Модуль numpy

1. Импортировать NumPy под именем `np`

```
import numpy as np
```

2. Напечатать версию и конфигурацию

```
print(np.__version__)
np.show_config()
```

3. Создать вектор (одномерный массив) размера 10, заполненный нулями

```
Z = np.zeros(10)
print(Z)
```

4. Создать вектор размера 10, заполненный единицами

```
Z = np.ones(10)
print(Z)
```

5. Создать вектор размера 10, заполненный числом 2.5

```
Z = np.full(10, 2.5)
print(Z)
```

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

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

7. Создать вектор размера 10, заполненный нулями, но пятый элемент равен 1

```
Z = np.zeros(10)
Z[4] = 1
print(Z)
```

8. Создать вектор со значениями от 10 до 49

```
Z = np.arange(10,50)
print(Z)
```

9. Развернуть вектор (первый становится последним)

```
Z = np.arange(50)
Z = Z[::-1]
```

10. Создать матрицу (двумерный массив) 3x3 со значениями от 0 до 8

```
Z = np.arange(9).reshape(3,3)
print(Z)
```

11. Найти индексы ненулевых элементов в [1,2,0,0,4,0]

```
nz = np.nonzero([1,2,0,0,4,0])
print(nz)
```

12. Создать 3x3 единичную матрицу

```
Z = np.eye(3)
print(Z)
```

13. Создать массив 3x3x3 со случайными значениями

```
Z = np.random.random((3,3,3))
print(Z)
```

14. Создать массив 10x10 со случайными значениями, найти минимум и максимум

```
Z = np.random.random((10,10))
Zmin, Zmax = Z.min(), Z.max()
print(Zmin, Zmax)
```

15. Создать случайный вектор размера 30 и найти среднее значение всех элементов

```
Z = np.random.random(30)
m = Z.mean()
print(m)
```

16. Создать матрицу с 0 внутри, и 1 на границах

```
Z = np.ones((10,10))
Z[1:-1,1:-1] = 0
```

17. Выяснить результат следующих выражений

```
0 * np.nan
np.nan == np.nan
np.inf > np.nan
np.nan - np.nan
0.3 == 3 * 0.1
```

18. Создать 5x5 матрицу с 1,2,3,4 под диагональю

```
Z = np.diag(np.arange(1, 5), k=-1)
print(Z)
```

19. Создать 8x8 матрицу и заполнить её в шахматном порядке

```
Z = np.zeros((8,8), dtype=int)
Z[1::2,::2] = 1
Z[::2,1::2] = 1
print(Z)
```

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

```
print(np.unravel_index(100, (6,7,8)))
```

21. Создать 8x8 матрицу и заполнить её в шахматном порядке, используя функцию `tile`

```
Z = np.tile(np.array([[0,1],[1,0]]), (4,4))
print(Z)
```

22. Перемножить матрицы 5x3 и 3x2

```
Z = np.dot(np.ones((5,3)), np.ones((3,2)))
print(Z)
```

23. Дан массив, поменять знак у элементов, значения которых между 3 и 8

```
Z = np.arange(11)
Z[(3 < Z) & (Z <= 8)] *= -1
```

24. Создать 5x5 матрицу со значениями в строках от 0 до 4

```
Z = np.zeros((5,5))
Z += np.arange(5)
print(Z)
```

25. Есть генератор, сделать с его помощью массив

```
def generate():
    for x in xrange(10):
        yield x
Z = np.fromiter(generate(),dtype=float,count=-1)
print(Z)
```

26. Создать вектор размера 10 со значениями от 0 до 1, не включая ни то, ни другое

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

27. Отсортировать вектор

```
Z = np.random.random(10)
Z.sort()
print(Z)
```

28. Проверить, одинаковы ли 2 numpy массива

```
A = np.random.randint(0,2,5)
B = np.random.randint(0,2,5)
equal = np.allclose(A,B)
print(equal)
```

29. Сделать массив неизменяемым

```
Z = np.zeros(10)
Z.flags.writeable = False
Z[0] = 1
```

30. Дан массив 10x2 (точки в декартовой системе координат), преобразовать в полярную

```
Z = np.random.random((10,2))
X,Y = Z[:,0], Z[:,1]
R = np.hypot(X, Y)
T = np.arctan2(Y,X)
print(R)
print(T)
```

31. Заменить максимальный элемент на ноль

```
Z = np.random.random(10)
Z[Z.argmax()] = 0
print(Z)
```

32. Создать структурированный массив с координатами x, y на сетке в квадрате [0,1]x[0,1]

```
Z = np.zeros((10,10), [('x',float),('y',float)])
Z['x'], Z['y'] = np.meshgrid(np.linspace(0,1,10),
                             np.linspace(0,1,10))
print(Z)
```

33. Из двух массивов сделать матрицу Коши C (Cij = 1/(xi - yj))

```
X = np.arange(8)
Y = X + 0.5
C = 1.0 / np.subtract.outer(X, Y)
print(np.linalg.det(C))
```

34. Найти минимальное и максимальное значение, принимаемое каждым числовым типом numpy

```
for dtype in [np.int8, np.int32, np.int64]:
   print(np.iinfo(dtype).min)
   print(np.iinfo(dtype).max)
for dtype in [np.float32, np.float64]:
   print(np.finfo(dtype).min)
   print(np.finfo(dtype).max)
   print(np.finfo(dtype).eps)
```

35. Напечатать все значения в массиве

```
np.set_printoptions(threshold=np.nan)
Z = np.zeros((25,25))
print(Z)
```

36. Найти ближайшее к заданному значению число в заданном массиве

```
Z = np.arange(100)
v = np.random.uniform(0,100)
index = (np.abs(Z-v)).argmin()
print(Z[index])
```

37. Создать структурированный массив, представляющий координату (x,y) и цвет (r,g,b)

```
 Z = np.zeros(10, [ ('position', [ ('x', float, 1),
                                   ('y', float, 1)]),
                    ('color',    [ ('r', float, 1),
                                   ('g', float, 1),
                                   ('b', float, 1)])])
print(Z)
```

38. Дан массив (100,2) координат, найти расстояние от каждой точки до каждой

```
import scipy.spatial

Z = np.random.random((10,2))
D = scipy.spatial.distance.cdist(Z,Z)
print(D)
```

39. Преобразовать массив из float в int

```
Z = np.arange(10, dtype=np.int32)
Z = Z.astype(np.float32, copy=False)
```

40. Дан файл:

```
1,2,3,4,5
6,,,7,8
,,9,10,11
```

Как прочитать его?

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

41. Каков эквивалент функции `enumerate` для numpy массивов?

```
Z = np.arange(9).reshape(3,3)
for index, value in np.ndenumerate(Z):
    print(index, value)
for index in np.ndindex(Z.shape):
    print(index, Z[index])
```

42. Сформировать 2D массив с распределением Гаусса

```
X, Y = np.meshgrid(np.linspace(-1,1,10), np.linspace(-1,1,10))
D = np.hypot(X, Y)
sigma, mu = 1.0, 0.0
G = np.exp(-((D - mu) ** 2 / (2.0 * sigma ** 2)))
print(G)
```

43. Случайно расположить p элементов в 2D массив

```
n = 10
p = 3
Z = np.zeros((n,n))
np.put(Z, np.random.choice(range(n*n), p, replace=False), 1)
```

44. Отнять среднее из каждой строки в матрице

```
X = np.random.rand(5, 10)
Y = X - X.mean(axis=1, keepdims=True)
```

45. Отсортировать матрицу по n-ому столбцу

```
Z = np.random.randint(0,10,(3,3))
n = 1  # Нумерация с нуля
print(Z)
print(Z[Z[:,n].argsort()])
```

46. Определить, есть ли в 2D массиве нулевые столбцы

```
Z = np.random.randint(0,3,(3,10))
print((~Z.any(axis=0)).any())
```

47. Дан массив, добавить 1 к каждому элементу с индексом, заданным в другом массиве (осторожно с повторами)

```
Z = np.ones(10)
I = np.random.randint(0,len(Z),20)
Z += np.bincount(I, minlength=len(Z))
print(Z)
```

48. Дан массив (w,h,3) (картинка) dtype=ubyte, посчитать количество различных цветов

```
w,h = 16,16
I = np.random.randint(0, 2, (h,w,3)).astype(np.ubyte)
F = I[...,0] * 256 * 256 + I[...,1] * 256 + I[...,2]
n = len(np.unique(F))
print(np.unique(I))
```

49. Дан четырехмерный массив, посчитать сумму по последним двум осям

```
A = np.random.randint(0,10, (3,4,3,4))
sum = A.reshape(A.shape[:-2] + (-1,)).sum(axis=-1)
print(sum)
```

50. Найти диагональные элементы произведения матриц

```
# Slow version
np.diag(np.dot(A, B))

# Fast version
np.sum(A * B.T, axis=1)

# Faster version
np.einsum("ij,ji->i", A, B).
```

51. Дан вектор [1, 2, 3, 4, 5], построить новый вектор с тремя нулями между каждым значением

```
Z = np.array([1,2,3,4,5])
nz = 3
Z0 = np.zeros(len(Z) + (len(Z)-1)*(nz))
Z0[::nz+1] = Z
print(Z0)
```

52. Поменять 2 строки в матрице

```
A = np.arange(25).reshape(5,5)
A[[0,1]] = A[[1,0]]
print(A)
```

53. Рассмотрим набор из 10 троек, описывающих 10 треугольников (с общими вершинами), найти множество уникальных отрезков, составляющих все треугольники

```
faces = np.random.randint(0,100,(10,3))
F = np.roll(faces.repeat(2,axis=1),-1,axis=1)
F = F.reshape(len(F)*3,2)
F = np.sort(F,axis=1)
G = F.view( dtype=[('p0',F.dtype),('p1',F.dtype)] )
G = np.unique(G)
print(G)
```

54. Дан массив C; создать массив A, что `np.bincount(A) == C`

```
C = np.bincount([1,1,2,3,4,4,6])
A = np.repeat(np.arange(len(C)), C)
print(A)
```

55. Посчитать среднее, используя плавающее окно

```
def moving_average(a, n=3):
    ret = np.cumsum(a, dtype=float)
    ret[n:] = ret[n:] - ret[:-n]
    return ret[n - 1:] / n

print(moving_average(np.arange(20), 3))
```

56. 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]))

```
from numpy.lib import stride_tricks

def rolling(a, window):
    shape = (a.size - window + 1, window)
    strides = (a.itemsize, a.itemsize)
    return stride_tricks.as_strided(a, shape=shape, strides=strides)
Z = rolling(np.arange(10), 3)
print(Z)
```

57. Инвертировать булево значение, или поменять знак у числового массива без создания нового

```
Z = np.random.randint(0,2,100)
np.logical_not(arr, out=arr)

Z = np.random.uniform(-1.0,1.0,100)
np.negative(arr, out=arr)
```

58. Рассмотрим 2 набора точек P0, P1 описания линии (2D) и точку p, как вычислить расстояние от p до каждой линии i (P0[i],P1[i])

```
def distance(P0, P1, p):
    T = P1 - P0
    L = (T**2).sum(axis=1)
    U = -((P0[:,0] - p[...,0]) * T[:,0] + (P0[:,1] - p[...,1]) * T[:,1]) / L
    U = U.reshape(len(U),1)
    D = P0 + U * T - p
    return np.sqrt((D**2).sum(axis=1))

P0 = np.random.uniform(-10,10,(10,2))
P1 = np.random.uniform(-10,10,(10,2))
p  = np.random.uniform(-10,10,( 1,2))
print(distance(P0, P1, p))
```

59. Дан массив. Написать функцию, выделяющую часть массива фиксированного размера с центром в данном элементе (дополненное значением `fill` если необходимо)

```
Z = np.random.randint(0,10, (10,10))
shape = (5,5)
fill  = 0
position = (1,1)

R = np.ones(shape, dtype=Z.dtype)*fill
P  = np.array(list(position)).astype(int)
Rs = np.array(list(R.shape)).astype(int)
Zs = np.array(list(Z.shape)).astype(int)

R_start = np.zeros((len(shape),)).astype(int)
R_stop  = np.array(list(shape)).astype(int)
Z_start = (P - Rs//2)
Z_stop  = (P + Rs//2)+Rs%2

R_start = (R_start - np.minimum(Z_start, 0)).tolist()
Z_start = (np.maximum(Z_start, 0)).tolist()
R_stop = np.maximum(R_start, (R_stop - np.maximum(Z_stop-Zs,0))).tolist()
Z_stop = (np.minimum(Z_stop,Zs)).tolist()

r = [slice(start,stop) for start,stop in zip(R_start,R_stop)]
z = [slice(start,stop) for start,stop in zip(Z_start,Z_stop)]
R[r] = Z[z]
print(Z)
print(R)
```

60. Посчитать ранг матрицы

```
Z = np.random.uniform(0,1,(10,10))
rank = np.linalg.matrix_rank(Z)
```

61. Найти наиболее частое значение в массиве

```
Z = np.random.randint(0,10,50)
print(np.bincount(Z).argmax())
```

62. Извлечь все смежные 3x3 блоки из 10x10 матрицы

```
Z = np.random.randint(0,5,(10,10))
n = 3
i = 1 + (Z.shape[0] - n)
j = 1 + (Z.shape[1] - n)
C = stride_tricks.as_strided(Z, shape=(i, j, n, n), strides=Z.strides + Z.strides)
print(C)
```

63. Создать подкласс симметричных 2D массивов (Z[i,j] == Z[j,i])

```
# Note: only works for 2d array and value setting using indices

class Symetric(np.ndarray):
    def __setitem__(self, (i,j), value):
        super(Symetric, self).__setitem__((i,j), value)
        super(Symetric, self).__setitem__((j,i), value)

def symetric(Z):
    return np.asarray(Z + Z.T - np.diag(Z.diagonal())).view(Symetric)

S = symetric(np.random.randint(0,10,(5,5)))
S[2,3] = 42
print(S)
```

64. Рассмотрим множество матриц (n,n) и множество из p векторов (n,1). Посчитать сумму p произведений матриц (результат имеет размерность (n,1))

```
p, n = 10, 20
M = np.ones((p,n,n))
V = np.ones((p,n,1))
S = np.tensordot(M, V, axes=[[0, 2], [0, 1]])
print(S)

# It works, because:
# M is (p,n,n)
# V is (p,n,1)
# 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.
```

65. Дан массив 16x16, посчитать сумму по блокам 4x4

```
Z = np.ones((16,16))
k = 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)
```

66. Написать игру "жизнь"

```
def iterate(Z):
    # Count neighbours
    N = (Z[0:-2,0:-2] + Z[0:-2,1:-1] + Z[0:-2,2:] +
         Z[1:-1,0:-2]                + Z[1:-1,2:] +
         Z[2:  ,0:-2] + Z[2:  ,1:-1] + Z[2:  ,2:])

    # Apply rules
    birth = (N == 3) & (Z[1:-1,1:-1]==0)
    survive = ((N == 2) | (N == 3)) & (Z[1:-1,1:-1] == 1)
    Z[...] = 0
    Z[1:-1,1:-1][birth | survive] = 1
    return Z

Z = np.random.randint(0,2,(50,50))
for i in range(100):
    print(Z)
    Z = iterate(Z)
```

67. Найти n наибольших значений в массиве

```
Z = np.arange(10000)
np.random.shuffle(Z)
n = 5

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

68. Построить прямое произведение массивов (все комбинации с каждым элементом)

```
def cartesian(arrays):
    arrays = [np.asarray(a) for a in arrays]
    shape = map(len, arrays)

    ix = np.indices(shape, dtype=int)
    ix = ix.reshape(len(arrays), -1).T

    for n, arr in enumerate(arrays):
        ix[:, n] = arrays[n][ix[:, n]]

    return ix

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

69. 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

```
A = np.random.randint(0,5,(8,3))
B = np.random.randint(0,5,(2,2))

C = (A[..., np.newaxis, np.newaxis] == B)
rows = (C.sum(axis=(1,2,3)) >= B.shape[1]).nonzero()[0]
print(rows)
```

70. Дана 10x3 матрица, найти строки из неравных значений (например [2,2,3])

```
Z = np.random.randint(0,5,(10,3))
E = np.logical_and.reduce(Z[:,1:] == Z[:,:-1], axis=1)
U = Z[~E]
print(Z)
print(U)
```

71. Преобразовать вектор чисел в матрицу бинарных представлений

```
I = np.array([0, 1, 2, 3, 15, 16, 32, 64, 128], dtype=np.uint8)
print(np.unpackbits(I[:, np.newaxis], axis=1))
```

72. Дан двумерный массив. Найти все различные строки

```
Z = np.random.randint(0, 2, (6,3))
T = np.ascontiguousarray(Z).view(np.dtype((np.void, Z.dtype.itemsize * Z.shape[1])))
_, idx = np.unique(T, return_index=True)
uZ = Z[idx]
print(uZ)
```

73. Даны векторы A и B, написать `einsum` эквиваленты функций `inner`, `outer`, `sum` и `mul`

```
# Make sure to read: http://ajcr.net/Basic-guide-to-einsum/

np.einsum('i->', A)       # np.sum(A)
np.einsum('i,i->i', A, B) # A * B
np.einsum('i,i', A, B)    # np.inner(A, B)
np.einsum('i,j', A, B)    # np.outer(A, B)
```