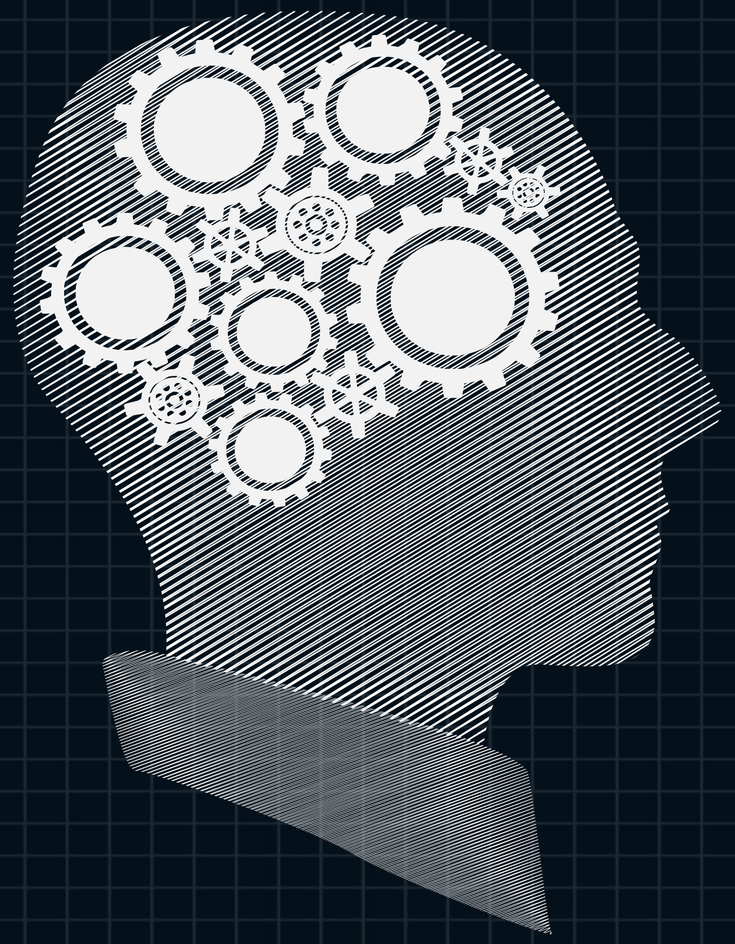


NumPy Master Class

Lecture.3
Making ndarrays

Lecture.3 Making ndarrays - from Python Lists

np.array

numpy.array(*object*, *dtype=None*, *, *copy=True*, *order='K'*, *subok=False*, *ndmin=0*, *like=None*)

```
import numpy as np

int_py = 3
float_py = 3.14

int_np = np.array(int_py)
float_np = np.array(float_py)


print("Integer case")
print(type(int_py), type(int_np))
print(int_py, int_np, sep=' - ')

print("Floating point case")
print(type(float_py), type(float_py))
print(float_py, float_np, sep=' - ')
```

```
Integer case
<class 'int'> <class 'numpy.ndarray'>
3 - 3

Floating point case
<class 'float'> <class 'float'>
3.14 - 3.14
```

Lecture.3 Making ndarrays - from Python Lists

Making Vector ndarrays

```
import numpy as np

vec_py = [1, 2, 3]
vec_np = np.array(vec_py)

print(type(vec_py), type(vec_np))
print(vec_py, vec_np, sep=' - ')
```

```
<class 'list'> <class 'numpy.ndarray'>
[1, 2, 3] - [1 2 3]
```

Lecture.3 Making ndarrays - from Python Lists

Making Matrix ndarrays

```
import numpy as np

mat_py = [[1, 2, 3],
          [4, 5, 6]]
mat_np = np.array(mat_py)

print(type(mat_py), type(mat_np))
print(mat_py, mat_np, sep='\n\n')
```

```
<class 'list'> <class 'numpy.ndarray'>
[[1, 2, 3], [4, 5, 6]]

[[1 2 3]
 [4 5 6]]
```

Lecture.3 Making ndarrays - from Python Lists

Making 3rd Order Tensor ndarrays

```
import numpy as np

tensor_py = [[[1, 2, 3],
              [4, 5, 6]],

             [[11, 12, 13],
              [14, 15, 16]]]
tensor_np = np.array(tensor_py)

print(type(tensor_py), type(tensor_np))
print(tensor_py, tensor_np, sep='\n\n')
```

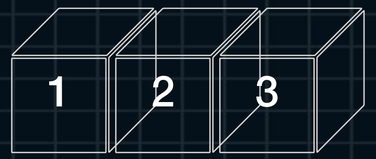
```
<class 'list'> <class 'numpy.ndarray'>
[[[1, 2, 3], [4, 5, 6]], [[11, 12, 13], [14, 15, 16]]]

[[[ 1  2  3]
  [ 4  5  6]]

 [[11 12 13]
  [14 15 16]]]
```

Lecture.3 Making ndarrays

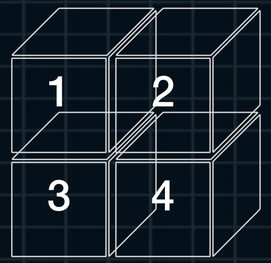
- Shapes of ndarrays

Shape Notation

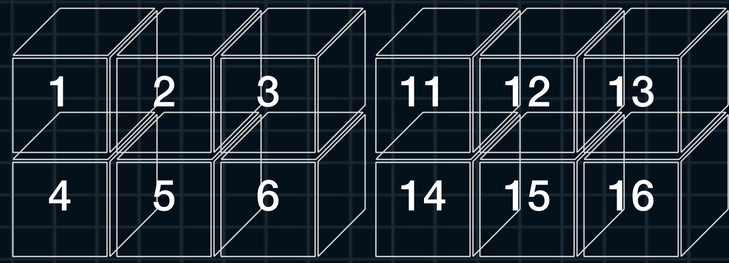
( )



(3,)



(2,3)



(2,2,3)

Lecture.3 Making ndarrays

- Shapes of ndarrays

Shapes of ndarrays

```
import numpy as np

scalar_np = np.array(3.14)
vec_np = np.array([1, 2, 3])
mat_np = np.array([[1, 2], [3, 4]])
tensor_np = np.array([[[1, 2, 3],
                       [4, 5, 6]],

                      [[11, 12, 13],
                       [14, 15, 16]]])

print(scalar_np.shape)    ()
print(vec_np.shape)       (3,)
print(mat_np.shape)       (2, 2)
print(tensor_np.shape)    (2, 2, 3)


print(len(()))            0
print(len((3,)))          1
print(len((2, 2)))        2
print(len((2, 2, 3)))     3
```

Lecture.3 Making ndarrays - with Shapes

ndarrays with Specific Values

`numpy.zeros`(*shape, dtype=float, order='C', *, like=None*)

```
import numpy as np

M = np.zeros(shape=(2, 3))


print(M.shape)  (2, 3)
print(M)        [[0. 0. 0.]
                 [0. 0. 0.]]
```

`numpy.ones`(*shape, dtype=None, order='C', *, like=None*)

```
import numpy as np

M = np.ones(shape=(2, 3))


print(M.shape)  (2, 3)
print(M)        [[1. 1. 1.]
                 [1. 1. 1.]]
```

Lecture.3 Making ndarrays - with Shapes

ndarrays with Specific Values

numpy.full(*shape, fill_value, dtype=None, order='C', *, like=None*)

```
import numpy as np

M = np.full(shape=(2, 3), fill_value=3.14)

print(M.shape)   (2, 3)
print(M)         [[3.14 3.14 3.14]
                  [3.14 3.14 3.14]]
```

numpy.empty(*shape, dtype=float, order='C', *, like=None*)

```
import numpy as np

M = np.empty(shape=(2, 3))

print(M.shape)   (2, 3)
print(M)         [[3.14 3.14 3.14]
                  [3.14 3.14 3.14]]
```

Lecture.3 Making ndarrays - from Existing ndarrays

numpy.zeros_like(*a, dtype=None, order='K', subok=True, shape=None*)

numpy.ones_like(*a, dtype=None, order='K', subok=True, shape=None*)

numpy.full_like(*a, fill_value, dtype=None, order='K', subok=True, shape=None*)

numpy.empty_like(*prototype, dtype=None, order='K', subok=True, shape=None*)

```
import numpy as np

M = np.full(shape=(2, 3), fill_value=3.14)

print(M, '\n')

zeros_like = np.zeros_like(M)
ones_like = np.ones_like(M)
full_like = np.full_like(M, fill_value=100)
empty_like = np.empty_like(M)

print("zeros_like: \n", zeros_like, '\n')
print("ones_like: \n", ones_like, '\n')
print("full_like: \n", full_like, '\n')
print("empty_like: \n", empty_like, '\n')
```

```
[[3.14 3.14 3.14]
 [3.14 3.14 3.14]]
```

```
zeros_like:
 [[0. 0. 0.]
 [0. 0. 0.]]

ones_like:
 [[1. 1. 1.]
 [1. 1. 1.]]
```

```
full_like:
 [[100. 100. 100.]
 [100. 100. 100.]]

empty_like:
 [[3.14 3.14 3.14]
 [3.14 3.14 3.14]]
```

Lecture.3
Making ndarrays

- with the Fixed Intervals/Points

Making ndarrays with Fixed Intervals

numpy.arange([*start*,]*stop*, [*step*,]*dtype=None*, *, *like=None*)

```
print(list(range(10)))          [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(list(range(2, 5)))        [2, 3, 4]
print(list(range(2, 10, 2)))    [2, 4, 6, 8]
```

```
import numpy as np

print(np.arange(10))          [0 1 2 3 4 5 6 7 8 9]
print(np.arange(2, 5))        [2 3 4]
print(np.arange(2, 10, 2))    [2 4 6 8]
```

Lecture.3 Making ndarrays

- with the Fixed Intervals/Points

Making ndarrays with Fixed Intervals

```
import numpy as np

print(np.arange(10.5))              [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
print(np.arange(1.5, 10.5))         [1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5]
print(np.arange(1.5, 10.5, 2.5))    [1.5 4.  6.5 9. ]
```

Lecture.3 Making ndarrays

- with the Fixed Intervals/Points

Making ndarrays with Fixed Points

numpy.linspace(*start*, *stop*, *num=50*, *endpoint=True*, *retstep=False*, *dtype=None*, *axis=0*)

```
print(np.linspace(0, 1, 5))
```

```
[0.   0.25 0.5  0.75 1.  ]
```

```
print(np.linspace(0, 1, 10))
```

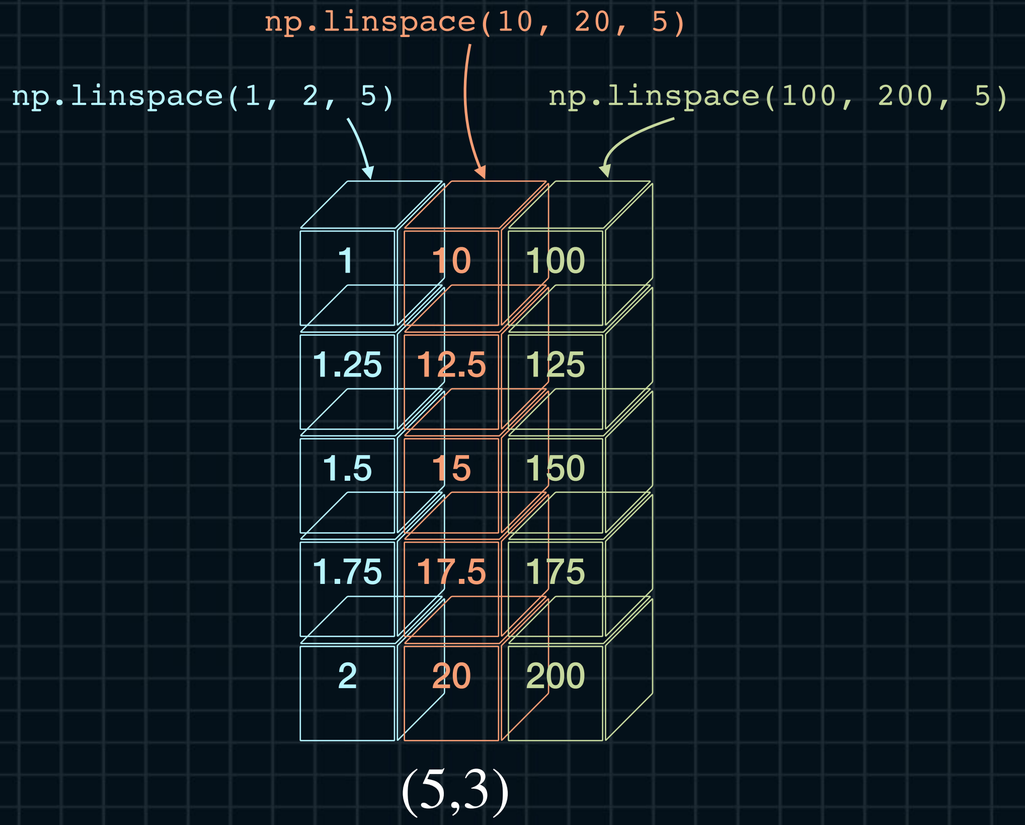
```
[0.         0.11111111 0.22222222 0.33333333 0.44444444 0.55555556 0.66666667 0.77777778 0.88888889 1.
```

Lecture.3 Making ndarrays - with the Fixed Intervals/Points

Making ndarrays with Fixed Points

```
import numpy as np

a = np.linspace([1, 10, 100], [2, 20, 200], 5)
print(a)
```

```
[[  1.    10.   100.  ]
 [  1.25  12.5  125.  ]
 [  1.5   15.   150.  ]
 [  1.75  17.5  175.  ]
 [  2.    20.   200.  ]]
```



Lecture.3
Making ndarrays

- with the Fixed Intervals/Points

np.arange and np.linspace

```
print(np.arange(0, 1 + 0.25, 0.25))
print(np.linspace(0, 1, 5))
```

```
[0.   0.25 0.5  0.75 1.  ]
[0.   0.25 0.5  0.75 1.  ]
```

```
print(np.arange(10))
print(np.linspace(0, 9, 10), '\n')
```

```
[0 1 2 3 4 5 6 7 8 9]
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

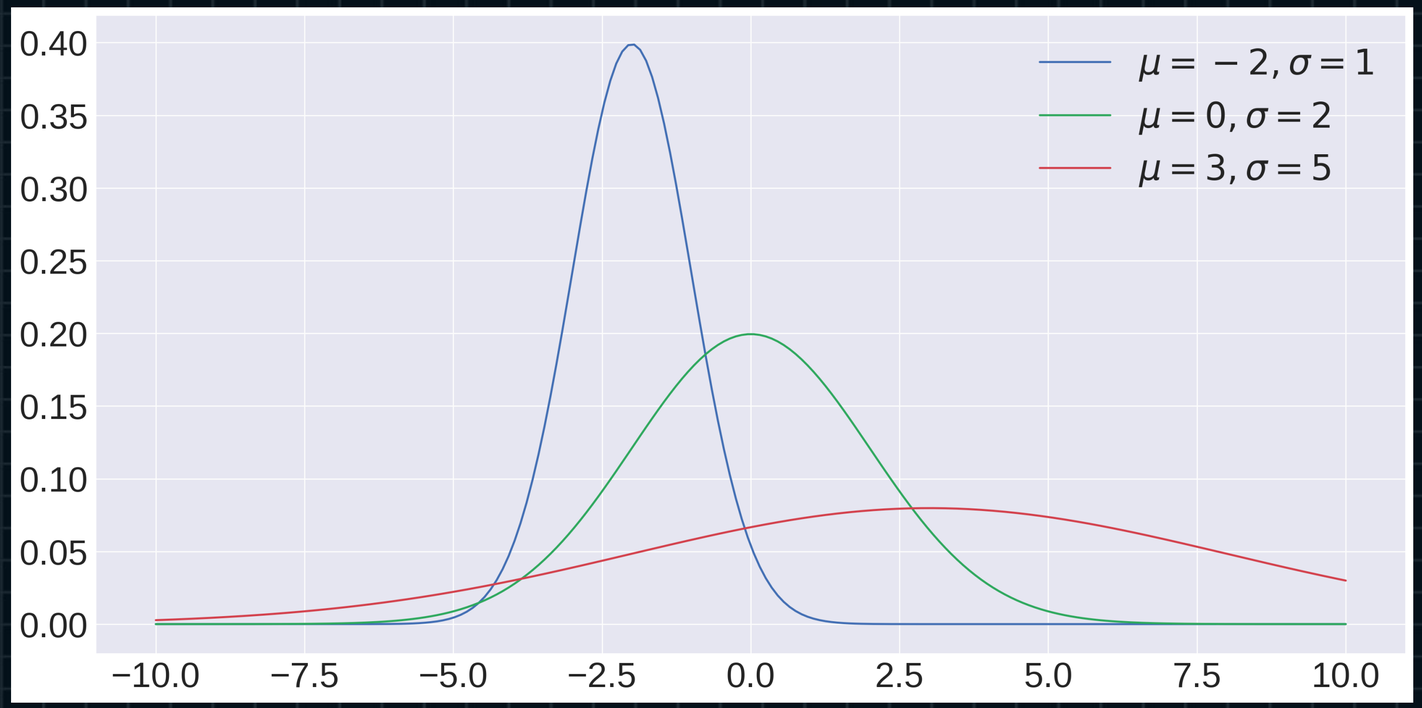
```
print(np.arange(5, 10))
print(np.linspace(5, 9, 9-5+1))
```

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

Lecture.3 Making ndarrays

- from Random Distributions

from Normal Distributions



`random.randn(d0, d1, ..., dn)`

`random.normal(loc=0.0, scale=1.0, size=None)`

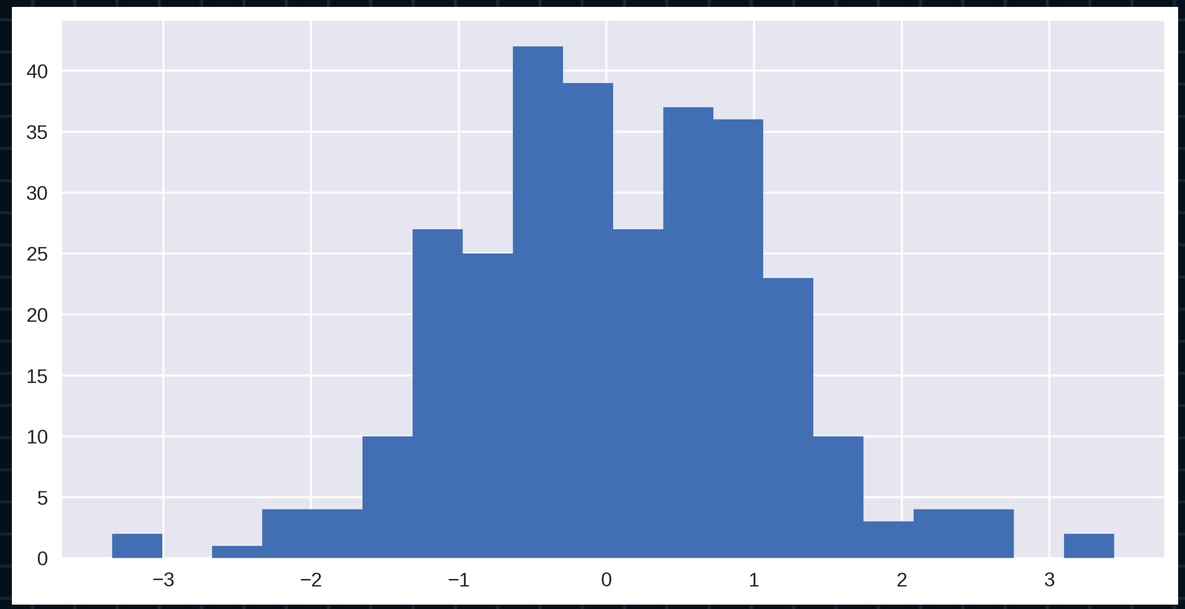
Lecture.3 Making ndarrays

- from Random Distributions

from Normal Distributions

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn')

fig, ax = plt.subplots(figsize=(10, 5))


random_values = np.random.randn(300)
ax.hist(random_values, bins=20)
print(random_values.shape)
```

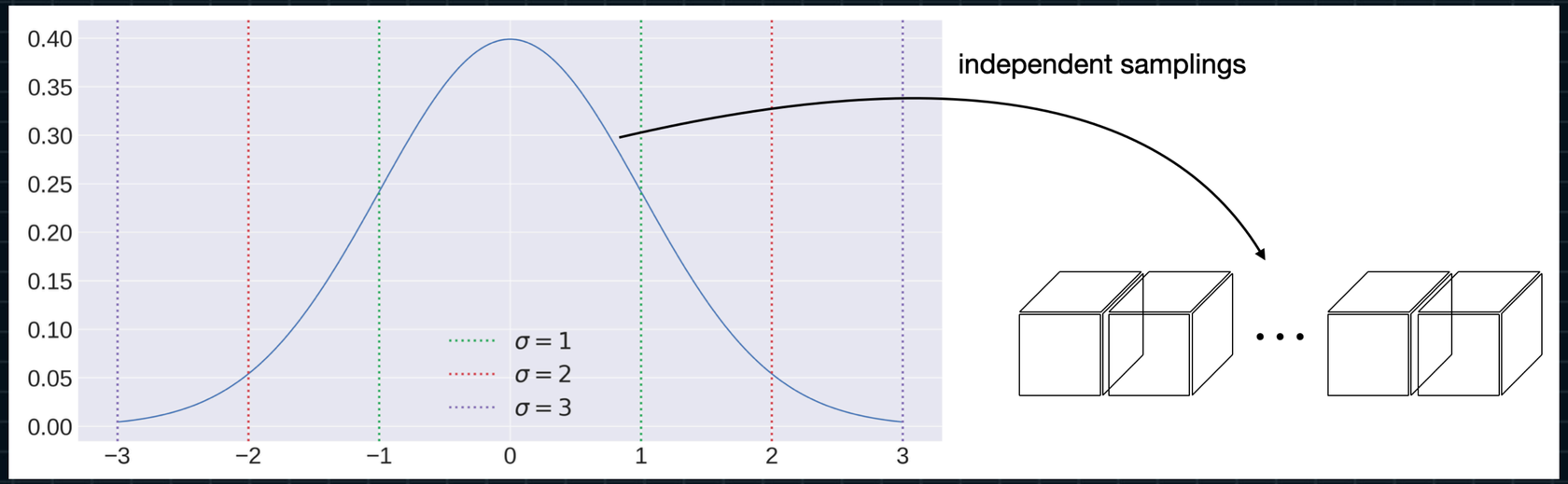
```
(300,)
```



Lecture.3 Making ndarrays - from Random Distributions

from Normal Distributions

```
import numpy as np

random_values = np.random.randn(300)
```

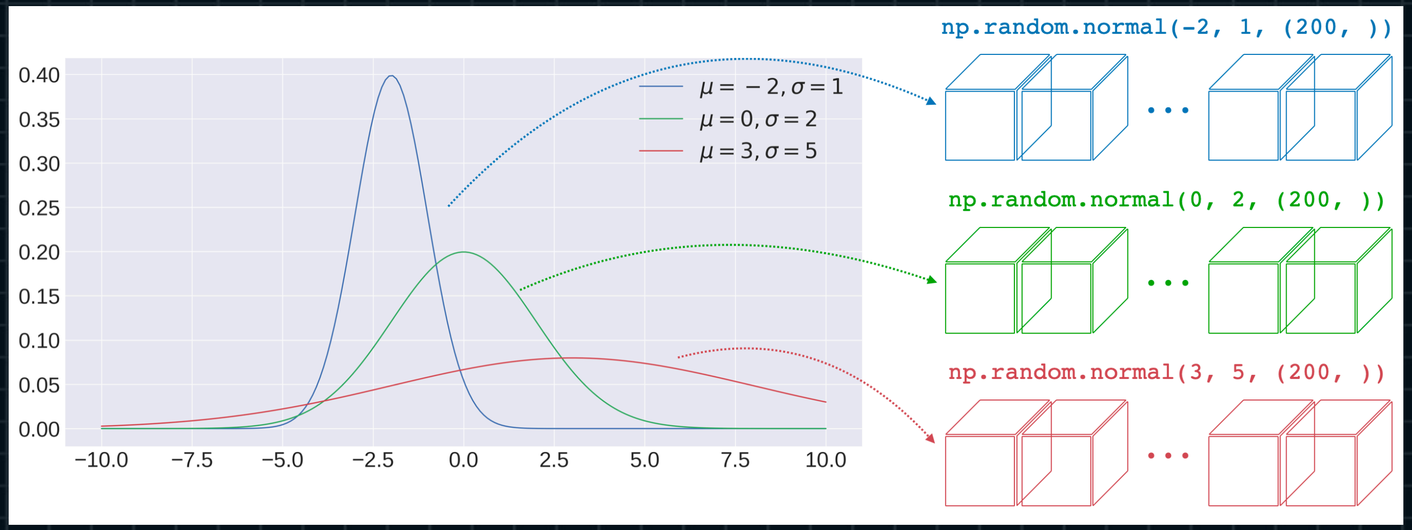


Lecture.3 Making ndarrays

- from Random Distributions

from Normal Distributions

```
import numpy as np

normal1 = np.random.normal(loc=-2, scale=1, size=(200, ))
normal2 = np.random.normal(loc=0, scale=2, size=(200, ))
normal3 = np.random.normal(loc=3, scale=5, size=(200, ))
```

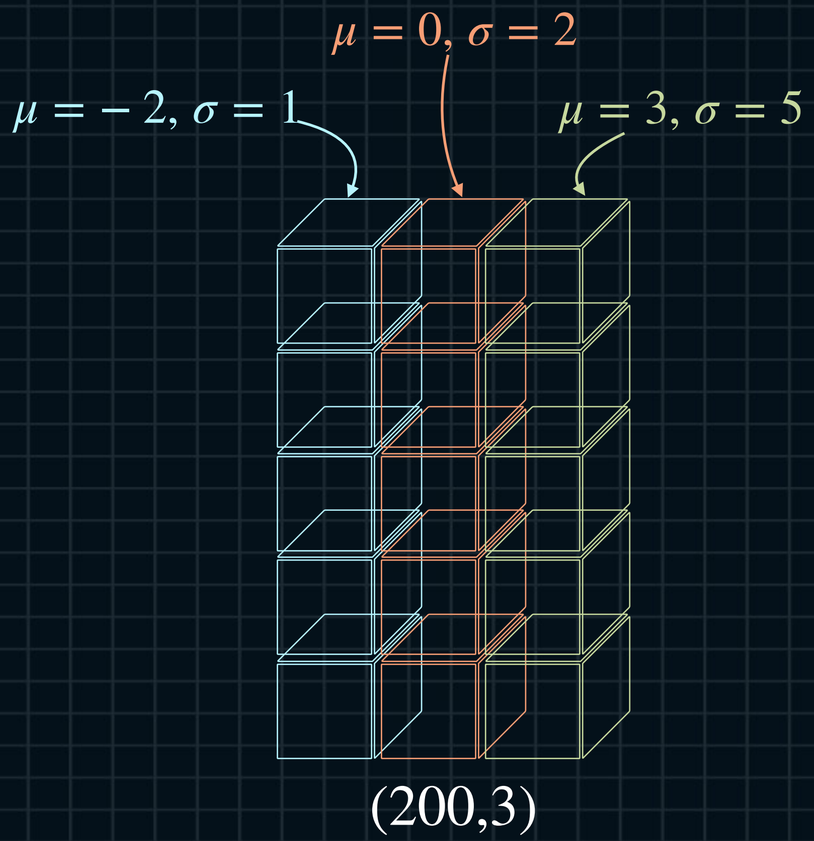


Lecture.3 Making ndarrays

- from Random Distributions

from Normal Distributions

```
import numpy as np

normal = np.random.normal(loc=[-2, 0, 3],
                          scale=[1, 2, 5],
                          size=(200, 3))
```



Lecture.3
Making ndarrays

- from Random Distributions

from Normal Distributions

```
import numpy as np

normal = np.random.normal(loc=-2, scale=1, size=(3, 3))
print(normal)
```

```
   [[-2.75087484 -0.3295038  -2.68580197]
    [-1.65031003 -4.18813124 -2.38557031]
    [-2.07713629 -2.75935169 -3.5294945 ]]
```

Lecture.3
Making ndarrays

- from Random Distributions

from Uniform Distributions

```
random.rand(d0, d1, ..., dn)
```

```
random.uniform(low=0.0, high=1.0, size=None)
```

```
random.randint(low, high=None, size=None, dtype=int)
```

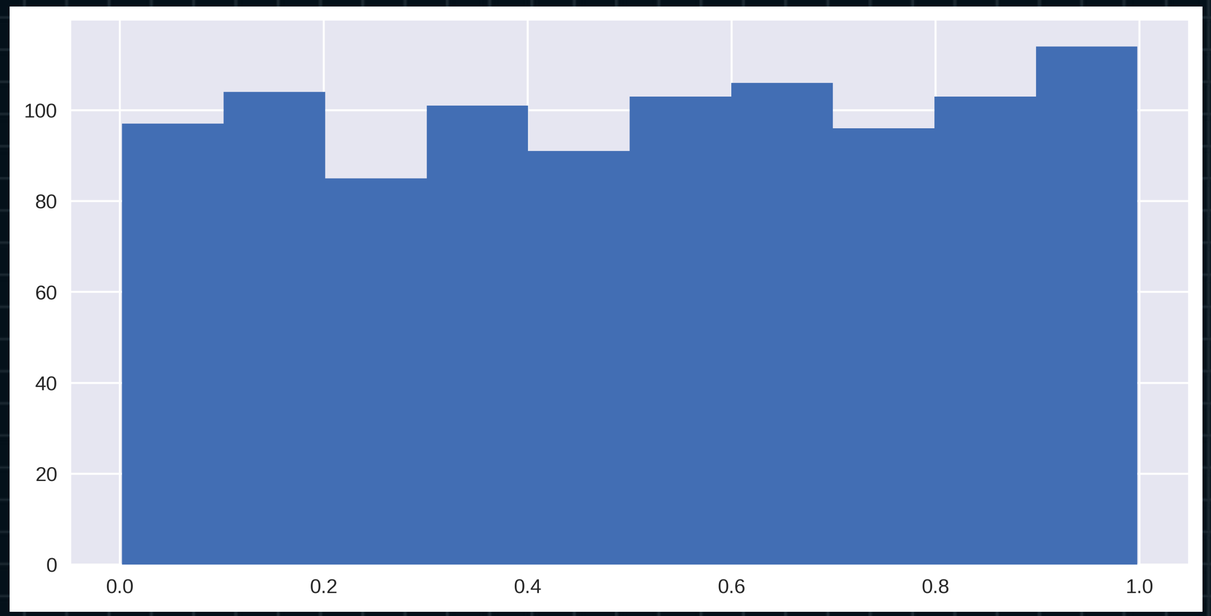
Lecture.3 Making ndarrays

- from Random Distributions

from Uniform Distributions

```
import numpy as np
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10, 5))

uniform = np.random.rand(1000)
ax.hist(uniform)
print(uniform.shape)
```

```
(1000,)
```



Lecture.3
Making ndarrays

- from Random Distributions

from Uniform Distributions

```
import numpy as np

uniform = np.random.rand(2, 3, 4)
print(uniform.shape)
```

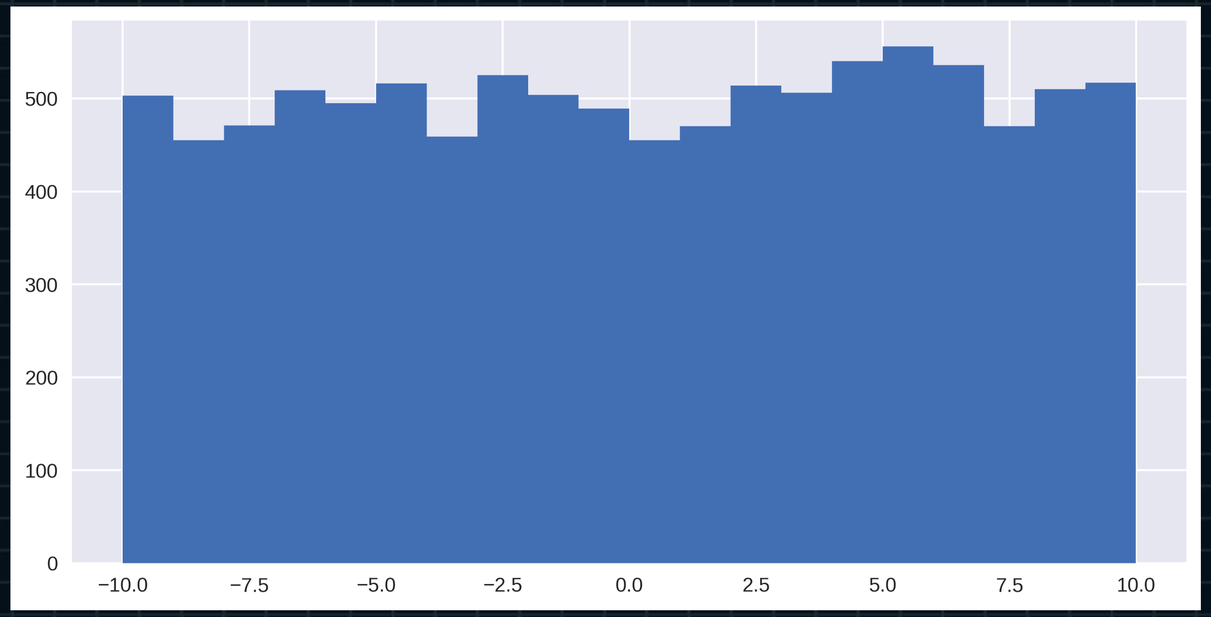
```
(2, 3, 4)
```

Lecture.3 Making ndarrays

- from Random Distributions

from Uniform Distributions

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn')

fig, ax = plt.subplots(figsize=(10, 5))


uniform = np.random.uniform(low=-10, high=10, size=(10000, ))

ax.hist(uniform, bins=20)
```



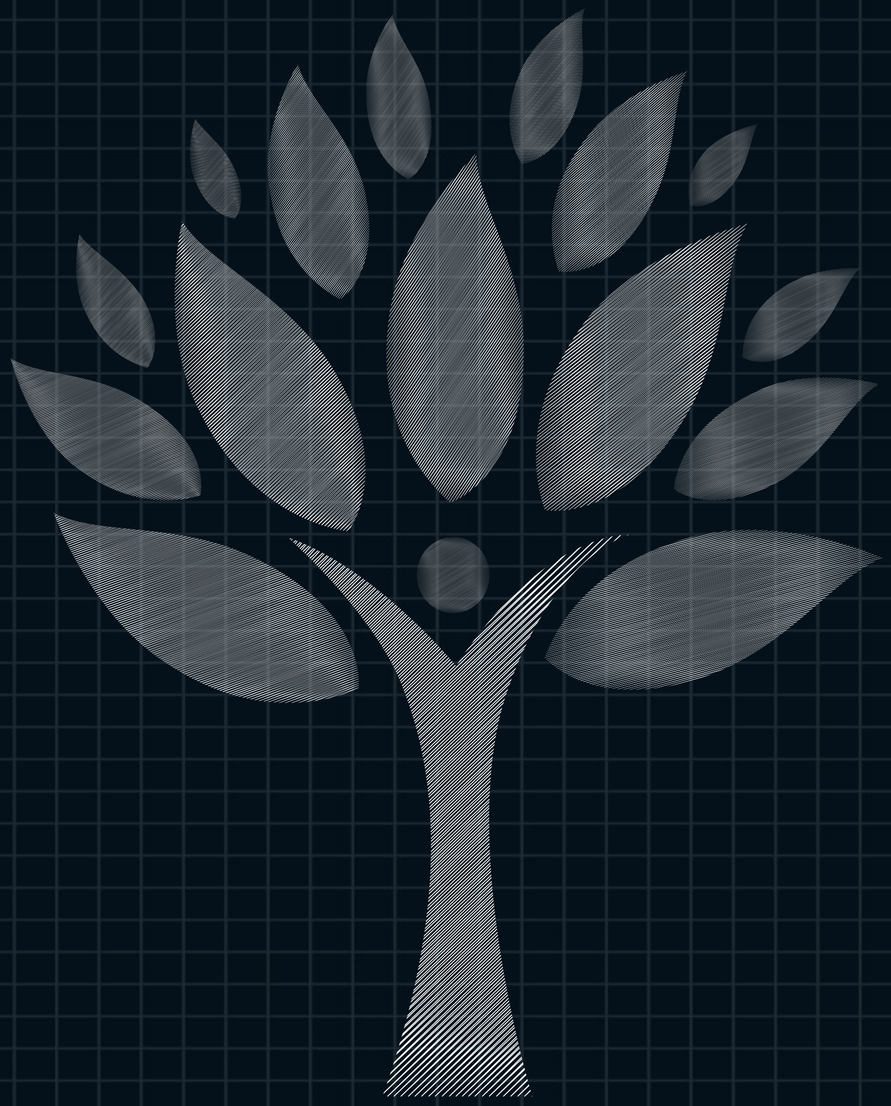
Lecture.3 Making ndarrays

- from Random Distributions

from Uniform Distributions

```
import numpy as np

randint = np.random.randint(low=0, high=7, size=(20, ))
print(randint)
```

```
[3 5 2 4 0 0 0 3 1 1 1 4 2 3 1 0 4 3 2 6]
```



NumPy Master Class

Lecture.3
Making ndarrays