1. Импортировать NumPy под именем np

**import** **numpy** **as** **np**

1. Напечатать версию и конфигурацию
2. print(np.\_\_version\_\_)

np.show\_config()

1. Создать вектор (одномерный массив) размера 10, заполненный нулями
2. Z = np.zeros(10)

print(Z)

1. Создать вектор размера 10, заполненный единицами
2. Z = np.ones(10)

print(Z)

1. Создать вектор размера 10, заполненный числом 2.5
2. Z = np.full(10, 2.5)

print(Z)

1. Как получить документацию о функции numpy.add из командной строки?

python3 -c "import numpy; numpy.info(numpy.add)"

1. Создать вектор размера 10, заполненный нулями, но пятый элемент равен 1
2. Z = np.zeros(10)
3. Z[4] = 1

print(Z)

1. Создать вектор со значениями от 10 до 49
2. Z = np.arange(10,50)

print(Z)

1. Развернуть вектор (первый становится последним)
2. Z = np.arange(50)

Z = Z[::-1]

1. Создать матрицу (двумерный массив) 3x3 со значениями от 0 до 8
2. Z = np.arange(9).reshape(3,3)

print(Z)

1. Найти индексы ненулевых элементов в [1,2,0,0,4,0]
2. nz = np.nonzero([1,2,0,0,4,0])

print(nz)

1. Создать 3x3 единичную матрицу
2. Z = np.eye(3)

print(Z)

1. Создать массив 3x3x3 со случайными значениями
2. Z = np.random.random((3,3,3))

print(Z)

1. Создать массив 10x10 со случайными значениями, найти минимум и максимум
2. Z = np.random.random((10,10))
3. Zmin, Zmax = Z.min(), Z.max()

print(Zmin, Zmax)

1. Создать случайный вектор размера 30 и найти среднее значение всех элементов
2. Z = np.random.random(30)
3. m = Z.mean()

print(m)

1. Создать матрицу с 0 внутри, и 1 на границах
2. Z = np.ones((10,10))

Z[1:-1,1:-1] = 0

1. Выяснить результат следующих выражений
2. 0 \* np.nan
3. np.nan == np.nan
4. np.inf > np.nan
5. np.nan - np.nan

0.3 == 3 \* 0.1

1. Создать 5x5 матрицу с 1,2,3,4 под диагональю
2. Z = np.diag(np.arange(1, 5), k=-1)

print(Z)

1. Создать 8x8 матрицу и заполнить её в шахматном порядке
2. Z = np.zeros((8,8), dtype=int)
3. Z[1::2,::2] = 1
4. Z[::2,1::2] = 1

print(Z)

1. Дан массив размерности (6,7,8). Каков индекс (x,y,z) сотого элемента?

print(np.unravel\_index(100, (6,7,8)))

1. Создать 8x8 матрицу и заполнить её в шахматном порядке, используя функцию tile
2. Z = np.tile(np.array([[0,1],[1,0]]), (4,4))

print(Z)

1. Перемножить матрицы 5x3 и 3x2
2. Z = np.dot(np.ones((5,3)), np.ones((3,2)))

print(Z)

1. Дан массив, поменять знак у элементов, значения которых между 3 и 8
2. Z = np.arange(11)

Z[(3 < Z) & (Z <= 8)] \*= -1

1. Создать 5x5 матрицу со значениями в строках от 0 до 4
2. Z = np.zeros((5,5))
3. Z += np.arange(5)

print(Z)

1. Есть генератор, сделать с его помощью массив
2. **def** generate():
3. **for** x **in** xrange(10):
4. **yield** x
5. Z = np.fromiter(generate(),dtype=float,count=-1)

print(Z)

1. Создать вектор размера 10 со значениями от 0 до 1, не включая ни то, ни другое
2. Z = np.linspace(0,1,12)[1:-1]

print(Z)

1. Отсортировать вектор
2. Z = np.random.random(10)
3. Z.sort()

print(Z)

1. Проверить, одинаковы ли 2 numpy массива
2. A = np.random.randint(0,2,5)
3. B = np.random.randint(0,2,5)
4. equal = np.allclose(A,B)

print(equal)

1. Сделать массив неизменяемым
2. Z = np.zeros(10)
3. Z.flags.writeable = **False**

Z[0] = 1

1. Дан массив 10x2 (точки в декартовой системе координат), преобразовать в полярную
2. Z = np.random.random((10,2))
3. X,Y = Z[:,0], Z[:,1]
4. R = np.hypot(X, Y)
5. T = np.arctan2(Y,X)
6. print(R)

print(T)

1. Заменить максимальный элемент на ноль
2. Z = np.random.random(10)
3. Z[Z.argmax()] = 0

print(Z)

1. Создать структурированный массив с координатами x, y на сетке в квадрате [0,1]x[0,1]
2. Z = np.zeros((10,10), [('x',float),('y',float)])
3. Z['x'], Z['y'] = np.meshgrid(np.linspace(0,1,10),
4. np.linspace(0,1,10))

print(Z)

1. Из двух массивов сделать матрицу Коши C (Cij = 1/(xi - yj))
2. X = np.arange(8)
3. Y = X + 0.5
4. C = 1.0 / np.subtract.outer(X, Y)

print(np.linalg.det(C))

1. Найти минимальное и максимальное значение, принимаемое каждым числовым типом numpy
2. **for** dtype **in** [np.int8, np.int32, np.int64]:
3. print(np.iinfo(dtype).min)
4. print(np.iinfo(dtype).max)
5. **for** dtype **in** [np.float32, np.float64]:
6. print(np.finfo(dtype).min)
7. print(np.finfo(dtype).max)

 print(np.finfo(dtype).eps)

1. Напечатать **все** значения в массиве
2. np.set\_printoptions(threshold=np.nan)
3. Z = np.zeros((25,25))

print(Z)

1. Найти ближайшее к заданному значению число в заданном массиве
2. Z = np.arange(100)
3. v = np.random.uniform(0,100)
4. index = (np.abs(Z-v)).argmin()

print(Z[index])

1. Создать структурированный массив, представляющий координату (x,y) и цвет (r,g,b)
2. Z = np.zeros(10, [ ('position', [ ('x', float, 1),
3. ('y', float, 1)]),
4. ('color', [ ('r', float, 1),
5. ('g', float, 1),
6. ('b', float, 1)])])

print(Z)

1. Дан массив (100,2) координат, найти расстояние от каждой точки до каждой
2. **import** **scipy.spatial**
3. Z = np.random.random((10,2))
4. D = scipy.spatial.distance.cdist(Z,Z)

print(D)

1. Преобразовать массив из float в int
2. Z = np.arange(10, dtype=np.int32)

Z = Z.astype(np.float32, copy=**False**)

1. Дан файл:
2. 1,2,3,4,5
3. 6,,,7,8
4. ,,9,10,11

Как прочитать его?

Z = np.genfromtxt("missing.dat", delimiter=",")

1. Каков эквивалент функции enumerate для numpy массивов?
2. Z = np.arange(9).reshape(3,3)
3. **for** index, value **in** np.ndenumerate(Z):
4. print(index, value)
5. **for** index **in** np.ndindex(Z.shape):

 print(index, Z[index])

1. Сформировать 2D массив с распределением Гаусса
2. X, Y = np.meshgrid(np.linspace(-1,1,10), np.linspace(-1,1,10))
3. D = np.hypot(X, Y)
4. sigma, mu = 1.0, 0.0
5. G = np.exp(-((D - mu) \*\* 2 / (2.0 \* sigma \*\* 2)))

print(G)

1. Случайно расположить p элементов в 2D массив
2. n = 10
3. p = 3
4. Z = np.zeros((n,n))

np.put(Z, np.random.choice(range(n\*n), p, replace=**False**), 1)

1. Отнять среднее из каждой строки в матрице
2. X = np.random.rand(5, 10)

Y = X - X.mean(axis=1, keepdims=**True**)

1. Отсортировать матрицу по n-ому столбцу
2. Z = np.random.randint(0,10,(3,3))
3. n = 1 *# Нумерация с нуля*
4. print(Z)

print(Z[Z[:,n].argsort()])

1. Определить, есть ли в 2D массиве нулевые столбцы
2. Z = np.random.randint(0,3,(3,10))

print((~Z.any(axis=0)).any())

1. Дан массив, добавить 1 к каждому элементу с индексом, заданным в другом массиве (осторожно с повторами)
2. Z = np.ones(10)
3. I = np.random.randint(0,len(Z),20)
4. Z += np.bincount(I, minlength=len(Z))

print(Z)

1. Дан массив (w,h,3) (картинка) dtype=ubyte, посчитать количество различных цветов
2. w,h = 16,16
3. I = np.random.randint(0, 2, (h,w,3)).astype(np.ubyte)
4. F = I[...,0] \* 256 \* 256 + I[...,1] \* 256 + I[...,2]
5. n = len(np.unique(F))

print(np.unique(I))

1. Дан четырехмерный массив, посчитать сумму по последним двум осям
2. A = np.random.randint(0,10, (3,4,3,4))
3. sum = A.reshape(A.shape[:-2] + (-1,)).sum(axis=-1)

print(sum)

1. Найти диагональные элементы произведения матриц
2. *# Slow version*
3. np.diag(np.dot(A, B))
4. *# Fast version*
5. np.sum(A \* B.T, axis=1)
6. *# Faster version*

np.einsum("ij,ji->i", A, B).

1. Дан вектор [1, 2, 3, 4, 5], построить новый вектор с тремя нулями между каждым значением
2. Z = np.array([1,2,3,4,5])
3. nz = 3
4. Z0 = np.zeros(len(Z) + (len(Z)-1)\*(nz))
5. Z0[::nz+1] = Z

print(Z0)

1. Поменять 2 строки в матрице
2. A = np.arange(25).reshape(5,5)
3. A[[0,1]] = A[[1,0]]

print(A)

1. Рассмотрим набор из 10 троек, описывающих 10 треугольников (с общими вершинами), найти множество уникальных отрезков, составляющих все треугольники
2. faces = np.random.randint(0,100,(10,3))
3. F = np.roll(faces.repeat(2,axis=1),-1,axis=1)
4. F = F.reshape(len(F)\*3,2)
5. F = np.sort(F,axis=1)
6. G = F.view( dtype=[('p0',F.dtype),('p1',F.dtype)] )
7. G = np.unique(G)

print(G)

1. Дан массив C; создать массив A, что np.bincount(A) == C
2. C = np.bincount([1,1,2,3,4,4,6])
3. A = np.repeat(np.arange(len(C)), C)

print(A)

1. Посчитать среднее, используя плавающее окно
2. **def** moving\_average(a, n=3):
3. ret = np.cumsum(a, dtype=float)
4. ret[n:] = ret[n:] - ret[:-n]
5. **return** ret[n - 1:] / n

print(moving\_average(np.arange(20), 3))

1. Дан вектор Z, построить матрицу, первая строка которой (Z[0],Z[1],Z[2]), каждая последующая сдвинута на 1 (последняя (Z[-3],Z[-2],Z[-1]))
2. **from** **numpy.lib** **import** stride\_tricks
3. **def** rolling(a, window):
4. shape = (a.size - window + 1, window)
5. strides = (a.itemsize, a.itemsize)
6. **return** stride\_tricks.as\_strided(a, shape=shape, strides=strides)
7. Z = rolling(np.arange(10), 3)

print(Z)

1. Инвертировать булево значение, или поменять знак у числового массива без создания нового
2. Z = np.random.randint(0,2,100)
3. np.logical\_not(arr, out=arr)
4. Z = np.random.uniform(-1.0,1.0,100)

np.negative(arr, out=arr)

1. Рассмотрим 2 набора точек P0, P1 описания линии (2D) и точку р, как вычислить расстояние от р до каждой линии i (P0[i],P1[i])
2. **def** distance(P0, P1, p):
3. T = P1 - P0
4. L = (T\*\*2).sum(axis=1)
5. U = -((P0[:,0] - p[...,0]) \* T[:,0] + (P0[:,1] - p[...,1]) \* T[:,1]) / L
6. U = U.reshape(len(U),1)
7. D = P0 + U \* T - p
8. **return** np.sqrt((D\*\*2).sum(axis=1))
9. P0 = np.random.uniform(-10,10,(10,2))
10. P1 = np.random.uniform(-10,10,(10,2))
11. p = np.random.uniform(-10,10,( 1,2))

print(distance(P0, P1, p))

1. Дан массив. Написать функцию, выделяющую часть массива фиксированного размера с центром в данном элементе (дополненное значением fill если необходимо)
2. Z = np.random.randint(0,10, (10,10))
3. shape = (5,5)
4. fill = 0
5. position = (1,1)
6. R = np.ones(shape, dtype=Z.dtype)\*fill
7. P = np.array(list(position)).astype(int)
8. Rs = np.array(list(R.shape)).astype(int)
9. Zs = np.array(list(Z.shape)).astype(int)
10. R\_start = np.zeros((len(shape),)).astype(int)
11. R\_stop = np.array(list(shape)).astype(int)
12. Z\_start = (P - Rs//2)
13. Z\_stop = (P + Rs//2)+Rs%2
14. R\_start = (R\_start - np.minimum(Z\_start, 0)).tolist()
15. Z\_start = (np.maximum(Z\_start, 0)).tolist()
16. R\_stop = np.maximum(R\_start, (R\_stop - np.maximum(Z\_stop-Zs,0))).tolist()
17. Z\_stop = (np.minimum(Z\_stop,Zs)).tolist()
18. r = [slice(start,stop) **for** start,stop **in** zip(R\_start,R\_stop)]
19. z = [slice(start,stop) **for** start,stop **in** zip(Z\_start,Z\_stop)]
20. R[r] = Z[z]
21. print(Z)

print(R)

1. Посчитать ранг матрицы
2. Z = np.random.uniform(0,1,(10,10))

rank = np.linalg.matrix\_rank(Z)

1. Найти наиболее частое значение в массиве
2. Z = np.random.randint(0,10,50)

print(np.bincount(Z).argmax())

1. Извлечь все смежные 3x3 блоки из 10x10 матрицы
2. Z = np.random.randint(0,5,(10,10))
3. n = 3
4. i = 1 + (Z.shape[0] - n)
5. j = 1 + (Z.shape[1] - n)
6. C = stride\_tricks.as\_strided(Z, shape=(i, j, n, n), strides=Z.strides + Z.strides)

print(C)

1. Создать подкласс симметричных 2D массивов (Z[i,j] == Z[j,i])
2. *# Note: only works for 2d array and value setting using indices*
3. **class** **Symetric**(np.ndarray):
4. **def** \_\_setitem\_\_(self, (i,j), value):
5. super(Symetric, self).\_\_setitem\_\_((i,j), value)
6. super(Symetric, self).\_\_setitem\_\_((j,i), value)
7. **def** symetric(Z):
8. **return** np.asarray(Z + Z.T - np.diag(Z.diagonal())).view(Symetric)
9. S = symetric(np.random.randint(0,10,(5,5)))
10. S[2,3] = 42

print(S)

1. Рассмотрим множество матриц (n,n) и множество из p векторов (n,1). Посчитать сумму p произведений матриц (результат имеет размерность (n,1))
2. p, n = 10, 20
3. M = np.ones((p,n,n))
4. V = np.ones((p,n,1))
5. S = np.tensordot(M, V, axes=[[0, 2], [0, 1]])
6. print(S)
7. *# It works, because:*
8. *# M is (p,n,n)*
9. *# V is (p,n,1)*
10. *# Thus, summing over the paired axes 0 and 0 (of M and V independently),*

*# and 2 and 1, to remain with a (n,1) vector.*

1. Дан массив 16x16, посчитать сумму по блокам 4x4
2. Z = np.ones((16,16))
3. k = 4
4. S = np.add.reduceat(np.add.reduceat(Z, np.arange(0, Z.shape[0], k), axis=0),

 np.arange(0, Z.shape[1], k), axis=1)

1. Написать игру "жизнь"
2. **def** iterate(Z):
3. *# Count neighbours*
4. N = (Z[0:-2,0:-2] + Z[0:-2,1:-1] + Z[0:-2,2:] +
5. Z[1:-1,0:-2] + Z[1:-1,2:] +
6. Z[2: ,0:-2] + Z[2: ,1:-1] + Z[2: ,2:])
7. *# Apply rules*
8. birth = (N == 3) & (Z[1:-1,1:-1]==0)
9. survive = ((N == 2) | (N == 3)) & (Z[1:-1,1:-1] == 1)
10. Z[...] = 0
11. Z[1:-1,1:-1][birth | survive] = 1
12. **return** Z
13. Z = np.random.randint(0,2,(50,50))
14. **for** i **in** range(100):
15. print(Z)

 Z = iterate(Z)

1. Найти n наибольших значений в массиве
2. Z = np.arange(10000)
3. np.random.shuffle(Z)
4. n = 5

print (Z[np.argpartition(-Z,n)[:n]])

1. Построить прямое произведение массивов (все комбинации с каждым элементом)
2. **def** cartesian(arrays):
3. arrays = [np.asarray(a) **for** a **in** arrays]
4. shape = map(len, arrays)
5. ix = np.indices(shape, dtype=int)
6. ix = ix.reshape(len(arrays), -1).T
7. **for** n, arr **in** enumerate(arrays):
8. ix[:, n] = arrays[n][ix[:, n]]
9. **return** ix

print(cartesian(([1, 2, 3], [4, 5], [6, 7])))

1. Даны 2 массива A (8x3) и B (2x2). Найти строки в A, которые содержат элементы из каждой строки в B, независимо от порядка элементов в B
2. A = np.random.randint(0,5,(8,3))
3. B = np.random.randint(0,5,(2,2))
4. C = (A[..., np.newaxis, np.newaxis] == B)
5. rows = (C.sum(axis=(1,2,3)) >= B.shape[1]).nonzero()[0]

print(rows)

1. Дана 10x3 матрица, найти строки из неравных значений (например [2,2,3])
2. Z = np.random.randint(0,5,(10,3))
3. E = np.logical\_and.reduce(Z[:,1:] == Z[:,:-1], axis=1)
4. U = Z[~E]
5. print(Z)

print(U)

1. Преобразовать вектор чисел в матрицу бинарных представлений
2. I = np.array([0, 1, 2, 3, 15, 16, 32, 64, 128], dtype=np.uint8)

print(np.unpackbits(I[:, np.newaxis], axis=1))

1. Дан двумерный массив. Найти все различные строки
2. Z = np.random.randint(0, 2, (6,3))
3. T = np.ascontiguousarray(Z).view(np.dtype((np.void, Z.dtype.itemsize \* Z.shape[1])))
4. \_, idx = np.unique(T, return\_index=**True**)
5. uZ = Z[idx]

print(uZ)

1. Даны векторы A и B, написать einsum эквиваленты функций inner, outer, sum и mul
2. *# Make sure to read: http://ajcr.net/Basic-guide-to-einsum/*
3. np.einsum('i->', A) *# np.sum(A)*
4. np.einsum('i,i->i', A, B) *# A \* B*
5. np.einsum('i,i', A, B) *# np.inner(A, B)*

np.einsum('i,j', A, B) *# np.outer(A, B)*