МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №3.2

Основы работы с библиотекой NumPy.

по дисциплине «Анализ данных»

Выполнил студент группы ИВТ-б-о-21-1

Лысенко И.А. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2023

Цель работы: исследовать базовые возможности библиотеки NumPy языка программирования Python.

Ход работы:

1. Создал репозиторий на GitHub: https://github.com/IsSveshuD/lab_3.2.git .

2. Проработал примеры:

```
In [2]: import numpy as np
        m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
        print(m)

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

In [5]: print(m[1,0])

        5

In [7]: print(m[1, :])

        [[5 6 7 8]]

In [8]: print(m[:, 2])

        [[3]
         [7]
         [5]]

In [9]: print(m[1, 2:])

        [[7 8]]

In [10]: cols = [0, 1, 3]
         print(m[:, cols])

         [[1 2 4]
          [5 6 8]
          [9 1 7]]

In [20]: m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
         print(m)

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

In [21]: type(m)

Out[21]: numpy.matrix

In [22]: m = np.array(m)
         type(m)
```

Out[22]:
```
numpy.ndarray
```

In [23]:
```
print(m.shape)
```
```
(3, 4)
```

In [24]:
```
print(m.max())
```
```
9
```

In [25]:
```
print(m.max())
```
```
9
```

In [26]:
```
print(m.max(axis=1))
```
```
[4 8 9]
```

In [27]:
```
print(m.max(axis=0))
```
```
[9 6 7 8]
```

In [28]:
```
m.mean()
```

Out[28]:
```
4.833333333333333
```

In [30]:
```
m.mean(axis=1)
```

Out[30]:
```
array([2.5, 6.5, 5.5])
```

In [31]:
```
m.sum()
```

Out[31]:
```
58
```

In [32]:
```
m.sum(axis=0)
```

Out[32]:
```
array([15,  9, 15, 19])
```

In [33]:
```
nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
letters = np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])
a = True
b = 5 > 7
print(b)
```
```
False
```

In [34]:
```
less_then_5 = nums < 5
less_then_5
```

Out[34]:
```
array([ True,  True,  True,  True, False, False, False, False, False,
       False])
```

In [35]:
```
pos_a = letters == 'a'
pos_a
```

Out[35]:
```
array([ True, False, False, False,  True, False, False])
```

In [36]:
```
nums[less_then_5]
```

Out[36]:
```
array([1, 2, 3, 4])
```

In [37]:
```
mod_m = np.logical_and(m>=3, m<=7)
mod_m
```

Out[37]:
```
array([[False, False,  True,  True],
       [ True,  True,  True, False],
       [False, False,  True,  True]])
```

In [38]:
```
m[mod_m]
```

Out[38]:
```
array([3, 4, 5, 6, 7, 5, 7])
```

In [39]:
```
nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
nums[nums < 5]
```

Out[39]:
```
array([1, 2, 3, 4])
```

In [40]:
```
nums[nums < 5] = 10
print(nums)
```
```
[10 10 10 10  5  6  7  8  9 10]
```

In [41]:
```
m[m > 7] = 25
print(m)
```
```
[[ 1  2  3  4]
 [ 5  6  7 25]
 [25  1  5  7]]
```

In [42]:
```
np.arange(10)
```

Out[42]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [43]:
```
np.arange(5, 12)
```

Out[43]:
```
array([ 5,  6,  7,  8,  9, 10, 11])
```

In [44]:
```
np.arange(1, 5, 0.5)
```

Out[44]:
```
array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
```

In [45]:
```
a = [[1, 2], [3, 4]]
np.matrix(a)
```

Out[45]:
```
matrix([[1, 2],
        [3, 4]])
```

In [46]:
```
b = np.array([[5, 6], [7, 8]])
np.matrix(b)
```

Out[46]:
```
matrix([[5, 6],
        [7, 8]])
```

In [47]:
```
np.matrix('[1, 2; 3, 4]')
```

Out[47]:
```
matrix([[1, 2],
        [3, 4]])
```

In [48]:
```
np.zeros((3, 4))
```

Out[48]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [49]:
```
np.eye(3)
```

Out[49]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [50]:
```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
A
```

Out[50]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [51]:
```
np.ravel(A)
```

Out[51]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [52]:
```
np.ravel(A, order='C')
```

Out[52]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [53]:
```
np.ravel(A, order='F')
```

Out[53]:
```
array([1, 4, 7, 2, 5, 8, 3, 6, 9])
```

In [54]:
```
a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
np.where(a % 2 == 0, a * 10, a / 10)
```

Out[54]:
```
array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])
```

In [55]:
```
a = np.random.rand(10)
a
```

Out[55]:
```
array([0.27478754, 0.23511538, 0.29925796, 0.10292996, 0.51583171,
       0.93412616, 0.63478666, 0.04978629, 0.35405863, 0.10207794])
```

In [56]:
```
np.where(a > 0.5, True, False)
```

Out[56]:
```
array([False, False, False, False,  True,  True,  True, False, False,
       False])
```

In [57]:
```
np.where(a > 0.5, 1, -1)
```

Out[57]:
```
array([-1, -1, -1, -1,  1,  1,  1, -1, -1, -1])
```

In [58]:
```
x = np.linspace(0, 1, 5)
x
```

Out[58]:
```
array([0.  , 0.25, 0.5 , 0.75, 1.  ])
```

In [59]:
```
y = np.linspace(0, 2, 5)
y
```

Out[59]:
```
array([0. , 0.5, 1. , 1.5, 2. ])
```

In [60]:
```
xg, yg = np.meshgrid(x, y)
xg
```

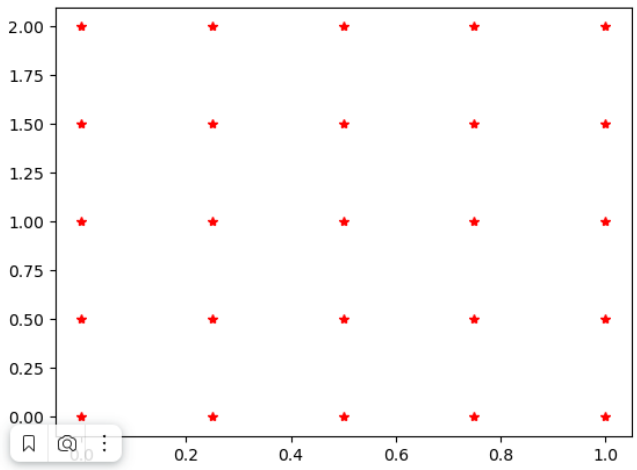
Out[60]:
```
array([[0.  , 0.25, 0.5 , 0.75, 1.  ],
       [0.  , 0.25, 0.5 , 0.75, 1.  ],
       [0.  , 0.25, 0.5 , 0.75, 1.  ],
       [0.  , 0.25, 0.5 , 0.75, 1.  ],
       [0.  , 0.25, 0.5 , 0.75, 1.  ]])
```

In [61]:
```
yg
```

Out[61]:
```
array([[0. , 0. , 0. , 0. , 0. ],
       [0.5, 0.5, 0.5, 0.5, 0.5],
       [1. , 1. , 1. , 1. , 1. ],
       [1.5, 1.5, 1.5, 1.5, 1.5],
       [2. , 2. , 2. , 2. , 2. ]])
```

In [62]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(xg, yg, color="r", marker="*", linestyle="none")
```

Out[62]:
```
[<matplotlib.lines.Line2D at 0x17720425000>,
 <matplotlib.lines.Line2D at 0x177204250c0>,
 <matplotlib.lines.Line2D at 0x177204251b0>,
 <matplotlib.lines.Line2D at 0x177204252a0>,
 <matplotlib.lines.Line2D at 0x17720425390>]
```



In [63]:
```
np.random.permutation(7)
```

Out[63]:
```
array([5, 0, 4, 3, 1, 6, 2])
```

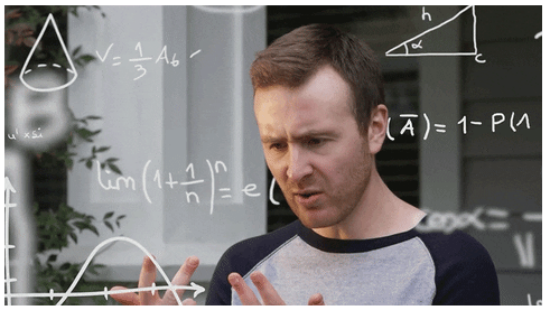
```
In [64]: a = ['a', 'b', 'c', 'd', 'e']
         np.random.permutation(a)
Out[64]: array(['c', 'a', 'b', 'e', 'd'], dtype='<U1')

In [65]: arr = np.linspace(0, 10, 5)
         arr
Out[65]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])

In [66]: arr_mix = np.random.permutation(arr)
         arr_mix
Out[66]: array([10. ,  0. ,  7.5,  5. ,  2.5])

In [67]: index_mix = np.random.permutation(len(arr_mix))
         index_mix
Out[67]: array([4, 0, 2, 1, 3])

In [68]: arr[index_mix]
Out[68]: array([10. ,  0. ,  5. ,  2.5,  7.5])
```

Рисунок 1 – Примеры

3. Решил заданиях в ноутбуках, выданных преподавателем.

Лабораторная работа 3.2. Знакомство с NumPy

Библиотека NumPy -- быстрая библиотека для математики в Python, основная структура данных -- массив `numpy.array`:



```
In [1]: # подключение модуля numpy под именем np
        import numpy as np

In [2]: # основная структура данных - массив
        a = np.array([1, 2, 3, 4, 5])
        b = np.array([0.1, 0.2, 0.3, 0.4, 0.5])

        print("a =", a)
        print("b =", b)

        a = [1 2 3 4 5]
        b = [0.1 0.2 0.3 0.4 0.5]
```

Создайте массив с 5 любыми числами:

```
In [4]: m = np.array([1, 2, 1, 2, 1])
        print(m)

        [1 2 1 2 1]
```

Арифметические операции, в отличие от операций над списками, применяются поэлементно:

In [5]:
```
list1 = [1, 2, 3]
array1 = np.array([1, 2, 3])

print("list1:", list1)
print('\tlist1 * 3:', list1 * 3)
print('\tlist1 + [1]:', list1 + [1])

print('array1:', array1)
print('\tarray1 * 3:', array1 * 3)
print('\tarray1 + 1:', array1 + 1)
```

```
list1: [1, 2, 3]
        list1 * 3: [1, 2, 3, 1, 2, 3, 1, 2, 3]
        list1 + [1]: [1, 2, 3, 1]
array1: [1 2 3]
        array1 * 3: [3 6 9]
        array1 + 1: [2 3 4]
```

Создайте массив из 5 чисел. Возведите каждый элемент массива в степень 3

In [9]:
```
import math
m = np.array([1, 2, 3, 4, 5])
print('m:', m)
print('\tm ^ 3:', m**3)
```

```
m: [1 2 3 4 5]
        m ^ 3: [  1   8  27  64 125]
```

Если в операции участвуют 2 массива (по умолчанию -- одинакового размера), операции считаются для соответствующих пар:

In [ ]:
```
print("a + b =", a + b)
print("a * b =", a * b)
```

In [ ]:
```
# вот это разность
print("a - b =", a - b)

# вот это деление
print("a / b =", a / b)

# вот это целочисленное деление
print("a // b =", a // b)

# вот это квадрат
print("a ** 2 =", a ** 2)
```

Создайте два массива одинаковой длины. Выведите массив, полученный делением одного массива на другой.

In [12]:
```
a = np.array([5, 10, 15, 20, 25])
b = np.array([1, 2, 3, 4, 5])
print("a / b = ", a / b)
```

```
a / b =  [5. 5. 5. 5. 5.]
```

Л — логика

К элементам массива можно применять логические операции.

Возвращаемое значение -- массив, содержащий результаты вычислений для каждого элемента (`True` -- "да" или `False` -- "нет"):

In [15]:
```
print("a =", a)
print("\ta > 1: ", a > 1)
print("\nb =", b)
print("\tb < 0.5:  ", b < 0.5)

print("\nОдновременная проверка условий:")
print("\t(a > 1) & (b < 0.5):  ", (a>1) & (b < 0.5))
print("А вот это проверяет, что a > 1 ИЛИ b < 0.5:  ", (a > 1) | (b < 0.5))
```

```
a = [1 2 3 4 5]
        a > 1:  [False  True  True  True  True]

b = [ 3  6  9 12 15]
        b < 0.5:   [False False False False False]

Одновременная проверка условий:
        (a > 1) & (b < 0.5):   [False False False False False]
А вот это проверяет, что a > 1 ИЛИ b < 0.5:   [False  True  True  True  True]
```

Создайте 2 массива из 5 элементов. Проверьте условие "Элементы первого массива меньше 6, элементы второго массива делятся на 3"

In [22]:
```
a = np.array([1, 2, 3, 4, 5])
b = np.array([3, 6, 9, 12, 15])
print("\t(a < 6 ) & (b // 3):", (a<6) & (b%3==0))
```

```
        (a < 6 ) & (b // 3): [ True  True  True  True  True]
```

Теперь проверьте условие "Элементы первого массива делятся на 2 или элементы второго массива больше 2"

In [23]:
```
print("\t(a / 2) или (b > 2)", (a%2==0) | (b > 2))
```

```
        (a / 2) или (b > 2) [ True  True  True  True  True]
```

Зачем это нужно? Чтобы выбирать элементы массива, удовлетворяющие какому-нибудь условию:

In [48]:
```
print("a =", a)
print("a > 2:", a > 2)
# индексация - выбираем элементы из массива в тех позициях, где True
print("a[a > 2]:", a[a > 2])
```

```
a = [1 2 3 4 5]
a > 2: [False False  True  True  True]
a[a > 2]: [3 4 5]
```

Создайте массив с элементами от 1 до 20. Выведите все элементы, которые больше 5 и не делятся на 2

Подсказка: создать массив можно с помощью функции np.arange(), действие которой аналогично функции range, которую вы уже знаете.

In [73]:
```
m = np.arange(20)
print("m[m > 5] & m[m%2!=0]", m[(m > 5) & (m%2==0)])
```

```
m[m > 5] & m[m%2!=0] [ 6  8 10 12 14 16 18]
```

А ещё NumPy умеет...

Все операции NumPy оптимизированы для быстрых вычислений над целыми массивами чисел и в методах `np.array` реализовано множество функций, которые могут вам понадобиться:

In [ ]:
```
# теперь можно считать средний размер котиков в одну строку!
print("np.mean(a) =", np.mean(a))
# минимальный элемент
print("np.min(a) =", np.min(a))
# индекс минимального элемента
print("np.argmin(a) =", np.argmin(a))
# вывести значения массива без дубликатов
print("np.unique(['male', 'male', 'female', 'female', 'male']) =", np.unique(['male', 'male', 'female', 'female', 'male']))

# и ещё много всяких методов
# Google в помощь
```

Пора еще немного потренироваться с NumPy.

Выполните операции, перечисленные ниже:

In [51]:
```
print("Разность между a и b:", a - b
      )
print("Квадраты элементов b:", b**2
      )
print("Половины произведений элементов массивов a и b:", (a * b)/2
      )

print()
print("Максимальный элемент b:", np.max(b)
      )
print("Сумма элементов массива b:", np.sum(b)
      )
print("Индекс максимального элемента b:", np.argmax(b)
      )
```

```
Разность между a и b: [ -2  -4  -6  -8 -10]
Квадраты элементов b: [  9  36  81 144 225]
Половины произведений элементов массивов a и b: [ 1.5  6.  13.5 24.  37.5]

Максимальный элемент b: 15
Сумма элементов массива b: 45
Индекс максимального элемента b: 4
```

Задайте два массива: [5, 2, 3, 12, 4, 5] и ['f', 'o', 'o', 'b', 'a', 'r']

Выведите буквы из второго массива, индексы которых соответствуют индексам чисел из первого массива, которые больше 1, меньше 5 и делятся на 2

In [74]:
```
a = np.array([5, 2, 3, 12, 4, 5])
b = np.array(['f', 'o', 'o', 'b', 'a', 'r'])
print(b[(a > 1) & (a < 5) & (a%2==0)])
```

```
['o' 'a']
```

Лабораторная работа 3.2. Домашнее задание

Задание №1

Создайте два массива: в первом должны быть четные числа от 2 до 12 включительно, а в другом числа 7, 11, 15, 18, 23, 29.

1. Сложите массивы и возведите элементы получившегося массива в квадрат:

In [3]:
```
import numpy as np
a = np.array([2, 4, 6, 8, 10, 12])
b = np.array([7, 11, 15, 18, 23, 29])
print("(a + b)^2:", (a+b)**2)
```

```
(a + b)^2: [  81  225  441  676 1089 1681]
```

2. Выведите все элементы из первого массива, индексы которых соответствуют индексам тех элементов второго массива, которые больше 12 и дают остаток 3 при делении на 5.

In [4]:
```
print(a[(b > 12) & (b%5==3)])
```

```
[ 8 10]
```

3. Проверьте условие "Элементы первого массива делятся на 4, элементы второго массива меньше 14". (Подсказка: в результате должен получиться массив с True и False)

In [7]:
```
print((a%4==0) & (b<14))
```

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

Задание №2

- Найдите интересный для вас датасет. Например, можно выбрать датасет тут. http://data.un.org/Explorer.aspx (выбираете датасет, жмете на view data, потом download, выбирайте csv формат)
- Рассчитайте подходящие описательные статистики для признаков объектов в выбранном датасете
- Проанализируйте и прокомментируйте содержательно получившиеся результаты
- Все комментарии оформляйте строго в ячейках формата markdown

Для выполнения задания был выбран датасет, по прогнозируемому числу жителей в России до 2100 года.

In [84]:
```
import csv
import numpy as np

with open('human.csv', newline='') as file:
    reader = csv.DictReader(file)
    data = [row for row in reader]

value = np.array([float(row['Value']) for row in data])
variant = np.array([str(row['Variant']) for row in data])

print(f"Описательные статистики общего числа жителей")
print(f"\nМаксимальное число жителей: ",np.max(value[(variant == "High")]))
print(f"Среднее число жителей: ",round(np.mean(value[(variant == "High")]), 3))
print(f"Минимальное число жителей: ",np.min(value[(variant == "High")]))
print(f"Среднее отклонение числа жителей: ",round(np.std(value[(variant == "High")]), 3))

print(f"\nОписательные статистики рождаемости")
print(f"\nМаксимальная рождаемость: ",np.max(value[(variant == "Constant fertility")]))
print(f"Средняя постоянная рожаемость: ",round(np.mean(value[(variant == "Constant fertility")]), 3))
print(f"Минимальная рождаемость: ",np.min(value[(variant == "Constant fertility")]))
print(f"Среднее отклонение рождаемости: ",round(np.std(value[(variant == "Constant fertility")]), 3))

print(f"\nОписательные статистики смертности")
print(f"\nМаксимальная смертность: ",np.max(value[(variant == "Constant mortality")]))
print(f"Средняя постоянная смертность: ",round(np.mean(value[(variant == "Constant mortality")]), 3))
print(f"Минимальная смертность: ",np.min(value[(variant == "Constant mortality")]))
print(f"Среднее отклонение смертности: ",round(np.std(value[(variant == "Constant mortality")]), 3))
```

```
Описательные статистики общего числа жителей

Максимальное число жителей:  162441.155
Среднее число жителей:  147337.055
Минимальное число жителей:  143258.068
Среднее отклонение числа жителей:  5393.453

Описательные статистики рождаемости

Максимальная рождаемость:  144713.314
Средняя постоянная рожаемость:  120630.861
Минимальная рождаемость:  94000.181
Среднее отклонение рождаемости:  15702.513

Описательные статистики смертности

Максимальная смертность:  144713.314
Средняя постоянная смертность:  112912.371
Минимальная смертность:  89460.587
Среднее отклонение смертности:  16587.278
```

Максимальное количество жителей в России за 100 лет достигнет 162 441 155 человек.

Максимальная рождаемость равна максимальной смертности. отсюда можно сделать вывод, что есть промежуток времени, когда либо смертность будет превышать рождаемость, либо наоборот.

Среднее отклонение числа жителей равна 5 393 453 человека. Среднее отклонение рождаемости 15 702 513 человек. Среднее отклонение смертности 16 587 278 человек.

Рисунок 2 – Задания от преподавателя

4. Выполнил индивидуальное задание 1:

Уплотнить заданную матрицу, удаляя из нее строки и столбцы, заполненные нулями. Найти номер первой из строк, содержащих хотя бы один положительный элемент.

In [42]:
```
import numpy as np
m = np.array([[0, 0, 1, 1, 0, 0, 1, 0, 1, 0],
              [0, 0, 1, 0, 1, 0, 1, 0, 1, 0],
              [0, 0, 1, 0, 1, 0, 1, 0, 1, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 1, 0, 1, 0, 1, 0, 1, 0],
              [0, 0, 1, 0, 1, 0, 1, 0, 1, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 1, 0, 1, 0, 1, 0, 1, 0],
              [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 1, 0, 1, 0, 1, 0, 1, 0]])
idx = np.all(m==0, axis=0)
a = np.delete(m, np.nonzero(idx), axis=1)
idx = np.all(a==0, axis=1)
a = np.delete(a, np.nonzero(idx), axis=0)
print(a)
b = np.where(m > 0)[0][0]
print("Строка вхождения первого положительного элемента: ", b+1)
```

```
[[1 1 0 1 1]
 [1 0 1 1 1]
 [1 0 1 1 1]
 [1 0 1 1 1]
 [1 0 1 1 1]
 [1 0 1 1 1]
 [1 0 0 0 0]
 [1 0 1 1 1]]
Строка вхождения первого положительного элемента:  1
```

Рисунок 3 – Индивидуальное задание 1.

5. Выполнил индивидуальное задание 2:

В билете 3 задачи. Вероятность правильного решения первой задачи равна 0,9, второй – 0,8, третьей – 0,7. Составить и изобразить графически закон распределения числа правильного решения задач в билете и вычислить математическое ожидание этой случайной величины

In [17]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import binom


Pa = 0.9
Pb = 0.8
Pc = 0.7
P0 = (1 - Pa) * (1 - Pb) * (1 - Pc)
P1 = (Pa * (1 - Pb) * (1 - Pc)) + ((1 - Pa) * Pb * (1 - Pc)) + ((1 - Pa) * (1 - Pb) * Pc)
P2 = ((1 - Pa) * Pb * Pc) + (Pa * (1 - Pb) * Pc) + (Pa * Pb * (1 - Pc))
P3 = Pa * Pb * Pc

p = np.array([0, 1, 2, 3])
x = np.array([P0, P1, P2, P3])

print("Математическое ожидание: ", np.sum(p * x))
print("Дисперсия: ", round((np.sum((p**2) * x) - np.sum(p * x)**2), 2))
plt.plot(x , p)
plt.ylabel('P(x)')
plt.xlabel('x')
plt.show()
```

```
Математическое ожидание:  2.4
Дисперсия:  0.46
```

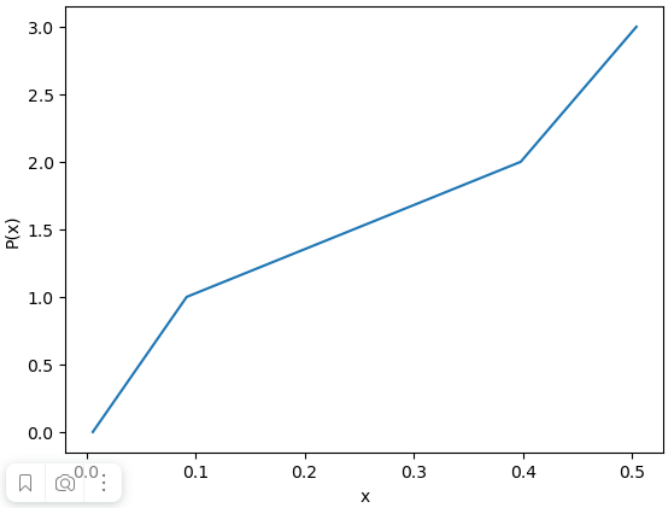


Рисунок 3 – Индивидуальное задание 2.

Ответы на вопросы:

1. Каково назначение библиотеки NumPy?

Numpy – это библиотека для языка программирования Python, которая предоставляет в распоряжение разработчика инструменты для эффективно работы с многомерными массивами и высокопроизводительные вычисли тельные алгоритмы.

2. Что такое массивы ndarray?

Ndarray — это (обычно фиксированный размер) многомерный контей нер элементов одного типа и размера. Количество измерений и элементов в массиве определяется его формой, которая является кортежем из N натураль ных чисел, которые определяют размеры каждого измерения.

3. Как осуществляется доступ к частям многомерного массива? Через срезы:

− Произвольный элемент (m[i,j])

− Строка (m[i, :])

− Столбец матрицы (m[:, j])

− Часть строки/столбца матрицы (m[i, j:], m[0:i, j])

− Непрерывная часть матрицы (m[i1:i2, j1:j2])

− Произвольные столбцы/строки матрицы (col = [0, 1, 2]; m[:, col])

4. Как осуществляется расчет статистик по данным?

shape – Размерность массива

argmax – Индексы элементов с максимальным значением (по осям) argmin – Индексы элементов с минимальным значением (по осям) max – Максимальные значения элементов (по осям)

min – Минимальные значения элементов (по осям)

mean – Средние значения элементов (по осям)

prod – Произведение всех элементов (по осям)

std – Стандартное отклонение (по осям)

sum – Сумма всех элементов (по осям)

var – Дисперсия (по осям)

5. Как выполняется выборка данных из массивов ndarray?

Если мы переменную, содержащую boolean-значение передадим в каче стве списка индексов для массива (nums), то получим массив, в котором бу дут содержаться элементы из nums с индексами равными индексам True по зиций boolean-массива, графически это будет выглядеть так.

Вывод: в результате выполнения лабораторной работы были получены были базовые навыки работы с библиотекой NumPy языка программирования Python.