calculus-in-python-1

April 17, 2025

0.0.1 What is a Derivative?

The **derivative** of a function at a certain point is the **slope of the tangent line** to the graph of the function at that point.

Think of it like this:

- If a curve is like a hill, and you're standing on a point on that hill…
- The **tangent line** is a straight path that *just touches* the hill at your feet and follows its slope at that exact spot.
- The **derivative** tells you how steep the hill is at that point.
  - Positive slope? You're walking uphill.
  - Negative slope? You're going downhill.
  - Slope = 0? You're on flat ground (like the top of a hill or bottom of a valley).

Let's Visualize It with Python Here's a code snippet using `matplotlib` that: - Plots a function. - Shows the tangent line at a specific point.

We'll do this for three functions:
1. $f(x) = x^2$
2. $f(x) = \sin(x)$
3. $f(x) = \ln(x)$

0.1 Introduction: What is Calculus?

Imagine you're riding a bike.
You look at your **speedometer**, it's telling you how fast you're going *right now*.
That's what **derivatives** are about: understanding how fast something is changing at any moment.

Now think of your **odometer**, it tells you how far you've gone in total.
That's what **integrals** help you calculate: adding up small bits of change to find the total.

0.1.1 The Big Ideas of Calculus

We'll focus on **two main ideas** in calculus:

1. Derivatives → Change

- How fast is something changing?

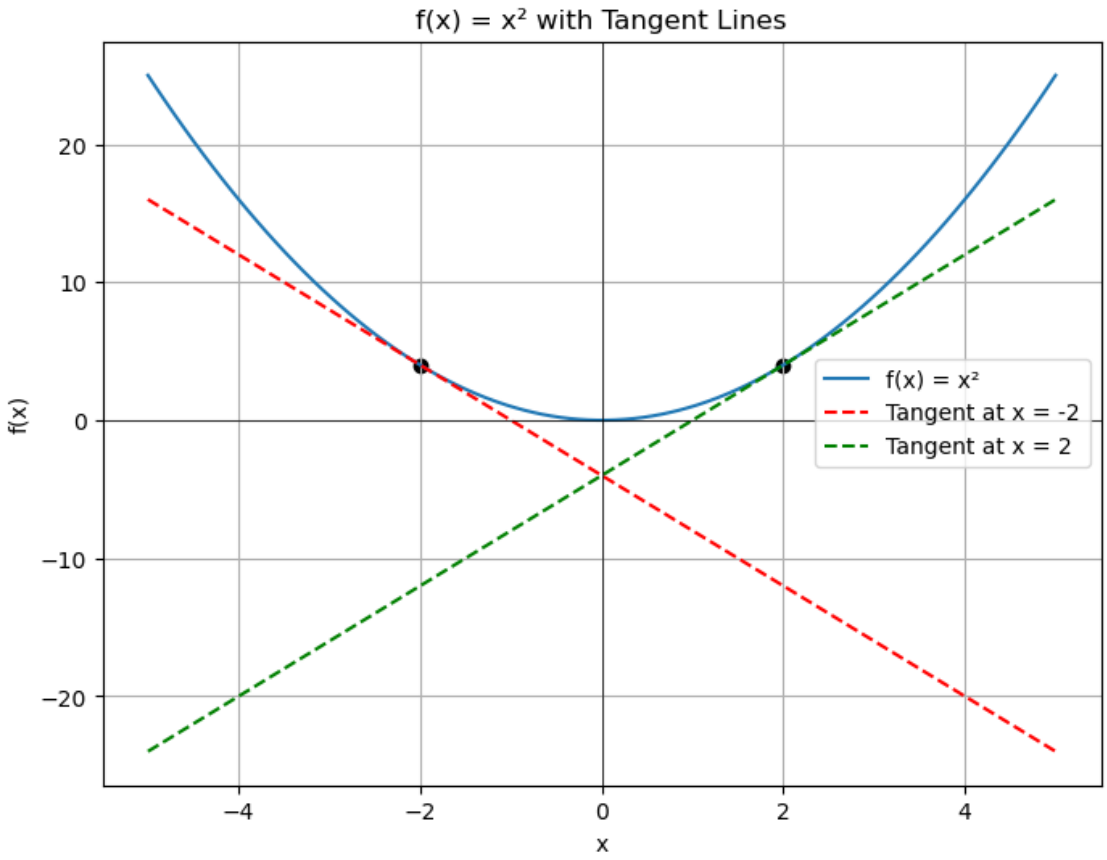
- Example: How fast is a car moving at a specific moment?

2. Integrals → Area

- How much has something changed in total?
- Example: How far has the car gone over time?

---

[10]:
```
import numpy as np
import matplotlib.pyplot as plt

# Define the function and its derivative
def f(x):
    return x**2

def df(x):
    return 2*x  # derivative of x^2 is 2x

# Main x range for the function
x = np.linspace(-5, 5, 100)
y = f(x)

# Tangent points
x1, x2 = -2, 2
y1, y2 = f(x1), f(x2)
m1, m2 = df(x1), df(x2)

# x range for tangent lines (small range around each point)
x_tangent = np.linspace(-5, 5, 100)
tangent1 = m1 * (x_tangent - x1) + y1
tangent2 = m2 * (x_tangent - x2) + y2

# Plot everything
plt.figure(figsize=(8,6))
plt.plot(x, y, label='f(x) = x²')
plt.plot(x_tangent, tangent1, '--r', label='Tangent at x = -2')
plt.plot(x_tangent, tangent2, '--g', label='Tangent at x = 2')
plt.scatter([x1, x2], [y1, y2], color='black')  # mark the tangent points
plt.title("f(x) = x² with Tangent Lines")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.grid(True)
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.legend()
plt.show()
```



0.1.2 Sine function

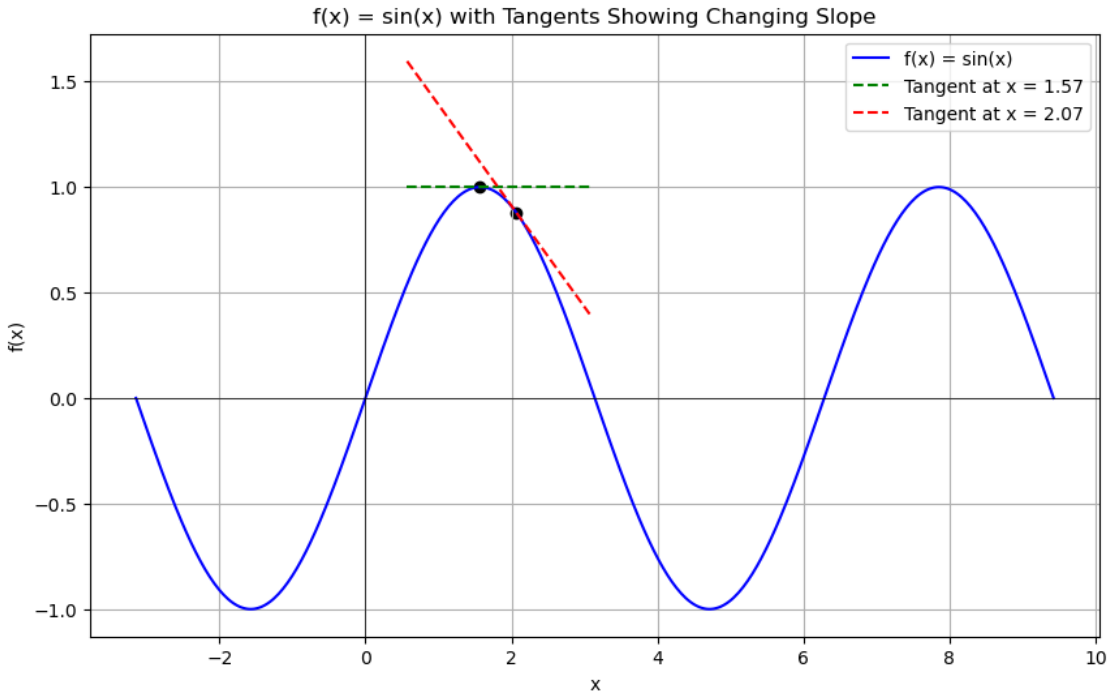
This graph shows the function $f(x) = \sin(x)$ along with two **tangent lines** at nearby points. A **tangent line** touches the curve at a single point and shows the **slope** of the function at that point — in other words, how fast the function is changing there.

By comparing the two tangents: - You can **see the change in direction** of the slope — from steeper to flatter. - This illustrates that the **derivative** of $\sin(x)$ tells us how the function is **moving** at every point.

Even though the points are close, their tangent lines are different. That shows how the rate of change itself is always changing — a core idea in **calculus**.

[16]:
```
import numpy as np
import matplotlib.pyplot as plt

# Define the function and its derivative
def f(x):
    return np.sin(x)

def df(x):
```

```
    return np.cos(x)

# Wider x range for full sine wave view
x = np.linspace(-np.pi, 3*np.pi, 400)
y = f(x)

# Two close points for tangents
x1 = np.pi / 2       #  1.57
x2 = np.pi / 2 + 0.5 #  2.07
y1, y2 = f(x1), f(x2)
m1, m2 = df(x1), df(x2)

# Tangent lines around each point
x_tangent = np.linspace(x1 - 1, x2 + 1, 100)
tangent1 = m1 * (x_tangent - x1) + y1
tangent2 = m2 * (x_tangent - x2) + y2

# Plot everything
plt.figure(figsize=(10, 6))
plt.plot(x, y, label='f(x) = sin(x)', color='blue')
plt.plot(x_tangent, tangent1, '--g', label=f'Tangent at x = {x1:.2f}')
plt.plot(x_tangent, tangent2, '--r', label=f'Tangent at x = {x2:.2f}')
plt.scatter([x1, x2], [y1, y2], color='black')  # Points of tangency
plt.title("f(x) = sin(x) with Tangents Showing Changing Slope")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(True)
plt.legend()
plt.show()
```



Follow-Up Tasks:

1. **Explore More Tangents:**
   - Pick a new point on the sine curve, e.g., $x = 0$, and plot the tangent there. How does the slope change compared to the points you've already used?
2. **Compare with Other Functions:**
   - Plot $f(x) = \cos(x)$ and find its tangents at two points, just like we did for $\sin(x)$.
3. **Find Derivatives:**
   - Choose any function you like (e.g., $f(x) = x^3$ or $f(x) = \tan(x)$ and compute its **derivative**. Then plot the tangent lines at different points to verify your calculation.

0.2 Concept of an Integral

In calculus, the **integral** of a function represents the **accumulated area** under the curve of that function over a specified interval. While a **derivative** gives us the rate of change (or slope) of a function, an **integral** tells us the total amount of change or the area accumulated between two points.

Key Ideas:

- The integral of a function $f(x)$ from $x = a$ to $x = b$ is written as:

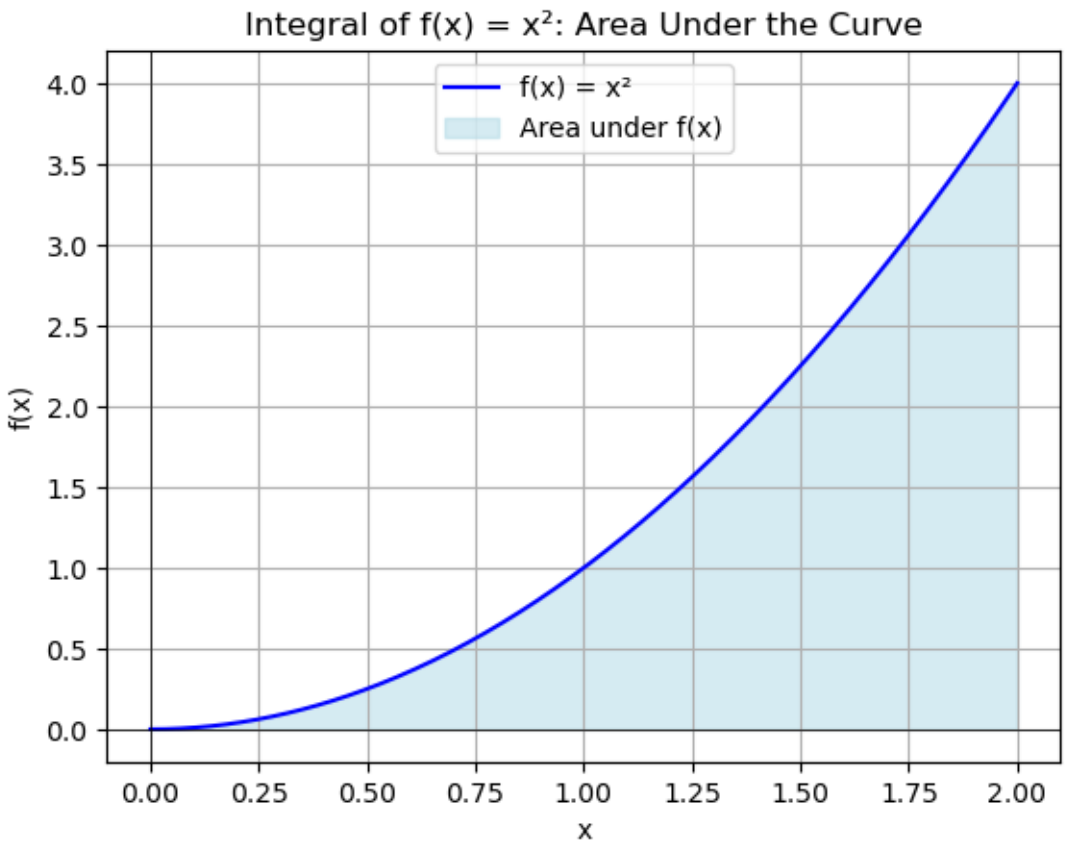
$$\int_a^b f(x)\,dx$$

  This represents the **area under** the curve of $f(x)$ between $x = a$ and $x = b$.

- If $f(x)$ is positive, the area is above the x-axis; if $f(x)$ is negative, the area is below the x-axis.
- The **definite integral** gives the **total area**, and the **indefinite integral** gives the general form of the area as a function of (x).

Visualizing the Integral as Area Under the Curve Now, let's visualize this concept by plotting the **area under the curve** for a polynomial function, say $f(x) = x^2$, from $x = 0$ to $x = 2$. We will shade the area under the curve to show the integral visually.

[24]:
```
# Define the function
def f(x):
    return x**2

# Define the x range for plotting
x = np.linspace(0, 2, 400)
y = f(x)

# Plot the function
plt.plot(x, y, label="f(x) = x²", color='blue')

# Fill the area under the curve (integral)
plt.fill_between(x, y, color='lightblue', alpha=0.5, label="Area under f(x)")

# Add labels and title
plt.title("Integral of f(x) = x²: Area Under the Curve")
plt.xlabel("x")
plt.ylabel("f(x)")

# Adding x and y axes
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)

# Display the plot
plt.legend()
plt.grid(True)
plt.show()
```

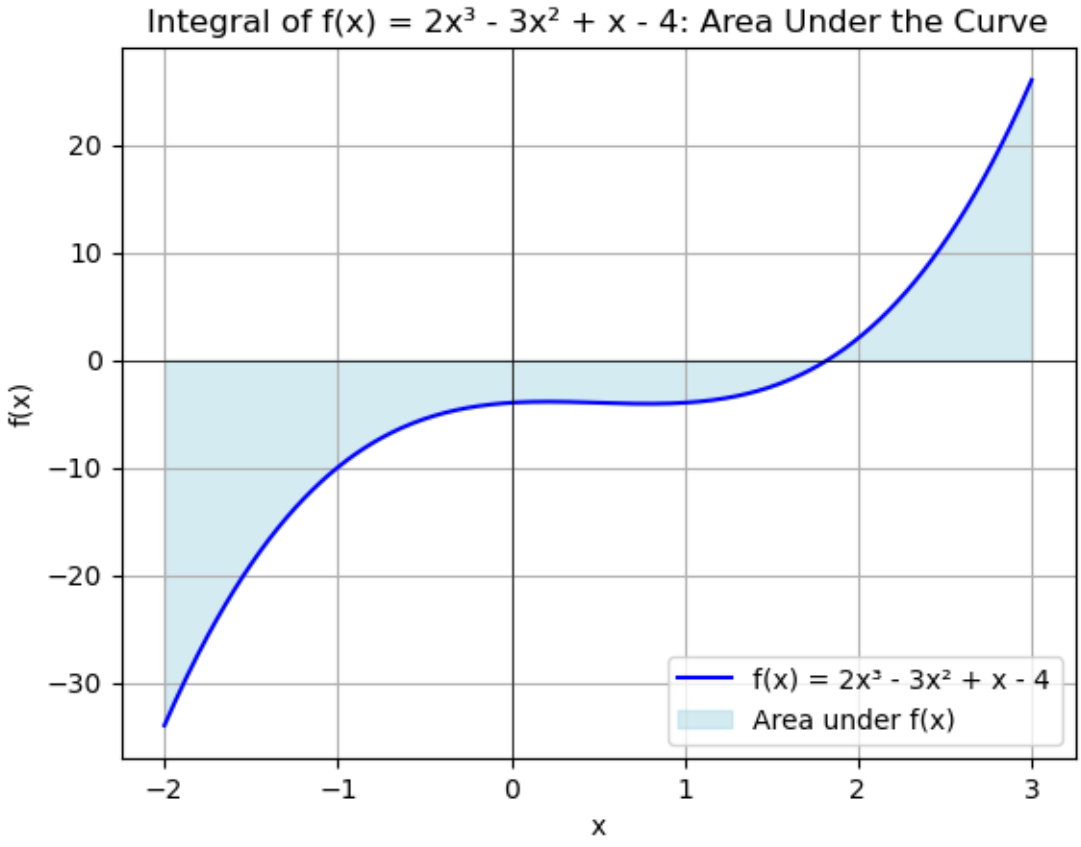


Concept of Integral for a Polynomial For more complex polynomials like $f(x) = 2x^3 - 3x^2 + x - 4$, the concept of the **integral** still holds: it represents the total **area under the curve** between two specific points.

In this case, the integral will calculate the accumulated area from $x = -2$ to $x = 2$. Some parts of the polynomial may be above the x-axis (positive area) and some parts may be below (negative area). The integral will account for both.

[38]:
```
import numpy as np
import matplotlib.pyplot as plt

# Define the complex polynomial function
def f(x):
    return 2*x**3 - 3*x**2 + x - 4

# Define the x range for plotting
x = np.linspace(-2, 3, 400)
y = f(x)

# Plot the function
```

```
plt.plot(x, y, label="f(x) = 2x³ - 3x² + x - 4", color='blue')

# Fill the area under the curve (integral)
plt.fill_between(x, y, color='lightblue', alpha=0.5, label="Area under f(x)")

# Add labels and title
plt.title("Integral of f(x) = 2x³ - 3x² + x - 4: Area Under the Curve")
plt.xlabel("x")
plt.ylabel("f(x)")

# Adding x and y axes
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)

# Display the plot
plt.legend()
plt.grid(True)
plt.show()
```



[ ]: