набор собственных значений) матриц $B$ и $C$, какие выводы вы можете сделать?
(подсказка: используйте готовые функции из библиотеки np.linalg).

In [151…

```
print('{}:\nDeterminant does not exist\nRank = {}\n\n'.format(A,np.linalg.matrix_rank(A)))
for let in [B,C]:
    print('{}:\nDeterminant = {}\nRank = {}\nSpectre = {}\n\n'.format(let,np.linalg.det(let),
```

```
[[ 1.00538421 -0.31193427  0.11775893 -1.73770851  0.92649964]
 [-1.19175613 -0.71551647  0.58997342  1.62095056 -0.30365137]
 [-0.1551741  -0.12908259  1.90325059 -0.69476849 -0.92422541]]:
Determinant does not exist
Rank = 3


[[ 2.45515909  0.55913763 -0.88004666 -3.57103239  1.43678233]
 [ 0.55913763  0.62592912 -0.70454526 -0.52808348  0.04756198]
 [-0.88004666 -0.70454526  3.98429861 -0.57063149 -1.8290752 ]
 [-3.57103239 -0.52808348 -0.57063149  6.12981483 -1.46006747]
 [ 1.43678233  0.04756198 -1.8290752  -1.46006747  1.80479835]]:
Determinant = 1.626400129023234e-30
Rank = 3
Spectre = (array([-9.79787103e-16, -1.23849563e-16,  8.71869955e-01,  5.12340298e+00,
        9.00472706e+00]), array([[-0.79846366, -0.27642884,  0.11296947,  0.0657647 , -0.5186
0948],
       [ 0.36422234, -0.54187348,  0.74336456,  0.10895661, -0.09619154],
       [-0.01869042, -0.45445337, -0.17969838, -0.86363323,  0.1223476 ],
       [-0.36035114, -0.37645122, -0.04383723,  0.32654952,  0.78732156],
       [ 0.31558771, -0.53077005, -0.63279841,  0.3623624 , -0.29486813]]))


[[ 5.         -4.00357652  0.45939105]
 [-4.00357652  5.          0.55461459]
 [ 0.45939105  0.55461459  5.        ]]:
Determinant = 40.223585676558585
Rank = 3
Spectre = (array([0.87186996, 5.12340298, 9.00472706]), array([[-0.69567071,  0.13310763, -0.
70592111],
       [-0.69767489,  0.10893232,  0.70808438],
       [ 0.17114906,  0.985097  ,  0.01708483]]))
```

Основные характеристики матриц разные, кроме ранга. Детерминант матрицы A не существует, т.к. она не квадратная.

1. Численно посчитать определённый интеграл 3-мя методами:
   a) Прямоугольников $\int_a^b f(x)dx \approx \Delta x \sum_{k=0}^{N} f(x_k)$
   b) Трапеций $\int_a^b f(x)dx \approx \frac{\Delta x}{2}\left(f(x_0) + 2\sum_{k=1}^{N-1} f(x_k) + f(x_N)\right)$
   c) По правилу Симпсона $\int_a^b f(x)dx \approx \frac{\Delta x}{3}\left(f(x_0) + 2\sum_{k=1}^{N/2-1} f(x_{2k}) + 4\sum_{k=1}^{N/2} f(x_{2k-1}) + f(x_N)\right)$

В реализации этих методов цикл for для суммирования использовать нельзя.
Нужно посчитать значение интеграла с некоторым шагом интегрирования (например начать с 0.1), потом уменьшить его в десять раз и ещё раз посчитать и так до тех пор пока отличие в ответах будет в 5-м знаке после запятой, т.е. чтобы ошибка была меньше 1e-5. Получить ответы для 3-х разных методов и сделать соответствующие выводы.

Далее необходимо сравнить полученные оценки с аналитическим решением (т.е. формульным, которое вы должны сами посчитать на листочке и вбить формулу-ответ).

В качестве интеграла взять один из следующих с номером, сгенерированным случайным образом с помощью функции np.random.randint(10), в качестве seed для генератора случайных чисел взять номер своего студенческого билета.

0) $\int_3^4 \frac{x^2+3}{x-2}dx$ 1) $\int_{-2}^{-1} \frac{x+1}{x^3-x^2}dx$ 2) $\int_1^2 \frac{e^{1/x^2}}{x^3}dx$ 3) $\int_1^e \frac{\cos(\ln x)}{x}dx$ 4) $\int_1^e \frac{dx}{x(1+\ln^2 x)}$

5) $\int_0^{\pi/2} \cos^3\alpha\, d\alpha$ 6) $\int_0^{1/3} ch^2 3x\, dx$ 7) $\int_2^3 \frac{dy}{y^2-2y-8}dy$ 8) $\int_{3/4}^2 \frac{dx}{\sqrt{2+3x-2x^2}}$ 9) $\int_0^2 \frac{2x-1}{2x+1}dx$

In [153…
```
np.random.seed(1032216493)
num = np.random.randint(10)
print(num)
```

9

Решаем интеграл под номером 9

In [177…
```
delta=0.00001
x = np.append(np.arange(0,2,delta),2)
k = len(x)
def f(x):
    return (2*x-1)/(2*x+1)
```

Метод прямоугольников

In [178…
```
I1 = delta*np.sum(f(x))
print(I1)
```

```
0.39056008753389976
```

Метод трапеций

In [179…
```
I2 = (delta/2)*(f(x[0])+2*np.sum(f(x[1:]))+f(x[-1]))
print(I2)
```

```
0.39056808753389977
```

По правилу Симпсона

In [180…
```
I3 = (delta/3)*(f(x[0])+2*np.sum(f(x[2:-3:2]))+4*np.sum(f(x[1:-2:2]))+f(x[-1]))
print(I3)
```

```
0.3905500876085667
```

Аналитическое решение: первообразная F(x) = x - ln|2x+1| + C, отсюда искомый интеграл I = 2-ln(5), что приблизительно равно 0,39056209