Practical Computing for Scientists

Armin Sobhani

CSCI 2000U

UOIT – Fall 2015

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Python

NumPy (Continue)

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Different Data Type Sizes

Integers (Signed)	Size
`int8`	8 bits
`int16`	16 bits
`int32`	32 bits (same as **`int`** on 32-bit platforms)
`int64`	64 bits (same as **`int`** on 64-bit platforms)

Different Data Type Sizes

Integers (Signed)	Size
int8	8 bits
int16	16 bits
int32	32 bits (same as **int** on 32-bit platforms)
int64	64 bits (same as **int** on 64-bit platforms)

```
>>> np.array([1], dtype=int).dtype
dtype('int64')
```

Different Data Type Sizes

Integers (Signed)	Size
int8	8 bits
int16	16 bits
int32	32 bits (same as **int** on 32-bit platforms)
int64	64 bits (same as **int** on 64-bit platforms)

```
>>> np.array([1], dtype=int).dtype
dtype('int64')
>>> np.iinfo(np.int32).max, 2**31 - 1
(2147483647, 2147483647)
```

Different Data Type Sizes

Integers (Signed)	Size
int8	8 bits
int16	16 bits
int32	32 bits (same as **int** on 32-bit platforms)
int64	64 bits (same as **int** on 64-bit platforms)

```
>>> np.array([1], dtype=int).dtype
dtype('int64')
>>> np.iinfo(np.int32).max, 2**31 - 1
(2147483647, 2147483647)
```

don't forget it

Different Data Type Sizes

Integers (Signed)	Size
int8	8 bits
int16	16 bits
int32	32 bits (same as **int** on 32-bit platforms)
int64	64 bits (same as **int** on 64-bit platforms)

```
>>> np.array([1], dtype=int).dtype
dtype('int64')
>>> np.iinfo(np.int32).max, 2**31 - 1
(2147483647, 2147483647)
```

don't forget it
also for dtype=...

Different Data Type Sizes

Unsigned Integers	Size
`uint8`	8 bits
`uint16`	16 bits
`uint32`	32 bits
`uint64`	64 bits

Different Data Type Sizes

Unsigned Integers	Size
uint8	8 bits
uint16	16 bits
uint32	32 bits
uint64	64 bits

```
>>> np.iinfo(np.uint32).max, 2**32 - 1
(4294967295, 4294967295)
```

Different Data Type Sizes

Floating-Point	Size
`float16`	16 bits
`float32`	32 bits
`float64`	64 bits (same as **`float`**)
`float96`	96 bits, platform-dependent
`float128`	128 bits, platform-dependent

Different Data Type Sizes

Floating-Point	Size
`float16`	16 bits
`float32`	32 bits
`float64`	64 bits (same as **`float`**)
`float96`	96 bits, platform-dependent
`float128`	128 bits, platform-dependent

```
>>> np.finfo(np.float32).eps
1.1920929e-07
```

Different Data Type Sizes

Floating-Point	Size
`float16`	16 bits
`float32`	32 bits
`float64`	64 bits (same as **`float`**)
`float96`	96 bits, platform-dependent
`float128`	128 bits, platform-dependent

```
>>> np.finfo(np.float32).eps
1.1920929e-07
>>> np.finfo(np.float64).eps
2.220446049250313e-16
```

Different Data Type Sizes

Complex Floating-Point	Size
`complex64`	two 32-bit floats
`complex128`	two 64-bit floats
`complex192`	two 96-bit floats, platform-dependent
`complex256`	two 128-bit floats, platform-dependent

Structured Data Types

```
>>> samples = np.zeros((6,), dtype=
... [('sensor_code', 'S4'),
...  ('position', float),
...  ('value', float)])
```

Structured Data Types

```
>>> samples = np.zeros((6,), dtype=
... [('sensor_code', 'S4'),
...  ('position', float),
...  ('value', float)])
>>> samples
array([(b'', 0.0, 0.0), (b'', 0.0, 0.0), (b'', 0.0, 0.0),
       (b'', 0.0, 0.0), (b'', 0.0, 0.0), (b'', 0.0, 0.0)],
       dtype=[('sensor_code', 'S4'), ('position', '<f8'),
              ('value', '<f8')])
```

Structured Data Types

```
>>> samples = np.zeros((6,), dtype=
... [('sensor_code', 'S4'),
...  ('position', float),
...  ('value', float)])
>>> samples
array([(b'', 0.0, 0.0), (b'', 0.0, 0.0), (b'', 0.0, 0.0),
       (b'', 0.0, 0.0), (b'', 0.0, 0.0), (b'', 0.0, 0.0)],
       dtype=[('sensor_code', 'S4'), ('position', '<f8'),
              ('value', '<f8')])
>>> samples.ndim
1
```

Structured Data Types

```
>>> samples = np.zeros((6,), dtype=
... [('sensor_code', 'S4'),
...  ('position', float),
...  ('value', float)])
>>> samples
array([(b'', 0.0, 0.0), (b'', 0.0, 0.0), (b'', 0.0, 0.0),
       (b'', 0.0, 0.0), (b'', 0.0, 0.0), (b'', 0.0, 0.0)],
       dtype=[('sensor_code', 'S4'), ('position', '<f8'),
              ('value', '<f8')])
>>> samples.ndim
1
>>> samples.shape
(6,)
```

Structured Data Types

```
>>> samples = np.zeros((6,), dtype=
... [('sensor_code', 'S4'),
...  ('position', float),
...  ('value', float)])
>>> samples
array([(b'', 0.0, 0.0), (b'', 0.0, 0.0), (b'', 0.0, 0.0),
       (b'', 0.0, 0.0), (b'', 0.0, 0.0), (b'', 0.0, 0.0)],
       dtype=[('sensor_code', 'S4'), ('position', '<f8'),
              ('value', '<f8')])
>>> samples.ndim
1
>>> samples.shape
(6,)
>>> samples.dtype.names # field names
('sensor_code', 'position', 'value')
```

Structured Data Types

```
>>> samples[:] = [('ALFA', 1, 0.37), ('BETA', 1, 0.11),
...               ('TAU', 1, 0.13),  ('ALFA', 1.5, 0.37),
...               ('ALFA', 3, 0.11), ('TAU', 1.2, 0.13)]
```

Structured Data Types

```
>>> samples[:] = [('ALFA', 1, 0.37), ('BETA', 1, 0.11),
...               ('TAU', 1, 0.13),  ('ALFA', 1.5, 0.37),
...               ('ALFA', 3, 0.11), ('TAU', 1.2, 0.13)]
>>> samples
array([('ALFA', 1.0, 0.37), ('BETA', 1.0, 0.11),
       ('TAU', 1.0, 0.13), ('ALFA', 1.5, 0.37),
       ('ALFA', 3.0, 0.11), ('TAU', 1.2, 0.13)],
      dtype=[('sensor_code', 'S4'), ('position', '<f8'),
             ('value', '<f8')])
```

Structured Data Types

```
>>> samples[:] = [('ALFA', 1, 0.37), ('BETA', 1, 0.11),
...               ('TAU', 1, 0.13),  ('ALFA', 1.5, 0.37),
...               ('ALFA', 3, 0.11), ('TAU', 1.2, 0.13)]
>>> samples
array([('ALFA', 1.0, 0.37), ('BETA', 1.0, 0.11),
       ('TAU', 1.0, 0.13), ('ALFA', 1.5, 0.37),
       ('ALFA', 3.0, 0.11), ('TAU', 1.2, 0.13)],
      dtype=[('sensor_code', 'S4'), ('position', '<f8'),
             ('value', '<f8')])
>>> samples['sensor_code']
array(['ALFA', 'BETA', 'TAU', 'ALFA', 'ALFA', 'TAU'], dtype='|S4')
```

Structured Data Types

```
>>> samples[:] = [('ALFA', 1, 0.37), ('BETA', 1, 0.11),
...               ('TAU', 1, 0.13),  ('ALFA', 1.5, 0.37),
...               ('ALFA', 3, 0.11), ('TAU', 1.2, 0.13)]
>>> samples
array([('ALFA', 1.0, 0.37), ('BETA', 1.0, 0.11),
       ('TAU', 1.0, 0.13), ('ALFA', 1.5, 0.37),
       ('ALFA', 3.0, 0.11), ('TAU', 1.2, 0.13)],
      dtype=[('sensor_code', 'S4'), ('position', '<f8'),
             ('value', '<f8')])
>>> samples['sensor_code']
array(['ALFA', 'BETA', 'TAU', 'ALFA', 'ALFA', 'TAU'], dtype='|S4')
>>> samples['value']
array([ 0.37, 0.11, 0.13, 0.37, 0.11, 0.13])
```

Structured Data Types

```
>>> samples[:] = [('ALFA', 1, 0.37), ('BETA', 1, 0.11),
...               ('TAU', 1, 0.13),  ('ALFA', 1.5, 0.37),
...               ('ALFA', 3, 0.11), ('TAU', 1.2, 0.13)]
>>> samples
array([('ALFA', 1.0, 0.37), ('BETA', 1.0, 0.11),
       ('TAU', 1.0, 0.13), ('ALFA', 1.5, 0.37),
       ('ALFA', 3.0, 0.11), ('TAU', 1.2, 0.13)],
      dtype=[('sensor_code', 'S4'), ('position', '<f8'),
             ('value', '<f8')])
>>> samples['sensor_code']
array(['ALFA', 'BETA', 'TAU', 'ALFA', 'ALFA', 'TAU'], dtype='|S4')
>>> samples['value']
array([ 0.37, 0.11, 0.13, 0.37, 0.11, 0.13])
>>> samples[0]
('ALFA', 1.0, 0.37)
```

Structured Data Types

```
>>> samples[0]['sensor_code'] = 'TAU'
```

Structured Data Types

```
>>> samples[0]['sensor_code'] = 'TAU'
>>> samples[0]
('TAU', 1.0, 0.37)
```

Structured Data Types

```
>>> samples[0]['sensor_code'] = 'TAU'
>>> samples[0]
('TAU', 1.0, 0.37)
>>> samples[['position', 'value']]
array([(1.0, 0.37), (1.0, 0.11),
       (1.0, 0.13), (1.5, 0.37),
       (3.0, 0.11), (1.2, 0.13)],
       dtype=[('position', '<f8'), ('value', '<f8')])
```

Structured Data Types

```
>>> samples[0]['sensor_code'] = 'TAU'
>>> samples[0]
('TAU', 1.0, 0.37)
>>> samples[['position', 'value']]
array([(1.0, 0.37), (1.0, 0.11),
       (1.0, 0.13), (1.5, 0.37),
       (3.0, 0.11), (1.2, 0.13)],
      dtype=[('position', '<f8'), ('value', '<f8')])
>>> samples[samples['sensor_code'] == 'ALFA']
array([('ALFA', 1.5, 0.37), ('ALFA', 3.0, 0.11)],
      dtype=[('sensor_code', 'S4'),
             ('position', '<f8'),
             ('value', '<f8')])
```

Structured Data Types

```
>>> samples[0]['sensor_code'] = 'TAU'
>>> samples[0]
('TAU', 1.0, 0.37)
>>> samples[['position', 'value']]
array([(1.0, 0.37), (1.0, 0.11),
       (1.0, 0.13), (1.5, 0.37),
       (3.0, 0.11), (1.2, 0.13)],
      dtype=[('position', '<f8'), ('value', '<f8')])
>>> samples[samples['sensor_code'] == 'ALFA']
array([('ALFA', 1.5, 0.37), ('ALFA', 3.0, 0.11)],
      dtype=[('sensor_code', 'S4'),
             ('position', '<f8'),
             ('value', '<f8')])
```

valid in Python2

Format of `typestr`

- A string consisting of three parts:

Format of `typestr`

- A string consisting of three parts:
  - byte order
    - <: little-endian
    - >: big-endian
    - | : not-relevant

Format of `typestr`

- A string consisting of three parts:
  - byte order
    - <: little-endian
    - >: big-endian
    - | : not-relevant
  - basic type

Character	Type
`b`	boolean
`i`	(signed) integer
`u`	unsigned integer
`f`	floating-point
`c`	complex float
`O`	(Python) object
`S, a`	(byte-)string
`U`	Unicode
`v`	raw data (void)

Format of `typestr`

- A string consisting of three parts:
  - byte order
    - <: little-endian
    - >: big-endian
    - | : not-relevant
  - basic type
  - number of bytes

Examples

```
>>> dt = np.dtype('i4')    # 32-bit signed integer
```

Examples

```
>>> dt = np.dtype('i4')    # 32-bit signed integer
>>> dt = np.dtype('f8')    # 64-bit floating-point number
```

Examples

```
>>> dt = np.dtype('i4')   # 32-bit signed integer
>>> dt = np.dtype('f8')   # 64-bit floating-point number
>>> dt = np.dtype('c16')  # 128-bit complex floating-point number
```

Examples

```
>>> dt = np.dtype('i4')   # 32-bit signed integer
>>> dt = np.dtype('f8')   # 64-bit floating-point number
>>> dt = np.dtype('c16')  # 128-bit complex floating-point number
>>> dt = np.dtype('a25')  # 25-character string
```

Examples

```
>>> dt = np.dtype('i4')   # 32-bit signed integer
>>> dt = np.dtype('f8')   # 64-bit floating-point number
>>> dt = np.dtype('c16')  # 128-bit complex floating-point number
>>> dt = np.dtype('a25')  # 25-character string
>>> dt = np.dtype([('big', '>i4'), ('little', '<i4')])
```

Examples

```
>>> dt = np.dtype('i4')   # 32-bit signed integer
>>> dt = np.dtype('f8')   # 64-bit floating-point number
>>> dt = np.dtype('c16')  # 128-bit complex floating-point number
>>> dt = np.dtype('a25')  # 25-character string
>>> dt = np.dtype([('big', '>i4'), ('little', '<i4')])
>>> dt = np.dtype([('R','u1'), ('G','u1'), ('B','u1'), ('A','u1')])
```