VS Code Setup Guide - Trading Strategy Simulator

Quick Start

1. Open the Workspace

```
# From command line
code trading-simulator.code-workspace

# Or from VS Code
File → Open Workspace from File... → Select trading-simulator.code-workspace
```

2. Install Recommended Extensions

VS Code will automatically prompt you to install recommended extensions. Click **"Install All"** when prompted, or:

1. Press `Cmd+Shift+P` (Mac) / `Ctrl+Shift+P` (Windows/Linux)
2. Type: "Extensions: Show Recommended Extensions"
3. Click "Install All Workspace Recommendations"

Essential Extensions

Python Development

- **Python** - Full Python language support
- **Pylance** - Fast, feature-rich language server
- **Black Formatter** - Code formatting
- **Pylint** - Code linting
- **isort** - Import sorting

Testing & Debugging

- **Python Test Explorer** - Visual test runner
- **debugpy** - Python debugger

DevOps & Infrastructure

- **Docker** - Container management
- **Kubernetes** - K8s resource management
- **Terraform** - IaC support

Database

- **SQLTools** - Database client
- **PostgreSQL Client** - PostgreSQL-specific tools

API Development

- **REST Client** - Test HTTP requests in VS Code
- **Thunder Client** - Alternative API client
- **OpenAPI (Swagger) Editor** - API documentation

Productivity

- **GitLens** - Advanced Git integration
- **Error Lens** - Inline error display
- **Todo Tree** - Track TODO comments
- **Better Comments** - Colorful comment highlighting

Keyboard Shortcuts

Development

Action	macOS	Windows/Linux
Run File	`Ctrl+Shift+P` → "Run Python File"	`F5`
Debug File	`F5`	`F5`
Run Tests	`Cmd+Shift+P` → "Test: Run All Tests"	`Ctrl+Shift+P` → "Test: Run All Tests"
Format Document	`Shift+Option+F`	`Shift+Alt+F`
Go to Definition	`F12`	`F12`

Find References	`Shift+F12`	`Shift+F12`
Rename Symbol	`F2`	`F2`

Custom Tasks (via Command Palette)

Task	Command
Start API Server	`Tasks: Run Task` → "Start FastAPI Server"
Run Tests	`Tasks: Run Task` → "Run Tests"
Format Code	`Tasks: Run Task` → "Format All Code"
Docker Compose Up	`Tasks: Run Task` → "Start Docker Compose"

Debugging Configurations

Available Debug Configurations

1. **Python: FastAPI Development Server**
   - Starts API server with hot-reload
   - Connects to local PostgreSQL/Redis
   - Best for API development
2. **Python: Run Backtest Simulator**
   - Executes the main backtest script
   - Useful for testing strategies
3. **Python: Current File**
   - Debug the currently open Python file
   - Quick debugging for any script
4. **Python: Debug Tests (Current File)**
   - Debug unit tests in current file
   - Step through test execution
5. **Docker: Python - FastAPI**
   - Debug inside Docker container
   - Full production environment simulation

How to Debug

1. **Set Breakpoints**: Click left of line numbers (red dot appears)

2. **Select Configuration**: Debug panel (Cmd+Shift+D) → Select from dropdown
3. **Start Debugging**: Press `F5` or click green play button
4. **Debug Controls**:
    - `F5` - Continue
    - `F10` - Step Over
    - `F11` - Step Into
    - `Shift+F11` - Step Out
    - `Shift+F5` - Stop

Example: Debug API Endpoint

```
# Set breakpoint on this line
@app.post("/api/v1/backtest")
async def create_backtest(request: BacktestRequest):
    backtest_id = str(uuid.uuid4())  # ← Set breakpoint here
    # ... rest of code
```

1. Click line 3 to set breakpoint
2. Select "Python: FastAPI Development Server"
3. Press `F5`
4. Send request using REST Client (see api-tests.http)
5. Debugger pauses at breakpoint
6. Inspect variables, step through code

Using REST Client

The `api-tests.http` file contains pre-configured API requests.

How to Use

1. Open `api-tests.http`
2. Click "Send Request" above any request
3. View response in split panel

Example Workflow

```
### Create Backtest
# @name createBacktest
POST http://localhost:8000/api/v1/backtest
Content-Type: application/json
```

```
{
  "strategy_type": "sma_cross",
  "symbol": "SPY",
  ...
}

### Get the backtest_id from above
@backtestId = {{createBacktest.response.body.backtest_id}}

### Check Status
GET http://localhost:8000/api/v1/backtest/{{backtestId}}/status
```

Tasks

Running Tasks

1. `Cmd+Shift+P` (Mac) / `Ctrl+Shift+P` (Windows/Linux)
2. Type: "Tasks: Run Task"
3. Select task from list

Available Tasks

Development

- **Install Python Dependencies** - Install/update requirements.txt
- **Create Virtual Environment** - Set up fresh venv
- **Start FastAPI Server** - Launch development server
- **Run Backtest Example** - Execute sample backtest

Code Quality

- **Format All Code** - Black + isort formatting
- **Lint: Pylint** - Run pylint checks
- **Lint: Flake8** - Run flake8 checks
- **Type Check: MyPy** - Type checking
- **Run All Linters** - Execute all linters in parallel

Testing

- **Run Tests** - Execute pytest suite
- **Run Tests with Coverage** - Generate coverage report
- **Open Coverage Report** - View HTML coverage

Docker

- **Start Docker Compose** - Launch all services
- **Stop Docker Compose** - Shutdown services
- **Build Docker Image** - Build production image
- **Docker Compose Logs** - Stream container logs

Kubernetes

- **Kubernetes: Apply Deployment** - Deploy to cluster
- **Kubernetes: Get Pods** - List running pods
- **Kubernetes: View Logs** - Stream pod logs
- **Kubernetes: Port Forward API** - Access cluster service locally

Database

- **Initialize Database** - Create schema and tables

Infrastructure

- **Terraform: Init** - Initialize Terraform
- **Terraform: Plan** - Preview infrastructure changes
- **Terraform: Apply** - Deploy infrastructure

Code Snippets

Type these prefixes and press `Tab` to expand:

Strategy Development

- `strategy-class` → Complete strategy class template
- `indicator` → Add technical indicator
- `backtest-setup` → Full backtest initialization

API Development

- `fastapi-endpoint` → FastAPI route with error handling
- `async-db-query` → Database query with connection management
- `redis-cache` → Redis caching pattern
- `k8s-health` → Kubernetes health endpoints

Testing

- `pytest-test` → Async test case template
- `logger` → Add logging statement

Data Structures

- `dataclass` → Create dataclass
- `enum` → Create enum
- `func-typed` → Typed function with docstring

Example Usage

Type `strategy-class` and press `Tab`:

```
class MyStrategy(TradingStrategy):
    """Description of the strategy"""

    def __init__(self, param1: int, param2: float):
        super().__init__("MyStrategy", {
            "param1": param1,
            "param2": param2
        })
        self.param1 = param1
        self.param2 = param2

    def generate_signals(self, data: pd.DataFrame) -> pd.Series:
        """
        Generate trading signals
        Returns: Series with 1 (BUY), -1 (SELL), 0 (HOLD)
        """
        signals = pd.Series(0, index=data.index)

        # Implement your strategy logic here

        return signals
```

Testing Workflow

Run Tests

1. **Open Test Explorer**: Testing icon in sidebar (beaker icon)
2. **Discover Tests**: Auto-discovers on file save

3. **Run Tests**: Click play button on test/suite
4. **Debug Test**: Right-click → "Debug Test"

Coverage Reports

Generate coverage
Tasks: Run Task → "Run Tests with Coverage"

Open HTML report
Tasks: Run Task → "Open Coverage Report"

Watch Mode

Add to `.vscode/tasks.json`:

```
{
   "label": "Watch Tests",
   "type": "shell",
   "command": "pytest-watch",
   "isBackground": true
}
```

Code Navigation

Go to Symbol

- `Cmd+Shift+O` (Mac) / `Ctrl+Shift+O` (Windows) - Symbols in file
- `Cmd+T` (Mac) / `Ctrl+T` (Windows) - Symbols in workspace

Find References

- `Shift+F12` - Find all references
- `Alt+Shift+F12` - Peek references

Rename Symbol

- F2 - Rename symbol across project

Themes & Appearance

Recommended Themes

- **GitHub Dark** - Clean, modern dark theme
- **Material Theme** - Popular material design theme

File Icons

- **VSCode Icons** - Comprehensive icon pack
- **Material Icon Theme** - Material design icons

Change Theme

1. `Cmd+K, Cmd+T` (Mac) / `Ctrl+K, Ctrl+T` (Windows)
2. Select theme from list

Settings Customization

User vs Workspace Settings

- **User Settings**: Apply to all projects
  - `Cmd+,` (Mac) / `Ctrl+,` (Windows)
- **Workspace Settings**: Apply to this project only
  - Already configured in `.vscode/settings.json`

Key Settings Configured

```
{
   "python.linting.enabled": true,
   "python.formatting.provider": "black",
   "editor.formatOnSave": true,
   "python.testing.pytestEnabled": true,
   "files.autoSave": "afterDelay"
}
```

Productivity Tips

Multi-Cursor Editing

- `Option+Click` (Mac) / `Alt+Click` (Windows) - Add cursor
- `Cmd+Option+Up/Down` (Mac) / `Ctrl+Alt+Up/Down` (Windows) - Add cursor above/below

Quick Fix

- `Cmd+.` (Mac) / `Ctrl+.` (Windows) - Show quick fixes
- Auto-import missing modules
- Add type hints
- Organize imports

Command Palette

- `Cmd+Shift+P` (Mac) / `Ctrl+Shift+P` (Windows)
- Access any command
- Search by name

Terminal

- `Ctrl+\`` - Toggle integrated terminal
- `Cmd+Shift+5` (Mac) / `Ctrl+Shift+5` (Windows) - Split terminal

File Navigation

- `Cmd+P` (Mac) / `Ctrl+P` (Windows) - Quick open file
- `Cmd+Shift+F` (Mac) / `Ctrl+Shift+F` (Windows) - Search in files

Docker Integration

View Containers

- Docker extension → Containers view
- Right-click container for options
- View logs, attach shell, stop/start

Build & Run

- Right-click `Dockerfile` → "Build Image"

- Right-click `docker-compose.yml` → "Compose Up"

Attach to Container

- Docker extension → Right-click container → "Attach Shell"
- Full terminal access inside container

Kubernetes Integration

Connect to Cluster

Set kubeconfig
doctl kubernetes cluster kubeconfig save trading-simulator-cluster

VS Code Kubernetes Extension

- Kubernetes icon in sidebar
- Browse clusters, namespaces, pods
- View logs, port-forward, exec into pods

Port Forwarding

- Right-click service → "Port Forward"
- Access cluster service on localhost

Resources

Official Documentation

- VS Code Python
- VS Code Debugging
- FastAPI

Extensions Marketplace

- Python Extension
- Docker Extension
- Kubernetes Extension

Troubleshooting

Python Interpreter Not Found

1. `Cmd+Shift+P` → "Python: Select Interpreter"
2. Choose `./venv/bin/python`
3. Reload window if needed

Tests Not Discovered

1. Check `pytest` is installed: `pip install pytest`
2. Verify test file naming: `test_*.py` or `*_test.py`
3. Reload window: `Cmd+Shift+P` → "Developer: Reload Window"

Linter Errors

Install linting tools
pip install pylint flake8 mypy black isort

Docker Commands Fail

- Ensure Docker Desktop is running
- Check Docker extension settings
- Restart VS Code

---

1. **Use GitLens Blame**: See who changed each line
2. **Peek Definition**: `Alt+F12` - View definition without leaving file
3. **Bracket Pair Colorization**: Enabled by default, helps with nested code
4. **Auto-save**: Already enabled (1 second delay)
5. **Error Lens**: See errors inline, no need to hover
6. **Todo Tree**: Track all TODO comments in project
