Python:
Data Structures and Functions

Day 2

FinTech

Lesson 2.2

Class Objectives

By the end of today's class, you will be able to:

01 Identify homogeneous and heterogeneous data.

02 Access and manipulate data within list and dict objects.

03 Iterate over lists and dicts.

04 Visualize and iterate over nested lists and dicts.

05 Define and call custom functions.

Class Refresher

Class Refresher

In the previous class, we learned how to...

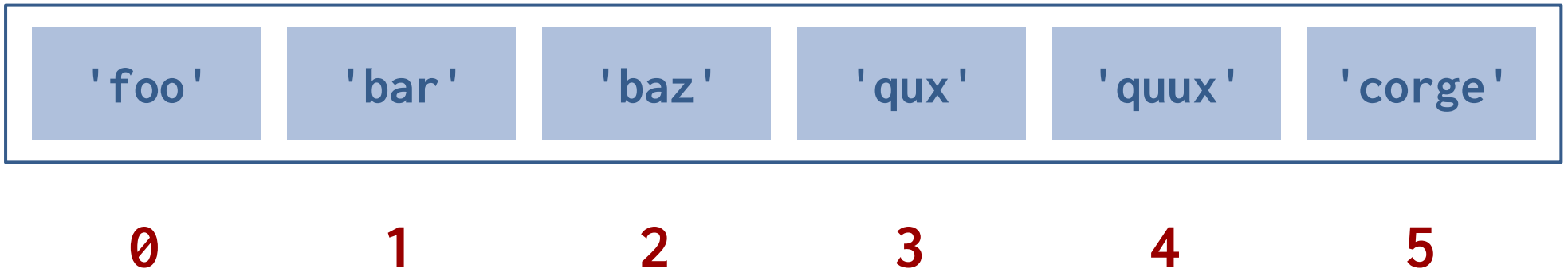
- Define Python and how it's used.
- Install JupyterLab.
- Use the terminal.
- Create and run Python files in JupyterLab (IDE).
- Use variables.
- Use conditionals.
- Use for loops.
- Pseudocode solutions.

Lists

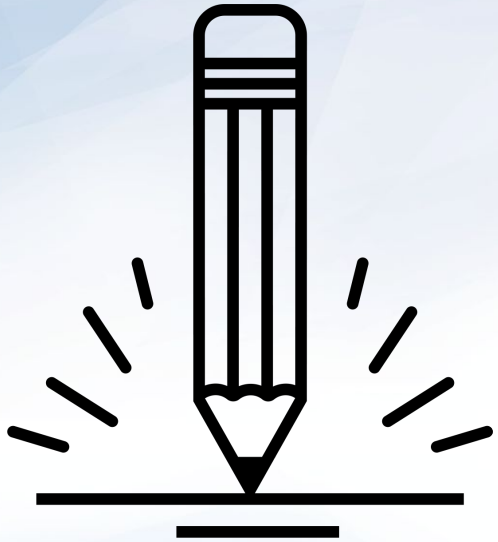
Lists

A **list** is a data structure with the following characteristics:

- It is a collection of elements.
- It is ordered.
- It is heterogenous (can hold different data types).
- It has a zero-based index.



Lists can hold different data types, **but they commonly hold elements of a single data type to represent a conceptual category or group**, e.g., a list of cars or a group of people.



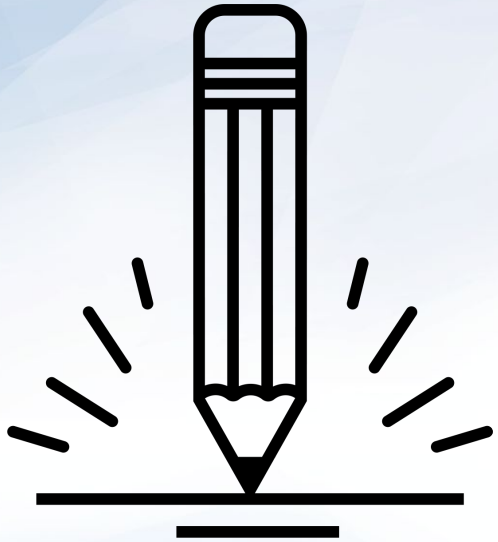
Activity: Sugar, Flour, Butter!

In this activity, you will work with lists to maintain a grocery list. You will create lists, append to lists, retrieve items from a list, and retrieve values by indexes.

(Instructions sent via Slack)

Suggested Time:
15 Minutes

Time's Up! Let's Review.



Activity: Trading Log

In this activity, you will use lists to maintain a grocery list. You will create a trading log that tracks profits and losses for each market day of the month. Then, you'll iterate over the list to calculate the highest and lowest profit and loss days.

(Instructions sent via Slack)

Suggested Time:
15 Minutes

Time's Up! Let's Review.

Dicts

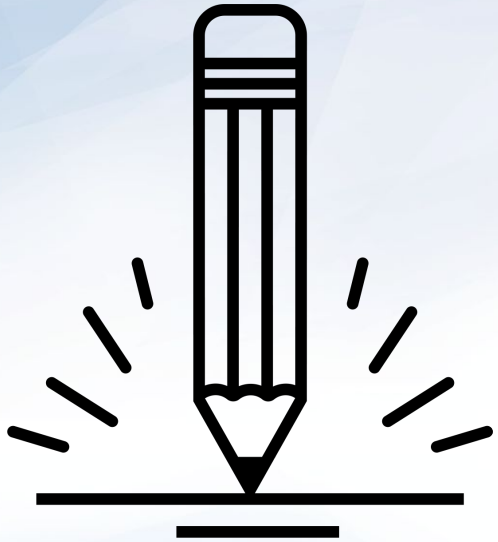
Dicts

Dicts are mutable (changeable), unordered data structures with the following key characteristics:

A collection of key-value pairs

Unordered

Heterogeneous (can contain different data types)



Activity: Market Capitalization

In this activity, you will create a dictionary, and then update, remove, and extract values from the dictionary.

(Instructions sent via Slack)

Suggested Time:
15 Minutes

Time's Up! Let's Review.

AdobeStock_56885471

Nested Lists and Dicts

Nested Lists and Dicts

A **nested** object is an iterable object that contains one or more iterable objects, thereby having “nested” levels of iteration.

Types of nested lists and dicts:

List of lists

List of dicts

Dictionary of lists

Dictionary of dicts

Nested Lists and Dicts

List of lists:

```
# List
ceo_list = ["Warren", "Jack", "Harry"]

# List of Lists
ceo_nested_list = [
    ["Warren Buffet", 88, "CEO of Berkshire Hathaway"],
    ["Jeff Bezos", 55, "CEO of Amazon"],
    ["Harry Markowitz", 91, "Professor of Finance"]
]

# Retrieve first entry of ceo_nested_list
first_entry = ceo_nested_list[0]

# Retrieve name of first entry
first_entry_name = ceo_nested_list[0][0]

# Retrieve age of first entry
first_entry_age = ceo_nested_list[0][1]

# Retrieve occupation of first entry
first_entry_occupation = ceo_nested_list[0][2]

# Print results to screen
print("The first entry in employees_nested_list is:", first_entry)
print(f"{first_entry_name} is {first_entry_age} years old, serving as {first_entry_occupation}.")
```

Nested Lists and Dicts

Dict of lists:

```
# Dictionary of Lists
stocks_nested_list = {
    "APPL": ["Apple", 101.32, "NASDAQ", 937.7],
    "MU": ["Micron Technology", 32.12, "NASDAQ", 48.03],
    "AMD": ["Advanced Micro Devices", 23.12, "NASDAQ", 29.94],
    "TWTR": ["Twitter", 34.40, "NASDAQ", 26.42]
}

# Retrieve entry for APPL
appl_entry = stocks_nested_list["APPL"]

# Retrieve name, stock_price, and exchange for APPL entry
appl_name = stocks_nested_list["APPL"][0]
appl_stock_price = stocks_nested_list["APPL"][1]
appl_exchange = stocks_nested_list["APPL"][2]

# Print results to screen
print(f"APPL ticker stands for {appl_name}. APPL stock price is currently {appl_stock_price}, and
it is available on {appl_exchange}.")
```

Nested Lists and Dicts

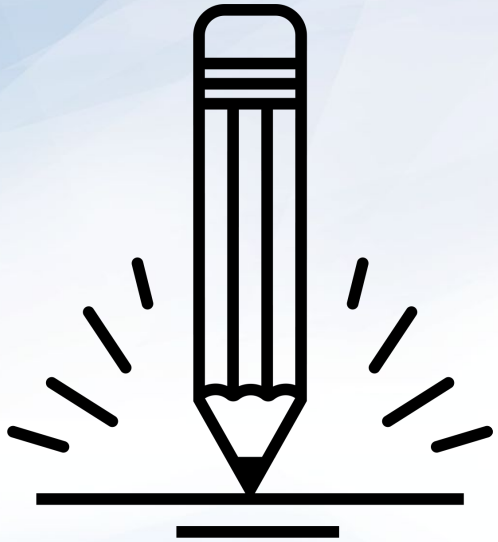
Dict of dicts:

```
# Dictionary of Dicts
stocks_nested_dict = {
    "APPL": {
        "name": "Apple",
        "exchange": "NASDAQ",
        "market_cap": 937.7
    },
    "MU": {
        "name": "Micron Technology",
        "exchange": "NASDAQ",
        "market_cap": 48.03
    },
    "AMD": {
        "name": "Advanced Micro Devices",
        "exchange": "NASDAQ",
        "market_cap": 29.94
    },
    "TWTR": {
        "name": "Twitter",
        "exchange": "NASDAQ",
        "market_cap": 26.42
    }
}

# Retrieve Twitter entry
twitter_entry = stocks_nested_dict["TWTR"]

# Retrieve TWTR name, exchange, and market_cap
twitter_name = stocks_nested_dict["TWTR"]["name"]
twitter_exchange = stocks_nested_dict["TWTR"]["exchange"]
twitter_market_cap = stocks_nested_dict["TWTR"]["market_cap"]

# Print results to screen
print(f"Name of TWTR ticker is {twitter_name}. TWTR is available on {twitter_exchange}, and it currently has a market capitalization of
{twitter_market_cap}.")
```



Activity: Weekly Gains

In this activity, you will work with nested data structures in a Python file. You will store daily stock data in a list and then store that list in a dictionary. The key of the dictionary will be the stock tickers. You will then retrieve stock data from the dictionary for specific days.

(Instructions sent via Slack)

Suggested Time:
20 Minutes

Time's Up! Let's Review.

Functions

Functions

A **function** is a block of reusable code that can be used to perform an action.

A function provides better:

Organization

- Functions make code more readable because they wrap blocks of operational code into a single, callable name.

Modularity

- Functions can be called multiple times and used over and over again.
- Functions take in optional inputs and generate output. Function inputs are not bound to a specific variable, but rather a specific data type.

Comprehension

- Functions are often annotated with docstrings, comments that help users understand the specific purpose of a defined function.
- Function names are often good indicators of what the function will try to achieve.

Functions

Calling a function:

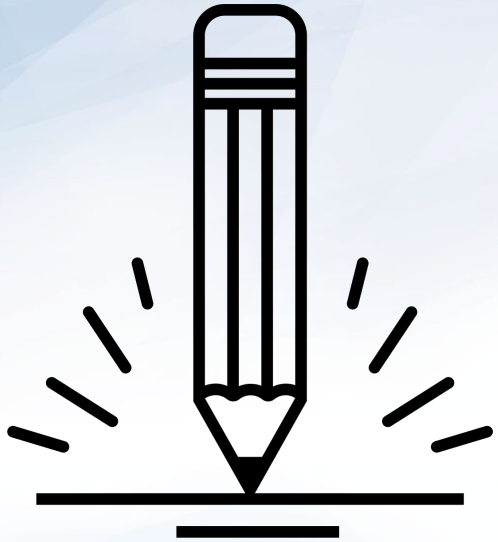
```
def calculate_market_cap(market_price, number_of_shares):
    cap = market_price * number_of_shares

    return cap

market_price = 76.06
number_of_shares = 1243600000

market_cap = calculate_market_cap(market_price, number_of_shares)
print(f"Market Capitalization: {market_cap}")
```

```
Market Capitalization: 94588216000.0
```



Activity: Finally Functioning

In this activity, you will define a function to calculate compound annual growth rate (CAGR) for an investment portfolio.

(Instructions sent via Slack)

Suggested Time:
20 Minutes

Time's Up! Let's Review.

You Made It!

Congratulations, you've completed Day 2 of Python! Let's recap what you learned:

- Lists
- Iterating lists
- Dictionaries
- Nested data structures
- Functions