Московский авиационный институт
(национальный исследовательский университет)

Библиотека для математических расчетов Numpy

Установка Numpy

```
pip install --user numpy

python -m pip install --user numpy
```

```python
# Пример расчета функции на сетке без использования Numpy
from array import array
import math

def func(x):
    return math.sin(x) * math.cos(3 * x)

minval = 0
maxval = 10
count = 101
step = (maxval - minval) / (count - 1)

xdata = array('d', [0] * count)
ydata = array('d', [0] * count)

for n in range(count):
    xdata[n] = minval + n * step
    ydata[n] = func(xdata[n])

print(f"{xdata=}")
print(f"{ydata=}")
```

src/15. Numpy/example_01/example_01.py

```python
# Пример расчета функции на сетке с использованием Numpy
import numpy as np

def func(x):
    return np.sin(x) * np.cos(3 * x)

minval = 0
maxval = 10
count = 101

xdata = np.linspace(minval, maxval, count)
ydata = func(xdata)

print(f"{type(xdata)=}")
print(f"{xdata=}")
print(f"{type(ydata)=}")
print(f"{ydata=}")
```

src/15. Numpy/example_02/example_02.py

Создание массивов Numpy

```python
# Создание массивов, заполненных нулями
import numpy as np

foo = np.zeros(5)
foo_2d = np.zeros((3, 4))
foo_3d = np.zeros((3, 4, 5))

print(f"{type(foo)=}")
print(f"{foo}")
print()
print(f"{type(foo_2d)=}")
print(f"{foo_2d}")
print()
print(f"{type(foo_3d)=}")
print(f"{foo_3d}")
```

```
type(foo)=<class 'numpy.ndarray'>
[0. 0. 0. 0. 0.]

type(foo_2d)=<class 'numpy.ndarray'>
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]

type(foo_3d)=<class 'numpy.ndarray'>
[[[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.]]]
```

```python
# Создание массивов, заполненных единицами
import numpy as np

foo = np.ones(5)
foo_2d = np.ones((3, 4))

print("foo:")
print(f"{foo}")
print()
print("foo_2d:")
print(f"{foo_2d}")
```

```
foo:
[1. 1. 1. 1. 1.]

foo_2d:
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

```python
# Создание массива, заполненного заданным значением
import numpy as np

value = 10

foo = np.full(5, value)
bar = np.full((3, 5), value)

print("foo:")
print(foo)
print()
print("bar:")
print(bar)
```

```
foo:
[10 10 10 10 10]

bar:
[[10 10 10 10 10]
 [10 10 10 10 10]
 [10 10 10 10 10]]
```

```python
# Создание единичной матрицы
import numpy as np

foo = np.eye(3)
bar = np.eye(5, k=2)
baz = np.eye(5, k=-2)

print("foo:")
print(foo)
print()
print("bar:")
print(bar)
print()
print("baz:")
print(baz)
```

```
foo:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

bar:
[[0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]

baz:
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]]
```

```python
# Создание массивов, заполненных последовательностью чисел
import numpy as np

foo = np.arange(1.0, 2.0, 0.1)
bar = np.arange(1.0, 10.0)
baz = np.arange(10.0)
spam = np.linspace(1.0, 2.0, 11)

print(f"{foo=}")
print()
print(f"{bar=}")
print()
print(f"{baz=}")
print()
print(f"{spam=}")
```

```
foo=array([1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9])

bar=array([1., 2., 3., 4., 5., 6., 7., 8., 9.])

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

spam=array([1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. ])
```

```python
# Создание массивов, заполненных псевдослучайными числами
import numpy as np
from numpy.random import rand

foo = rand(3, 4)
bar = rand(3, 4) * 2 - 1
baz = np.round(rand(3, 4) * 10)

print("foo:")
print(f"{foo}")
print()
print("bar:")
print(f"{bar}")
print()
print("baz:")
print(f"{baz}")
```

```
foo:
[[0.21764368 0.74868811 0.58075236 0.3912998 ]
 [0.6336203  0.49800793 0.35559929 0.79551925]
 [0.82125084 0.59544188 0.94007288 0.90175849]]

bar:
[[-0.70476497 -0.8555988   0.01903383  0.05124231]
 [-0.71897426  0.49959979 -0.55091192  0.73555402]
 [-0.84338374 -0.14433317 -0.2284243   0.73844891]]

baz:
[[ 5.  4.  8.  7.]
 [10.  0.  7.  4.]
 [ 2.  7.  5.  6.]]
```

Методы и свойства массивов Numpy

```
# Методы и свойства массивов ndarray
import numpy as np

foo = np.zeros((3, 4))
print(dir(foo))
```

['T', '__abs__', '__add__', '__and__', '__array__', '__array_finalize__', '__array_function__', '__array_interface__', '__array_prepare__', '__array_priority__', '__array_struct__', '__array_ufunc__', '__array_wrap__', '__bool__', '__class__', '__class_getitem__', '__complex__', '__contains__', '__copy__', '__deepcopy__', '__delattr__', '__delitem__', '__dir__', '__divmod__', '__dlpack__', '__dlpack_device__', '__doc__', '__eq__', '__float__', '__floordiv__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__', '__iand__', '__ifloordiv__', '__ilshift__', '__imatmul__', '__imod__', '__imul__', '__index__', '__init__', '__init_subclass__', '__int__', '__invert__', '__ior__', '__ipow__', '__irshift__', '__isub__', '__iter__', '__itruediv__', '__ixor__', '__le__', '__len__', '__lshift__', '__lt__', '__matmul__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__or__', '__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rlshift__', '__rmatmul__', '__rmod__', '__rmul__', '__ror__', '__rpow__', '__rrshift__', '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__setitem__', '__setstate__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__truediv__', '__xor__', 'all', 'any', 'argmax', 'argmin', 'argpartition', 'argsort', 'astype', 'base', 'byteswap', 'choose', 'clip', 'compress', 'conj', 'conjugate', 'copy', 'ctypes', 'cumprod', 'cumsum', 'data', 'diagonal', 'dot', 'dtype', 'dump', 'dumps', 'fill', 'flags', 'flat', 'flatten', 'getfield', 'imag', 'item', 'itemset', 'itemsize', 'max', 'mean', 'min', 'nbytes', 'ndim', 'newbyteorder', 'nonzero', 'partition', 'prod', 'ptp', 'put', 'ravel', 'real', 'repeat', 'reshape', 'resize', 'round', 'searchsorted', 'setfield', 'setflags', 'shape', 'size', 'sort', 'squeeze', 'std', 'strides', 'sum', 'swapaxes', 'take', 'tobytes', 'tofile', 'tolist', 'tostring', 'trace', 'transpose', 'var', 'view']

```python
# Свойство shape и транспонирование матрицы
import numpy as np
from numpy.random import rand

foo = np.round(rand(3, 4) * 10)

print(f"{foo}")
print()
print(f"{foo.shape=}")
print()
print(f"{foo.T}")
print()
print(f"{foo.T.shape=}")
```

```
[[8. 4. 6. 7.]
 [1. 3. 3. 0.]
 [9. 3. 2. 9.]]

foo.shape=(3, 4)

[[8. 1. 9.]
 [4. 3. 3.]
 [6. 3. 2.]
 [7. 0. 9.]]

foo.T.shape=(4, 3)
```

```python
# Расчет суммы и произведения элементов массива
import numpy as np
from numpy.random import rand

foo = np.round(rand(3, 4) * 10)

print(f"{foo}")
print()
print(f"{foo.sum()=}")
print(f"{foo.sum(axis=0)=}")
print(f"{foo.sum(axis=1)=}")
print()
print(f"{foo.prod()=}")
print(f"{foo.prod(axis=0)=}")
print(f"{foo.prod(axis=1)=}")
```

```
[[ 5.  5.  5.  7.]
 [ 4. 10.  6.  2.]
 [ 9.  3.  2.  0.]]

foo.sum()=58.0
foo.sum(axis=0)=array([18., 18., 13.,  9.])
foo.sum(axis=1)=array([22., 22., 14.])

foo.prod()=0.0
foo.prod(axis=0)=array([180., 150.,  60.,   0.])
foo.prod(axis=1)=array([875., 480.,   0.])
```

```python
# Нахождение минимального, максимального и среднего значений
import numpy as np
from numpy.random import rand

foo = np.round(rand(3, 4) * 10)

print(f"{foo}")
print()
print(f"{foo.min()=}")
print(f"{foo.min(axis=0)=}")
print(f"{foo.min(axis=1)=}")
print()
print(f"{foo.max()=}")
print(f"{foo.max(axis=0)=}")
print(f"{foo.max(axis=1)=}")
print()
print(f"{foo.mean()=}")
print(f"{foo.mean(axis=0)=}")
print(f"{foo.mean(axis=1)=}")
```

```
[[ 3.  9.  8.  8.]
 [ 9.  7.  4.  1.]
 [ 3.  5. 10. 10.]]

foo.min()=1.0
foo.min(axis=0)=array([3., 5., 4., 1.])
foo.min(axis=1)=array([3., 1., 3.])

foo.max()=10.0
foo.max(axis=0)=array([ 9.,  9., 10., 10.])
foo.max(axis=1)=array([ 9.,  9., 10.])

foo.mean()=6.416666666666667
foo.mean(axis=0)=array([5.        , 7.        , 7.33333333, 6.33333333])
foo.mean(axis=1)=array([7.  , 5.25, 7.  ])
```

```python
# Нахождение индексов минимального и максимального значений
import numpy as np
from numpy.random import rand

foo = np.round(rand(3, 4) * 10)

print(f"{foo}")
print()
print(f"{foo.argmin()=}")
print(f"{foo.argmin(axis=0)=}")
print(f"{foo.argmin(axis=1)=}")
print()
print(f"{foo.argmax()=}")
print(f"{foo.argmax(axis=0)=}")
print(f"{foo.argmax(axis=1)=}")
```

```
[[8. 9. 2. 0.]
 [8. 1. 2. 9.]
 [2. 8. 4. 3.]]

foo.argmin()=3
foo.argmin(axis=0)=array([2, 1, 0, 0])
foo.argmin(axis=1)=array([3, 1, 0])

foo.argmax()=1
foo.argmax(axis=0)=array([0, 0, 2, 1])
foo.argmax(axis=1)=array([1, 3, 1])
```

```python
# Изменение формы массива
import numpy as np
from numpy.random import rand

foo = np.round(rand(12) * 10)

print("foo:")
print(f"{foo}")
print()

bar = foo.reshape((3, 4))
baz = foo.reshape((2, 6))

print("bar:")
print(f"{bar}")
print()
print("baz:")
print(f"{baz}")
```

```
foo:
[ 2.  1.  9.  4. 10.  4.  7.  6.  4.  3.  6.  3.]

bar:
[[ 2.  1.  9.  4.]
 [10.  4.  7.  6.]
 [ 4.  3.  6.  3.]]

baz:
[[ 2.  1.  9.  4. 10.  4.]
 [ 7.  6.  4.  3.  6.  3.]]
```

```python
# Ошибка при изменении формы массива
import numpy as np
from numpy.random import rand

foo = np.round(rand(3, 4) * 10)

print(f"{foo}")
print()

# Ошибка!
bar = foo.reshape((3, 7))
```

```
[[8. 9. 8. 5.]
 [6. 7. 6. 7.]
 [6. 6. 0. 3.]]

Traceback (most recent call last):
  File "example.py", line 11, in <module>
    bar = foo.reshape((3, 7))
          ^^^^^^^^^^^^^^^^^^^
ValueError: cannot reshape array of size 12 into shape (3,7)
```

```python
# Изменение размера массива
import numpy as np
from numpy.random import rand

foo = np.round(rand(3, 4) * 10)

print(f"{foo}")
print()

foo.resize((3, 5))
print(f"{foo}")
print()

foo.resize((3, 2))
print(f"{foo}")
```

```
[[ 4.  4.  5. 10.]
 [ 0.  9.  9.  2.]
 [ 2.  8.  1.  1.]]

[[ 4.  4.  5. 10.  0.]
 [ 9.  9.  2.  2.  8.]
 [ 1.  1.  0.  0.  0.]]

[[ 4.  4.]
 [ 5. 10.]
 [ 0.  9.]]
```

Срезы массивов Numpy

Сравнение работы со срезами массивов Numpy и списков

`src/15. Numpy/example_03/`

```python
# Выделение срезов для одномерных массивов
import numpy as np

foo_ndarray = np.arange(0, 10)
slice_ndarray = foo_ndarray[2: 7: 2]

print(f"{foo_ndarray=}")
print(f"{slice_ndarray=}")
```

```
foo_ndarray=array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
slice_ndarray=array([2, 4, 6])
```

```python
# Выделение срезов для одномерных массивов
import numpy as np

foo_ndarray = np.arange(0, 10)
slice_ndarray = foo_ndarray[3:]

print(f"{foo_ndarray=}")
print(f"{slice_ndarray=}")
```

```
foo_ndarray=array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
slice_ndarray=array([3, 4, 5, 6, 7, 8, 9])
```

```python
# Выделение срезов для одномерных массивов
import numpy as np

foo_ndarray = np.arange(0, 10)
slice_ndarray = foo_ndarray[:5]

print(f"{foo_ndarray=}")
print(f"{slice_ndarray=}")
```

```
foo_ndarray=array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
slice_ndarray=array([0, 1, 2, 3, 4])
```

Создание копии массивов

src/15. Numpy/example_04/
src/15. Numpy/example_05/

```python
# Создание копии среза
import numpy as np

foo = np.arange(0, 10)
foo_slice_copy = foo[3: 7].copy()

print("Первоначальное состояние")
print(f"{foo=}")
print(f"{foo_slice_copy=}")
print()

foo_slice_copy[:] = 0

print("После изменения элементов копии среза")
print(f"{foo=}")
print(f"{foo_slice_copy=}")
```

```
Первоначальное состояние
foo=array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
foo_slice_copy=array([3, 4, 5, 6])

После изменения элементов копии среза
foo=array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
foo_slice_copy=array([0, 0, 0, 0])
```

```python
# Выделение срезов для двумерных массивов
import numpy as np

# Создание матрицы из последовательности целых чисел
matrix = np.arange(0, 25).reshape((5, 5))
print("matrix:")
print(matrix)
print()

bar = matrix[1:4, 1:3]

print("bar:")
print(bar)
```

```
matrix:
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]

bar:
[[ 6  7]
 [11 12]
 [16 17]]
```

```python
# Выделение строк двумерных массивов
import numpy as np

# Создание матрицы из последовательности целых чисел
matrix = np.arange(0, 25).reshape((5, 5))
print("matrix:")
print(matrix)
print()

bar = matrix[1:4, :]

print("bar:")
print(bar)
```

```
matrix:
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]

bar:
[[ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

```python
# Выделение строки двумерного массива
import numpy as np

# Создание матрицы из последовательности целых чисел
matrix = np.arange(0, 25).reshape((5, 5))
print("matrix:")
print(matrix)
print()

bar = matrix[2, :]

print("bar:")
print(bar)
```

```
matrix:
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]

bar:
[10 11 12 13 14]
```

```python
# Выделение столбцов двумерных массивов
import numpy as np

# Создание матрицы из последовательности целых чисел
matrix = np.arange(0, 25).reshape((5, 5))
print("matrix:")
print(matrix)
print()

bar = matrix[:, 1:3]

print("bar:")
print(bar)
```

```
matrix:
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]

bar:
[[ 1  2]
 [ 6  7]
 [11 12]
 [16 17]
 [21 22]]
```

```python
# Выделение столбца двумерного массива
import numpy as np

# Создание матрицы из последовательности целых чисел
matrix = np.arange(0, 25).reshape((5, 5))
print("matrix:")
print(matrix)
print()

bar = matrix[:, 2]

print("bar:")
print(bar)
```

```
matrix:
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]

bar:
[ 2  7 12 17 22]
```

```python
# Выделение срезов для двумерных массивов
import numpy as np

# Создание матрицы из последовательности целых чисел
matrix = np.arange(0, 36).reshape((6, 6))
print("matrix:")
print(matrix)
print()

bar = matrix[1:7:2, 1:7:2]

print("bar:")
print(bar)
```

```
matrix:
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]
 [24 25 26 27 28 29]
 [30 31 32 33 34 35]]

bar:
[[ 7  9 11]
 [19 21 23]
 [31 33 35]]
```

Модификация срезов двумерных массивов

src/15. Numpy/example_06/

Булевы массивы

```python
# Булевы массивы
import numpy as np

matrix = np.arange(0, 25).reshape((5, 5))
matrix_bool_1 = matrix > 10
matrix_bool_2 = matrix % 2 == 0

print("matrix:")
print(matrix)
print()

print("matrix_bool_1:")
print(matrix_bool_1)
print()

print("matrix_bool_2:")
print(matrix_bool_2)
print()
```

```
matrix:
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]

matrix_bool_1:
[[False False False False False]
 [False False False False False]
 [False  True  True  True  True]
 [ True  True  True  True  True]
 [ True  True  True  True  True]]

matrix_bool_2:
[[ True False  True False  True]
 [False  True False  True False]
 [ True False  True False  True]
 [False  True False  True False]
 [ True False  True False  True]]
```

```python
# Булевы массивы
import numpy as np

matrix = np.arange(0, 25).reshape((5, 5))
matrix_bool = matrix > 10

bar = matrix[matrix_bool]

print("matrix:")
print(matrix)
print()

print("matrix_bool:")
print(matrix_bool)
print()

print("bar:")
print(bar)
print()
```

```
matrix:
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]

matrix_bool:
[[False False False False False]
 [False False False False False]
 [False  True  True  True  True]
 [ True  True  True  True  True]
 [ True  True  True  True  True]]

bar:
[11 12 13 14 15 16 17 18 19 20 21 22 23 24]
```

```python
# Булевы массивы
import numpy as np

matrix = np.arange(0, 25).reshape((5, 5))
matrix_bool = matrix > 10
matrix[matrix_bool] = 10

print("matrix:")
print(matrix)
print()
```

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

```python
# Булевы массивы
import numpy as np

matrix = np.arange(0, 25).reshape((5, 5))
matrix[matrix > 10] = 10

print("matrix:")
print(matrix)
print()
```

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

```python
# Булевы массивы
import numpy as np

matrix = np.arange(0, 25).reshape((5, 5))
matrix[matrix % 2 == 0] = 0

print("matrix:")
print(matrix)
print()
```

```
matrix:
[[ 0  1  0  3  0]
 [ 5  0  7  0  9]
 [ 0 11  0 13  0]
 [15  0 17  0 19]
 [ 0 21  0 23  0]]
```

```python
# Булевы массивы
import numpy as np
from numpy.random import rand

foo = np.round(rand(3, 3) * 10)

print("foo:")
print(f"{foo}")
print()
print(f"{(foo < 10).all()=}")
print(f"{(foo >= 10).any()=}")
```

```
foo:
[[4. 5. 7.]
 [2. 4. 8.]
 [3. 3. 7.]]

(foo < 10).all()=True
(foo >= 10).any()=False
```