1 Influence_param_on_option_price.py

Purpose

This script is a visual intuition builder. It shows how each key input in the Black-Scholes model changes the value of a European call or put option. If you've ever wondered *"How does more time affect an option?"* or *"What happens if volatility jumps?"*, this is the code that answers it — not in words, but in pictures.

Core idea

The script fixes a "base case" (spot, strike, time, rate, volatility, dividend yield) and then changes **one parameter at a time**, plotting the option price against that parameter. The four plots are:

1. Time to maturity → the effect of giving the option more (or less) time to be in the money.
2. Volatility → the market's view of uncertainty in the underlying.
3. Risk-free rate → how the cost of carrying money impacts option values.
4. Dividend yield → the effect of expected payouts on equity options.

What you'll learn by running it

- **Time** usually increases an option's value because optionality is worth more when you have longer to wait for favourable moves. But for deep-in-the-money puts, the pattern can be subtler — the discounting effect can offset some of that gain.
- **Volatility** is always "long vol" for options. Higher vol means more extreme possible outcomes, which makes both calls and puts more valuable.
- **Interest rates** lift calls and depress puts for equities, because they raise the forward price.
- **Dividends** pull the forward price down, so calls lose value and puts gain.

Learning tip

Try changing the base spot price to be below the strike and see how the curves shift. You'll notice that calls become much less sensitive to some parameters when they are far out-of-the-money.

2 Heatmap_Param_Influence.py

Purpose

This takes the same idea as the first script — "how does a parameter affect option price?" — but adds a second dimension: **spot price**. The result is a grid (heatmap) where you can instantly see the joint effect of spot and another parameter.

Core idea

Instead of a line plot, you get a coloured matrix:

- X axis = spot price
- Y axis = one chosen parameter (time, vol, rate, or dividend yield)
- Colour = option price

It does this for calls and puts separately, so you can compare side by side.

What you'll learn by running it

- For calls, moving right (higher spot) almost always moves you into warmer colours (higher prices). For puts, it's the opposite.
- Sensitivity is not uniform. At low volatility and short maturities, prices change quickly near the strike but are flat far away. At high vol or long maturity, the colour gradients are smoother — reflecting a softer sensitivity.
- Rates and dividends shift the whole "hot zone" up or down depending on their economic effect on forward prices.

Learning tip

Change the ranges to extreme values — like volatility up to 100% — and see how the map changes. This is a quick way to visualise what "high vol regime" really means for option pricing.

3 BS_model.py

Purpose

This script is about **validation**, not visualisation. It compares your own Black-Scholes function against a reference library's function to check whether they match.

Core idea

- You write your own black_scholes function.
- You import a known-good function from a blackscholes module.
- You run both with the same inputs and see if the prices match.

It prints out both prices and their difference for calls and puts.

What you'll learn by running it

- Even with a "standard" formula like Black-Scholes, implementation details matter — e.g., whether dividends are continuous, whether rates are compounded continuously, or how you handle edge cases.
- A tiny difference (like 1e-10) is fine and due to floating-point precision. A larger systematic gap means your formula or input handling is different.

Learning tip

Try deliberately changing your function (e.g., remove the dividend term) and see how the difference changes. This is a powerful way to understand exactly what each term in the formula is doing.