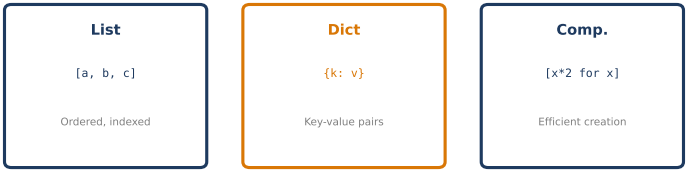
Lesson 02 Summary: Data Structures

Data Science with Python – Key Concepts

Data Science Program

Data Structures Overview

Data Structures Summary



Key Operations:

`list[0]` `list[1:3]` | `dict["key"]` | `[x for x in items]`

Finance: Store prices in lists, portfolios in dicts

Three fundamental structures: lists, dictionaries, comprehensions

Lists – Key Concepts

Ordered sequences accessed by position

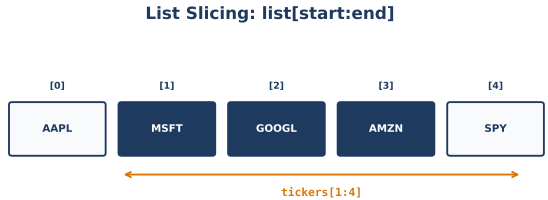
- **Creation:** `prices = [150.0, 165.5, 148.25]`
- **Positive index:** `prices[0]` → first element
- **Negative index:** `prices[-1]` → last element
- **Methods:** `append()`, `insert()`, `pop()`, `sort()`

Finance use: Store daily prices, transaction history, ticker lists

---

Python indexing starts at 0 – remember this!

List Slicing



Syntax: list[start:end:step] – end is EXCLUSIVE

Dictionaries – Key Concepts

Key-value pairs for fast lookups

- **Creation:** `portfolio = {"AAPL": 150.50, "MSFT": 340.00}`
- **Access:** `portfolio["AAPL"]` → `150.50`
- **Update:** `portfolio["AAPL"] = 155.00`
- **Check:** `"AAPL" in portfolio` → `True`

Methods: `.keys()`, `.values()`, `.items()`

O(1) lookup time – much faster than searching a list

Nested Structures

Complex data with dictionaries inside dictionaries

```
portfolio = {
  "AAPL":  {"shares":  100, "price":  150.50},
  "MSFT":  {"shares":  50, "price":  340.00}
}
```

Accessing nested data:

- `portfolio["AAPL"]["price"]` → `150.50`
- `portfolio["MSFT"]["shares"]` → `50`

Nested structures model real financial data elegantly

List Comprehensions

Concise syntax: `[expression for item in iterable if condition]`

Examples:

- **Double prices:** `[p * 2 for p in prices]`
- **Filter:** `[p for p in prices if p > 150]`
- **Transform:** `[t.upper() for t in tickers]`
- **Calculate:** `[shares[t] * prices[t] for t in tickers]`

Advantages: More readable, often faster, Pythonic style

Comprehensions replace loops with a single, expressive line

List vs Dictionary

Choose based on how you need to access the data

Finance Applications

Real-world data structure patterns:

- **Price history:** List of closing prices over time
- **Portfolio:** Dict mapping ticker $\rightarrow$ holdings
- **Multi-asset:** Nested dict with price, shares, sector
- **Filtering:** Comprehension to find stocks above threshold

Portfolio Value Calculation:

```
total = sum([shares[t] * prices[t] for t in tickers])
```

Data structures + comprehensions = efficient financial analysis

Quick Reference

Essential Syntax:

List	Dictionary
`prices = [150, 165, 148]`	`port = {"AAPL": 150}`
`prices[0]` → first	`port["AAPL"]` → value
`prices[-1]` → last	`port["NEW"] = 100` add
`prices[1:3]` → slice	`"AAPL" in port` check
`prices.append(175)`	`port.keys()` all keys

Comprehension Pattern:

```
[f(x) for x in items if condition]
```

Master these patterns for efficient Python programming