Tutorial 4: Building a DCF Model in VS Code - Real PE Work! 💼

> **Personal Note to Mauricio**: This is it - the big one! 🎯 DCF valuation is THE foundation of investment banking and private equity. At PE Club Brussels, you'll value companies this way weekly. But here's your advantage: while others build DCFs in Excel (slow, error-prone, not reproducible), you'll build them in Python (fast, scalable, professional). This tutorial will make you the valuation expert on your team! - Dad ❤️

📊 Progress Tracker

Where you are in the course:

- ✅ Tutorial 1: VS Code Basics (Complete!)
- ✅ Tutorial 2: GitHub & Copilot (Complete!)
- ✅ Tutorial 3: Data Analysis (Complete!)
- 🔥 **Tutorial 4: Building DCF Models (You are here!)**
- ⬜ Tutorial 5: Power User Skills

Course completion: 60% → 80% 🎯

🎯 What You'll Learn (120 minutes)

Build a professional Discounted Cash Flow (DCF) valuation model from scratch:

✅ DCF Fundamentals

- ☐ Understand DCF methodology deeply
- ☐ Calculate Free Cash Flows correctly
- ☐ Determine WACC (cost of capital)
- ☐ Compute terminal value (perpetuity & multiples)
- ☐ Bridge from enterprise to equity value

✅ Python Implementation

- ☐ Build DCF model programmatically
- ☐ Create reusable valuation functions
- ☐ Implement sensitivity analysis
- ☐ Generate scenario tables
- ☐ Automate entire valuation process

✅ Professional Skills

- ☐ Use Copilot to accelerate model building
- ☐ Create presentation-ready output
- ☐ Build valuation dashboards

- ☐ Export results to Excel for clients
- ☐ Version control your models with Git

✅ **Real-World Application**

- ☐ Value an actual company
- ☐ Test different scenarios (base/bull/bear)
- ☐ Perform sensitivity analysis
- ☐ Generate investment recommendation
- ☐ Create executive summary

💡 **Why This Matters at PE Club Brussels**: DCF is the gold standard for company valuation. Master this, and you'll evaluate any deal confidently. Python DCFs are superior to Excel: they're faster, reproducible, scalable to multiple companies, and impress tech-forward investors. This is the skill that separates good analysts from great ones! 🚀

📝 Prerequisites

Before starting:

- ✅ Completed Tutorial 1 (VS Code Basics)
- ✅ Completed Tutorial 2 (GitHub & Copilot)
- ✅ Completed Tutorial 3 (Data Analysis with Pandas)
- ✅ Understanding of corporate finance basics
- ✅ Familiarity with DCF concepts (if not, we'll review!)

New Concepts (we'll learn together!):

- ⬜ Free Cash Flow calculations
- ⬜ WACC computation
- ⬜ Terminal value methods
- ⬜ Sensitivity analysis
- ⬜ Scenario modeling

⏱️ **Estimated Time**: 120 minutes (longest tutorial - worth it!) ☕ **Suggested**: Have coffee/snacks ready. This is intense but incredibly rewarding!

Part 1: DCF Theory Review (15 minutes)

What is DCF?

Core Concept: A company is worth the present value of all its future cash flows.

$$\text{Enterprise Value} = \sum_{t=1}^{n} \frac{FCF_t}{(1 + WACC)^t} + \frac{Terminal\ Value}{(1 + WACC)^n}$$

Key Components

1. Free Cash Flow (FCF)

```
FCF = EBITDA
      - Taxes
      - CapEx
      - Change in NWC
      + D&A
```

2. Weighted Average Cost of Capital (WACC)

```
WACC = (E/V × Re) + (D/V × Rd × (1 - Tax Rate))

Where:
E = Equity Value
D = Debt Value
V = Total Value (E + D)
Re = Cost of Equity (from CAPM)
Rd = Cost of Debt
```

3. Terminal Value

```
Method 1 - Perpetuity Growth:
TV = FCF(n+1) / (WACC - g)

Method 2 - Exit Multiple:
TV = EBITDA(final) × Exit Multiple
```

4. Enterprise to Equity Value

```
Enterprise Value = PV(Cash Flows) + PV(Terminal Value)
Equity Value = Enterprise Value + Cash - Debt
Share Price = Equity Value / Shares Outstanding
```

When to Use DCF

✅ **Good for:**

- Mature, stable companies
- Predictable cash flows
- Long-term valuation
- Investment banking / PE

❌ **Challenging for:**

- Early-stage startups
- Cyclical industries

- Financial institutions
- Negative FCF companies

Part 2: Project Setup (10 minutes)

Create Project Structure

Open VS Code terminal:

```
# Create project folder
mkdir dcf-valuation-model
cd dcf-valuation-model

# Create virtual environment
python -m venv venv
venv\Scripts\activate

# Install packages
pip install pandas numpy matplotlib openpyxl jupyter plotly

# Open in VS Code
code .
```

Create Initial Files

File 1: `README.md`

```
# DCF Valuation Model

Professional discounted cash flow model built in Python.

## Features
- Complete DCF calculation
- WACC computation
- Terminal value (perpetuity & multiple)
- Sensitivity analysis
- Scenario testing

## Usage
```python
from dcf_model import DCFModel

model = DCFModel("Company Name")
# ... configure and run
```

Author

[Your Name]
```

```
**File 2: `.gitignore`**
```

venv/ *.pyc **pycache**/ .ipynb_checkpoints/ *.xlsx !template.xlsx

```
### Initialize Git

```bash
git init
git add .
git commit -m "Initial commit: DCF project setup"
```

Part 3: Build DCF Model Class (30 minutes)

Create Model File

File: `dcf_model.py`

Let's build this step-by-step using **GitHub Copilot**!

Step 1: Basic Structure

Type this comment and let Copilot complete:

```
# DCF Valuation Model Class
# Calculates enterprise value using discounted cash flow methodology
#
# Features:
# - Free cash flow projection
# - WACC calculation using CAPM
# - Terminal value (perpetuity growth method)
# - Sensitivity analysis
# - Comprehensive valuation output

import pandas as pd
import numpy as np
from typing import Dict, List, Tuple
from dataclasses import dataclass
```

Copilot will suggest imports. Press `Tab`.

Step 2: Data Classes for Inputs

```

```
@dataclass
class CompanyAssumptions:
    """Store all company-specific assumptions"""
    # Let Copilot suggest fields
    company_name: str
    # Type more comments, Copilot fills in:
    # - Base year revenue
    # - Revenue growth rates (list)
    # - EBITDA margins (list)
    # - Tax rate
    # - CapEx as % of revenue
    # - NWC as % of revenue
    # - D&A as % of revenue
```

Ask **Copilot Chat** (`Ctrl+Shift+I`):

```
"Complete the CompanyAssumptions dataclass with all fields needed
for DCF model including revenue, margins, capex, nwc, and tax assumptions"
```

Copy Copilot's response!

Step 3: DCF Model Class

```
class DCFModel:
    """Complete DCF Valuation Model"""

    def __init__(self, company_name: str):
        """Initialize DCF model"""
        self.company_name = company_name
        self.projection_years = 5
        self.projections = None
        self.dcf_results = {}

    def set_assumptions(self,
                        revenue: float,
                        revenue_growth: List[float],
                        ebitda_margins: List[float],
                        tax_rate: float,
                        capex_pct_revenue: float,
                        nwc_pct_revenue: float,
                        da_pct_revenue: float):
        """Set operating assumptions"""
        # Let Copilot complete this method

    def set_wacc_assumptions(self,
                             risk_free_rate: float,
                             equity_risk_premium: float,
                             beta: float,
                             debt_cost: float,
```

```
                            market_value_equity: float,
                            market_value_debt: float,
                            tax_rate: float):
        """Set WACC calculation assumptions"""
        # Let Copilot complete

    def calculate_wacc(self) -> float:
        """Calculate weighted average cost of capital using CAPM"""
        # Ask Copilot Chat to write this method
        # Formula: WACC = (E/V × Re) + (D/V × Rd × (1-T))
        # where Re = Rf + Beta × (Rm - Rf)
```

Use Copilot Chat:

```
"Write the calculate_wacc method using CAPM formula.
Include cost of equity = risk_free_rate + beta * equity_risk_premium.
Include after-tax cost of debt."
```

Step 4: Build Projections

```
    def build_projections(self) -> pd.DataFrame:
        """Build 5-year financial projections"""
        # Let Copilot write this - it should create DataFrame with:
        # - Revenue
        # - EBITDA
        # - D&A
        # - EBIT
        # - Taxes
        # - NOPAT
        # - CapEx
        # - Change in NWC
        # - Free Cash Flow
```

Use Copilot Chat:

```
"Write build_projections method that:
1. Creates DataFrame with years 1-5
2. Calculates Revenue using growth rates
3. Calculates EBITDA using margins
4. Calculates taxes on EBIT
5. Calculates FCF = NOPAT + D&A - CapEx - ΔNW C
Return DataFrame with all line items"
```

Step 5: Terminal Value

```
    def calculate_terminal_value(self,
                                 terminal_growth_rate: float) -> float:
        """Calculate terminal value using perpetuity growth"""
        # FCF(n+1) = FCF(n) × (1 + g)
        # TV = FCF(n+1) / (WACC - g)
        # Let Copilot complete

    def calculate_terminal_value_multiple(self,
                                          exit_multiple: float) -> float:
        """Calculate terminal value using exit multiple method"""
        # TV = Final Year EBITDA × Exit Multiple
        # Let Copilot complete
```

Step 6: DCF Valuation

```
    def calculate_dcf(self,
                      terminal_growth_rate: float = 0.025,
                      use_multiple: bool = False,
                      exit_multiple: float = 10.0) -> Dict:
        """Calculate enterprise value using DCF"""
        # 1. Get projections
        # 2. Calculate WACC
        # 3. Discount FCFs to present value
        # 4. Calculate terminal value
        # 5. Discount terminal value
        # 6. Sum for enterprise value
        # 7. Bridge to equity value
        # Let Copilot write this!
```

Use Copilot Chat:

```
"Write calculate_dcf method that:
1. Builds projections DataFrame
2. Calculates WACC
3. Discounts each year FCF using PV = FCF / (1+WACC)^year
4. Calculates terminal value
5. Sums PV of FCFs + PV of terminal value = Enterprise Value
6. Returns dictionary with all results"
```

Step 7: Sensitivity Analysis

```
    def sensitivity_analysis(self,
                             wacc_range: List[float],
                             growth_range: List[float]) -> pd.DataFrame:
        """Create sensitivity table for enterprise value"""
```

```
        # Create 2-way table varying WACC and terminal growth
        # Let Copilot complete
```

Step 8: Summary Output

```
    def print_summary(self):
        """Print valuation summary"""
        # Print formatted output of key results
        # Let Copilot complete

    def export_to_excel(self, filename: str):
        """Export model to Excel"""
        # Export projections and results to Excel
        # Let Copilot complete
```

Part 4: Test the Model (15 minutes)

Create Test File

File: `test_dcf.py`

```
"""
Test DCF Model with Sample Company
"""

from dcf_model import DCFModel

# Create model instance
model = DCFModel("TechCorp Inc.")

# Set operating assumptions
model.set_assumptions(
    revenue=1000,  # $1B base year
    revenue_growth=[0.15, 0.12, 0.10, 0.08, 0.06],  # Declining growth
    ebitda_margins=[0.25, 0.26, 0.27, 0.28, 0.28],  # Margin expansion
    tax_rate=0.25,
    capex_pct_revenue=0.05,
    nwc_pct_revenue=0.10,
    da_pct_revenue=0.03
)

# Set WACC assumptions
model.set_wacc_assumptions(
    risk_free_rate=0.04,      # 4% risk-free
    equity_risk_premium=0.06, # 6% market premium
    beta=1.2,                 # Higher risk tech company
    debt_cost=0.05,           # 5% cost of debt
    market_value_equity=5000, # $5B equity
```

```
    market_value_debt=1000,   # $1B debt
    tax_rate=0.25
)

# Run DCF
results = model.calculate_dcf(
    terminal_growth_rate=0.025,  # 2.5% perpetual growth
    use_multiple=False
)

# Print results
model.print_summary()

# Show projections
print("\n📊 Financial Projections:")
print(model.projections.round(2))

# Sensitivity analysis
print("\n📈 Sensitivity Analysis:")
wacc_range = [0.06, 0.07, 0.08, 0.09, 0.10]
growth_range = [0.015, 0.020, 0.025, 0.030, 0.035]
sensitivity = model.sensitivity_analysis(wacc_range, growth_range)
print(sensitivity.round(0))

# Export to Excel
model.export_to_excel('techcorp_dcf_valuation.xlsx')
print("\n✅ Model exported to Excel!")
```

Run the Model

In terminal:

```
python test_dcf.py
```

Expected output:

```
============================================================
DCF VALUATION SUMMARY - TechCorp Inc.
============================================================

WACC Calculation:
  Cost of Equity (CAPM): 11.20%
  After-tax Cost of Debt: 3.75%
  Weighted Average Cost of Capital: 9.83%

Valuation Results:
  PV of Projected FCF (5 years): $XXX M
  Terminal Value: $XXX M
  PV of Terminal Value: $XXX M
```

```
  Enterprise Value: $XXX M
  + Cash: $XX M
  - Debt: $XXX M
  = Equity Value: $XXX M

  Shares Outstanding: XX M
  Implied Share Price: $XX.XX

============================================================
```

Part 5: Build Interactive Notebook (20 minutes)

Create Jupyter Notebook

File: `dcf_interactive.ipynb`

Cell 1: Setup

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from dcf_model import DCFModel

print("✅ DCF Model Loaded!")
```

Cell 2: Company Inputs

```
# Company to value
COMPANY_NAME = "TechCorp Inc."
BASE_REVENUE = 1000  # millions

# Create model
model = DCFModel(COMPANY_NAME)

print(f"📊 Building DCF for: {COMPANY_NAME}")
print(f"Base Year Revenue: ${BASE_REVENUE}M")
```

Cell 3: Operating Assumptions

```
# Set assumptions (you can adjust these!)
model.set_assumptions(
    revenue=BASE_REVENUE,
    revenue_growth=[0.15, 0.12, 0.10, 0.08, 0.06],
    ebitda_margins=[0.25, 0.26, 0.27, 0.28, 0.28],
```

```
    tax_rate=0.25,
    capex_pct_revenue=0.05,
    nwc_pct_revenue=0.10,
    da_pct_revenue=0.03
)

# Show assumption table
assumptions_df = pd.DataFrame({
    'Year': ['1', '2', '3', '4', '5'],
    'Revenue Growth': ['15%', '12%', '10%', '8%', '6%'],
    'EBITDA Margin': ['25%', '26%', '27%', '28%', '28%']
})

assumptions_df
```

Cell 4: WACC Inputs

```
model.set_wacc_assumptions(
    risk_free_rate=0.04,
    equity_risk_premium=0.06,
    beta=1.2,
    debt_cost=0.05,
    market_value_equity=5000,
    market_value_debt=1000,
    tax_rate=0.25
)

wacc = model.calculate_wacc()
print(f"📌 Calculated WACC: {wacc*100:.2f}%")
```

Cell 5: Build Projections

```
# Build model
results = model.calculate_dcf(terminal_growth_rate=0.025)

# Show projections
model.projections.round(2)
```

Cell 6: Visualize Projections

```
# Plot revenue and FCF
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

# Revenue growth
years = model.projections.index
ax1.bar(years, model.projections['Revenue'], color='lightblue', alpha=0.8)
ax1.set_title('Revenue Projection', fontsize=14, fontweight='bold')
```

```
ax1.set_ylabel('Revenue ($M)')
ax1.set_xlabel('Year')
ax1.grid(True, alpha=0.3)

# Free Cash Flow
colors = ['green' if x > 0 else 'red' for x in model.projections['FCF']]
ax2.bar(years, model.projections['FCF'], color=colors, alpha=0.8)
ax2.set_title('Free Cash Flow', fontsize=14, fontweight='bold')
ax2.set_ylabel('FCF ($M)')
ax2.set_xlabel('Year')
ax2.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```

Cell 7: Valuation Waterfall

```
# Create waterfall chart
waterfall_data = [
    ('PV of FCF', results['pv_fcf_total']),
    ('PV of TV', results['pv_terminal_value']),
    ('Enterprise Value', results['enterprise_value']),
    ('+ Cash', results.get('cash', 0)),
    ('- Debt', -results.get('debt', 1000)),
    ('Equity Value', results['equity_value'])
]

labels = [x[0] for x in waterfall_data]
values = [x[1] for x in waterfall_data]

# Plot
plt.figure(figsize=(12, 6))
plt.bar(labels, values, color=['blue', 'blue', 'green', 'lightgreen',
'red', 'darkgreen'])
plt.title('DCF Valuation Waterfall', fontsize=16, fontweight='bold')
plt.ylabel('Value ($M)')
plt.xticks(rotation=45, ha='right')
plt.grid(True, alpha=0.3, axis='y')
plt.tight_layout()
plt.show()

print(f"\n🎯 Implied Equity Value: ${results['equity_value']:,.0f}M")
```

Cell 8: Sensitivity Analysis

```
# 2-way sensitivity table
wacc_range = np.arange(0.07, 0.13, 0.01)
growth_range = np.arange(0.015, 0.036, 0.005)
```

```
sensitivity = model.sensitivity_analysis(
    wacc_range.tolist(),
    growth_range.tolist()
)

print("📊 Enterprise Value Sensitivity ($M)")
print("Rows = Terminal Growth Rate | Columns = WACC")
sensitivity.round(0)
```

Cell 9: Interactive Sensitivity Heatmap

```
# Create interactive heatmap
fig = go.Figure(data=go.Heatmap(
    z=sensitivity.values,
    x=[f"{x*100:.1f}%" for x in sensitivity.columns],
    y=[f"{y*100:.1f}%" for y in sensitivity.index],
    colorscale='RdYlGn',
    text=sensitivity.round(0).values,
    texttemplate='%{text}',
    textfont={"size": 10},
    colorbar=dict(title="EV ($M)")
))

fig.update_layout(
    title='Enterprise Value Sensitivity Analysis',
    xaxis_title='WACC',
    yaxis_title='Terminal Growth Rate',
    height=500
)

fig.show()
```

Cell 10: Scenario Analysis

```
# Test different scenarios
scenarios = {
    'Base Case': {'growth': 0.025, 'wacc_adjust': 0.00},
    'Bull Case': {'growth': 0.035, 'wacc_adjust': -0.01},
    'Bear Case': {'growth': 0.015, 'wacc_adjust': 0.01}
}

scenario_results = []

for scenario_name, params in scenarios.items():
    # Recalculate with adjusted parameters
    temp_results = model.calculate_dcf(
        terminal_growth_rate=params['growth']
    )
```

```
    scenario_results.append({
        'Scenario': scenario_name,
        'Terminal Growth': f"{params['growth']*100:.1f}%",
        'Enterprise Value': temp_results['enterprise_value'],
        'Equity Value': temp_results['equity_value']
    })

scenario_df = pd.DataFrame(scenario_results)
scenario_df
```

Cell 11: Export Results

```
# Export comprehensive report
model.export_to_excel(f'{COMPANY_NAME.replace(" ", "_")}_DCF_Model.xlsx')
print(f"✅ Complete model exported to Excel!")
```

Part 6: Real Company Example - Apple Inc. (15 minutes)

Get Real Financial Data

New notebook: `apple_dcf.ipynb`

Cell 1:

```
import yfinance as yf
import pandas as pd
from dcf_model import DCFModel

# Get Apple data
aapl = yf.Ticker("AAPL")
info = aapl.info
financials = aapl.financials
balance_sheet = aapl.balance_sheet

print("📱 Apple Inc. - DCF Valuation")
print(f"Market Cap: ${info.get('marketCap')/1e9:.1f}B")
print(f"Current Price: ${info.get('currentPrice'):.2f}")
```

Cell 2:

```
# Extract key metrics
latest_revenue = financials.loc['Total Revenue'].iloc[0] / 1e9  # billions
latest_ebitda = financials.loc['EBITDA'].iloc[0] / 1e9 if 'EBITDA' in
financials.index else None
total_debt = balance_sheet.loc['Total Debt'].iloc[0] / 1e9 if 'Total Debt'
in balance_sheet.index else 0
```

```
cash = balance_sheet.loc['Cash'].iloc[0] / 1e9 if 'Cash' in
balance_sheet.index else 0
shares = info.get('sharesOutstanding', 0) / 1e9

print(f"Latest Revenue: ${latest_revenue:.1f}B")
print(f"Total Debt: ${total_debt:.1f}B")
print(f"Cash: ${cash:.1f}B")
print(f"Shares Outstanding: {shares:.2f}B")
```

Cell 3:

```
# Build DCF for Apple
apple_model = DCFModel("Apple Inc.")

# Set assumptions (analyst estimates)
apple_model.set_assumptions(
    revenue=latest_revenue,
    revenue_growth=[0.08, 0.07, 0.06, 0.05, 0.04],  # Mature growth
    ebitda_margins=[0.30, 0.30, 0.31, 0.31, 0.32],  # Strong margins
    tax_rate=0.15,  # Apple's effective tax rate
    capex_pct_revenue=0.03,
    nwc_pct_revenue=0.05,
    da_pct_revenue=0.025
)

# WACC assumptions
apple_model.set_wacc_assumptions(
    risk_free_rate=0.045,
    equity_risk_premium=0.055,
    beta=1.25,
    debt_cost=0.03,
    market_value_equity=info.get('marketCap') / 1e9,
    market_value_debt=total_debt,
    tax_rate=0.15
)

# Calculate
apple_results = apple_model.calculate_dcf(
    terminal_growth_rate=0.025
)

apple_model.print_summary()
```

Cell 4:

```
# Compare to market
implied_price = apple_results['equity_value'] * 1000 / shares  # Convert B
to M, then per share
current_price = info.get('currentPrice')
```

```
print(f"\n🎯 VALUATION COMPARISON")
print(f"{'='*50}")
print(f"DCF Implied Price: ${implied_price:.2f}")
print(f"Current Market Price: ${current_price:.2f}")
print(f"Upside/(Downside): {((implied_price/current_price - 1) *
100):.1f}%")

if implied_price > current_price:
    print(f"\n✅ Stock appears UNDERVALUED by
{((implied_price/current_price - 1) * 100):.1f}%")
else:
    print(f"\n⚠ Stock appears OVERVALUED by {((1 -
implied_price/current_price) * 100):.1f}%")
```

Part 7: Best Practices & Tips (10 minutes)

Code Organization

```
# Good: Modular structure
dcf-valuation-model/
├── dcf_model.py          # Core model class
├── utils.py              # Helper functions
├── data_loader.py        # Data import functions
├── test_dcf.py           # Unit tests
├── examples/
│   ├── tech_company.ipynb
│   ├── apple_dcf.ipynb
│   └── custom_valuation.ipynb
└── outputs/              # Excel exports
```

Error Handling

```
def calculate_wacc(self) -> float:
    """Calculate WACC with validation"""
    try:
        # Validation
        if self.market_value_equity <= 0:
            raise ValueError("Equity value must be positive")

        if not 0 <= self.tax_rate <= 1:
            raise ValueError("Tax rate must be between 0 and 1")

        # Calculation
        total_value = self.market_value_equity + self.market_value_debt
        # ...

    except Exception as e:
```

```
        print(f"❌ Error calculating WACC: {e}")
        return None
```

Documentation

```
def calculate_dcf(self, terminal_growth_rate: float = 0.025) -> Dict:
    """
    Calculate enterprise value using DCF methodology.

    Parameters:
    -----------
    terminal_growth_rate : float, default=0.025
        Perpetual growth rate for terminal value (e.g., 0.025 = 2.5%)
        Should be <= long-term GDP growth rate

    Returns:
    --------
    Dict containing:
        - enterprise_value : float
        - equity_value : float
        - pv_fcf_total : float
        - pv_terminal_value : float
        - wacc : float

    Example:
    --------
    >>> model = DCFModel("Company")
    >>> results = model.calculate_dcf(terminal_growth_rate=0.03)
    >>> print(f"EV: ${results['enterprise_value']:.0f}M")
    """
```

Version Control

```
# Commit after each major milestone
git add dcf_model.py
git commit -m "Add terminal value calculation methods"

git add test_dcf.py
git commit -m "Create test case for DCF model"

git add apple_dcf.ipynb
git commit -m "Add Apple valuation example"

# Push to GitHub
git push origin main
```

Part 8: Enhancements & Extensions (10 minutes)

1. Add Multiple Valuation Methods

```
class EnhancedDCFModel(DCFModel):
    """Extended DCF with additional methods"""

    def comparable_companies_check(self, comps_multiples: List[float]):
        """Compare DCF result to comps"""
        # Calculate implied EV/EBITDA from DCF
        # Compare to comparable companies
        pass

    def precedent_transactions_check(self, transaction_multiples:
List[float]):
        """Compare to precedent transactions"""
        pass

    def football_field(self):
        """Create football field chart with all methods"""
        pass
```

2. Monte Carlo Simulation

```
def monte_carlo_valuation(self, iterations: int = 10000):
    """Run Monte Carlo simulation on key assumptions"""
    # Randomly vary: growth rates, margins, WACC
    # Run DCF 10,000 times
    # Show distribution of outcomes
    # Return probability of positive return
    pass
```

3. Historical Analysis

```
def backtest_model(self, historical_data: pd.DataFrame):
    """Test model accuracy using historical data"""
    # Use past financials
    # Calculate what DCF would have predicted
    # Compare to actual stock performance
    pass
```

4. API Integration

```
def auto_populate_from_api(self, ticker: str):
    """Automatically populate assumptions from financial APIs"""
    import yfinance as yf
```

```
    stock = yf.Ticker(ticker)
    # Extract all needed data
    # Populate model automatically
    # Return ready-to-run model
    pass
```

✅ Skills Checklist

After this tutorial, you can:

- ☐ Understand DCF methodology and theory
- ☐ Build complete DCF model from scratch
- ☐ Use GitHub Copilot for financial modeling
- ☐ Calculate WACC using CAPM
- ☐ Project free cash flows
- ☐ Calculate terminal value
- ☐ Create sensitivity analysis
- ☐ Value real companies
- ☐ Export models to Excel
- ☐ Build interactive valuation notebooks

🎯 DCF Formula Quick Reference

```
# Free Cash Flow
FCF = NOPAT + D&A - CapEx - Δ NWC

# NOPAT
NOPAT = EBIT × (1 - Tax Rate)

# WACC
WACC = (E/V × Re) + (D/V × Rd × (1 - Tax Rate))

# Cost of Equity (CAPM)
Re = Rf + β × (Rm - Rf)

# Terminal Value (Perpetuity)
TV = FCF(n+1) / (WACC - g)

# Terminal Value (Multiple)
TV = EBITDA(n) × Exit Multiple

# Present Value
PV = FV / (1 + r)^n

# Enterprise Value
EV = Σ PV(FCF) + PV(TV)

# Equity Value
```

```
Equity Value = EV + Cash - Debt

# Share Price
Share Price = Equity Value / Shares Outstanding
```

💡 Pro Tips

1. Sanity Check Your Results

- WACC typically 7-12% for mature companies
- Terminal growth ≤ GDP growth (2-3%)
- FCF margins should be realistic
- Compare implied multiple to peers

2. Sensitivity is Critical

- Always show sensitivity tables
- Test multiple scenarios
- Document key assumptions
- Show range of outcomes

3. Use Real Data

- Start with yfinance for actuals
- Use analyst estimates for projections
- Research industry benchmarks
- Validate assumptions

4. Keep Learning

- Read equity research reports
- Study investment banking models
- Practice with different industries
- Build template library

🚀 What's Next?

You now have:

- ✅ Complete DCF model built in VS Code
- ✅ Professional valuation framework
- ✅ Interactive analysis capabilities
- ✅ Real company valuation skills

Next steps:

1. ✅ Value 5 companies to practice
2. ✅ Build industry-specific templates

3. ✅ Continue to: `Tutorials/05_VS_Code_Power_User.md`
4. ✅ Start Module 4: DCF Modeling

🎁 Bonus: Export to Investment Memo

```
def create_investment_memo(model, output_file='memo.md'):
    """Generate investment recommendation memo"""

    memo = f"""
# Investment Recommendation: {model.company_name}

## Executive Summary
Based on our DCF analysis, {model.company_name} has an implied equity
value of
${model.dcf_results['equity_value']:.0f}M, suggesting the stock is
[OVERVALUED/UNDERVALUED] by XX%.

## Valuation Methodology
- Discounted Cash Flow (DCF)
- 5-year projection period
- WACC: {model.wacc*100:.2f}%
- Terminal growth: 2.5%

## Key Assumptions
- Revenue CAGR: X%
- EBITDA margins expanding to X%
- Modest CapEx requirements

## Investment Thesis
[Your thesis here]

## Risks
- [Risk 1]
- [Risk 2]

## Recommendation
[BUY/HOLD/SELL]

## Price Target
$XX.XX per share

---
*Analysis Date: {datetime.now().strftime('%Y-%m-%d')}*
    """

    with open(output_file, 'w') as f:
        f.write(memo)
```

🎉 **You're now a DCF modeling expert in VS Code!**

Next: `Tutorials/05_VS_Code_Power_User.md` - Advanced VS Code techniques

Estimated completion time: 120 minutes Difficulty: Intermediate-Advanced Next: Power User Techniques