Calculus

Understanding the Nature of Functions

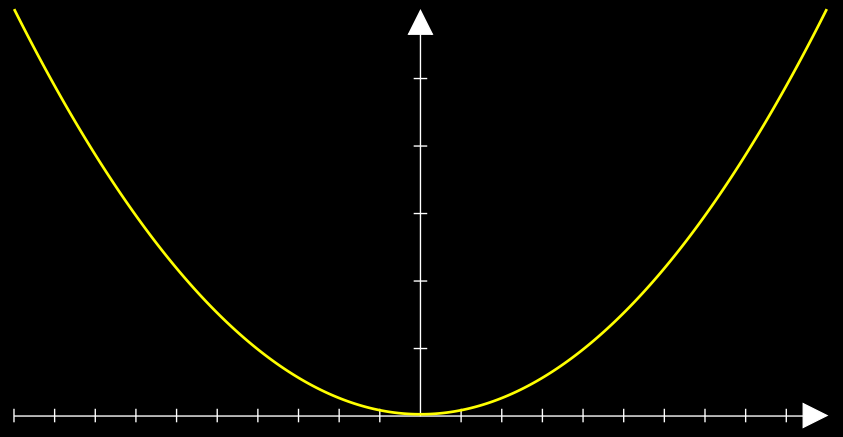
Functions

Functions take one or more numeric inputs (e.g., x) and provide an output(e.g., y).

To the right we plot the function:

$$f(x) = 3x^2 + 1$$

We can use SymPy as shown to quickly plot the function.



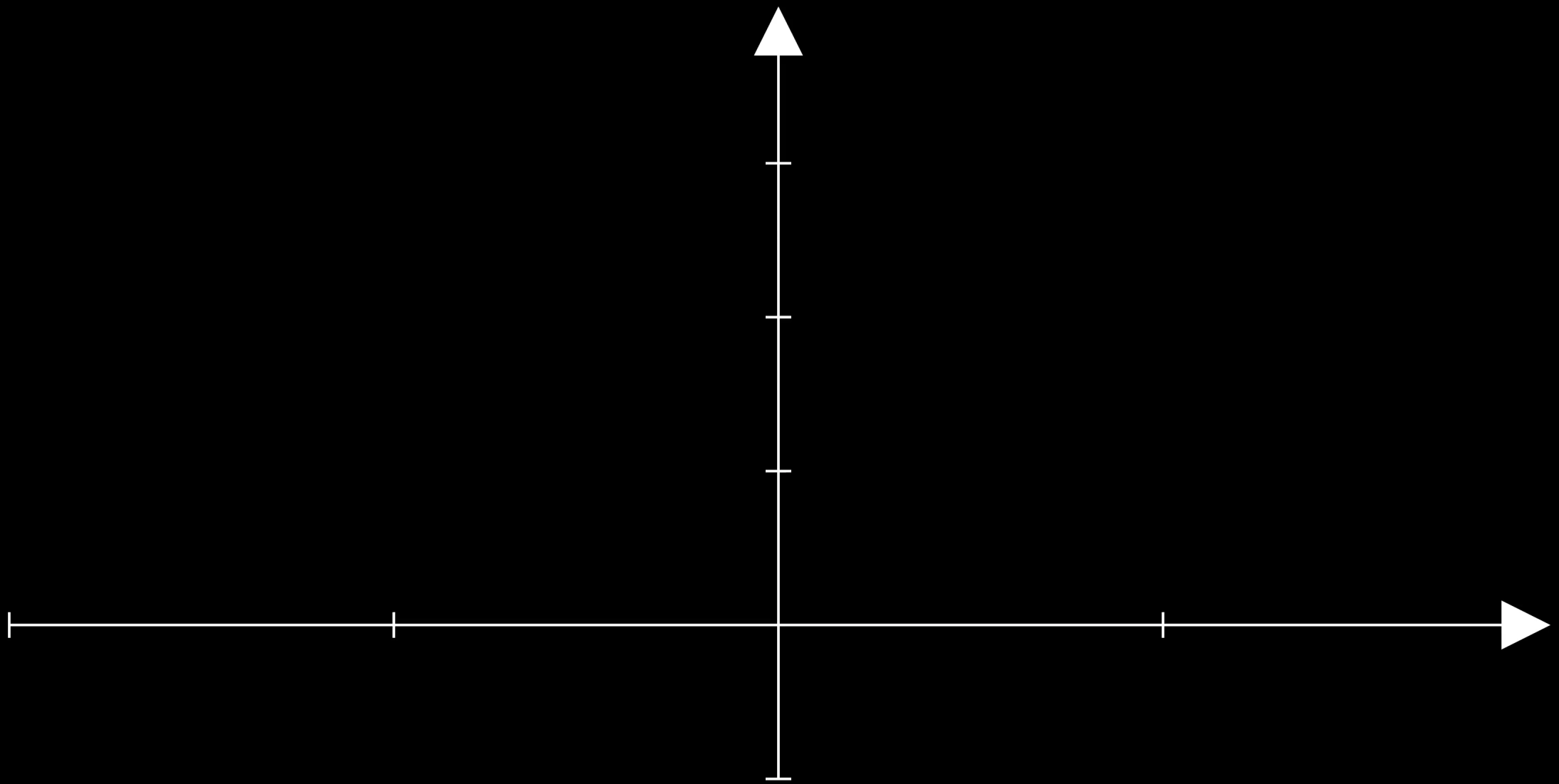
```
1  from sympy import *
2
3  x = symbols('x')
4  f = 3*x**2 + 1
5
6  plot(f)
```

SymPy

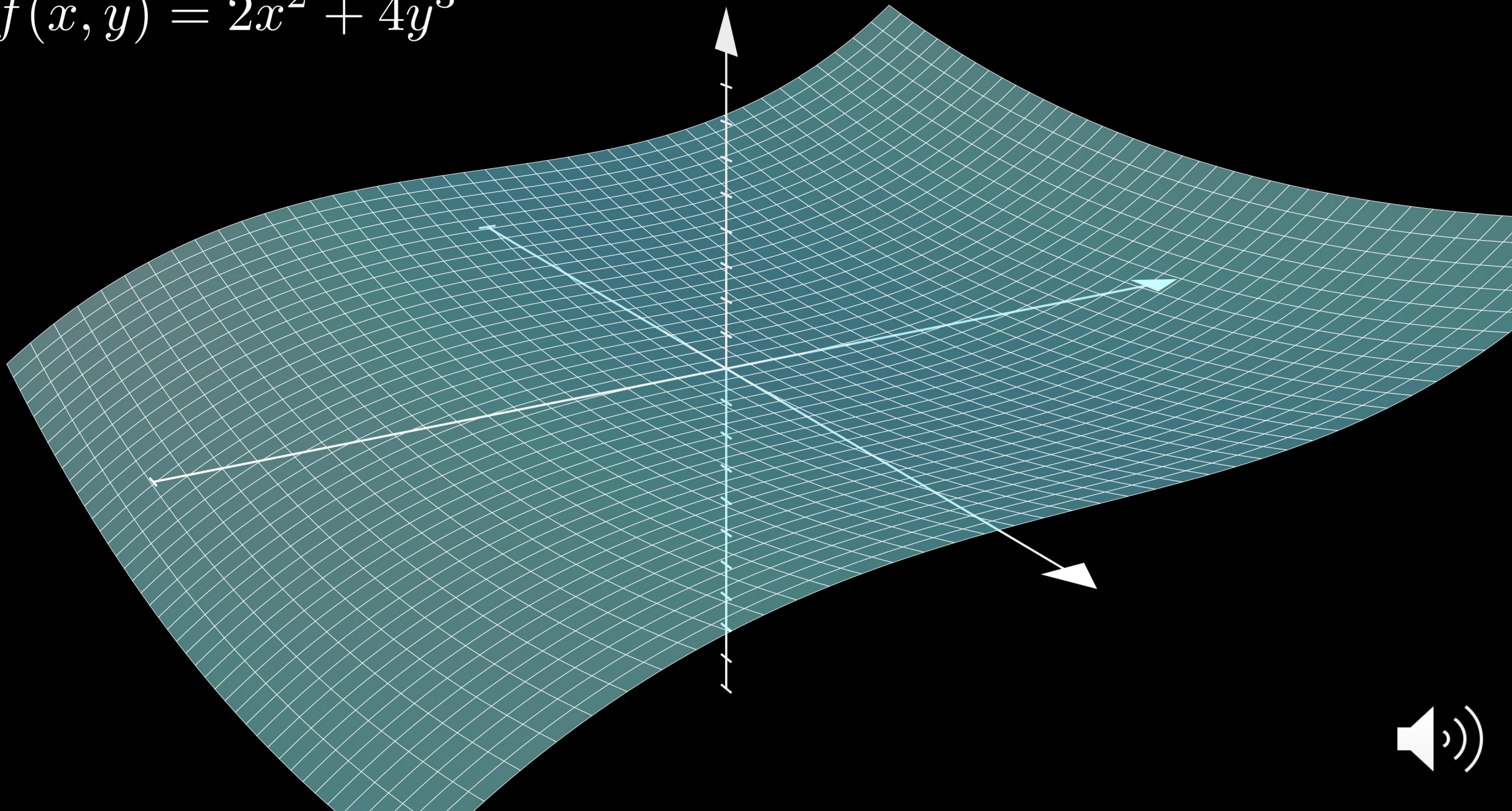
SymPy performs symbolic math operations, much like having Python do pencil-and-paper math work for you!

It performs algebraic and calculus operations, while using symbols and avoiding the need for floating point arithmetic.

```
1  from sympy import *
2
3  x = symbols('x')
4  f = 2*x + 3*x + pi
5  print(f) # 5*x + pi
```



$$f(x, y) = 2x^2 + 4y^3$$



Summations and Products

$$x_i = \begin{bmatrix} 5 & 2 & 9 & 6 & 1 \end{bmatrix}$$

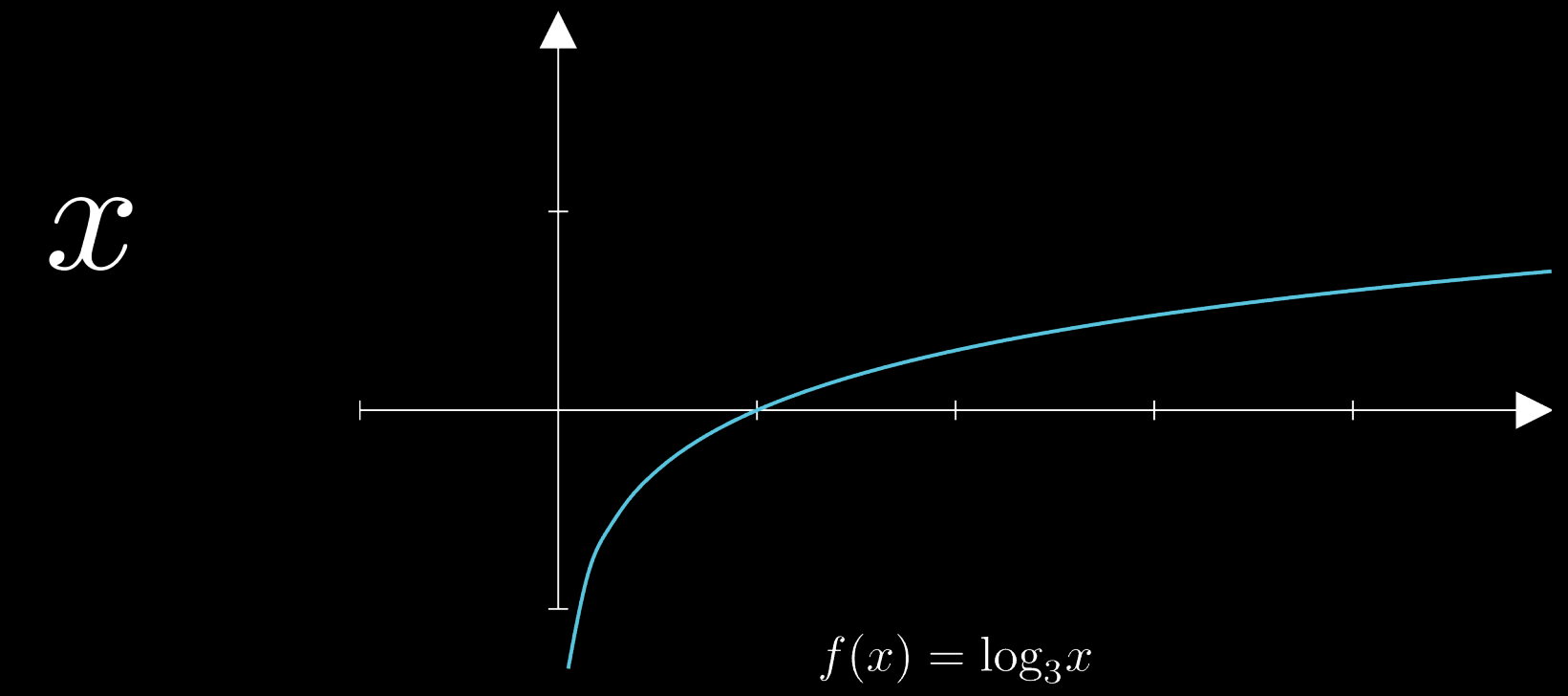
$$\sum_{i=1}^{5} x_i = 5 + 2 + 9 + 6 + 1 = 23$$

$$\prod_{i=1}^{5} x_i = 5 \times 2 \times 9 \times 6 \times 1 = 540$$

Logarithms

$$a^x = b$$

$$\log_a b = x$$

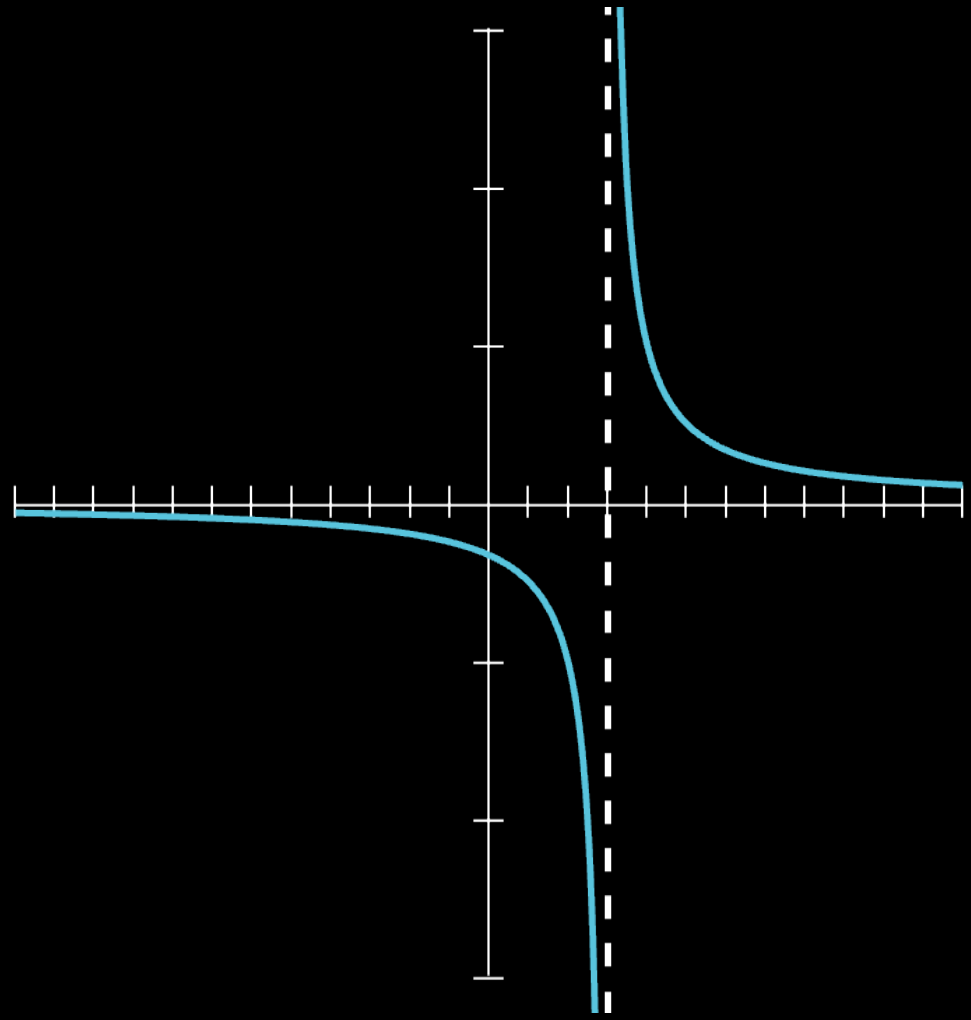


$$3^x = 9$$

$$\log_3 9 = 2$$

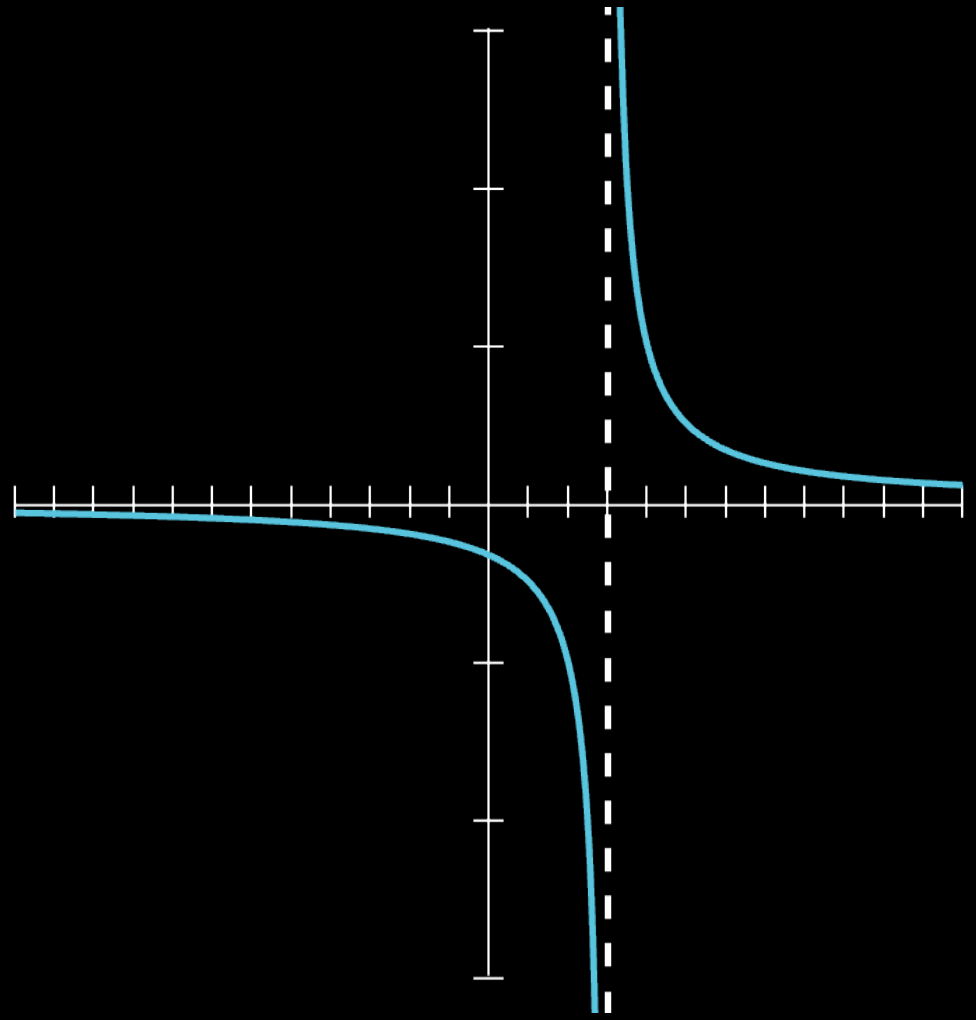
Infinity and Limits

$$f(x) = \frac{1}{x-3}$$



Infinity and Limits

$$\lim_{x \to 3} \frac{1}{x - 3} = \infty$$

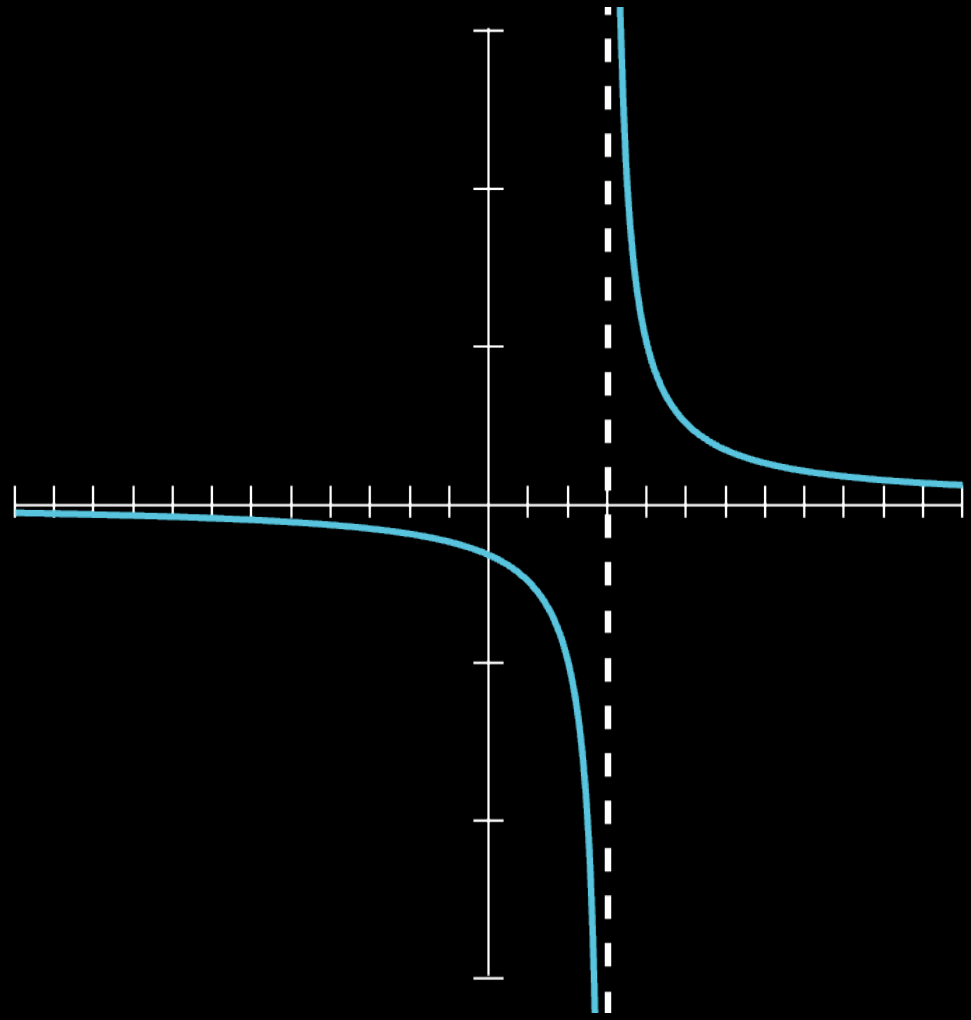


```python
from sympy import *

x = symbols('x')
f = 1 / (x-3)

result = limit(f, x, 3)

print(result) # oo
```

Infinity and Limits

$$\lim_{x \to \infty} \frac{1}{x - 3} = 0$$

```python
from sympy import *

x = symbols('x')
f = 1 / (x-3)

result = limit(f, x, oo)

print(result) # 0
```

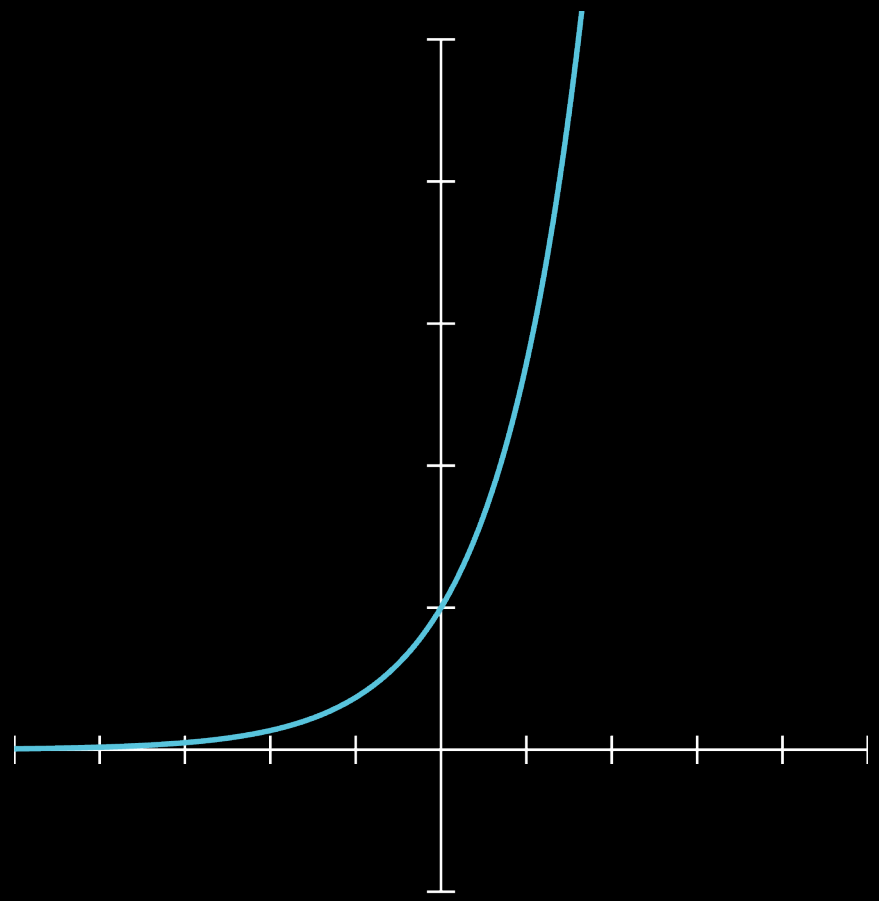


Euler's Number

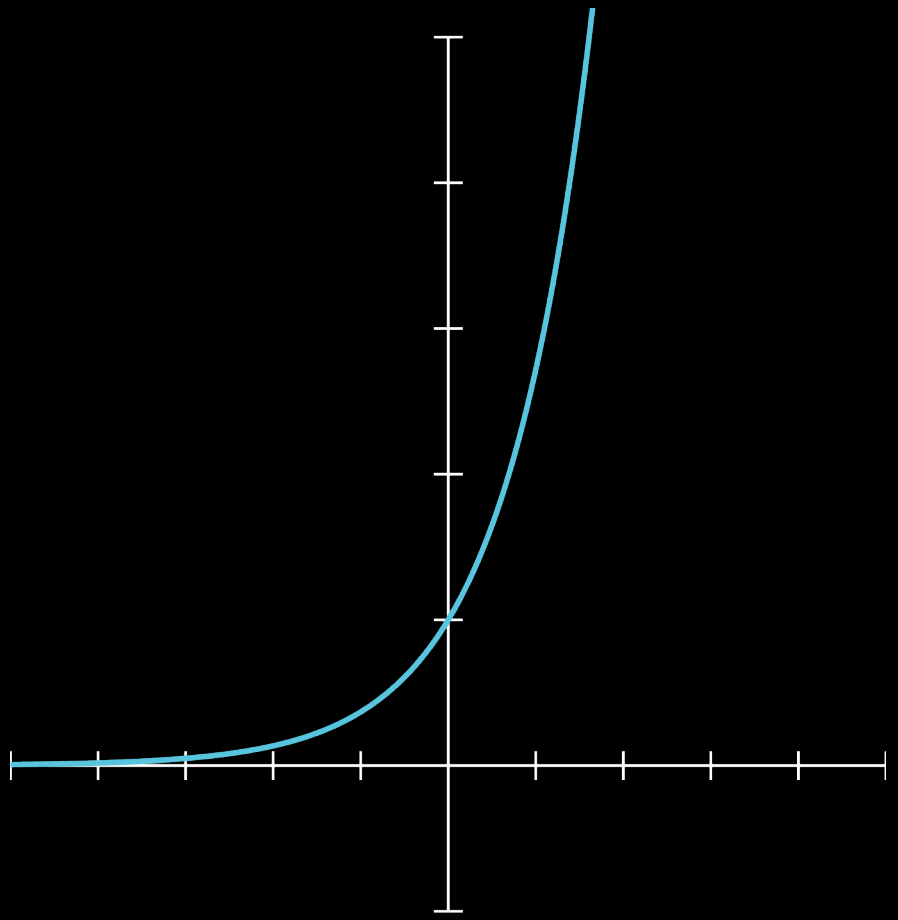
$$e = \lim_{n \to \infty} (1 + \frac{1}{n})^n$$

$$= 2.71828182846...$$

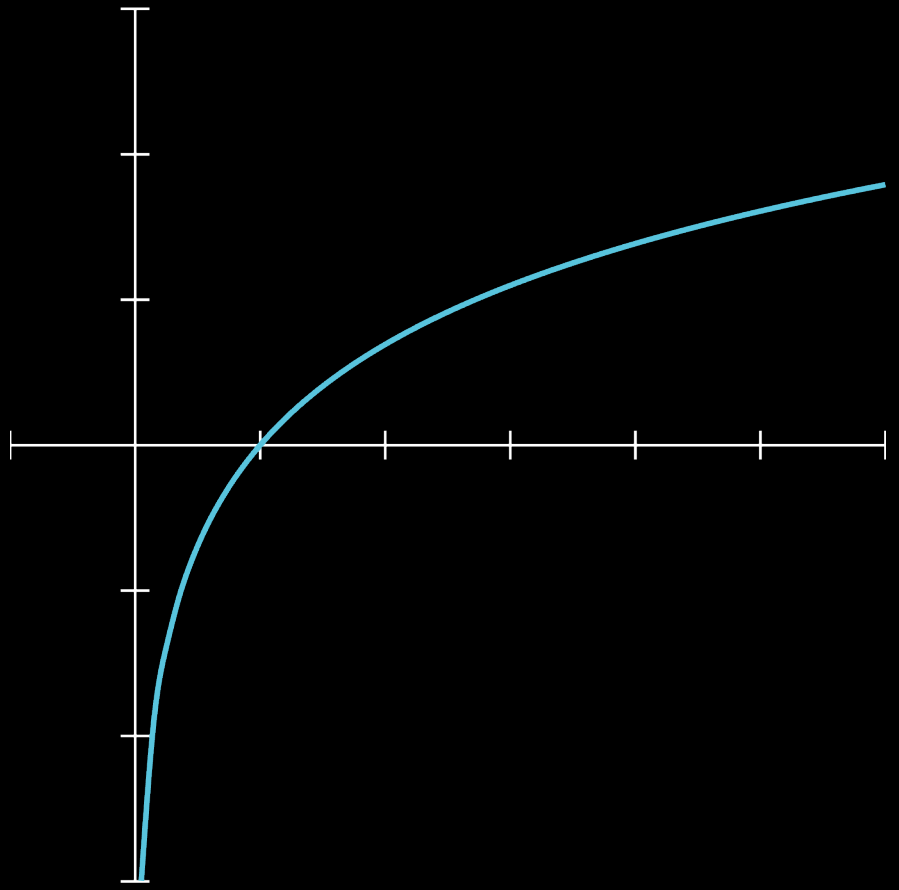
$$\log_e x = \ln(x)$$



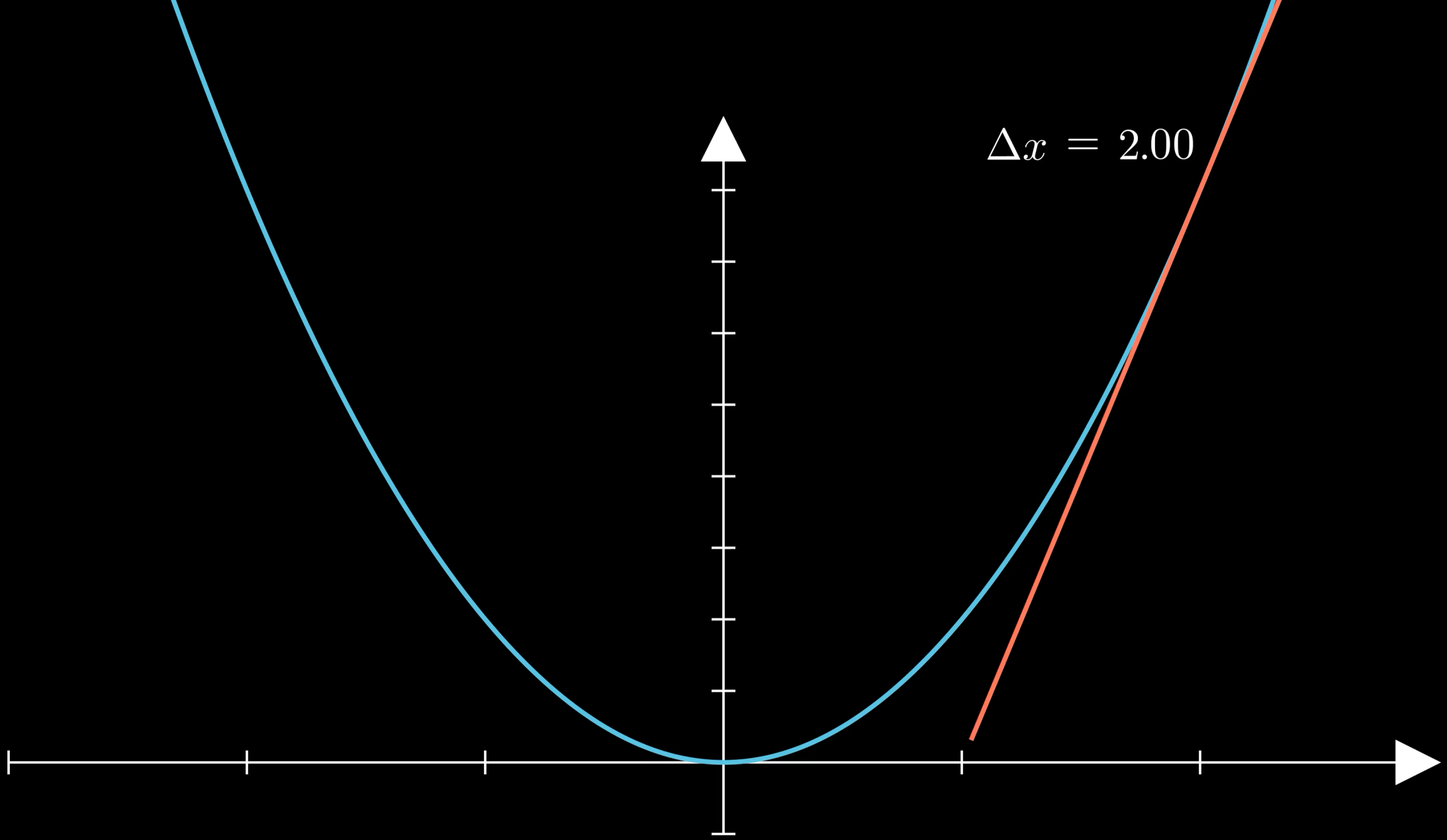
Euler's Number

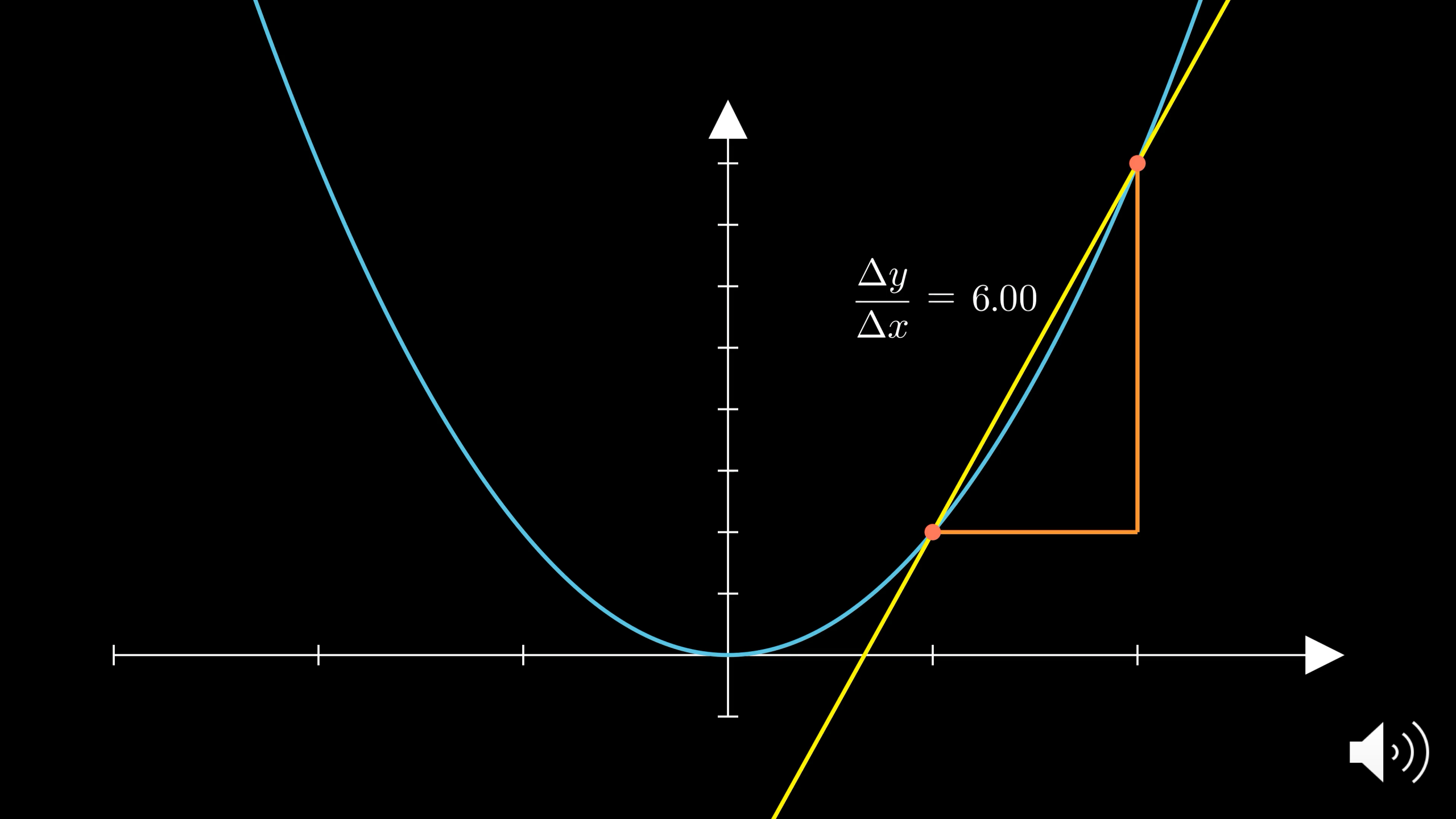


$f(x) = e^x$



$f(x) = ln(x)$


$\Delta x = 2.00$


$\frac{\Delta y}{\Delta x} = 6.00$

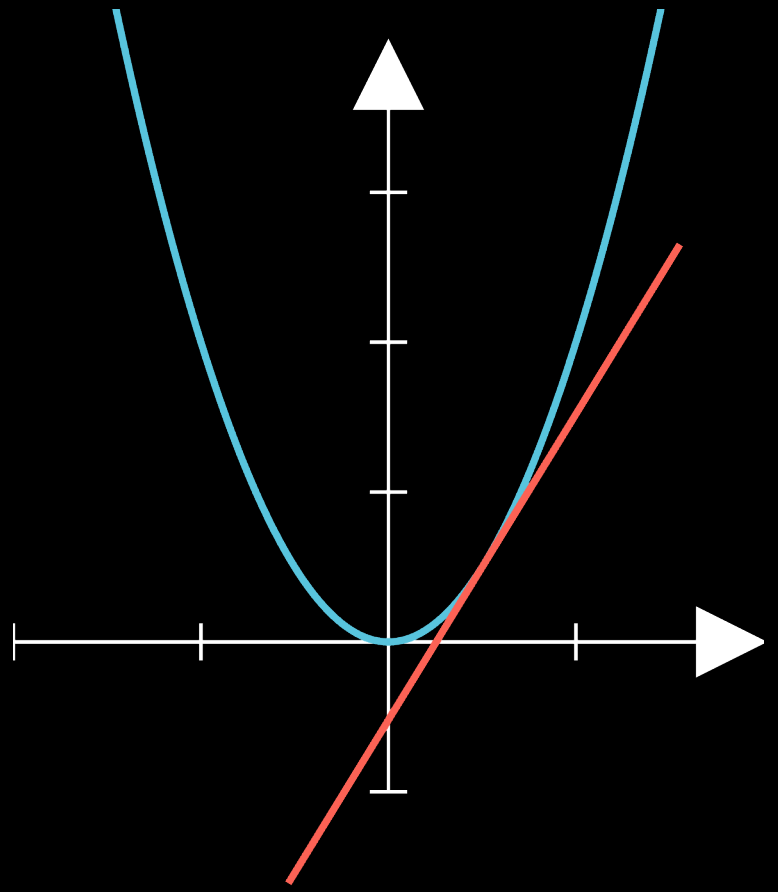
Derivatives

The derivative of a function $f'(x)$ allows us to calculate the slope at any given point of a function.

$$f(x) = 2x^2$$

$$f'(x) = 4x$$

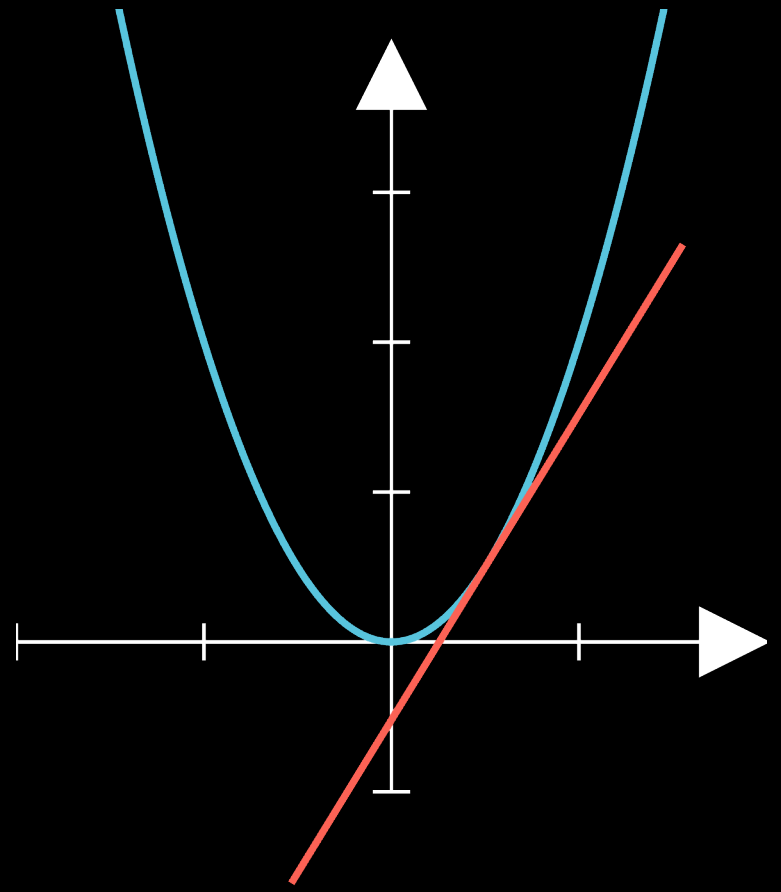
At $x = 2$, the slope will be 8.



Derivatives

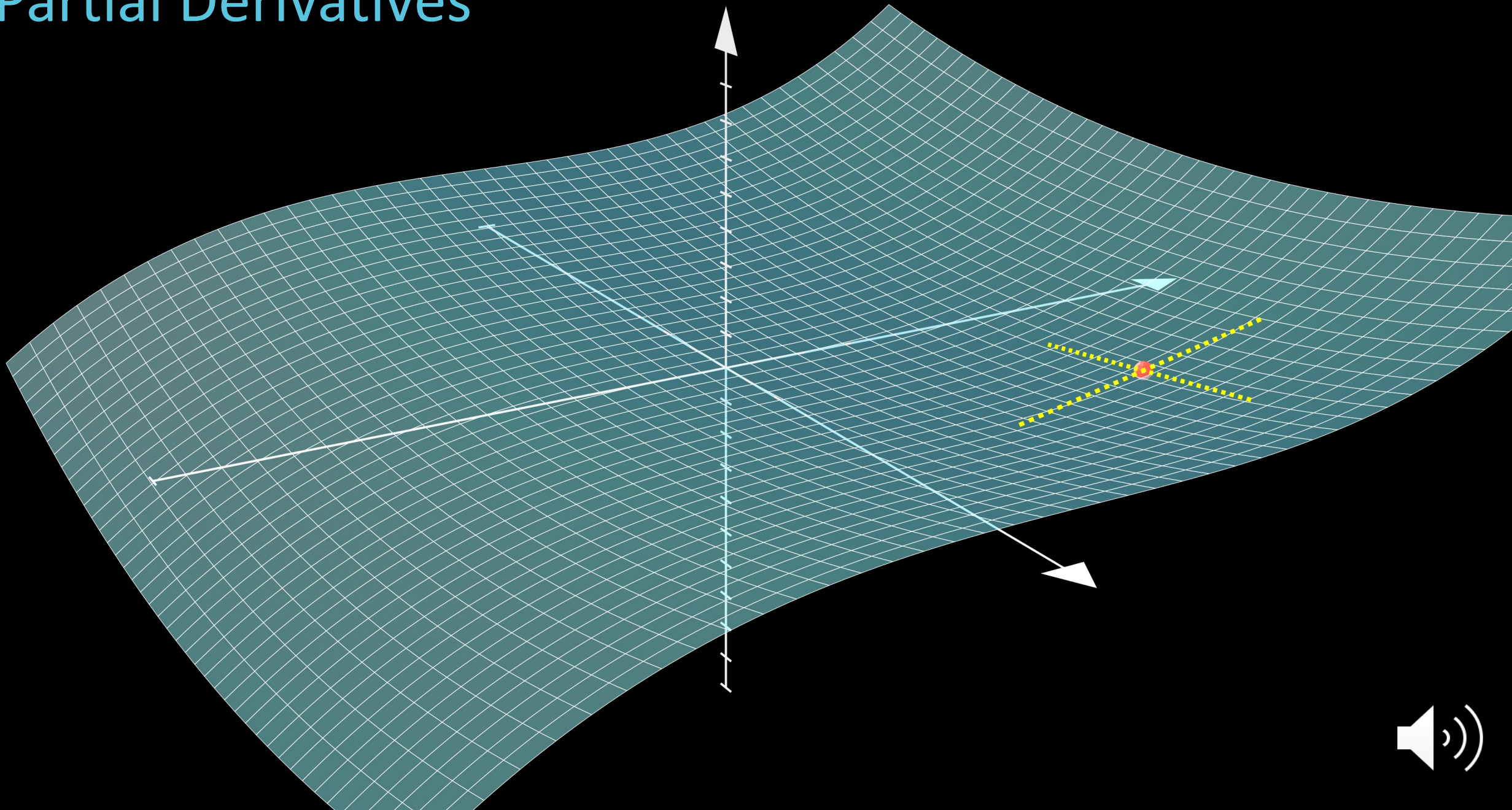
The derivative of a function $f'(x)$ allows us to calculate the slope at any given point of a function.

```
1  from sympy import *
2  x = symbols('x')
3  f = 2*x**2
4  dx_f = diff(f,x)
5  print(dx_f) # 4*x
```

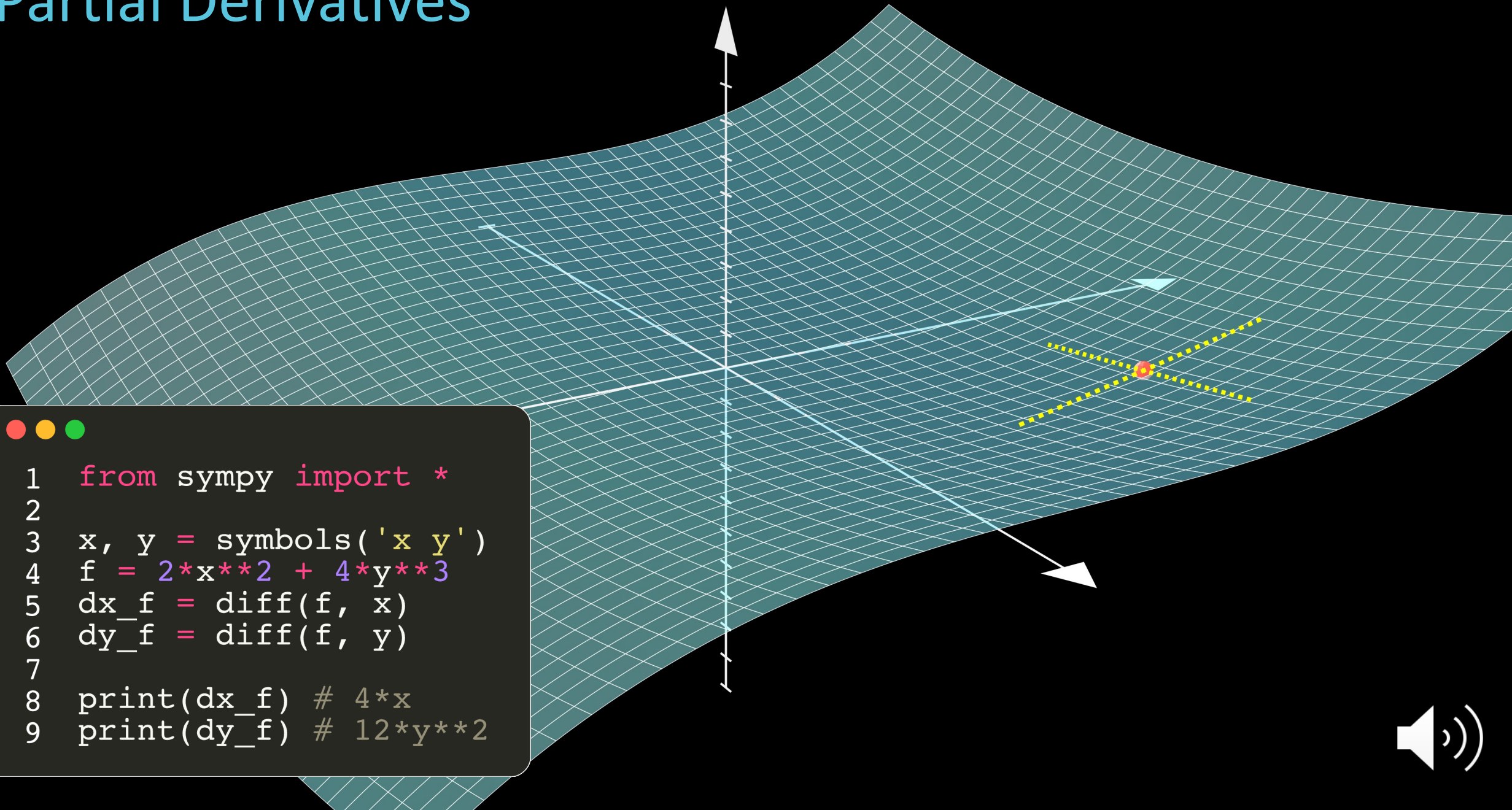


You can use SymPy as shown above to calculate the derivative function

Partial Derivatives



Partial Derivatives

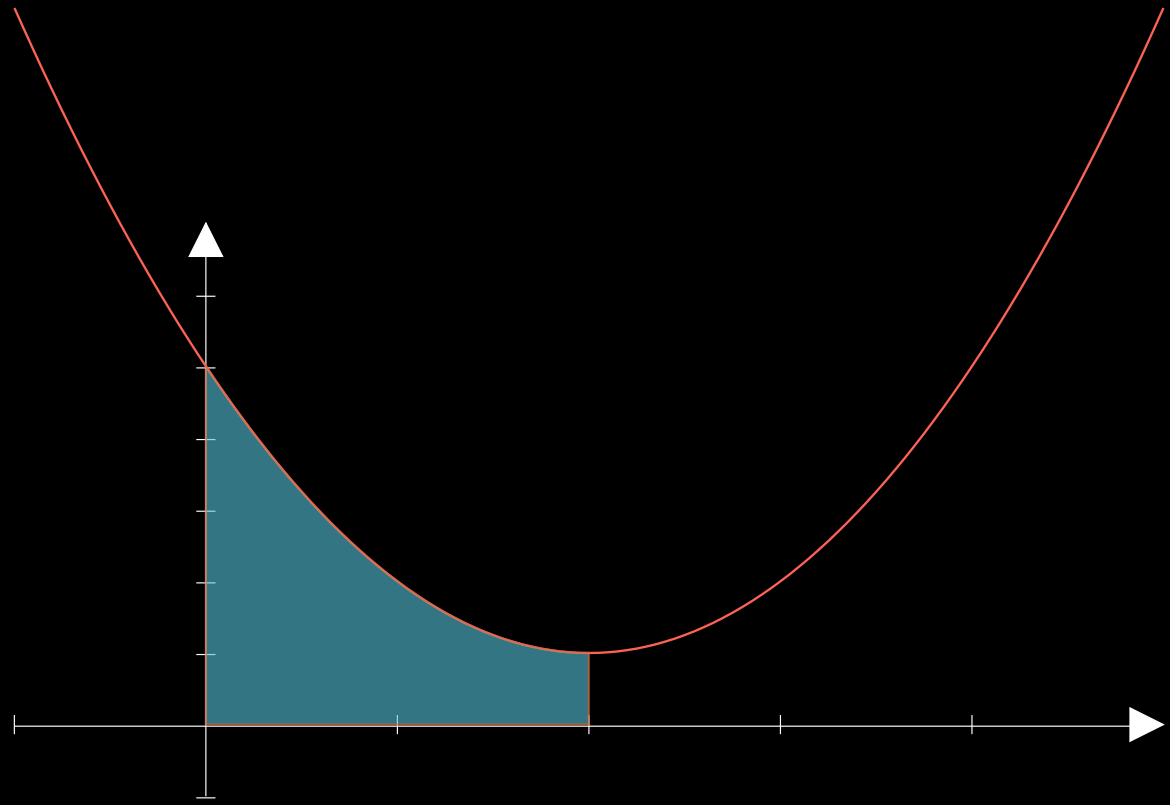


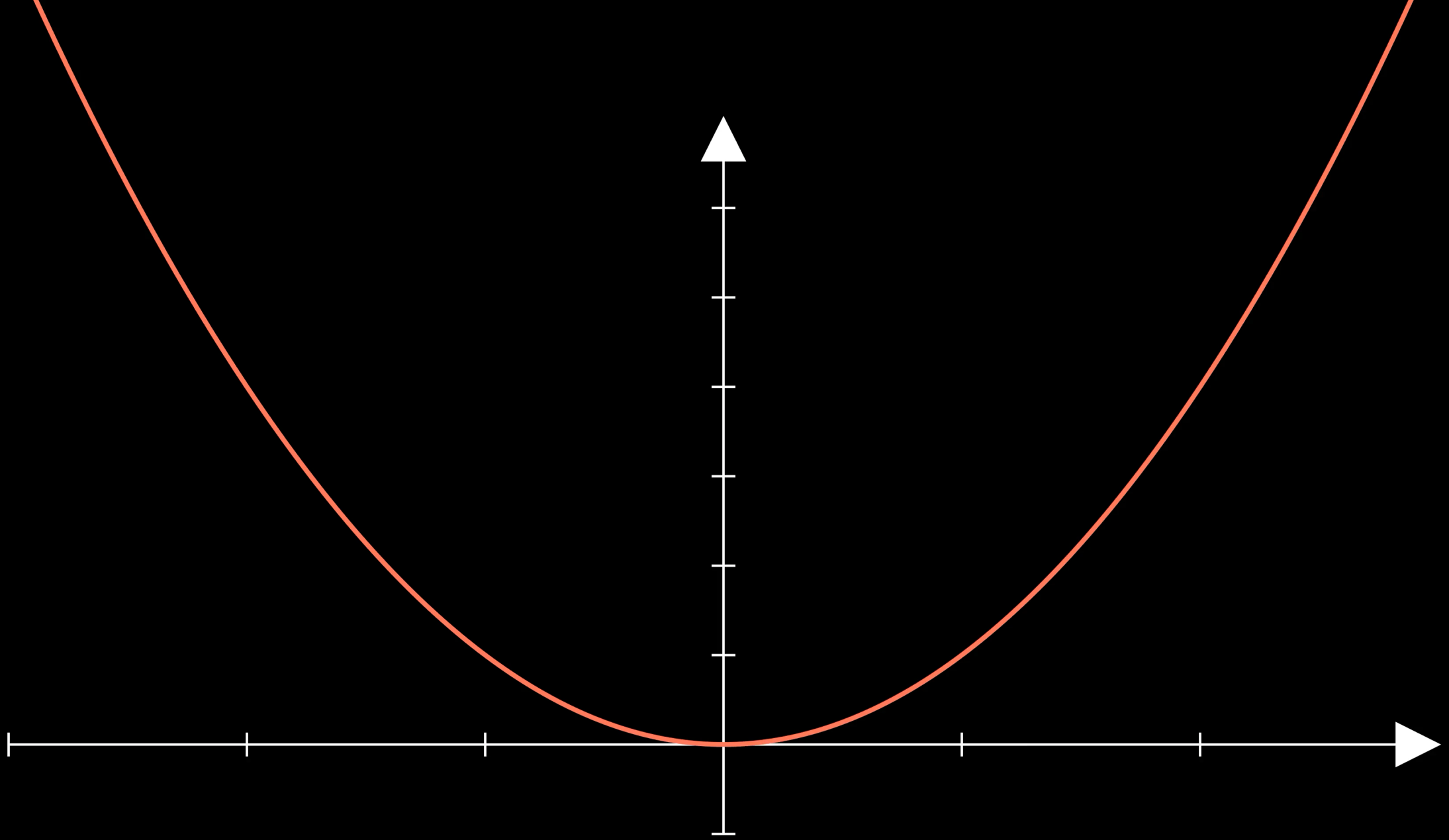
```python
from sympy import *

x, y = symbols('x y')
f = 2*x**2 + 4*y**3
dx_f = diff(f, x)
dy_f = diff(f, y)

print(dx_f) # 4*x
print(dy_f) # 12*y**2
```

Integrals

Integrals will help you find the area in a range under a curve.

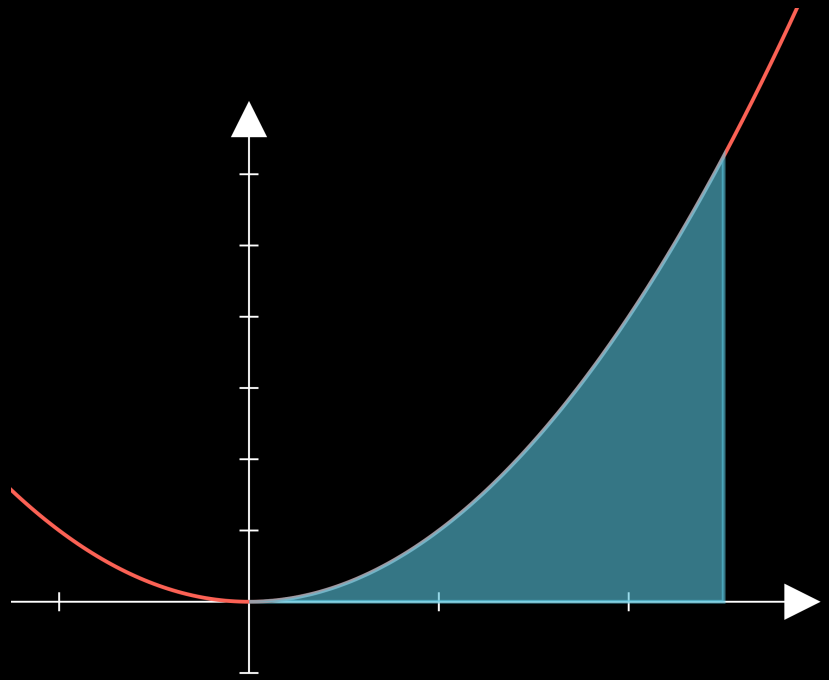
This concept is useful for tasks like finding probabilities under probability distributions.





Integrals

```python
from sympy import *

x = symbols('x')
f = x**2

# area from x = 0 to 2.5
area = integrate(f, (x, 0, 2.5))
print(area) # 5.208333333333333
```



Probability

Measuring Uncertainty

What is Probability?

Probability is a measure of how likely an event will happen, acting as a quantification of certainty.

$$P(X)$$

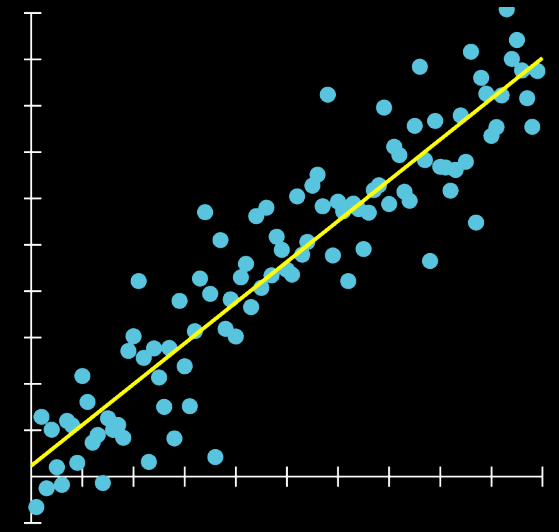
Here are some questions that are answered with a probability:

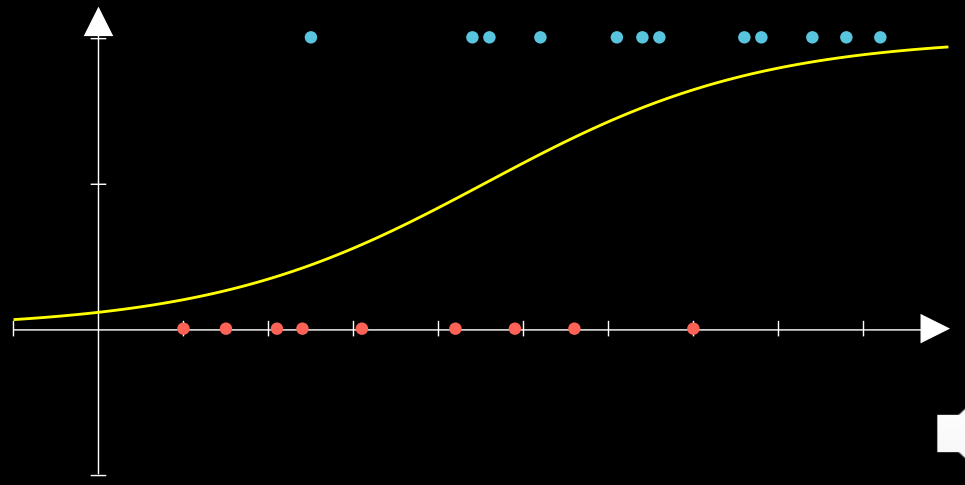
- How likely will my flight be late?
- Will a stock price increase?
- Will the Dallas Cowboys win the Super Bowl?

Probability and Machine Learning

You will find probability plays a critical role in machine learning.

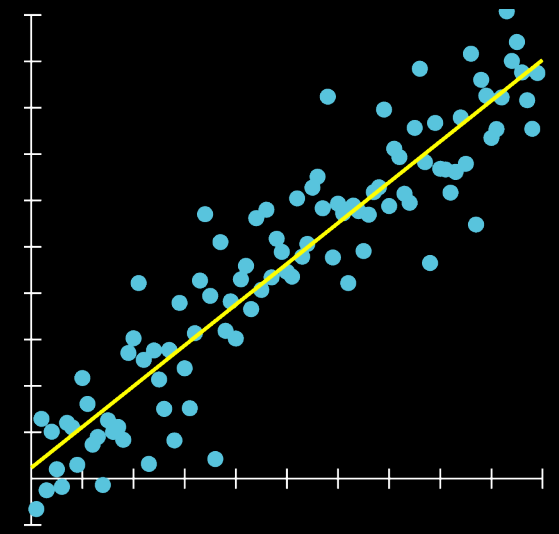
- Linear regression
- Logistic regression
- Naïve Bayes
- Neural networks
- Model validation

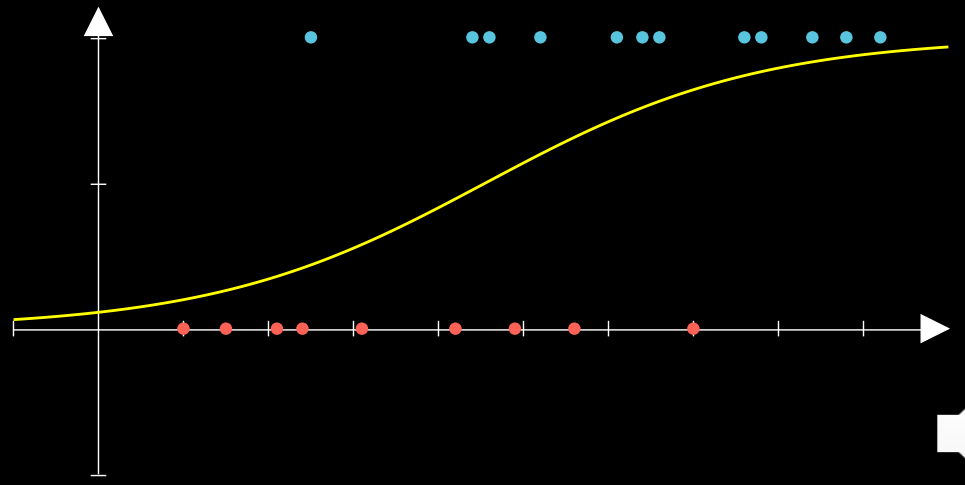




Probability and Machine Learning

Not understanding probability before machine learning is like learning to read without first learning the alphabet!





Probability Rules

There are a couple of rules that a probability $P(X)$ of an event X must follow.

$$P(X)$$

- It must be between 0 and 1, or 0% and 100%
- The probability of an event happening $P(X)$ and not happening $P(\prime X)$ must add up to 1.
- If there are three or more possibilities of an outcome of an event they all must add up to 1.

Where Does Probability Come From?

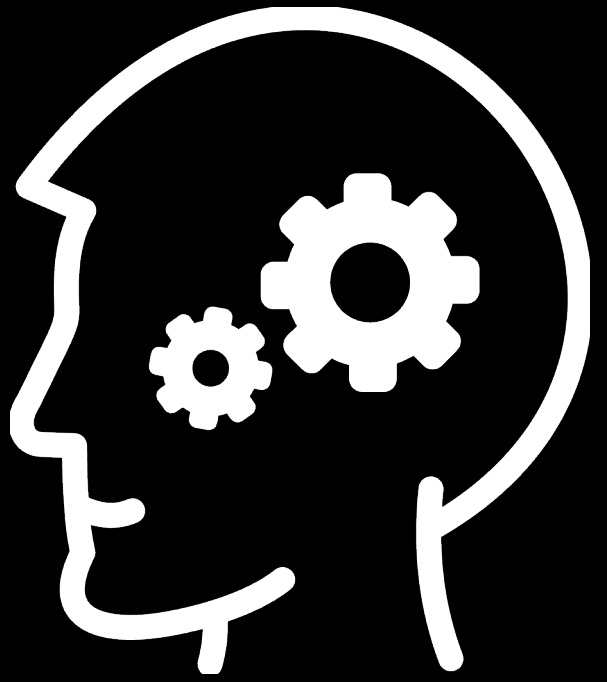
Probability can be based off data, a belief, or both!

Does data trump belief? Not always.

Both data and beliefs are shaped by…

- Bias
- Decisions
- Humans!

We will discuss data and bias when we talk about statistics.



Joint Probability

What is the probability event A *and* B will occur? This is what we call **joint probability**.

$$P(A \cap B)$$

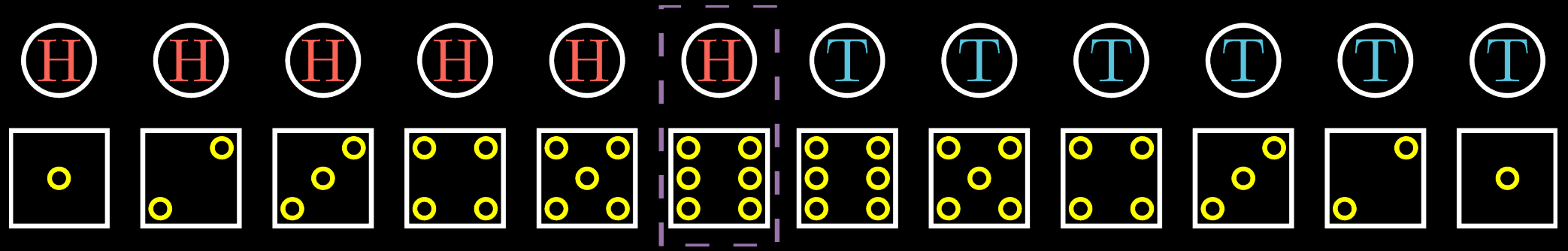
It is as simple as multiplying the two probabilities together.

$$P(A \text{ and } B)$$

$$P(A \; and \; B) = \; P(A) \times P(B)$$

Joint Probability

$$P(H \text{ and } 6) = \frac{1}{2} \times \frac{1}{6}$$
$$= \frac{1}{12}$$



Union Probability

What is the probability event A *or* B will occur? This is what we call **union probability**.

$$P(A \cup B)$$

$$P(A \text{ or } B)$$

If A and B cannot occur simultaneously (mutually exclusive) we add them together

$$P(A \; or \; B) = P(A) + P(B)$$

Union Probability

However, there is a catch if A and B can occur simultaneously.

Double-counting with the joint probability will occur, so you have to subtract it.

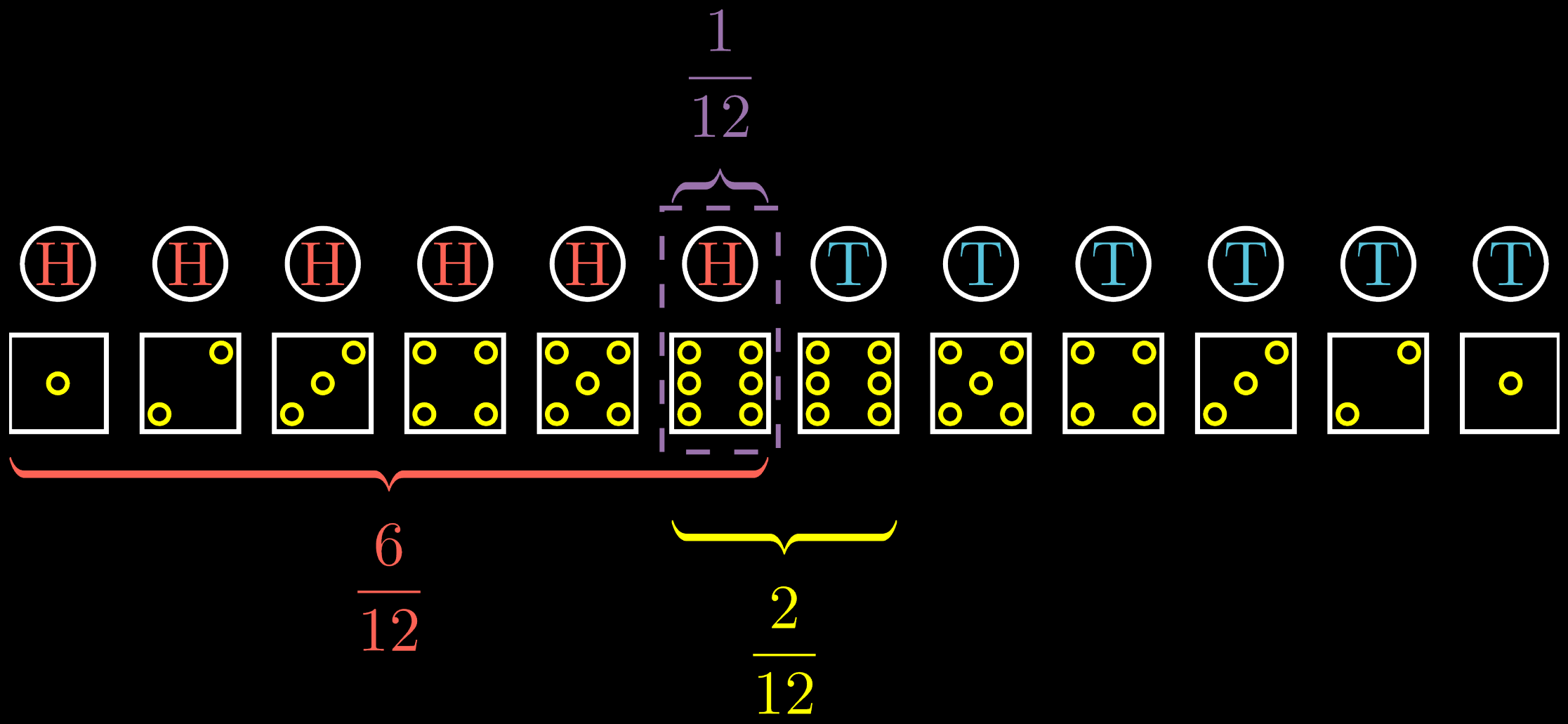
$$P(A \text{ or } B) = P(A) + P(B) - P(A) \times P(B)$$

While this is not intuitive, let's reason why.

$$P(A \cup B)$$

$$P(A \text{ or } B)$$

Union Probability



Union Probability

$$P(A \text{ or } B) = P(A) + P(B) - P(A) \times P(B)$$

$$P(H \text{ or } 6) = \frac{1}{2} + \frac{1}{6} - \frac{1}{2} \times \frac{1}{6}$$

$$= \frac{7}{12}$$

Conditional Probability

When we consider how much event B affects A, and whether A's probability changes when B occurs, we call that conditional probability.

$$P(A|B)$$

$$P(A \text{ given } B)$$

P(flood) = .01

P(flood given rain) = .40

If A does not care what B does, then P(A) = P(A given B).

Conditional Probability

Keep in mind that the $P(A\ given\ B)$ is not the $P(B\ given\ A)$!

$$P(A\ given\ B) \neq P(B\ given\ A)$$

The probability of a colorblind person being male is not the same as the probability of a male being colorblind!

Bayes Theorem

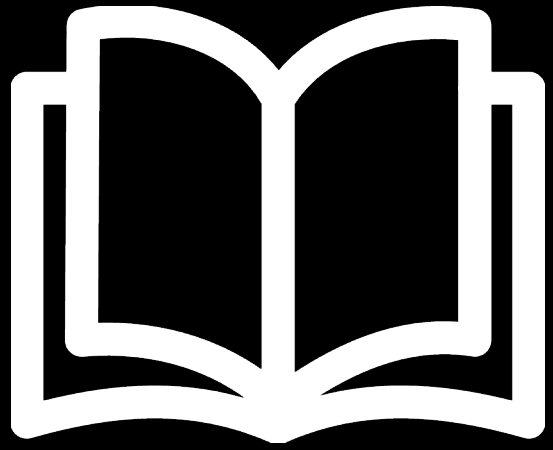
Bayes Theorem allows us to flip a conditional probability.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

It is one of the most famous formulas in statistics and math.

Bayes Theorem

Let’s say a study came out and claimed that 95% of successful Fortune 500 company leaders read avidly.

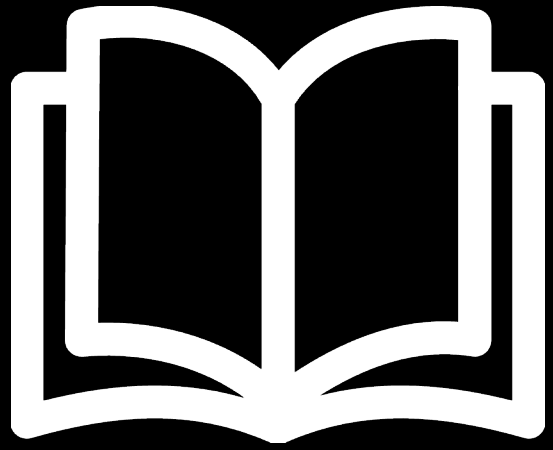


Is it right to presume that reading will help make you a successful Fortune 500 CEO?

Bayes Theorem

It’s hard to define a *person who reads*, but let’s say it is someone who reads at least 10 books a year.

I gather these stats:



P(read|CEO) = .95

P(read) = .27

P(CEO) = .0000074

Does something already feel off?

Bayes Theorem

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

P(read|CEO) = .95

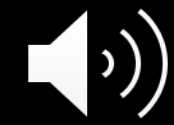
P(read) = .27

P(CEO) = .0000074

$$P(CEO|read) = \frac{P(read|CEO) \times P(CEO)}{P(read)}$$

$$= .000026$$

Well that's disappointing! If you read, you only have a .000026 probability of becoming a successful Fortune 500 CEO!



Bayes Theorem

Let's say a machine learning system has the following performance specs.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

P(spam|positive) = .95

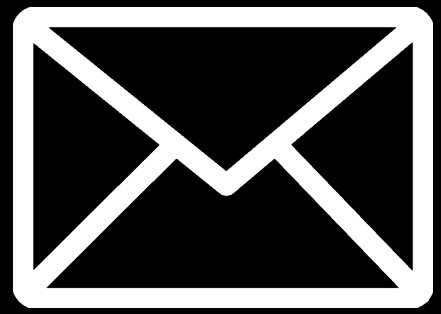
P(spam) = .06

P(positive) = .03

$$P(positive|spam) = \frac{P(spam|positive) \times P(positive)}{P(spam)}$$

$$= .2375$$

What is the probability a given spam email will be identified as positive?



Statistics and Hypothesis Testing

Summarizing and Predicting with Data

Samples and Populations

Population – A group of elements we are interested in studying.

Sample – A randomly selected subset of the population

$$\mu$$

$$\bar{x}$$

Biased Data

Imagine you are a student at a college and you want to find the average time college students watch TV per week.

You walk outside your dorm and collect responses from over 100 students.

You believe your work is done. Are you?

Biased Data

Consider that you are polling students at one university, outside your dorm, and using that to represent *ALL* students across the country!

Is this fair? No!

What if I create a social media poll that is shared with other colleges across the country?

Biased Data

Even the social media poll is going to be prone to heavy bias.

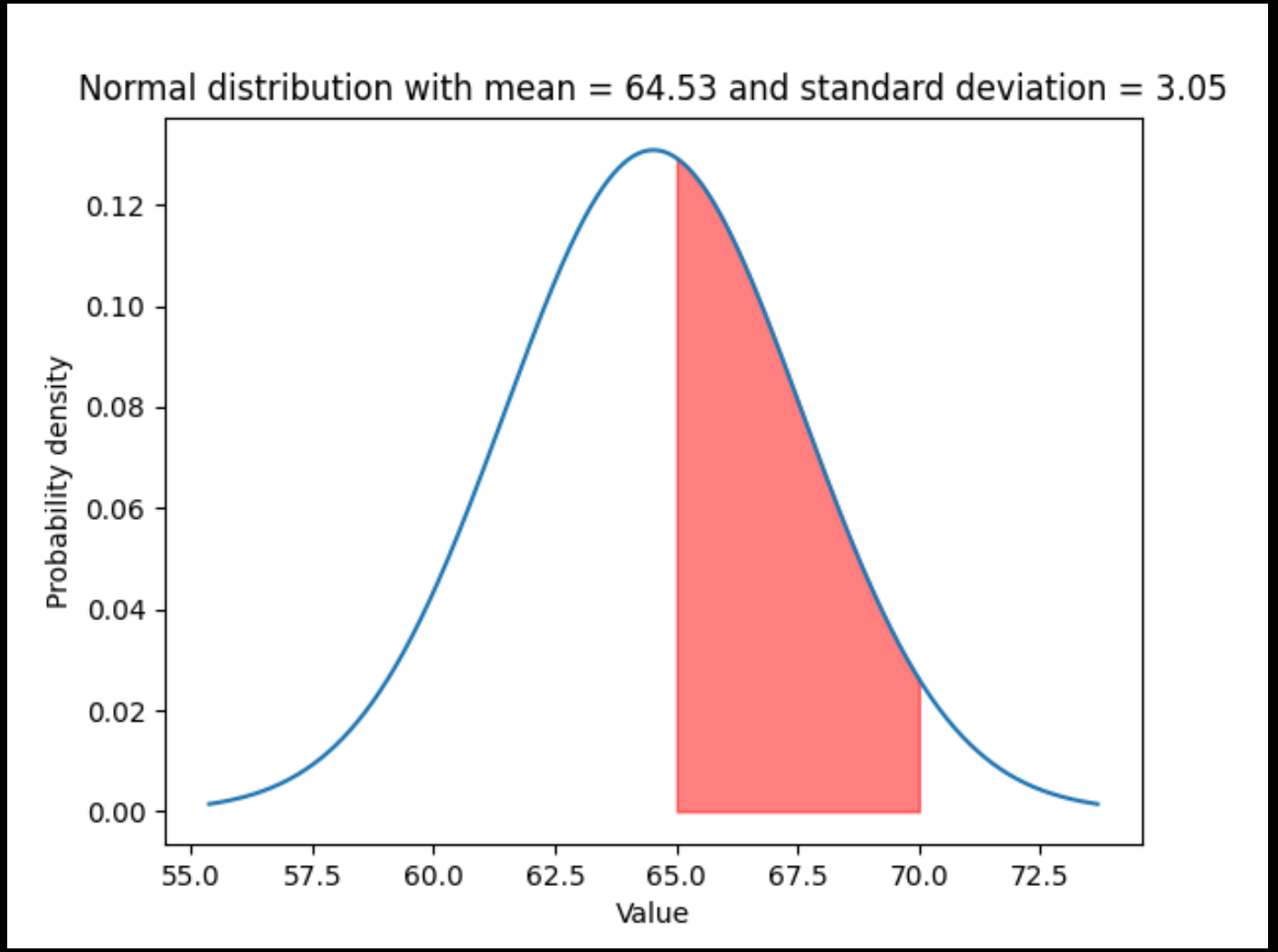
Students who are online enough to see the poll on social media may not mind recreational screen time.

Therefore they may watch more TV than other students!

Using SciPy for Normal Distribution

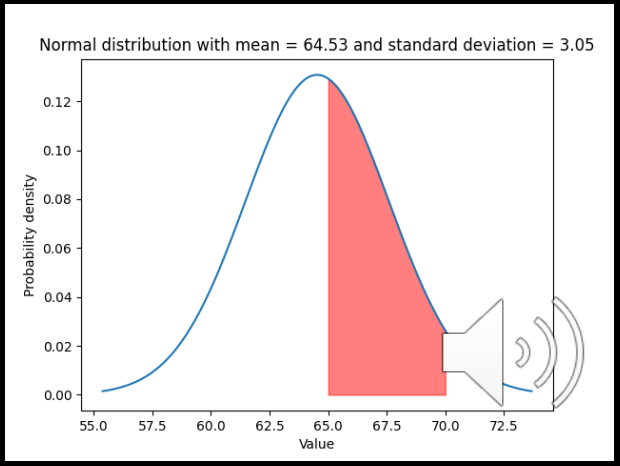
```
from scipy.stats import norm

mean = 64.53
std = 3.05

area_up_to_70 = norm.cdf(70, mean, std)
area_up_to_65 = norm.cdf(65, mean, std)
area_65_to_70 = area_up_to_70 - area_up_to_65

print(area_up_to_70) # prints 0.9635489113038331
print(area_up_to_65) # prints 0.5612339095753831
print(area_65_to_70) # prints 0.40231500172845003
```

Using matplotlib



Using matplotlib

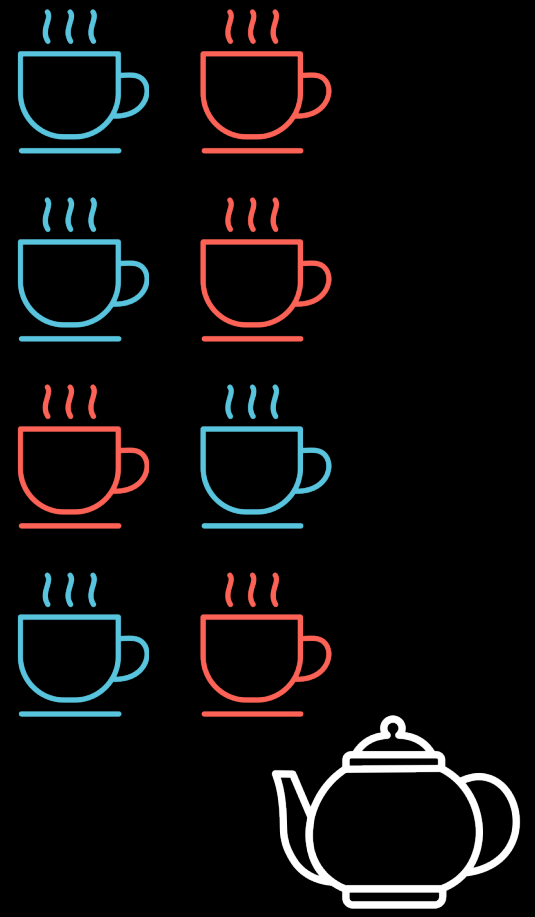
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

# define mean and standard deviation
mean = 64.53
std = 3.05
# lower and upper bounds of area
lower_x, upper_x = -3*std+mean, 3*std+mean
lower_area_x, upper_area_x = 65, 70

# have axis capture ± 3 standard deviations from mean
x = np.arange(lower_x, upper_x, .01)
y = norm.pdf(x, mean, std)

# plot the normal PDF
plt.plot(x, y) # bell curve

# Calculate the shaded area under the curve
shaded_area = norm.cdf(upper_area_x, mean, std) -
norm.cdf(lower_area_x, mean, std)

# Plot the normal distribution and the shaded area
x_area_range = (x >= lower_area_x) & (x <= upper_area_x)
plt.fill_between(x[x_area_range], y[x_area_range], 0,
alpha=0.5, color='red')
plt.xlabel('Value')
plt.ylabel('Probability density')
plt.title('Normal distribution with mean = {} and standard
deviation = {}'.format(mean, std))
plt.text(11.4, 0.1, 'Area = {:.2f}'.format(shaded_area))
plt.show()
```



P-Values

When someone says something is statistically significant, what does that mean?

This has everything to do with the p-value, which is the probability something happened by random chance rather than because of a hypothesized variable.



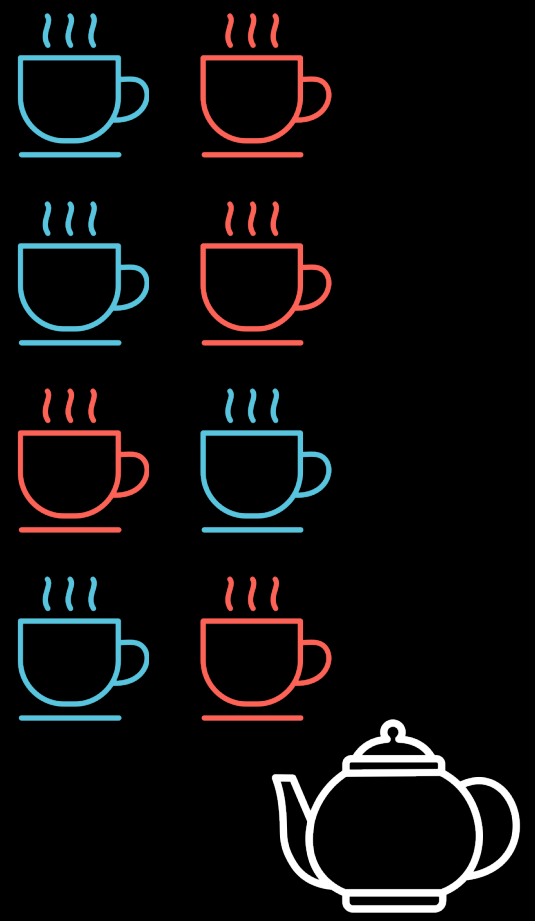
P-Values

Ronald Fisher was at a tea party, and his colleague Muriel said she could detect whether tea was poured before milk in her tea.

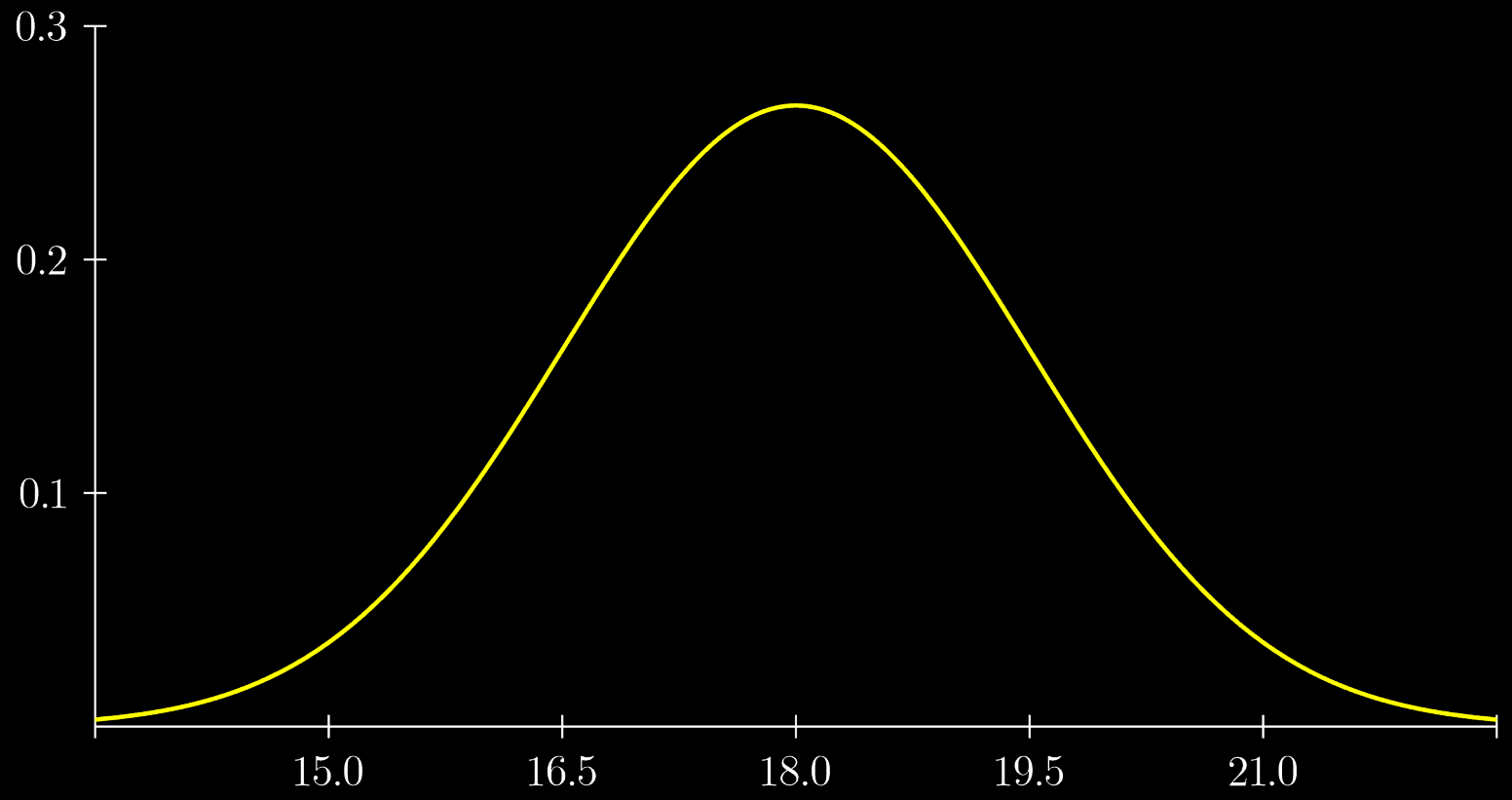
He made an experiment, 4 cups had milk poured first, the other 4 had tea poured first.

He made her guess which was which, and she was correct on all of them.

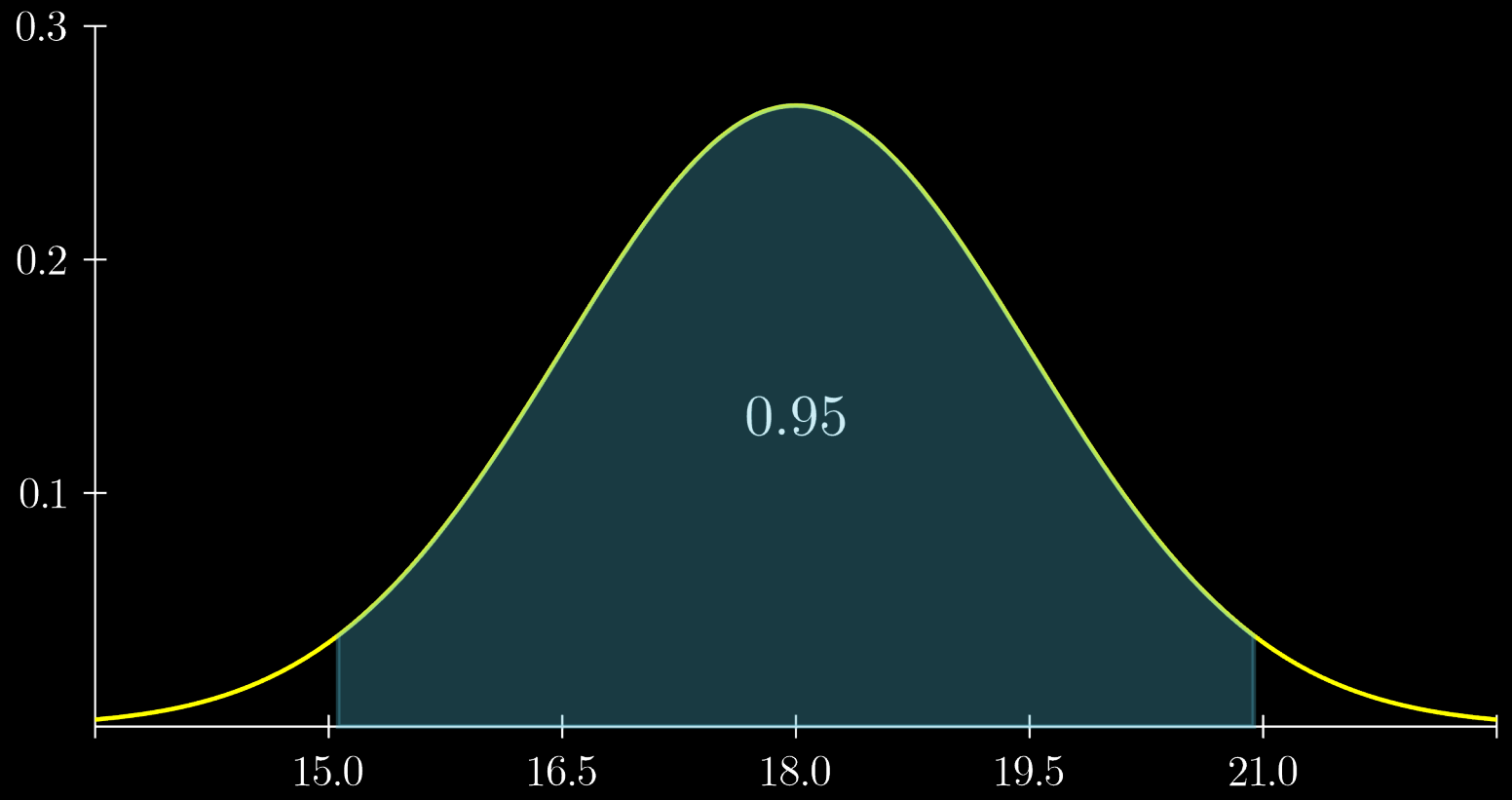
The probability of this happening by chance, the **p-value**, is 0.01428571.



Hypothesis Testing

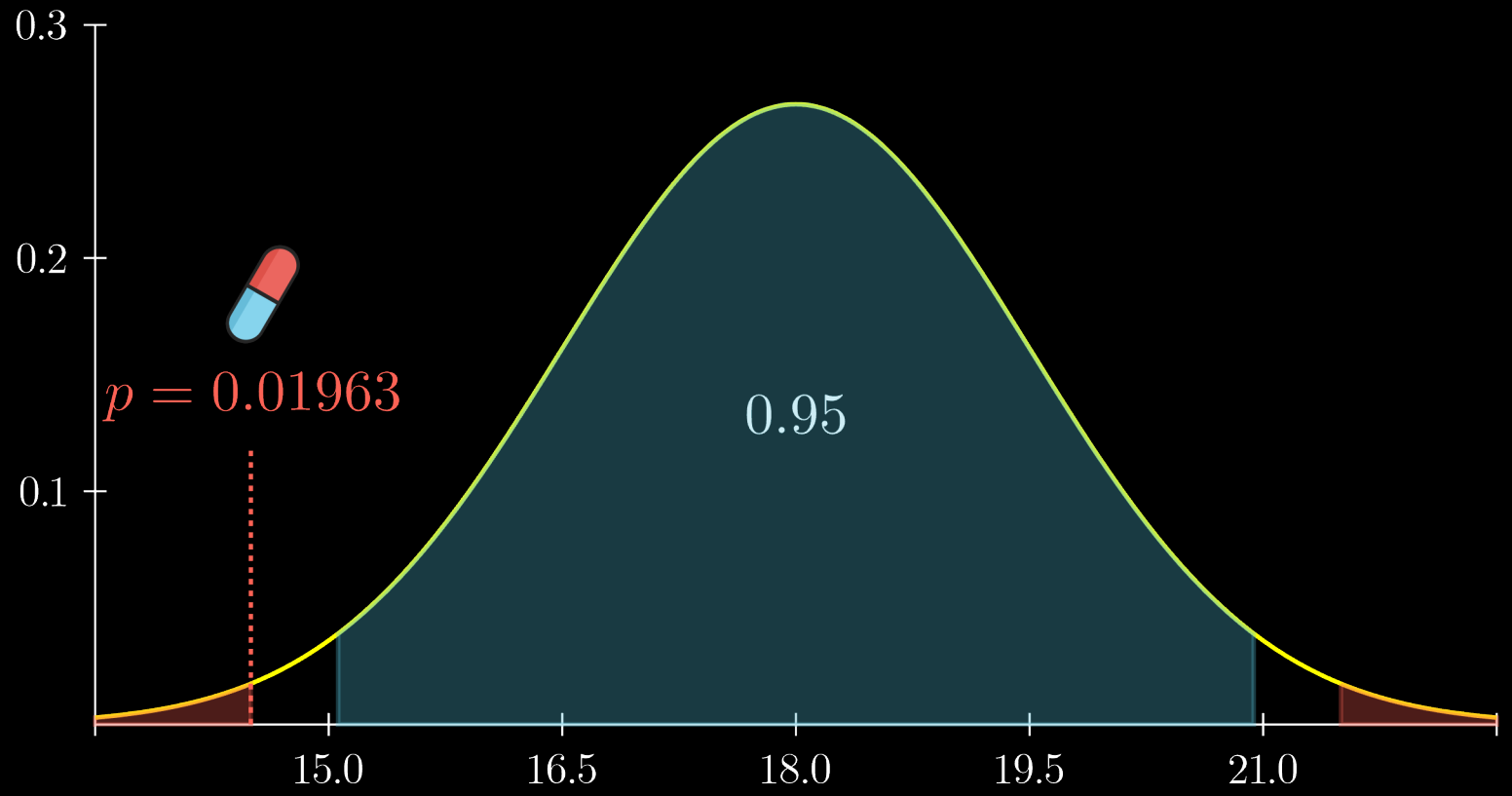


Hypothesis Testing



Hypothesis Testing

With drug average cold duration was 14 days. Coincidence?



Hypothesis Testing

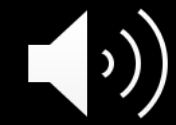
```
from scipy.stats import norm

# Cold has mean recovery of 18 days
# with 1.5 standard deviations
mean = 18
std = 1.5

# Experimental drug has 14.5 days of recovery
x = 14.5

# Probability of 14.5
tail_p = norm.cdf(x, mean, std)

# Get p-value of both tails
p_value = 2*tail_p

print(p_value) # 0.019630657257290667
```

Linear Models

Correlation, Linear Regression, and Logistic Regression

Calculating R

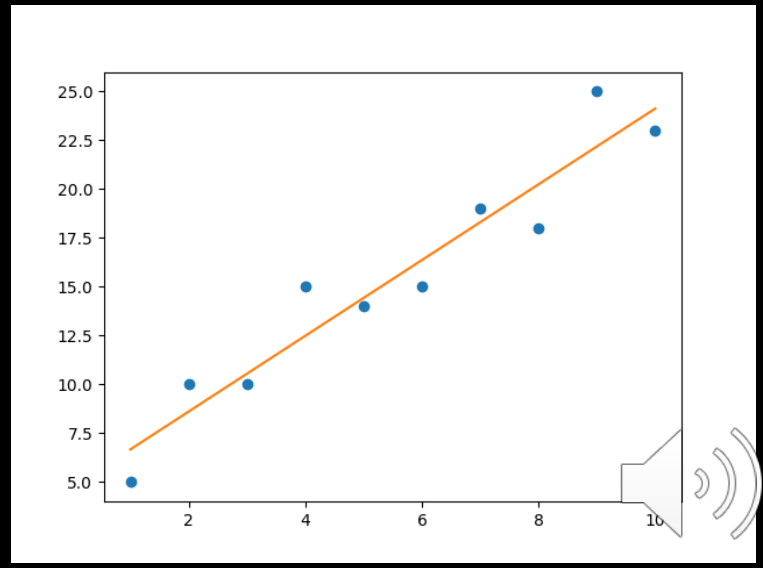
```
import pandas as pd

# Read data into Pandas dataframe
df = pd.read_csv('https://bit.ly/2KF29Bd', delimiter=",")

# Print correlations between variables
correlations = df.corr(method='pearson')
print(correlations)

# OUTPUT:
#          x         y
# x  1.000000  0.957586
# y  0.957586  1.000000
```



Linear Regression with scikit-learn

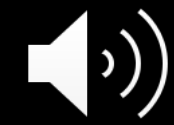
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Import points
df = pd.read_csv('https://bit.ly/3goOAnt',
delimiter=",")

# Extract input variables (all rows, all columns
but last column)
X = df.values[:, :-1]

# Extract output column (all rows, last column)
Y = df.values[:, -1]

# Fit a line to the points
fit = LinearRegression().fit(X, Y)
```

```
m = fit.coef_.flatten()
b = fit.intercept_.flatten()
print("m = {0}".format(m)) # m = [1.93939394]
print("b = {0}".format(b)) # b = [4.73333333]

# show in chart
plt.plot(X, Y, 'o') # scatterplot
plt.plot(X, m*X+b) # line
plt.show()
```



Calculating Sum of Squares

```
import pandas as pd

# Import points
points = pd.read_csv("https://bit.ly/2KF29Bd").itertuples()

# Test with a given line
m = 1.93939
b = 4.73333

sum_of_squares = 0.0

# calculate sum of squares
for p in points:
    y_actual = p.y
    y_predict = m*p.x + b
    residual_squared = (y_predict - y_actual)**2
    sum_of_squares += residual_squared

print("sum of squares = {}".format(sum_of_squares))
# sum of squares = 28.096969704500005
```



Using Inverse Matrices

```
import pandas as pd
from numpy.linalg import inv
import numpy as np

# Import points
df = pd.read_csv('https://bit.ly/3goOAnt',
delimiter=",")

# Extract input variables (all rows, all columns but
last column)
X = df.values[:, :-1].flatten()

# Add placeholder "1" column to generate intercept
X_1 = np.vstack([X, np.ones(len(X))]).T

# Extract output column (all rows, last column)
Y = df.values[:, -1]
```

```
# Calculate coefficients for slope and
intercept
b = inv(X_1.transpose() @ X_1) @
     (X_1.transpose() @ Y)
print(b) # [1.93939394, 4.73333333]

# Predict against the y-values
y_predict = X_1 @ b

print(y_predict)
# [ 6.67272727  8.61212121 10.55151515
12.49090909 14.43030303 16.36969697
18.30909091 20.24848485 22.18787879
24.12727273]
```

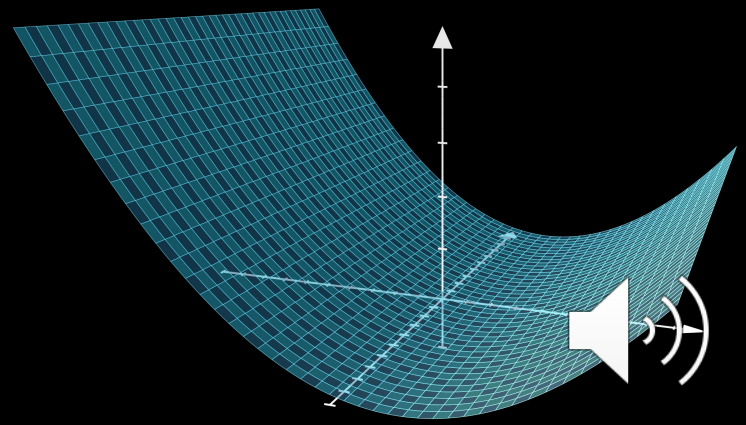
Using Gradient Descent

```python
import pandas as pd

# Import points from CSV
points = list(pd.read_csv("https://bit.ly/2KF29Bd").itertuples())

# Building the model
m = 0.0
b = 0.0


# The learning Rate
L = .001

# The number of iterations
iterations = 100_000

n = float(len(points))  # Number of elements in X
```

```python
# Perform Gradient Descent
for i in range(iterations):

    # slope with respect to m
    D_m = sum(2 * p.x * ((m * p.x + b) - p.y) for p in points)

    1# slope with respect to b
    D_b = sum(2 * ((m * p.x + b) - p.y) for p in points)

    # update m and b
    m -= L * D_m
    b -= L * D_b

print("y = {0}x + {1}".format(m, b))
# y = 1.9393939393939548x + 4.733333333333227
```

$$\frac{\partial}{\partial m} = \sum_{i=0}^{n} 2(b + mx_i - y_i)x_i$$

$$\frac{\partial}{\partial b} = \sum_{i=0}^{n} 2b + 2mx_i - 2y_i$$



Logistic Regression with scikit-learn

```
import pandas as pd
from sklearn.linear_model import LogisticRegression

# Load the data
df = pd.read_csv('https://bit.ly/33ebs2R', delimiter=",")

# Extract input variables (all rows, all columns but last column)
X = df.values[:, :-1]

# Extract output column (all rows, last column)
Y = df.values[:, -1]
```

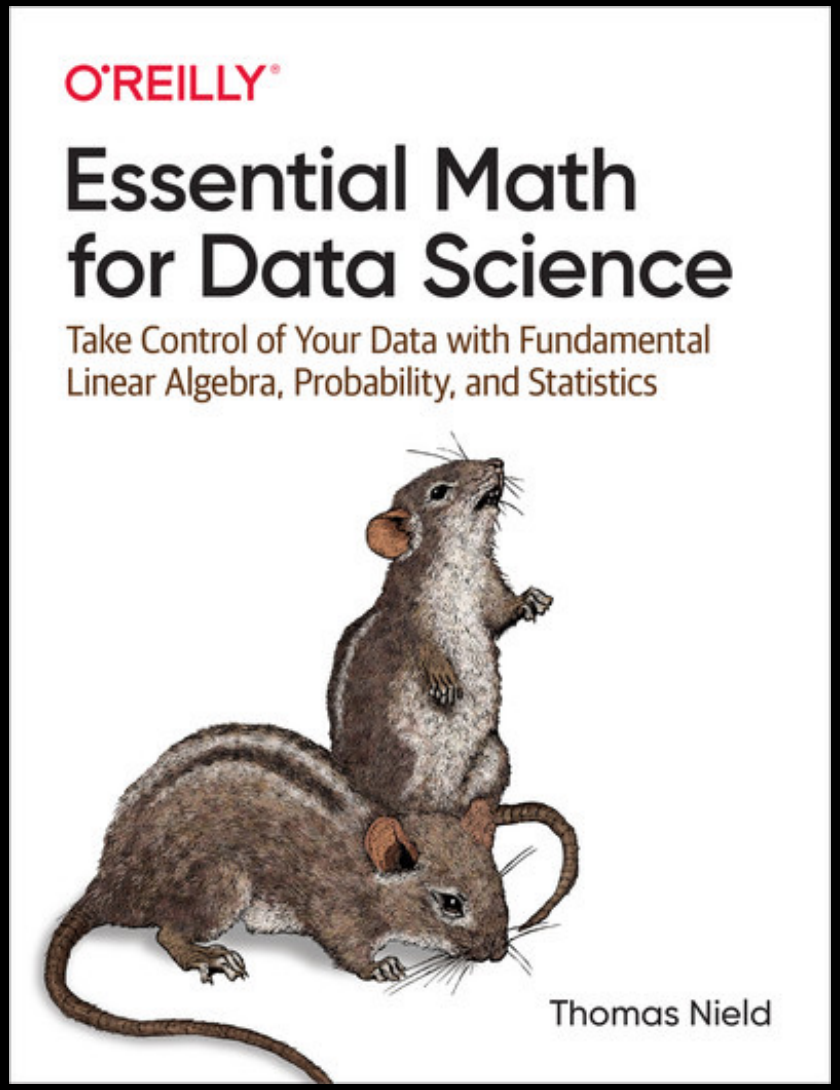
```
# Perform logistic regression
# Turn off penalty
model = LogisticRegression(penalty='none')
model.fit(X, Y)

# print beta1
print(model.coef_.flatten()) # 0.69267212

# print beta0
print(model.intercept_.flatten()) # -3.17576395
```

Neural Network in scikit-learn

```
import pandas as pd
# load data
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

df = pd.read_csv('https://bit.ly/3GsNzGt', delimiter=",")

# Extract input variables (all rows, all columns
# but last column)
# Note we should do some linear scaling here
X = (df.values[:, :-1] / 255.0)

# Extract output column (all rows, last column)
Y = df.values[:, -1]

# Separate training and testing data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
    test_size=1/3)
```

```
nn = MLPClassifier(solver='sgd',
                   hidden_layer_sizes=(3, ),
                   activation='relu',
                   max_iter=100_000,
                   learning_rate_init=.05)

nn.fit(X_train, Y_train)

# Print weights and biases
print(nn.coefs_ )
print(nn.intercepts_)

print("Training set score: %f" % nn.score(X_train, Y_train))
print("Test set score: %f" % nn.score(X_test, Y_test))
```

Thank you!


O'REILLY
Essential Math for Data Science
Take Control of Your Data with Fundamental Linear Algebra, Probability, and Statistics
Thomas Nield