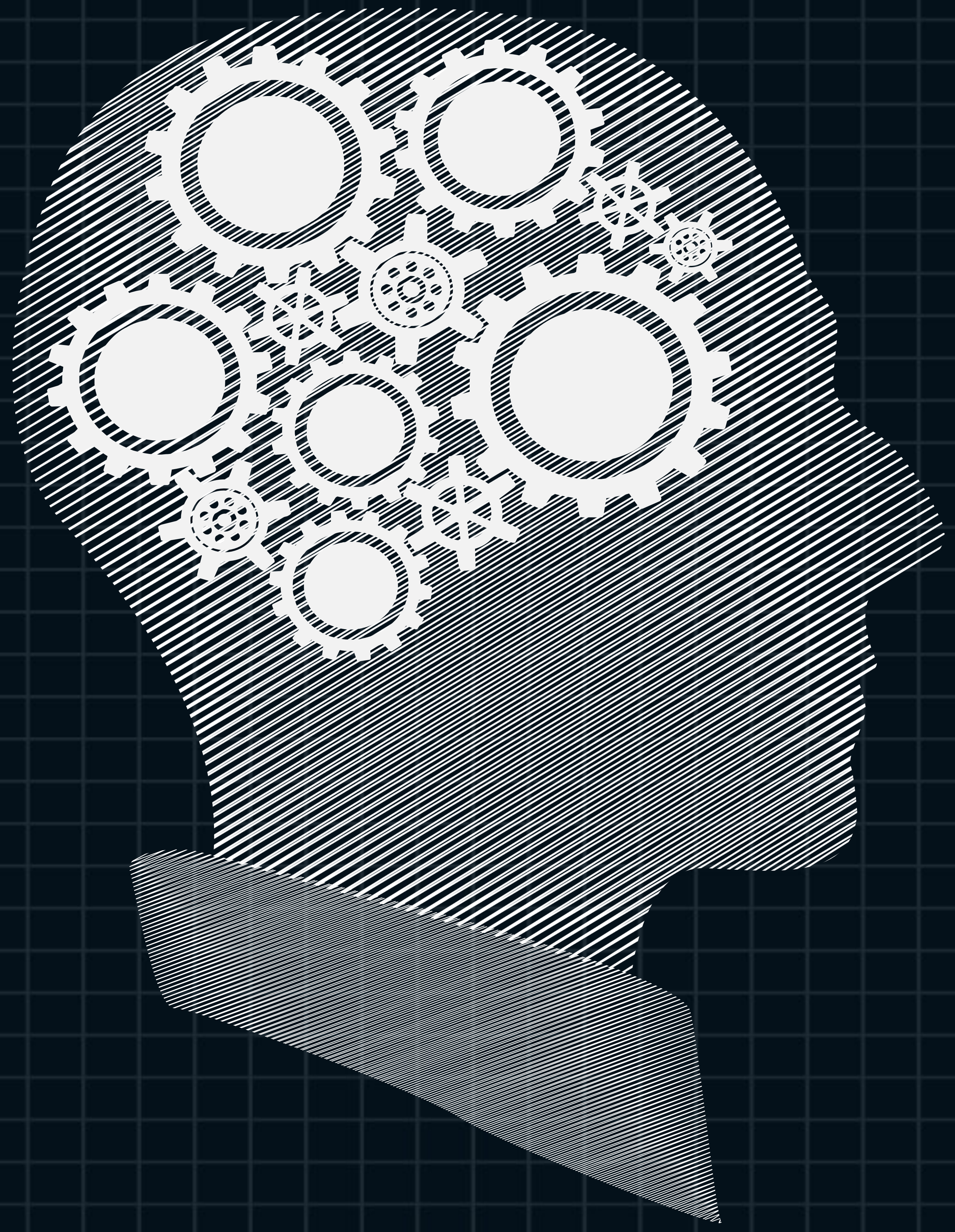


NumPy Master Class

Lecture.11
Mathematical Functions

- Trigonometric/Hyperbolic Functions

Constants

```python
import numpy as np

PI = np.pi
E = np.e

print("pi: ", PI)
print("natural constant: ", E)
```

```
pi:  3.141592653589793
natural constant:  2.718281828459045
```

- Trigonometric/Hyperbolic Functions

deg2rad and rad2deg

numpy.rad2deg(*x*)   **numpy.deg2rad(*x*)**

```
import numpy as np

degree = np.array([30, 45, 60, 90, 180, 360])
rad = np.deg2rad(degree)
degree = np.rad2deg(rad)

print("radian: ", rad.round(3))
print("degree again: ", degree)
```

```
radian:   [0.524 0.785 1.047 1.571 3.142 6.283]
degree again:  [ 30.  45.  60.  90. 180. 360.]
```

- Trigonometric/Hyperbolic Functions

Trigonometric Functions

numpy.sin(*x*)   **numpy.cos**(*x*)   **numpy.tan**(*x*)

```
import numpy as np

x = np.deg2rad(np.linspace(0, 360, 11))

sin, cos = np.sin(x), np.cos(x)
tan = np.tan(x)

print(f"np.tan: \n {tan.round(2)}")
print(f"np.sin/np.cos: \n {(sin/cos).round(2)}")
```

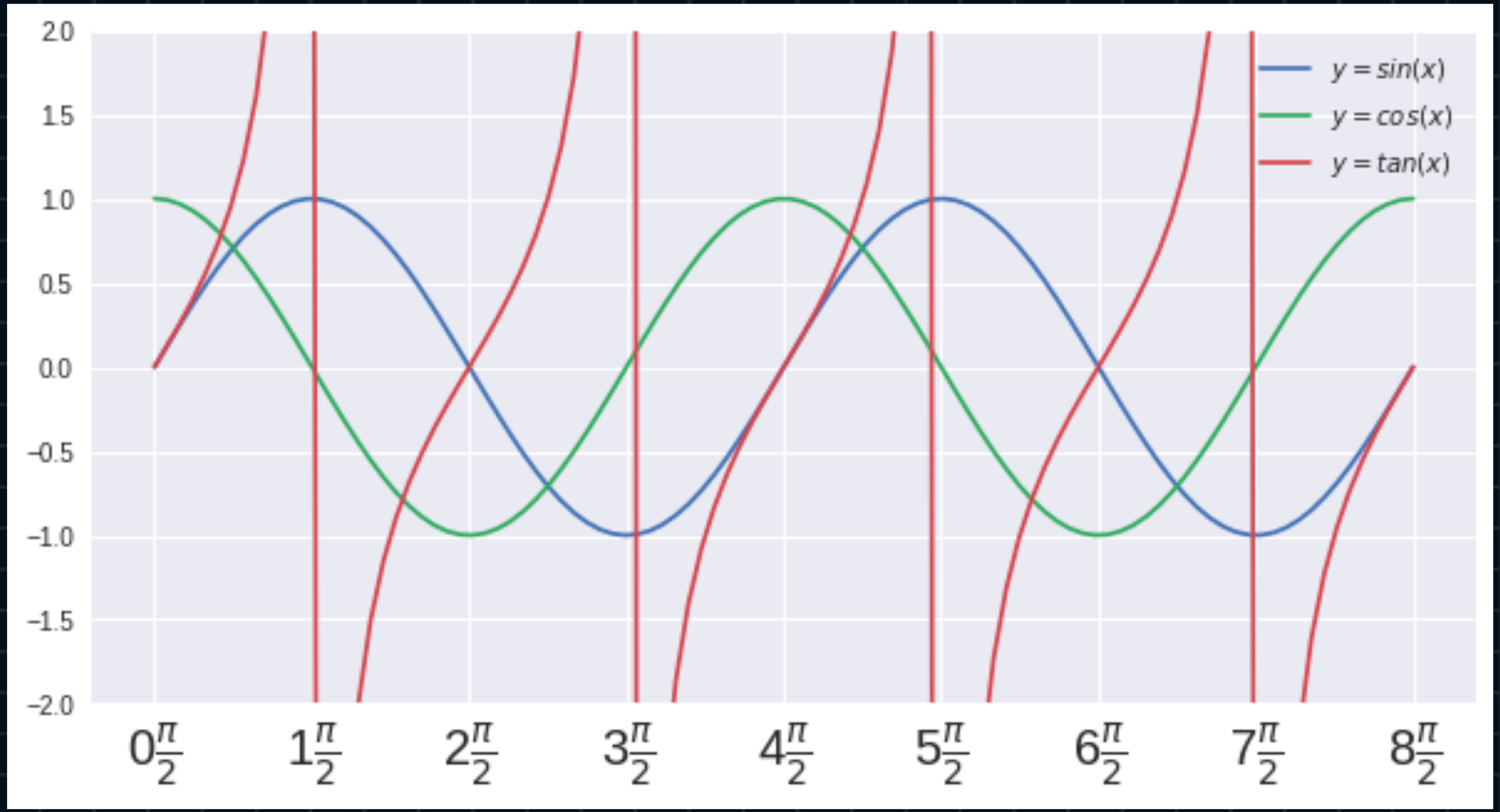
```
np.tan:
 [ 0.    0.73  3.08 -3.08 -0.73 -0.    0.73  3.08 -3.08 -0.73 -0.  ]
np.sin/np.cos:
 [ 0.    0.73  3.08 -3.08 -0.73 -0.    0.73  3.08 -3.08 -0.73 -0.  ]
```

Lecture.11
Mathematical Functions

- Trigonometric/Hyperbolic Functions

Trigonometric Functions

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

PI = np.pi

x = np.linspace(0, 4*PI, 100)


sin, cos, tan = np.sin(x), np.cos(x), np.tan(x)


fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(x, sin, label=r'$y = sin(x)$')
ax.plot(x, cos, label=r'$y = cos(x)$')
ax.plot(x, tan, label=r'$y = tan(x)$')

xticks = np.arange(0, 4*PI+0.1, 0.5*PI)
xticklabels = [str(xtick)+r'$\frac{\pi}{2}$' for xtick in range(9)]
ax.set_xticks(xticks)
ax.set_xticklabels(xticklabels)
ax.tick_params(axis='x', labelsize=20)
ax.set_ylim([-2, 2])
ax.legend()
```



- Trigonometric/Hyperbolic Functions

Exponential Functions

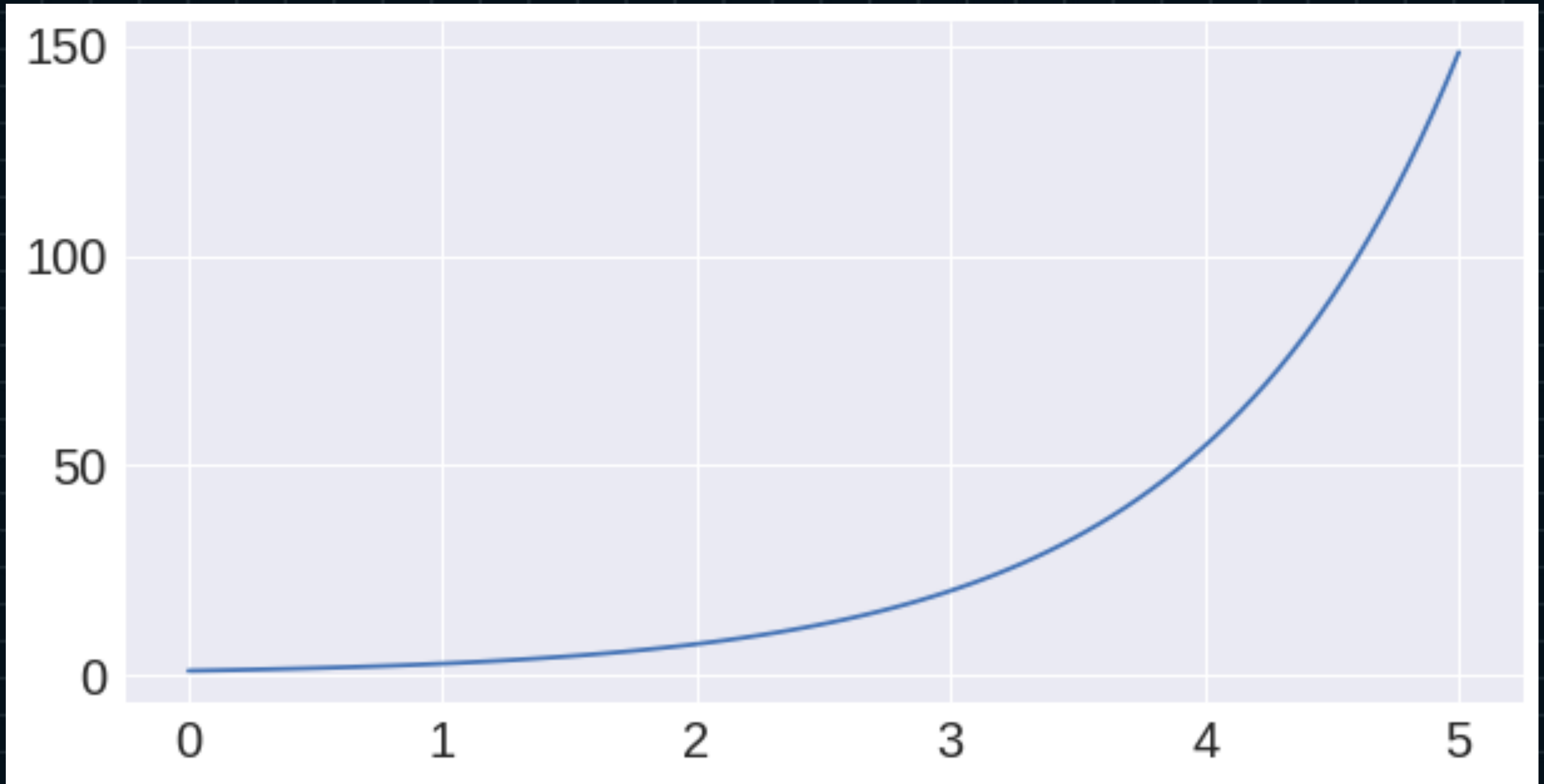
`numpy.exp(x)`

```python
import numpy as np

E = np.e
x = np.arange(1, 7)


print(f"E**x: \n {(E**x).round(2)}")
print(f"np.exp(x): \n {np.exp(x).round(2)}")
```

```
E**x:
 [  2.72   7.39  20.09  54.6  148.41 403.43]
np.exp(x):
 [  2.72   7.39  20.09  54.6  148.41 403.43]
```

Exponential Functions

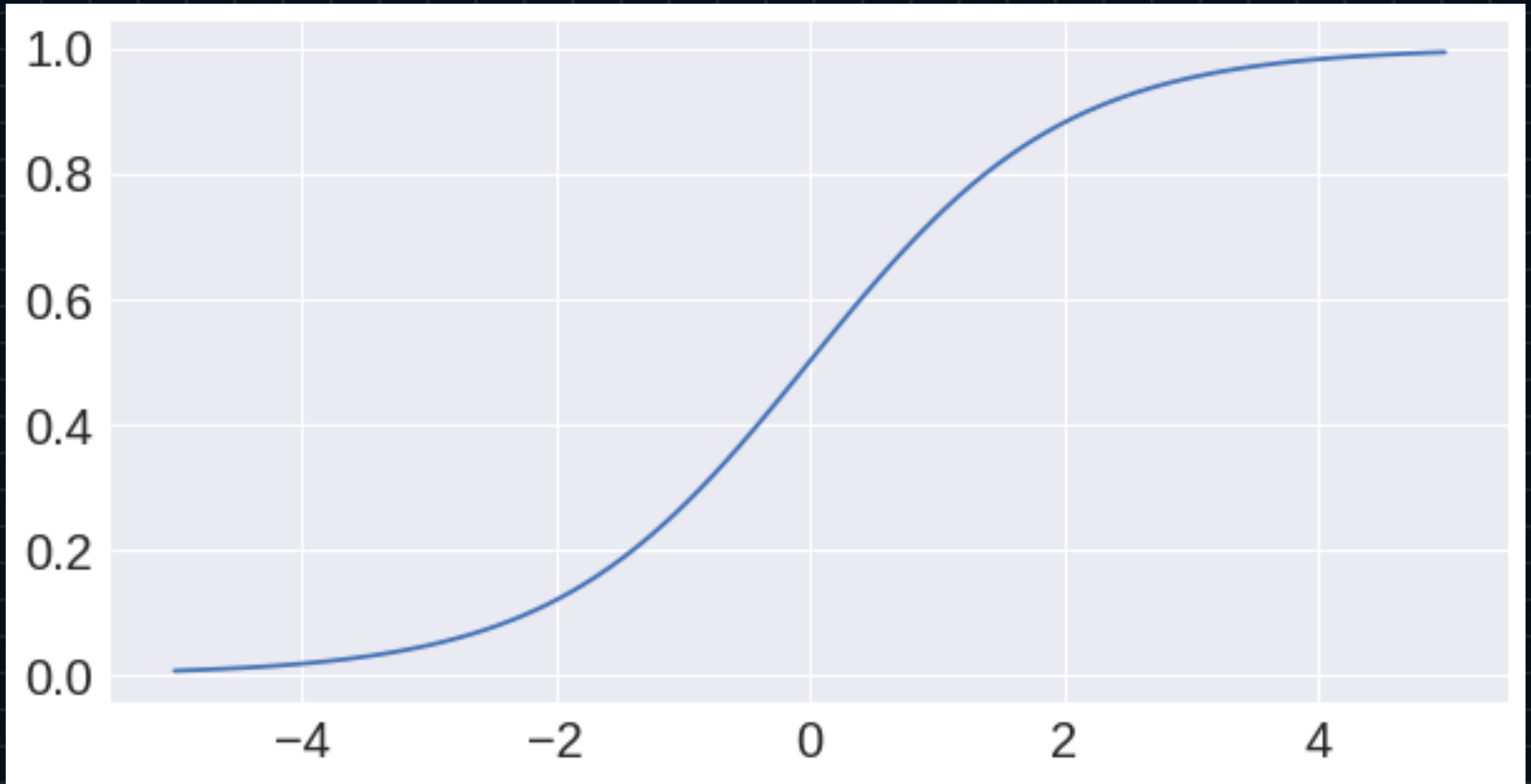
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 5, 100)
exp = np.exp(x)

fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(x, exp)
ax.tick_params(labelsize=20)
```



Exponential Functions

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

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

x = np.linspace(-5, 5, 100)
sigmoid = 1/(1 + np.exp(-x))

fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(x, sigmoid)
ax.tick_params(labelsize=20)
```



Lecture.11
Mathematical Functions

- Trigonometric/Hyperbolic Functions

Hyperbolic Functions

numpy.sinh(*x*) **numpy.cosh**(*x*) **numpy.tanh**(*x*)

```python
import numpy as np

x = np.linspace(0, 1, 5)

sinh, cosh = np.sinh(x), np.cosh(x)
tanh = np.tanh(x)

print(f"np.tanh: \n {tanh.round(2)}")
print(f"np.sinh/np.cosh: \n {(sinh/cosh).round(2)}")
```

```
np.tanh:
 [0.   0.24 0.46 0.64 0.76]
np.sinh/np.cosh:
 [0.   0.24 0.46 0.64 0.76]
```

- Trigonometric/Hyperbolic Functions

Hyperbolic Functions

```python
import numpy as np

x = np.linspace(0, 1, 5)

sinh = np.sinh(x)
sinh_exp = (np.exp(x) - np.exp(-x)) / 2

cosh = np.cosh(x)
cosh_exp = (np.exp(x) + np.exp(-x)) / 2

tanh = np.tanh(x)
tanh_exp = (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))

print(f"sinh: {sinh.round(2)}")
print(f"sinh_exp: {sinh_exp.round(2)}\n")

print(f"cosh: {cosh.round(2)}")
print(f"cosh_exp: {cosh_exp.round(2)}\n")

print(f”tanh: {tanh.round(2)}")
print(f"tanh_exp: \n {tanh_exp.round(2)}\n")
```

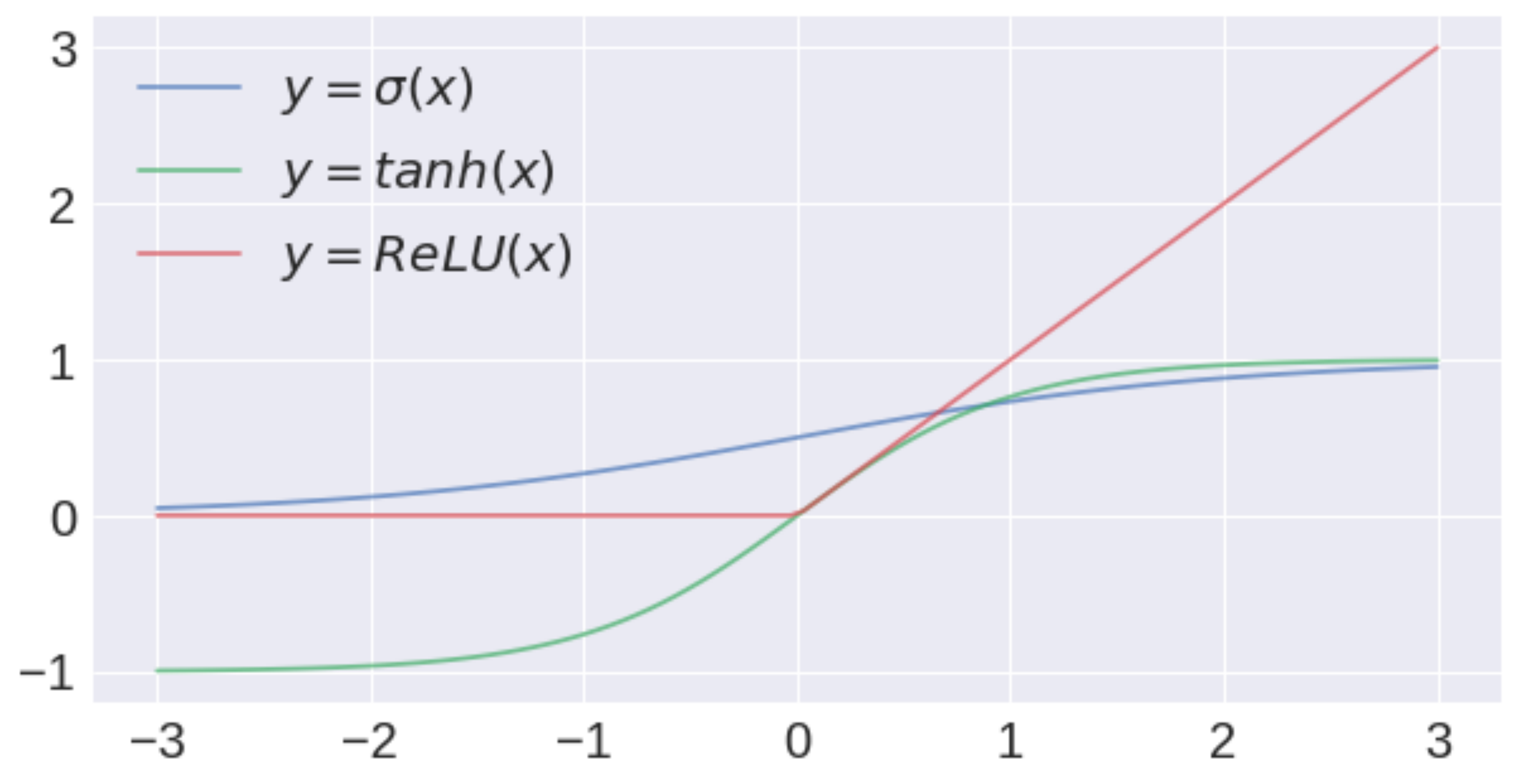
```
sinh: [0.   0.25 0.52 0.82 1.18]
sinh_exp: [0.   0.25 0.52 0.82 1.18]

cosh: [1.   1.03 1.13 1.29 1.54]
cosh_exp: [1.   1.03 1.13 1.29 1.54]

tanh: [0.   0.24 0.46 0.64 0.76]
tanh_exp: [0.   0.24 0.46 0.64 0.76]
```

- Trigonometric/Hyperbolic Functions

Hyperbolic Functions

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad ReLU(x) = max(0,x)$$

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


x = np.linspace(-3, 3, 100)
sigmoid = 1/(1 + np.exp(-x))
tanh = np.tanh(x)
relu = np.maximum(x, 0)


fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(x, sigmoid, label=r'$y = \sigma(x)$', alpha=0.7)
ax.plot(x, tanh, label=r'$y = tanh(x)$', alpha=0.7)
ax.plot(x, relu, label=r'$y = ReLU(x)$', alpha=0.7)

ax.tick_params(labelsize=20)
ax.legend(fontsize=20)
```



- Quadratic, Irrational, Rational Functions

Quadratic Functions

$$y = ax^2 + bx + c$$

`numpy.square(x)`

```
import numpy as np

a = np.random.randint(0, 10, (10, ))

square1 = a*a
square2 = a**2
square3 = np.square(a)

print(f"a: \n{a}\n")

print(f"a*a: \n {square1}")
print(f"a**2: \n {square2}")
print(f"np.square(a): \n {square3}")
```

```
a:
[4 7 2 3 0 9 1 4 8 3]

a*a:
 [16 49  4  9  0 81  1 16 64  9]
a**2:
 [16 49  4  9  0 81  1 16 64  9]
np.square(a):
 [16 49  4  9  0 81  1 16 64  9]
```

Irrational Functions

$$y = \sqrt{x}, y = \sqrt[3]{x}$$

`numpy.sqrt(x)` `numpy.cbrt(x)`

```python
import numpy as np

a = np.random.randint(0, 10, (4, ))

sqrt1 = a**(1/2)
sqrt2 = np.sqrt(a)

cbrt1 = a**(1/3)
cbrt2 = np.cbrt(a)

print(f"a: \n {a}\n")

print(f"a**(1/2): \n {sqrt1.round(2)}")
print(f"np.sqrt(a): \n {sqrt2.round(2)}\n")

print(f"a**(1/3): \n {cbrt1.round(2)}")
print(f"np.cbrt(a): \n {cbrt2.round(2)}")
```

```
a:
 [7 0 7 0]

a**(1/2):
 [2.65 0.   2.65 0.  ]
np.sqrt(a):
 [2.65 0.   2.65 0.  ]

a**(1/3):
 [1.91 0.   1.91 0.  ]
np.cbrt(a):
 [1.91 0.   1.91 0.  ]
```

- Quadratic, Irrational, Rational Functions

Rational Functions

$$y = \frac{1}{x} = x^{-1} \qquad y = \frac{1}{x^2} = x^{-2}$$

`numpy.reciprocal(x)`

```python
import numpy as np

a = np.random.uniform(0, 10, (4, ))

recip1 = 1/a
recip2 = a**(-1)
recip3 = np.reciprocal(a)

print(f"a: \n {a.round(2)}\n")

print(f"1/a: \n {recip1.round(2)}")
print(f"a**(-1): \n {recip2.round(2)}")
print(f"np.reciprocal(a): \n {recip3.round(2)}")
```

```
a:
 [9.   5.01 9.   0.66]

1/a:
 [0.11 0.2  0.11 1.53]
a**(-1):
 [0.11 0.2  0.11 1.53]
np.reciprocal(a):
 [0.11 0.2  0.11 1.53]
```

Rational Functions

$$y = \frac{1}{x^2}, \quad z = \frac{1}{\sqrt{x}}$$

```python
import numpy as np

a = np.random.uniform(0, 10, (4, ))

y1 = a**(-2)
y2 = np.reciprocal(np.square(a))

z1 = a**(-1/2)
z2 = np.reciprocal(np.sqrt(a))

print(f"a: \n {a.round(2)}\n")

print(f"y1: \n {y1.round(2)}")
print(f"y2: \n {y2.round(2)}\n")

print(f"z1: \n {z1.round(2)}")
print(f"z2: \n {z2.round(2)}")
```

```
a:
 [8.34 5.42 7.25 2.62]

y1:
 [0.01 0.03 0.02 0.15]
y2:
 [0.01 0.03 0.02 0.15]

z1:
 [0.35 0.43 0.37 0.62]
z2:
 [0.35 0.43 0.37 0.62]
```

Power Functions

`numpy.power(x1, x2)`

```
import numpy as np

a = np.random.uniform(0, 5, (4, ))

s1 = np.square(a).round(2)
s2 = (a**2).round(2)
s3 = np.power(a, 2).round(2)

re1 = np.reciprocal(a).round(2)
re2 = (a**(-1)).round(2)
re3 = np.power(a, -1).round(2)

print("square")
print(f"{s1}\n{s2}\n{s3}\n")

print("reciprocal")
print(f"{re1}\n{re2}\n{re3}")
```

```
square
[ 1.95 14.44  4.82  0.57]
[ 1.95 14.44  4.82  0.57]
[ 1.95 14.44  4.82  0.57]

reciprocal
[0.72 0.26 0.46 1.33]
[0.72 0.26 0.46 1.33]
[0.72 0.26 0.46 1.33]
```

Power Functions

$$y = 3x^3 - 2x^2 + x - 2$$

```
import numpy as np

x = np.random.uniform(0, 5, (4, ))

y1 = 3*x**3 - 2*x**2 + x - 2
y2 = 3*np.power(x, 3) -2*np.power(x, 2) + x - 2

print(f"y1: \n {y1.round(2)}")
print(f"y2: \n {y2.round(2)}")
```

```
y1:
 [ 20.61  98.64 173.22  -1.72]
y2:
 [ 20.61  98.64 173.22  -1.72]
```

Power Functions

```python
import numpy as np

a = np.random.uniform(0, 5, (4, ))

sqrt1 = np.sqrt(a).round(2)
sqrt2 = (a**(1/2)).round(2)
sqrt3 = np.power(a, (1/2)).round(2)

cbrt1 = np.cbrt(a).round(2)
cbrt2 = (a**(1/3)).round(2)
cbrt3 = np.power(a, (1/3)).round(2)

print("sqrt")
print(f"{sqrt1}\n{sqrt2}\n{sqrt3}\n")

print("cbrt")
print(f"{cbrt1}\n{cbrt2}\n{cbrt3}")
```

```
sqrt
[2.15 1.88 2.08 0.47]
[2.15 1.88 2.08 0.47]
[2.15 1.88 2.08 0.47]

cbrt
[1.66 1.52 1.63 0.61]
[1.66 1.52 1.63 0.61]
[1.66 1.52 1.63 0.61]
```

Power Functions

```
import numpy as np

a = np.random.uniform(0, 5, (4, ))

exp1 = np.exp(a).round(2)
exp2 = (np.e**a).round(2)
exp3 = np.power(np.e, a).round(2)

print(f"{exp1}\n{exp2}\n{exp3}\n")
```

```
[  1.26 105.55  60.57   2.75]
[  1.26 105.55  60.57   2.75]
[  1.26 105.55  60.57   2.75]
```

Power Functions

```python
import numpy as np

a = np.random.uniform(0, 5, (4, ))
b = np.random.uniform(0, 5, (4, ))

power1 = (a**b).round(2)
power2 = np.power(a, b).round(2)

print(f"{power1}\n{power2}")
```

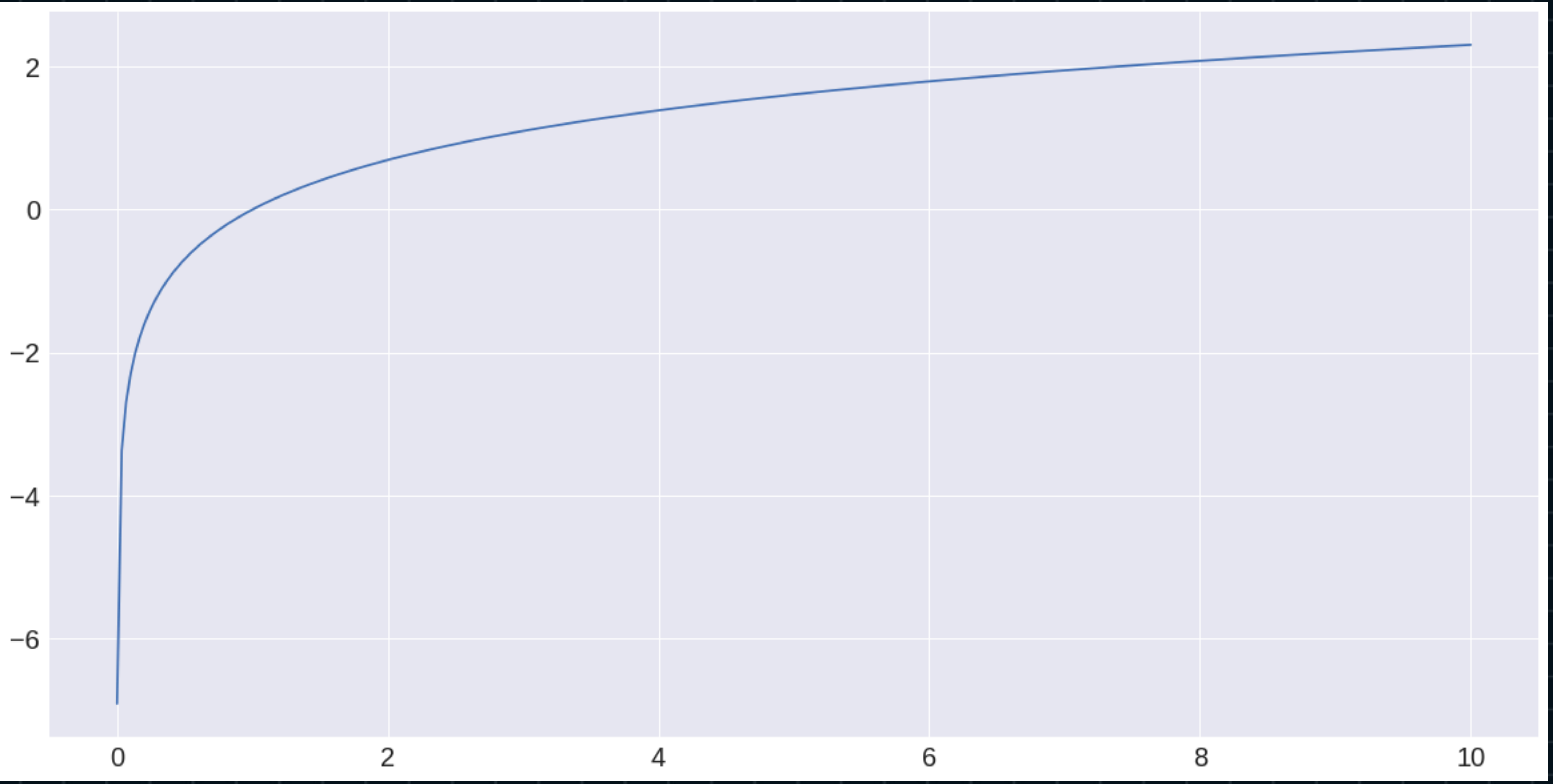
```
[0.44 6.62 0.75 9.43]
[0.44 6.62 0.75 9.43]
```

- Logarithmic Functions

Log Functions

`numpy.log(x)`

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

x = np.linspace(0.001, 10, 300)
log = np.log(x)


fig, ax = plt.subplots(figsize=(20, 10))
ax.plot(x, log)
ax.tick_params(labelsize=20)
```



- Logarithmic Functions

Log Functions

```
import numpy as np

a = np.random.uniform(1, 5, (4, ))


log = np.log(a)
exp = np.exp(log)

print(f"a: \n {a.round(3)}")
print(f"log: \n {log.round(3)}")
print(f"exp: \n {exp.round(3)}")
```

```
a:
 [1.67  2.406 1.06  4.983]
log:
 [0.513 0.878 0.059 1.606]
exp:
 [1.67  2.406 1.06  4.983]
```

- Logarithmic Functions

Properties of Log

$$log(a) + log(b) = log(ab)$$

```
import numpy as np

a = np.random.uniform(1, 5, (4, ))
b = np.random.uniform(1, 5, (4, ))


print((np.log(a) + np.log(b)).round(3))
print(np.log(a*b).round(3))
```

```
[1.854 1.767 2.461 1.536]
[1.854 1.767 2.461 1.536]
```

- Logarithmic Functions

Properties of Log

$$log_a b = \frac{log_c b}{log_c a}$$

```python
import numpy as np

a = np.random.uniform(1, 5, (4, ))

log2 = np.log(a)/np.log(2)
log3 = np.log(a)/np.log(3)
log5 = np.log(a)/np.log(5)

print(f"log2: \n {log2.round(3)}")
print(f"log3: \n {log3.round(3)}")
print(f"log5: \n {log5.round(3)}")
```

```
log2:
 [1.196 1.422 1.569 2.23 ]
log3:
 [0.755 0.897 0.99  1.407]
log5:
 [0.515 0.612 0.676 0.96 ]
```

- Logarithmic Functions

Binary Entropy

$$\mathcal{L}_e = -\left[plog_e(p) + (1-p)log_e(1-p)\right]$$

$$\mathcal{L}_2 = -\left[plog_2(p) + (1-p)log_2(1-p)\right]$$

```
import numpy as np

p = np.random.uniform(0, 1, (4, ))

be_e = -(p*np.log(p) + (1-p)*np.log(1-p))
be_2 = -(p*np.log(p)/np.log(2) + (1-p)*np.log(1-p)/np.log(2))

print(f"probability: \n {p.round(2)}")
print(f"binary entropy with base e: \n {be_e.round(2)}")
print(f"binary entropy with base 2: \n {be_2.round(2)}")
```

```
probability:
 [0.32 0.88 0.1  0.88]
binary entropy with base e:
 [0.62 0.38 0.32 0.37]
binary entropy with base 2:
 [0.9  0.54 0.45 0.54]
```

NumPy Master Class

Lecture.11
Mathematical Functions