# Lesson 01: Python Setup

## Data Science with Python – BSc Course

Data Science Program

45 Minutes

# Learning Objectives

**After this lesson, you will be able to:**

- Install Anaconda and launch Jupyter Notebook
- Create and execute Python code cells
- Understand and use basic data types (int, float, str, bool)
- Store stock prices and financial data in variables

**Finance Application:** Store and manipulate stock prices using Python variables.

**Foundation lesson – everything builds on these basics**

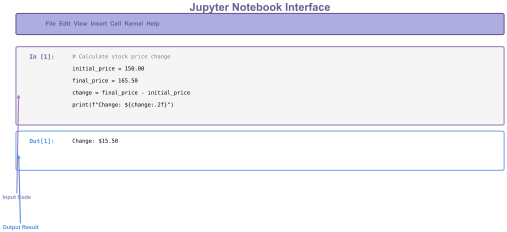
# The Jupyter Notebook Environment

**Why Jupyter?**

- Interactive code execution
- Mix code, output, and documentation
- Industry standard for data science
- Perfect for financial analysis

**Getting Started:**

1. Install Anaconda from anaconda.com
2. Launch Jupyter Notebook
3. Create New → Python 3



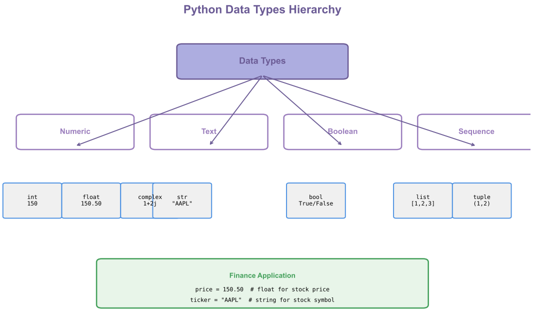
Jupyter Notebook Interface

```
# Calculate stock price change
initial_price = 150.00
final_price = 165.50
change = final_price - initial_price
print(f"Change: ${change:.2f}")
```

```
Change: $15.50
```

**Jupyter = Julia + Python + R – supports multiple languages**

# Python Data Types



```
price = 150.50  # float for stock price
ticker = "AAPL"  # string for stock symbol
```

**Four Basic Types:**

`int` – Integers (whole numbers)

```
price_shares = 100
```

`float` – Decimals

```
stock_price = 185.50
```

`str` – Text strings

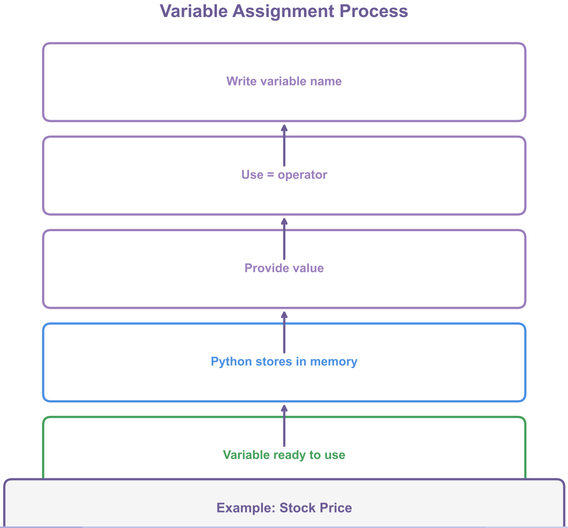
```
ticker = "AAPL"
```

`bool` – True/False

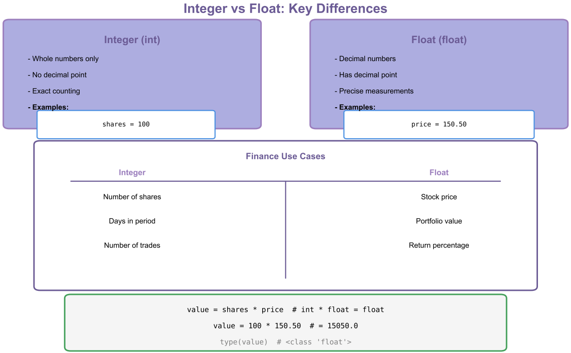
```
is_profitable = True
```

**Use type(variable) to check any variable's type**

# Variable Assignment



# Integers vs Floats in Finance



### Integer vs Float: Key Differences

**Integer (int)**

- Whole numbers only
- No decimal point
- Exact counting
- **Examples:**

```
shares = 100
```

**Float (float)**

- Decimal numbers
- Has decimal point
- Precise measurements
- **Examples:**

```
price = 150.50
```

**Finance Use Cases**

| Integer | Float |
|---|---|
| Number of shares | Stock price |
| Days in period | Portfolio value |
| Number of trades | Return percentage |

```
value = shares * price  # int * float = float
value = 100 * 150.50  # = 15050.0
type(value)  # <class 'float'>
```

## When to use integers:

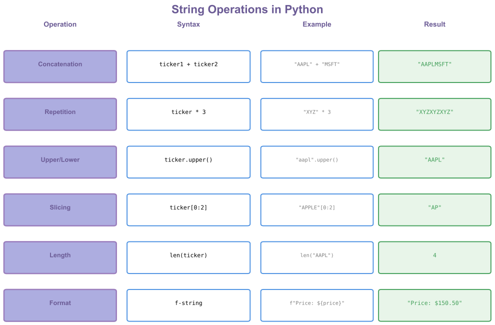
- Number of shares: `shares = 100`
- Trading days: `days = 252`
- Position count: `positions = 5`

## When to use floats:

- Stock prices: `price = 185.50`
- Returns: `ret = 0.0523`
- Percentages: `pct = 5.23`

**Division always returns float: 10 / 3 = 3.333...**

# String Operations for Finance



String Operations in Python

| Operation | Syntax | Example | Result |
|---|---|---|---|
| Concatenation | ticker1 + ticker2 | "AAPL" + "MSFT" | "AAPLMSFT" |
| Repetition | ticker * 3 | "XYZ" * 3 | "XYZXYZXYZ" |
| Upper/Lower | ticker.upper() | "aapl".upper() | "AAPL" |
| Slicing | ticker[0:2] | "APPLE"[0:2] | "AP" |
| Length | len(ticker) | len("AAPL") | 4 |
| Format | f-string | f"Price: ${price}" | "Price: $150.50" |

## Common String Operations:

```
ticker = "AAPL"
ticker.upper() → "AAPL"
ticker.lower() → "aapl"
```

## String Formatting:

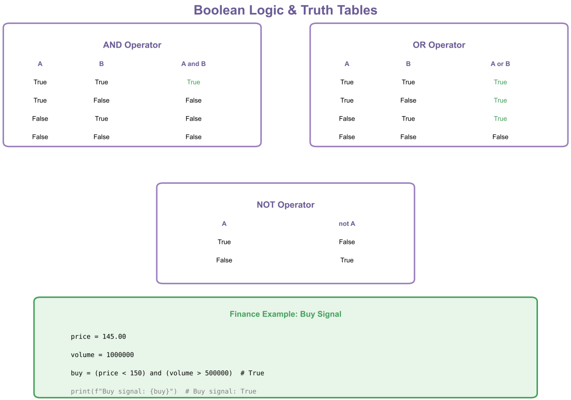
```
f"Price:  ${price}"
```

## Concatenation:

```
"NASDAQ:" + ticker
```

**F-strings (f"...") are the modern way to format strings**

# Boolean Logic for Trading Decisions



### Boolean Logic & Truth Tables

AND Operator

| A | B | A and B |
|---|---|---|
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

OR Operator

| A | B | A or B |
|---|---|---|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

NOT Operator

| A | not A |
|---|---|
| True | False |
| False | True |

Finance Example: Buy Signal

```
price = 145.00
volume = 1000000
buy = (price < 150) and (volume > 500000)  # True
print(f"Buy signal: {buy}")  # Buy signal: True
```

**Comparison Operators:**

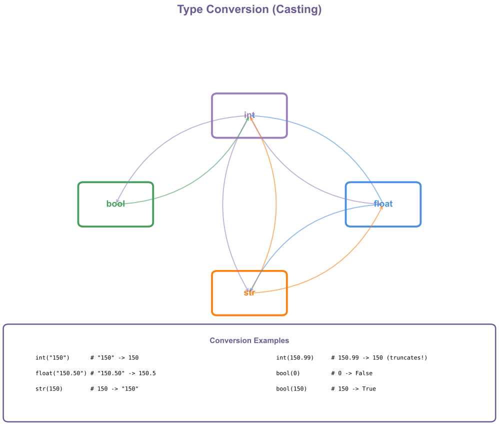
- > greater than
- < less than
- >= greater or equal
- == equal to
- != not equal

**Example:**

`price > 200` → `True/False`

**Booleans are essential for trading rule logic**

# Type Conversion



```
int("150")      # "150" -> 150                int(150.99)     # 150.99 -> 150 (truncates!)
float("150.50") # "150.50" -> 150.5           bool(0)         # 0 -> False
str(150)        # 150 -> "150"                bool(150)       # 150 -> True
```

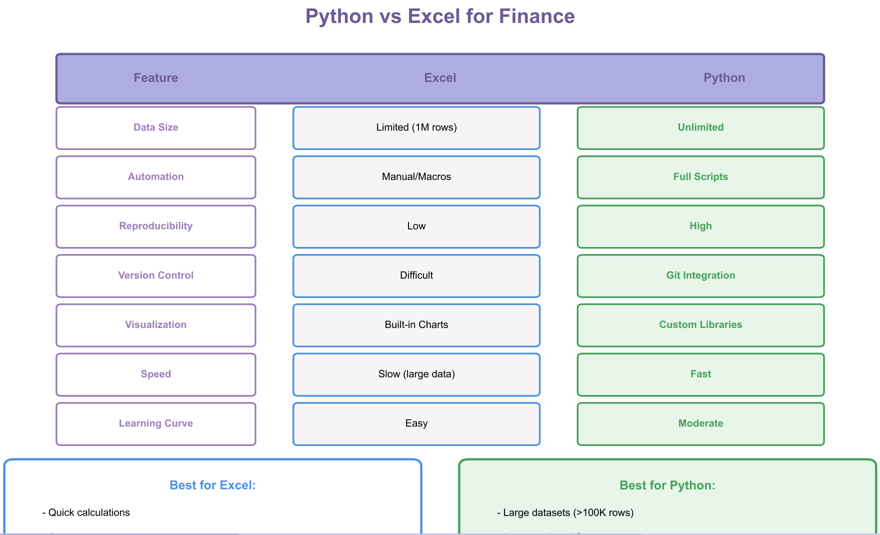
**Converting Between Types:**

`int("100")` → `100`
`float("185.5")` → `185.5`
`str(185.5)` → `"185.5"`
`bool(1)` → `True`

**Finance Use Case:**
Reading prices from CSV files
(data comes as strings)

**Be careful: int(”185.5”) fails – convert to float first**

# Why Python Instead of Excel?



## Python vs Excel for Finance

| Feature | Excel | Python |
|---|---|---|
| Data Size | Limited (1M rows) | Unlimited |
| Automation | Manual/Macros | Full Scripts |
| Reproducibility | Low | High |
| Version Control | Difficult | Git Integration |
| Visualization | Built-in Charts | Custom Libraries |
| Speed | Slow (large data) | Fast |
| Learning Curve | Easy | Moderate |

**Best for Excel:**

- Quick calculations

**Best for Python:**

- Large datasets (>100K rows)

# Hands-on Exercise (25 min)

**Create a Jupyter notebook and complete:**

1. Create variables for a stock portfolio:
   - `ticker = "AAPL"` (string)
   - `shares = 50` (integer)
   - `buy_price = 150.25` (float)
   - `current_price = 185.50` (float)
2. Calculate portfolio metrics:
   - Total investment: `shares * buy_price`
   - Current value: `shares * current_price`
   - Profit: current value - investment
   - Return %: `(profit / investment) * 100`
3. Create a boolean: `is_profitable = profit > 0`
4. Print results using f-strings

---

**Save your notebook – we'll build on this next lesson**

# Lesson Summary

**Key Takeaways:**

- Jupyter Notebook is our development environment
- Four basic types: `int`, `float`, `str`, `bool`
- Variables store values with descriptive names
- Type conversion needed when reading external data
- Python handles large datasets better than Excel

**Next Lesson:** Data Structures (Lists and Dictionaries)

**Practice: Experiment with different variable types in Jupyter**