Topic

Introduction to NumPy

Jan 28, 2025

Topics

- Covered in
  - `python_intro_pt2.ipynb`
  - `python_intro_pt3.ipynb`

- Introduction to **NumPy**.
  - Arrays (creation, assignment, access). (part 2)
  - Loading data from a file into an array. (part 3)
- Introduction to **Matplotlib**.
  - Creating simple line plots. (part 3)

NumPy

- “NumPy is the fundamental package needed for scientific computing with Python.”
- NumPy provides (non exhaustive list):
  - a powerful N-dimensional array object (**`ndarray`**);
  - useful for linear algebra and so much more
- It is freely available and open-source

  https://github.com/numpy/numpy

NumPy

- We will learn about N-dimensional arrays (**`ndarray`**).

```
#python

>>> import numpy as np
>>> x = np.array([1.0, 2.0, 3.0])

>>> type(x)
<class ‘numpy.ndarray'>
```

- “An array object represents a multidimensional, homogeneous array of fixed-size items.”

NumPy

- Each **`ndarray`** has these important attributes:
  - **`ndim`** (the number of dimensions),
  - **`shape`** (the size of each dimension),
  - **`size`** (the total size of the array),
  - **`dtype`** (the data type of the entries)

```
#python

>>> import numpy as np
>>> x = np.array([1.0, 2.0, 3.0])

>>> x.ndim
1

>>> x.shape
(3,)

>>> x.size
3

>>> x.dtype
dtype('float64')
```

`ndarray` Pointwise Operations

```
#python

>>> import numpy as np

>>> x = np.array([1.0, 2.0, 3.0])
>>> y = np.array([4.0, 5.0, 6.0])
>>> z = x + y
>>> z
array([5., 7., 9.])
```

`ndarray` Pointwise Operations

```
#python

>>> import numpy as np

>>> x = np.array([1.0, 2.0, 3.0])
>>> y = np.array([4.0, 5.0, 6.0])
>>> z = x + y
>>> z
array([5., 7., 9.])
```

`z = x + y` ⟶

```
for i in range(3):
    z[i] = x[i] + y[i]
```

`ndarray` Creation

- Create a multi-dimensional **`ndarray`** using a **`list`** or **`tuple`** filled with values.
- Pass list / tuple as argument to the constructor **`np.array()`**

```
#python

>>> import numpy as np
>>> A = np.array(
          [ [1.0, 2.0, 3.0],
            [4.0, 2.0, 0.0],
              [1.0, 1.0, 2.0] ])

>>> A.ndim
2

>>> A.shape
(3, 3)

>>> A.size
9
```

`ndarray` Attributes

- Each **`ndarray`** has these important attributes:
  - **`ndim`** (the number of dimensions),
  - **`shape`** (the size of each dimension),
  - **`size`** (the total size of the array),
  - **`dtype`** (the data type of the entries)

```
#python

>>> import numpy as np
>>> A = np.array(
          [ [1.0, 2.0, 3.0],
            [4.0, 2.0, 0.0],
            [1.0, 1.0, 2.0] ])


>>> A.ndim
2

>>> A.shape
(3, 3)

>>> A.size
9
```

`ndarray` Creation

- Other creation short-cuts / helpers
  - **`np.zeros(shape)`**
    - Example: **`shape = 3`**
    - Example: **`shape = (3, 2)`**
  - **`np.ones(shape)`**
    - As above

```
#python

>>> import numpy as np

>>> A = np.zeros(4)
>>> print(A)
[0. 0. 0. 0.]

>>> A = np.zeros((3,2))
>>> print(A)
[[0. 0.]
 [0. 0.]
 [0. 0.]]

>>> A = np.ones((2,3))
>>> print(A)
[[1. 1. 1.]
 [1. 1. 1.]]
```

`ndarray` Access

- Access a single element `A[i, j]`

- Access a single row, `A[i, :]`

- Access a single column `A[:, j]`

```
#python

>>> import numpy as np

>>> A = np.zeros((2,3))

>>> A[1, 2] = 33.0
>>> print(A)
[[ 0.  0.  0.]
 [ 0.  0. 33.]]

>>> A[0, :] = 22.0
>>> print(A)
[[22. 22. 22.]
 [ 0.  0. 33.]]

>>> A[:, 1] = 11.0
>>> print(A)
[[22. 11. 22.]
 [ 0. 11. 33.]]
```