Options Trading and Plotting DSL

This document outlines the design of our DSL regarding option trading. It describes how we will handle modeling and visualizing option trading strategies. This DSL will also be able to detect and name certain strategies that the user inputs.

Purpose and Concepts

This DSL will be used to allow traders to easily test and examine option strategies and portfolios that they come up with, as well as plot them in an easy-to-understand way. Additionally, this DSL will detect option strategies (if they are pre-existing) and name them accordingly, allowing users to experiment and learn existing strategies. We want all skill levels from financial students to options trader to have the understanding to use this DSL.

Option

A contract giving the right to buy (**call**) or sell (**put**) at a fixed price.

Represented in Racket as:

```
(define-option <option-name>
  #:type <call | put>
  #:action <buy | sell>
  #:strike <Number>
  #:expiration <Number>
  #:volatility <Number>
  #:risk-free-rate <Number>
  #:quantity <Number>)
```

These variables represent:

- **Type (`#:type <call | put>`)** → Whether the option is a **call** or **put**.

- **Action (`#:action <buy | sell>`)** → Whether the option is being **bought or sold**.

- **Strike Price (`#:strike <Number>`)** → The price at which an option is exercised.

- **Expiration (`#:expiration <Number>`)** → How long the option has until expiration.

- **Volatility (`#:volatility <Number>`)** → The standard deviation of the asset's returns. A decimal value from 0-1 inclusive.

- **Risk Free Interest Rate (`#:Risk-Free Interest Rate <Number>`)** The continuously compounded risk-free rate. A decimal from 0-1 inclusive.
- **Quantity (`#:quantity <Number>`)** → The number of options purchased.

Strategy

A combination of options that are part of the same stock.

Represented in Racket as:

```
(define-option-strategy <strategy-name>
  #:ticker <String>
  #:current-price <Number>
  (<option1> <option2> ...))
```

These variables represent:

- **Ticker (`#:ticker <String>`)** → The name of the stock (e.g., "AAPL").

- **Current-price (`#:current-price <Number>`)** → The current price of the stock.

- **Option list (`(<option1> <option2> ...)`)** → List of options that make up the strategy.

Portfolio

A combination of strategies, making up a portfolio. Each strategy represents an individual stock, making a portfolio a collection of options for different stocks.

Represented in Racket as:

```
(define-portfolio <portfolio-name>
   #:account-balance <Number>
  (<strategy1> <strategy2> ...))
```

List of Strategies (`(<strategy1> <strategy2> ...)`)

- A portfolio consists of multiple strategies, each containing different options. We then can confirm if a someone has the available balance to make that trade and determine basic things about their strategies like max potential max loss, average risk etc..

Basic Strategy Functions

```
option-payoff: Option Number -> Number
; Computes the profit or loss for a single option at a given stock price.
```

```
strategy-payoff: Strategy Number -> Number
; Computes the total profit/loss for an option strategy with days till expiration.

plot-strategy: Strategy Number -> Void
; Plots the combined payoff function of one or more options with days till expiration.

compute-breakeven: Strategy -> (Listof Number)
; Finds breakeven prices where net payoff is zero.

compute-max-loss: Strategy -> Number
; Determines the maximum loss for a strategy.
```

Example Usages

Defining an Option

```
(define-option call1
  #:type call
  #:action buy
  #:strike 100
  #:expiration 30
  #:volatility 0.20
  #:risk-free-rate 0.03
  #:quantity 1)

(define-option call2
  #:type call
  #:action sell
  #:strike 110
  #:expiration 30
  #:volatility 0.10
  #:risk-free-rate 0.03
  #:quantity 1)
```

Defining a Strategy (e.g., Call Debit Spread)

```
(define-option-strategy call-debit-spread
  #:ticker "AAPL"
  #:current-price 105
  (call1 call2))
```

Defining a Portfolio

```
(define-portfolio my-portfolio
   #:balance 100,000
  (bull-call-spread bear-put-spread long-straddle))
```

Example Usage with Payoff Functions

```
(strategy-payoff bull-call-spread 110) ;; Evaluate strategy at $110

(strategy-payoff bear-put-spread 85) ;; Evaluate strategy at $85

(portfolio-payoff my-portfolio 100) ;; Evaluate entire portfolio at $100
```

Implementation Milestones

Phase 1: Core DSL Syntax

Implement basic option payoff functions
- Create `option-payoff` to compute the profit/loss for a single option at a given stock price.
- Implemnt the black-scholes model for option payoff as well. - Example: Calculate how much a trader gains or loses for a **single call or put option**.

Implement `define-option` macro
- Introduce a macro to define **option contracts** in a structured way.
- Ensures each option has the correct attributes (strike price, expiration, premium, etc.).
Catch errors **before runtime** for better user experience

Phase 2: Multi-Option Strategies

Implement `define-option-strategy` macro for multiple-leg strategies
- Allow users to define **strategies** that contain multiple options.
- Example: A **bull call spread** (buying one call, selling another).

Extend `option-payoff` to handle single-leg & multi-leg strategies
- Modify `option-payoff` so it can compute payoffs for **an entire strategy**, not just single options.
- Example: If a strategy has two options, their payoffs should be **added together**.

Implement `compute-breakeven`
- Calculate the **breakeven point** where the strategy makes $0 profit/loss.
- Helps traders understand at what stock price they stop losing money.

Implement `plot-strategy`
- Create a function to **visualize** option payoffs using Racket's `plot` library.
- This allows traders to see **profit/loss across different stock prices**.

Phase 3: Advanced

- Create option detection algorithms
- Allow users to define a **portfolio** of strategies across different stocks.

- Example: Compare the total profit/loss of **AAPL, TSLA, and GOOG options together**.