פונקציות מתמטיות

Array

List Comprehension

```python
import numpy as np
x = np.linspace(-5,5,11)

y = []
for val in x:
    y.append(val**2)

y = [val**2 for val in x]
```

Numpy - when all elements of the list have the same type

```
import numpy as np
x = np.linspace(-5,5,11)

y = []
for val in x:
   y.append(val**2)

y = x**2
```

Numpy - Array

```
import numpy as np
x = np.linspace(-5,5,11)
```

array([-5., -4., -3., -2., -1., 0., 1., 2., 3., 4., 5.])

```
y = x**2
```

array([25., 16., 9., 4., 1., 0., 1., 4., 9., 16., 25.])

Numpy - Functions

```
import numpy as np
x = np.linspace(-5,5,11)

y = np.sin(x)
```

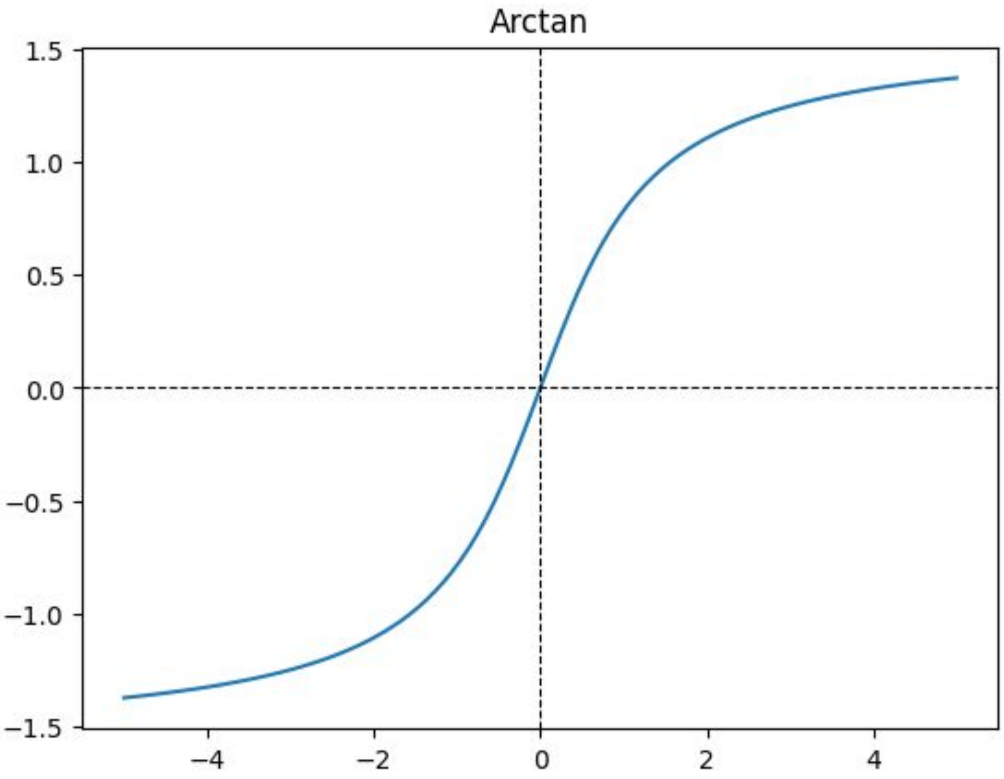
array([-5., -4., -3., -2., -1., 0., 1., 2., 3., 4., 5.])

array([0.95892427, 0.7568025 , -0.14112001, -0.90929743, -0.84147098,   0., 0.84147098, …])

Integration

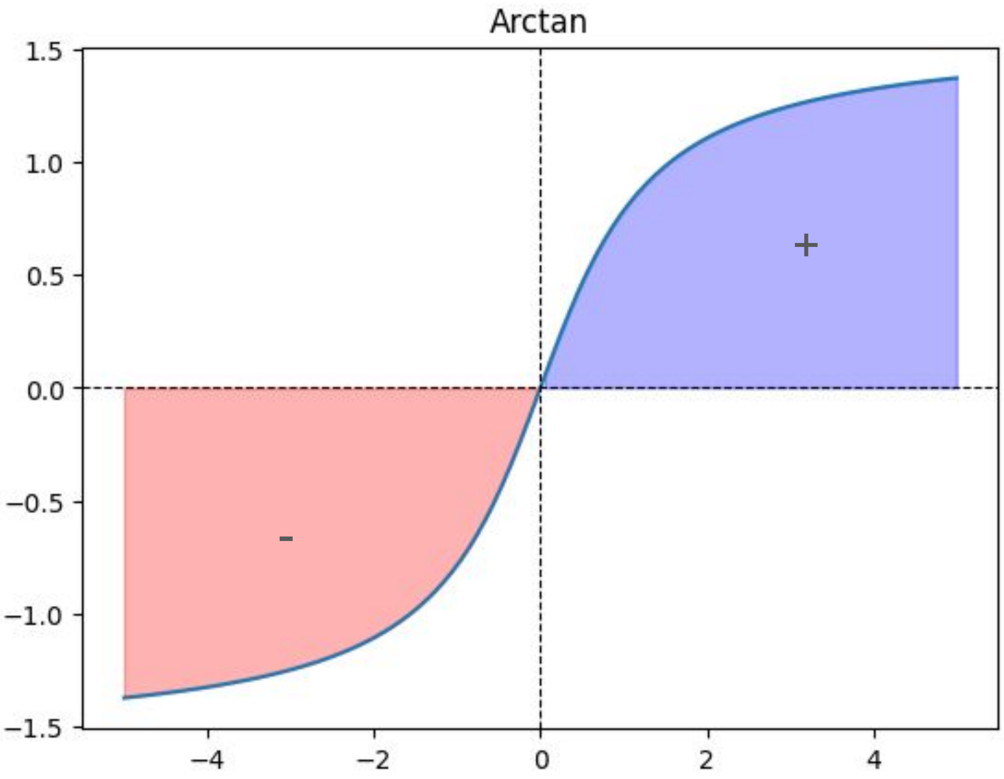
ArcTan(x)

Integrate ArcTan



$$f(x) = \arctan(x)$$

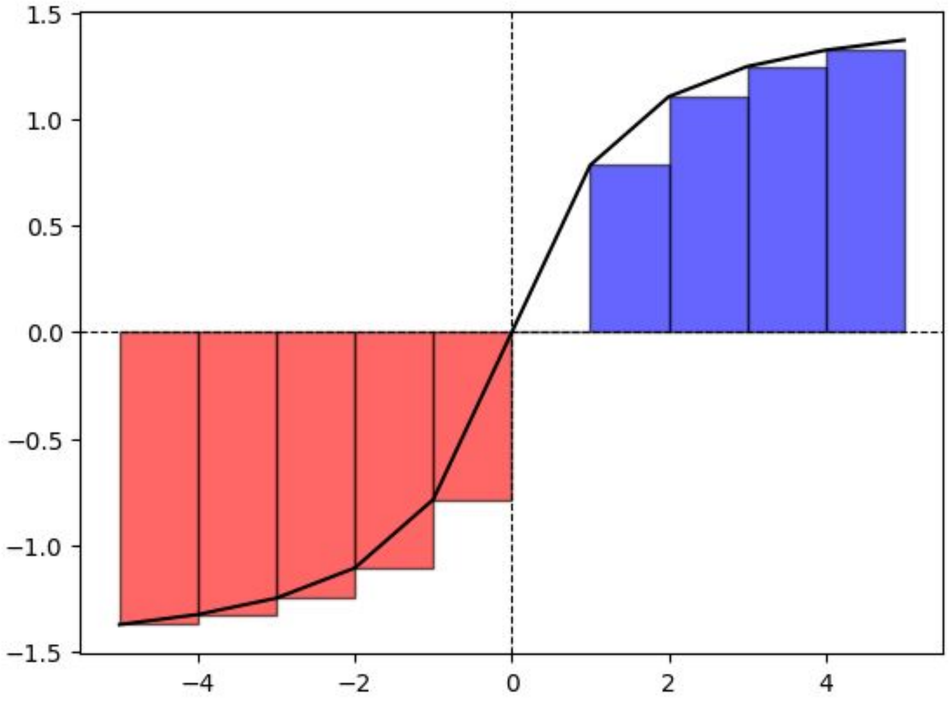
Integrate ArcTan



$$f(x) = \arctan(x)$$

$$\int f(x)dx =$$

Integrate ArcTan



$$f(x) = \arctan(x)$$

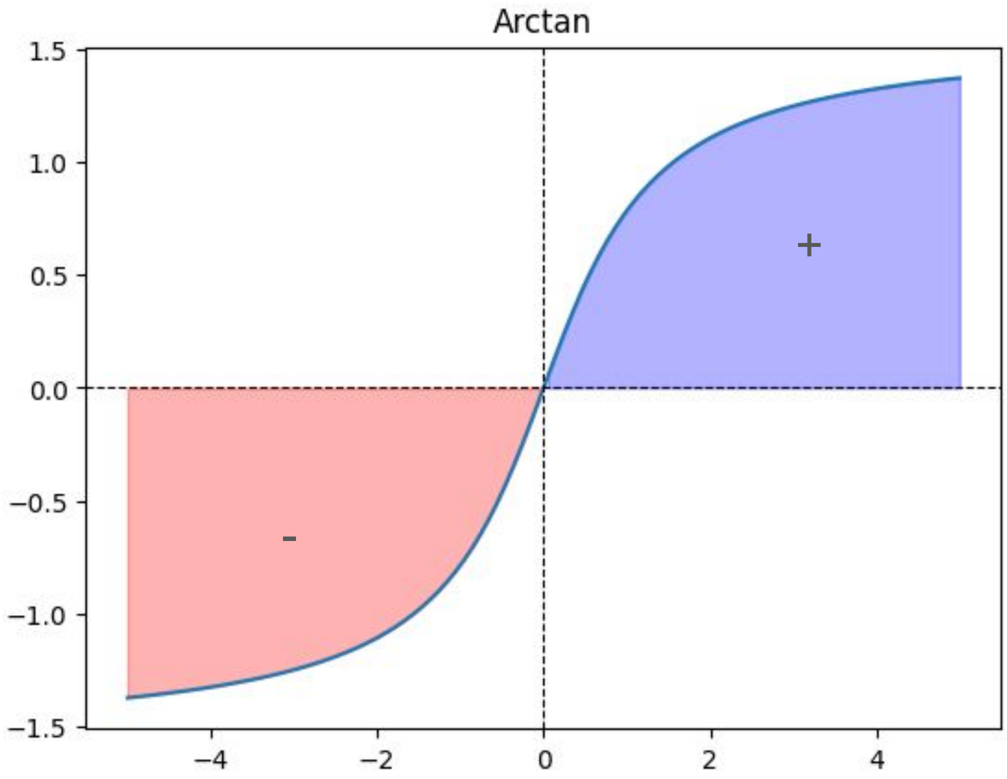
$$\int f(x)dx$$

```python
import numpy as np
from scipy.integrate import trapezoid

x = np.linspace(-5, 5, 10)
y = np.arctan(x)
integral = trapezoid(y, x)
```

↓

2.220446049250313e-16

Integrate ArcTan



$$f(x) = \arctan(x)$$

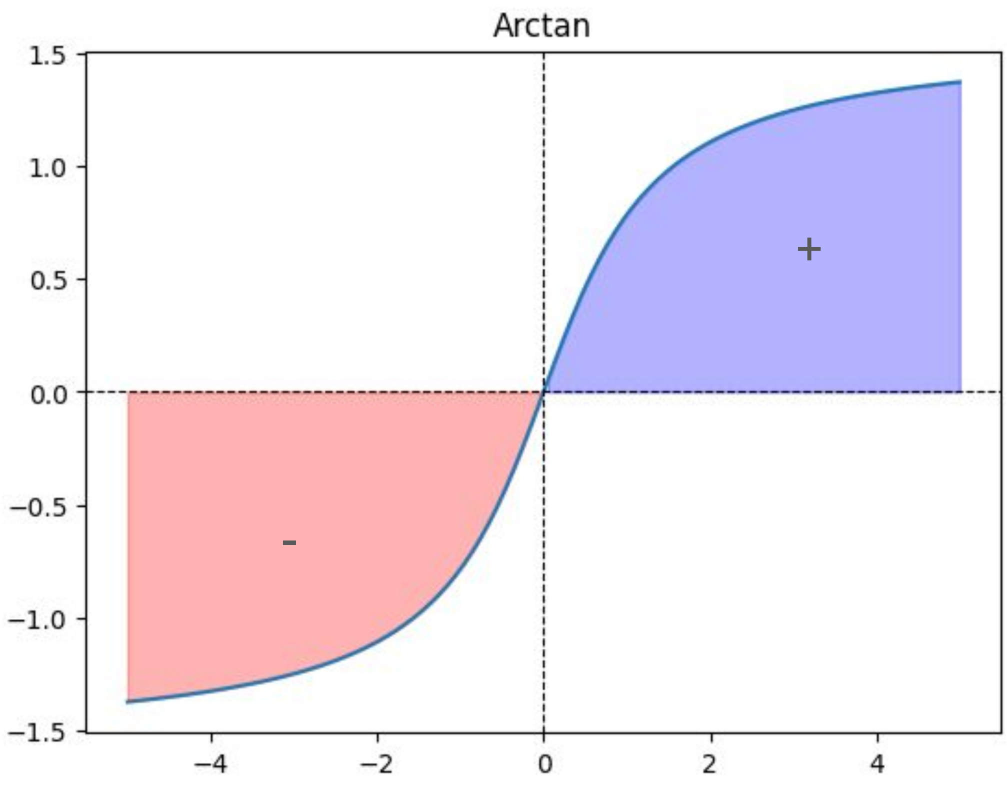
$$\int f(x)dx$$

```
from scipy.integrate import quad
```

```
integral, error = quad(np.arctan, -5, 5)
```

0.0 1.1524703870869896e-13

Integrate ArcTan



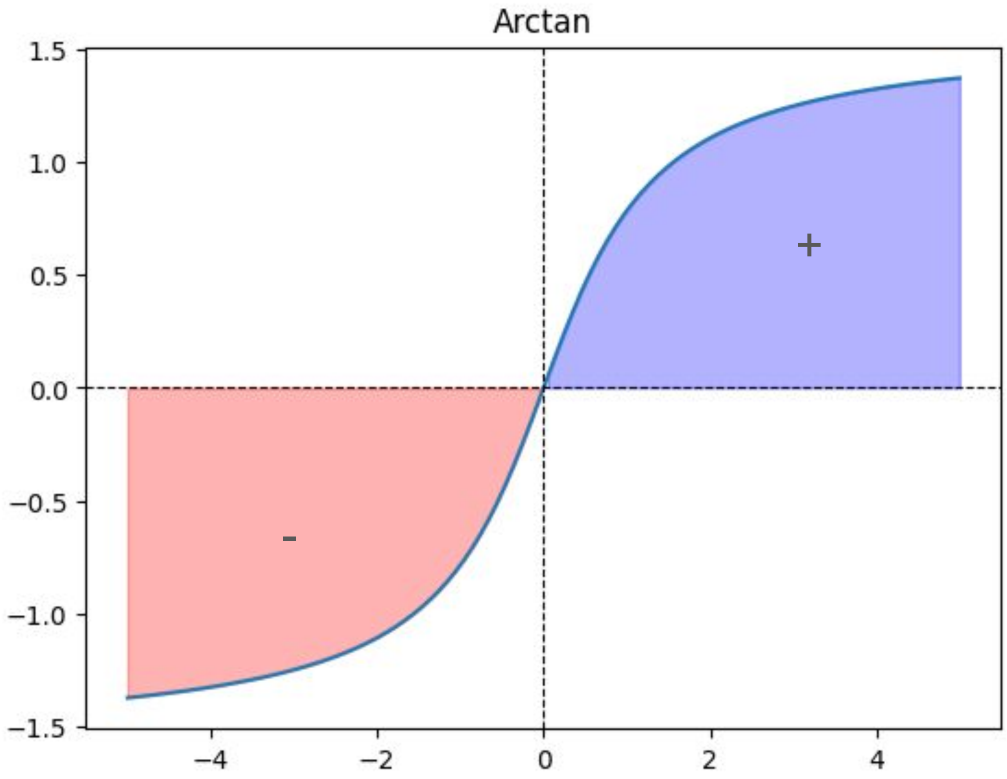
$$f(x) = \arctan(x)$$

$$\int f(x)dx$$

```
import numpy as np
from scipy.integrate import quad

integral, error = quad(np.arctan, -5, 5)
```

0.0 1.152470387086989 6e-13

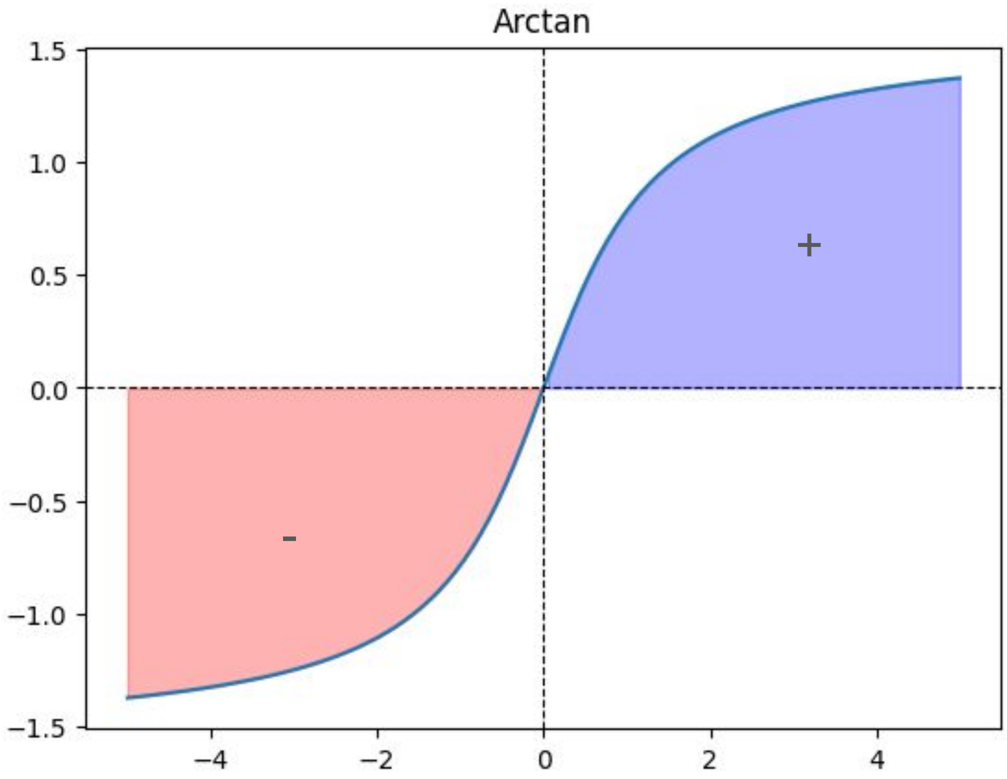
Integrate ArcTan




```
import numpy as np
from scipy.integrate import quad

integral1, _ = quad(np.arctan, -5, 0)
integral2, _ = quad(np.arctan, 0, 5)
integral = integral1 + integral2
```

0.0 -5.237 5.237

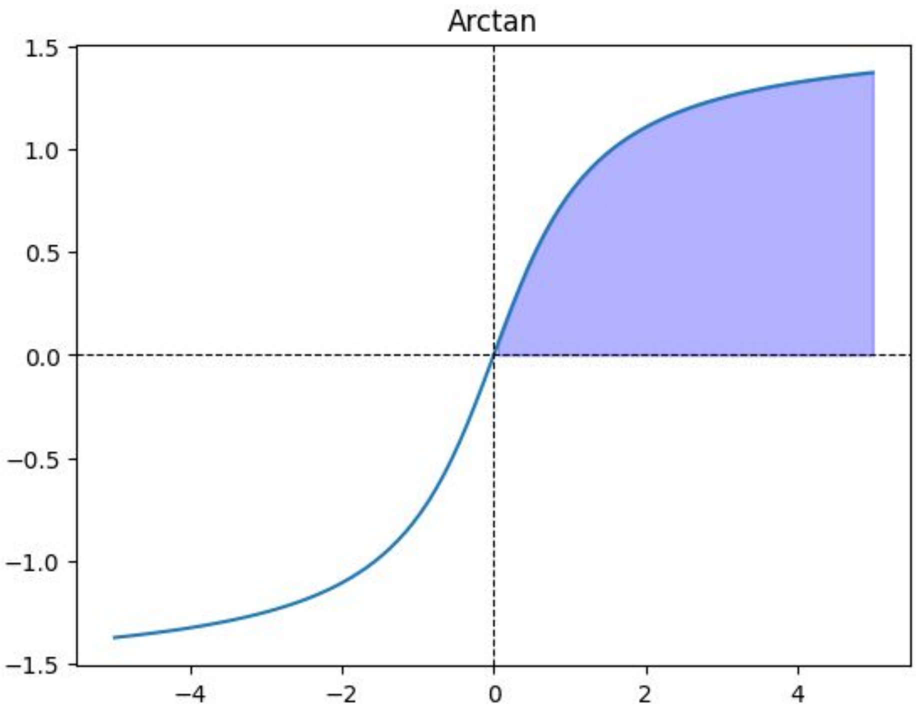
Integrate ArcTan - Exact Solution




```
import sympy as sp

x = sp.symbols('x')
f = sp.atan(x)

integral_f = sp.integrate(f, x)
```

```
x*atan(x) - log(x**2 + 1)/2
```

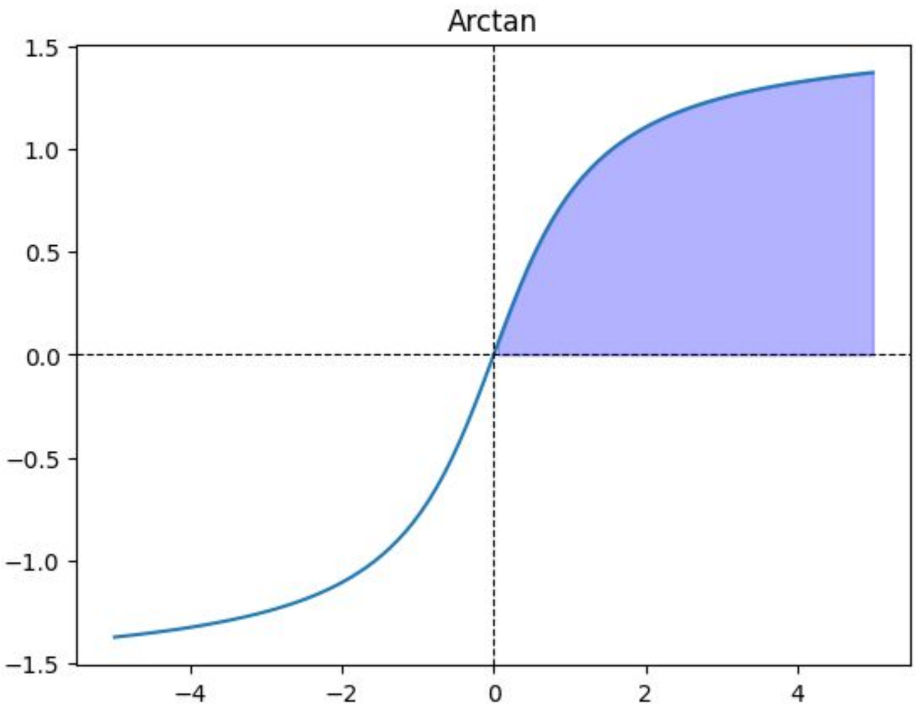
Integrate ArcTan - Exact Solution




```python
import sympy as sp

x = sp.symbols('x')
f = sp.atan(x)

integral = sp.integrate(f, (x, 0, 5))
```

$$-\frac{\log(26)}{2} + 5\,\mathrm{atan}(5)$$

Integrate ArcTan - Exact Solution




```python
import sympy as sp

x = sp.symbols('x')
f = sp.atan(x)

integral = sp.integrate(f, (x, 0, 5))
```

$$-\frac{\log(26)}{2} + 5\,\mathrm{atan}(5)$$

```python
numeric_value = integral.evalf()
```

$$5.23795556571434$$

Optimization

X^X

Define the function

```
def my_func(x):
    return x**x
```

```
my_func(0.5)
```

0.7071067811865476

Define the function: Lambda

```
def my_func(x):
    return x**x
```

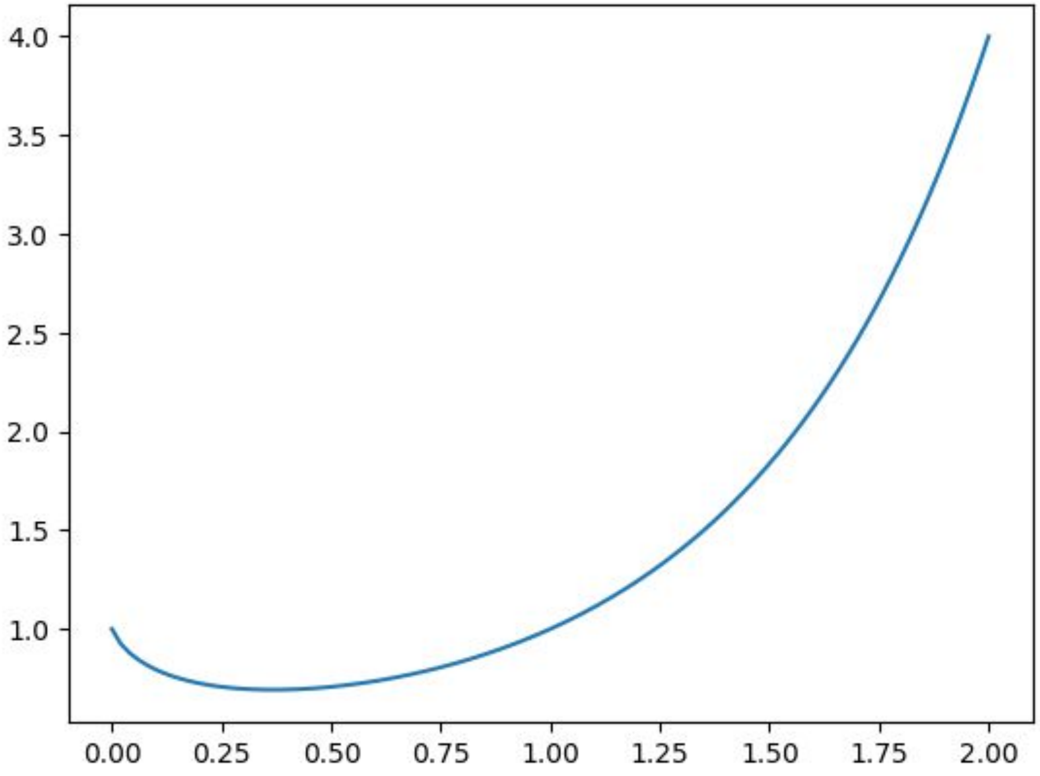
```
my_func = lambda x: x**x
```

```
my_func(0.5)
```

0.7071067811865476

Define the function

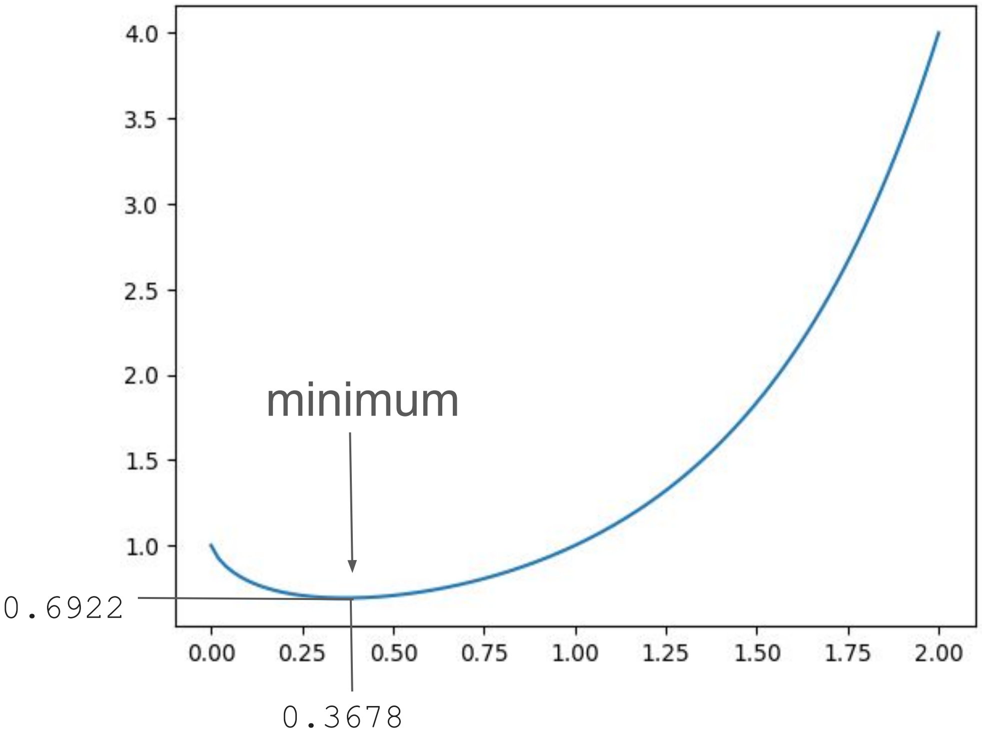
```
my_func = lambda x: x**x

x = np.linspace(0, 2, 100)
y = my_func(x)

plt.plot(x, y)
```



Find the minimum

```python
from scipy.optimize import minimize

my_func = lambda x: x**x
result = minimize(my_func, 0.5)
minimum_x = result.x[0]
minimum_y = result.fun
```

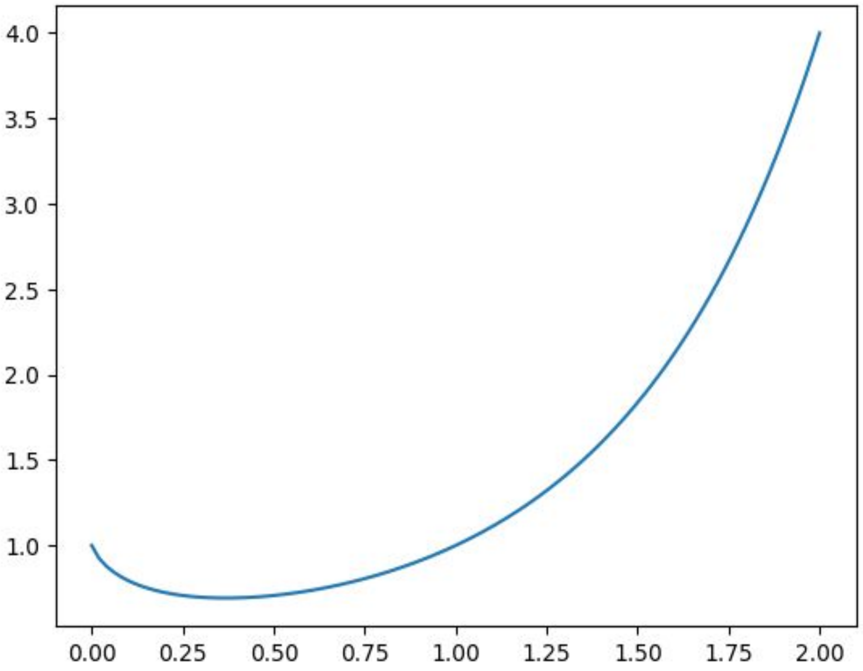
0.3678 0.6922



Find the minimum: Different initial value

```
from scipy.optimize import minimize

my_func = lambda x: x**x
result = minimize(my_func, 1)
minimum_x = result.x[0]
```

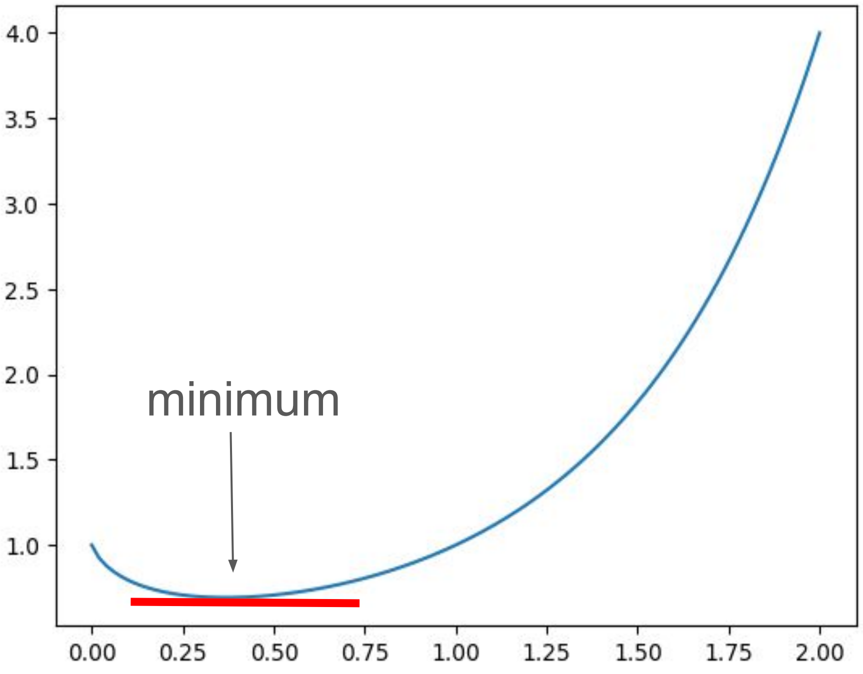
```
-1023.00001525
```



Find the minimum: Derivative value

```
from scipy.misc import derivative
from scipy.optimize import minimize

my_func = lambda x: x**x
result = minimize(my_func, 0.5)
minimum_x = result.x[0]

dfdx = lambda x:
     derivative(my_func, x, dx=1e-6)
minimum_df = dfdx(minimum_x)
```

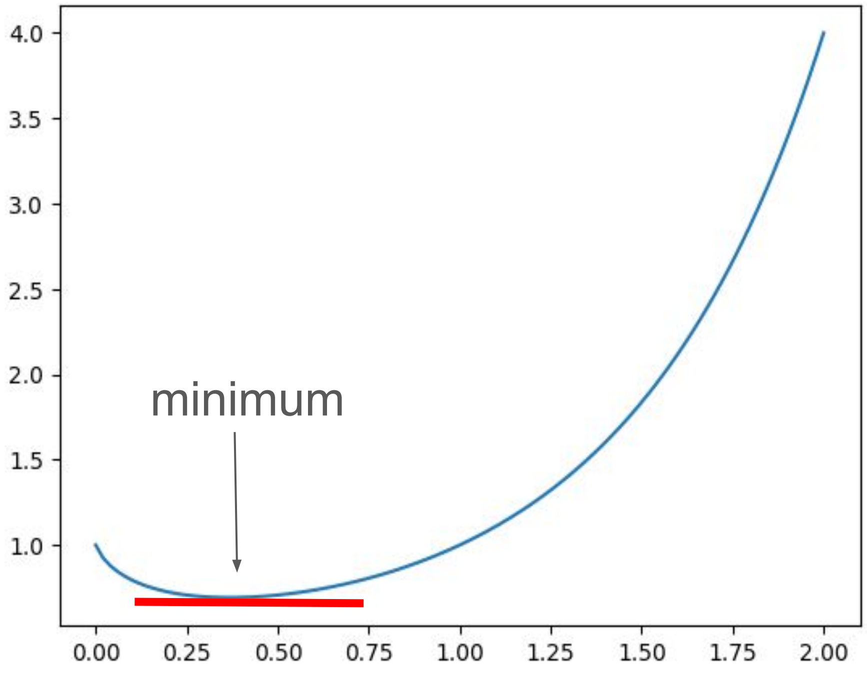
-1.717e-06



Find the minimum: Second Derivative value

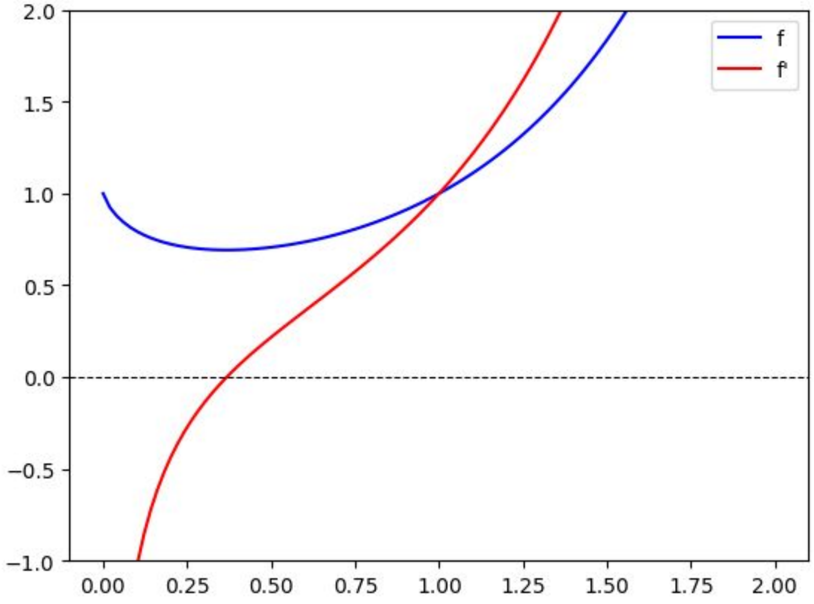
```
from scipy.misc import derivative
from scipy.optimize import minimize

my_func = lambda x: x**x
result = minimize(my_func, 0.5)
minimum_x = result.x[0]

dfdx = lambda x:
     derivative(my_func, x, dx=1e-6)
df2dx = lambda x:
     derivative(dfdx, x, dx=1e-6)
minimum_df2 = df2dx(minimum_x
```

1.8816059



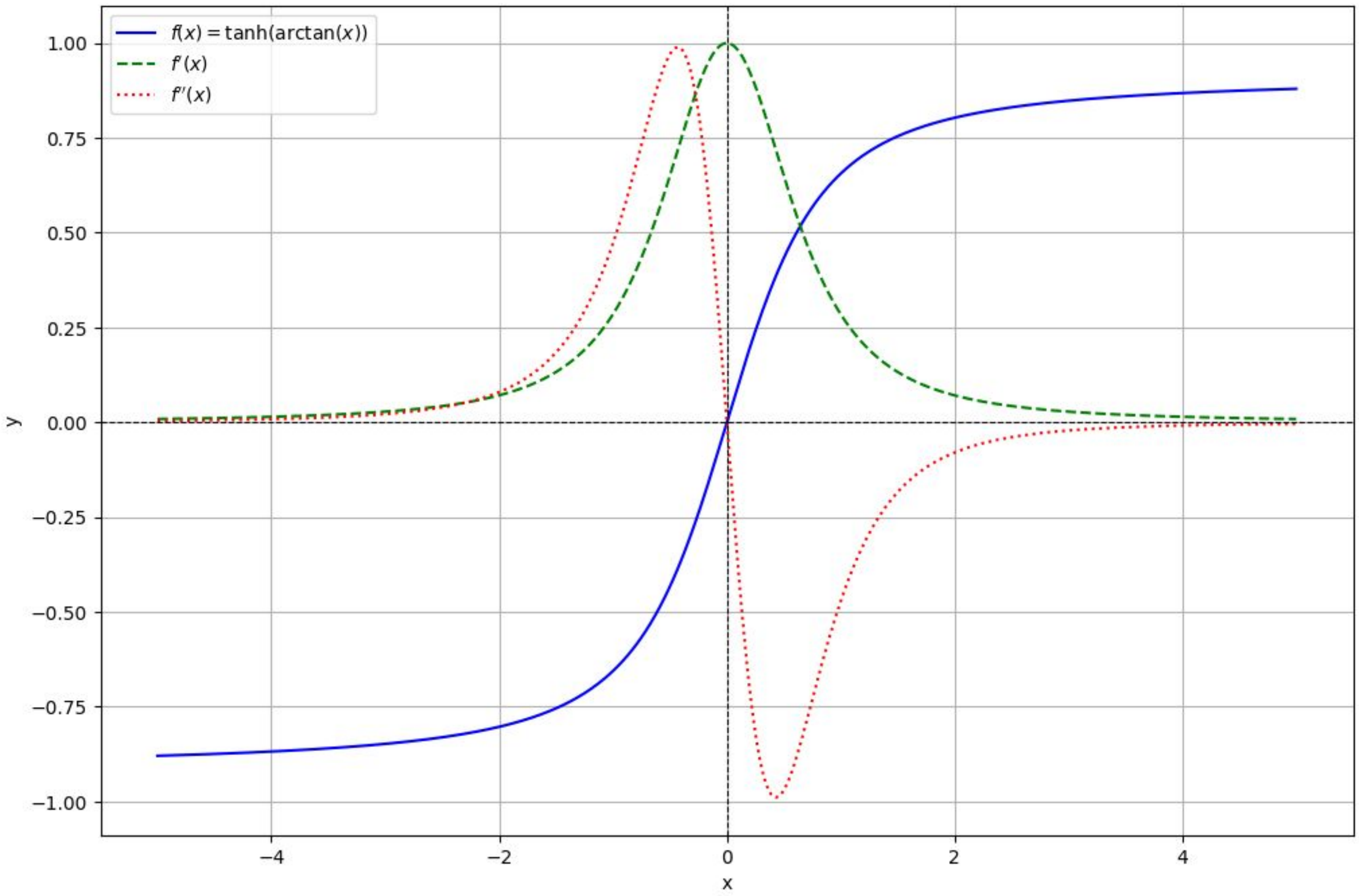
Illustrate Derivative

```
from scipy.misc import derivative
import matplotlib.pyplot as plt
import numpy as np

my_func = lambda x: x**x
dfdx = lambda x: derivative(my_func, x, dx=1e-6)
x = np.linspace(0,2,100)
plt.plot(x, my_func(x), c = 'blue', label = 'f')
plt.plot(x, dfdx(x), c='red', label = 'f\'')
plt.legend()

plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.ylim(-1,2)
```



Root

Tanh(ArcTan(x))



$f(x) = \tanh(\arctan(x))$
$f'(x)$
$f''(x)$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−4
−2
0
2
4
x
y

Find the root of the second derivative

```
import numpy as np
from scipy.optimize import minimize

def f(x):
    return np.tanh(np.arctan(x))

dx=1e-6
def dfdx(x):
    return (f(x + dx) - f(x-dx)) / (2*dx)
def df2dx2(x):
    return (f(x + dx) - 2 * f(x) + f(x - dx)) / (dx**2)

inflection_point_result = minimize(df2dx2, 0.1)
inflection_point = inflection_point_result.x[0]
```

-6.0742e-11

Curve Fit

Clausius–Clapeyron relation

Read the Data

```
import pandas as pd

df = pd.read_csv("clausius_clapeyron_data.csv")
T = df['T']
lnP = df['lnP']
```

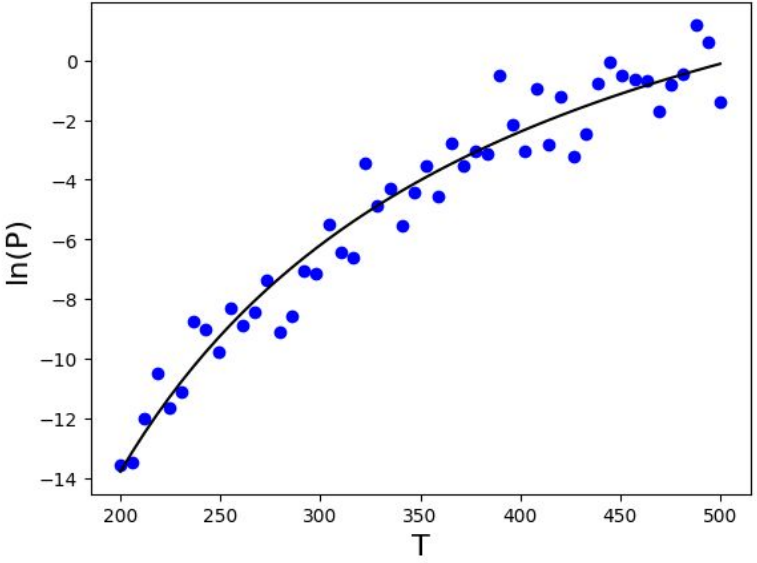
Curve-Fit

```
from scipy.optimize import curve_fit

R = 8.314  # Universal gas constant in J/mol·K
def clausius_clapeyron(T, delta_H, C):
    return -delta_H / (R * T) + C

popt, pcov = curve_fit(clausius_clapeyron, T, lnP)
delta_H_fit, C_fit = popt

plt.scatter(T, lnP, color='blue')
plt.plot(T, clausius_clapeyron(T, delta_H_fit, C_fit), color='black')
plt.xlabel('T', fontsize=16)
plt.ylabel('ln(P)', fontsize=16);
```

Visualization

```
import matplotlib.pyplot as plt
```

```
plt.scatter(T, lnP, color='blue')
plt.plot(T,
         clausius_clapeyron(T, delta_H_fit, C_fit),
         color='black')
plt.xlabel('T', fontsize=16)
plt.ylabel('ln(P)', fontsize=16);
```



Visualization

```
import matplotlib.pyplot as plt

plt.scatter(1 / T, lnP, color='blue')
plt.plot(1 / T,
          clausius_clapeyron(T, delta_H_fit, C_fit),
          color='black')
plt.xlabel('1/T', fontsize=16)
plt.ylabel('ln(P)', fontsize=16);
```