Data structures example

We will do a large example reviewing data structure concepts.

Represent a customer's financial situation

Suppose you are at a bank, and you want to analyze each customer's financial situation to offer targeted investment guidance. How do you represent the current financial situation?

What are the things we need to consider?

- Bank accounts: customers could have any number of bank accounts, or no bank accounts too
- Real estate:
  - Current value of property or multiple properties
  - Mortgages on those properties
- Retirement funds and IRA
- Stocks
- Foreign currency
- Credit cards

How do we organize all this?

Companies have a similar problem when they report their quarterly earnings. A nice organizational principle is to think hierarchically.

At the top, what we have are **Assets** and **Liabilities**.

Assets:

- Bank accounts

- Real estate properties
- Retirement funds
- Stocks
- Forex

Liabilities:

- Mortgages
- Credit card debts

How should we represent the fact that the financial situation consists of Assets and Liabilities?

Tuple of exactly two items: **(Assets, Liabilities)**

Assets:

- Bank accounts
- Real estate properties
- Retirement funds
- Stocks
- Forex

How do we represent assets?

Tuple of (banks, real-estate, retirement-funds, stocks, forex)

How do we represent bank accounts?

- We can have any number of accounts...
- The names of the banks are unimportant; we just want to be able to add up the total balance.

banks = **List** of account balances

- `banks = []` if no bank accounts

Real-estate and retirement funds can be considered similarly, and represented as lists...

Assets:

- Bank accounts
- Real estate properties
- Retirement funds
- Stocks
- Forex

How do we represent stocks?

- I want to be able to calculate the value of a customer's stock portfolio at any time.

Will a list work?

Better:

- a **dictionary**
- key = stock ticker symbol
- value = number of shares

At any time, we can look up the current stock prices to calculate the value of a customer's stock holdings.

Assets:

- Bank accounts
- Real estate properties
- Retirement funds
- Stocks
- Forex

How do we represent foreign currencies?

This may look like:

- 500 of EUR
- 10,000 of JPY
- 50,000 of INR
- ...

Again best represented as a **dictionary**

- key = currency symbol
- value = amount

The current value of all forex assets can be calculated using the current rates of USD:EUR, USD:JPY, ...

Liabilities:

- Mortgages
- Credit card debts

How do we represent these liabilities?

Tuple of (mortgages, credit-cards)

Lists should work for both.

To summarize:

Each customer represented as **(Assets, Liabilities)**.

Assets = **tuple** of (banks, real_estate, retirement, stocks, forex)

- banks, real_estate, and retirement are **lists**
- stocks and forex are **dictionaries**

Liabilities = **tuple** of (mortgages, credit_cards)

- both mortgages and credit_cards are lists

In [1]:
```
banks = [10000, 20000]              # 2 bank accounts
real_estates = [50000, 80000]       # 2 properties
retirements = [100000]              # one retirement account
stocks = {'AAPL': 50, 'SPY': 500}   # Apple and S&P 500 held
forex = {'INR': 50000}              # Some Indian Rupees too

assets = (banks, real_estates, retirements, stocks, forex)
print('Assets =', assets)
```

```
Assets = ([10000, 20000], [50000, 80000], [100000], {'AAP
L': 50, 'SPY': 500}, {'INR': 50000})
```

In [2]:
```
mortgages = []                      # no current or previous mortgages
credit_cards = [2350, 4000]         # two credit cards

liabilities = (mortgages, credit_cards)
print('Liabilities =', liabilities)
```

```
Liabilities = ([], [2350, 4000])
```

In [3]:
```
customer_record = (assets, liabilities)
print('Overall:', customer_record)
```

```
Overall: (([10000, 20000], [50000, 80000], [100000], {'AAP
L': 50, 'SPY': 500}, {'INR': 50000}), ([], [2350, 4000]))
```

What is the current value of a customer's stock portfolio?

We will need:

- the customer's record, and
- current prices of all stocks.

How do we represent the latter?

As a dictionary:

- AAPL is 110.12
- GOOG is 496.04

- FB is 76.46
- YHOO is 48.33
- SPY is 201.84
- ...

What is the current value of a customer's stock portfolio?

Step 1: A function to get stocks from the customer record

In [4]:
```
def get_customer_stocks(customer_record):
    """Return the stocks part of the customer record."""

    # customer_record = (assets, liabilities)
    assets = customer_record[0]

    # assets = (banks, real_estates, retirements, stocks, forex)
    stocks = assets[3]

    return stocks
```

Step 2: A function to compute the value of shares of one stock held by the customer

In [5]:
```
def get_value_of_one_stock(stock_ticker, num_shares, current_stock_pric
    """How much is num_shares shares of ticker worth
       at current prices?"""

    if stock_ticker in current_stock_prices:
        value_per_share = current_stock_prices[stock_ticker]
    else:
        print('ERROR: Stock ticker', stock_ticker, 'not recognized')
        return -1  # Use -1 to represent error signal (do you foresee c

    this_stock_value = num_shares * value_per_share

    return this_stock_value
```

Step 3: A glue function that computes the total value of all stocks

In [6]:
```
def stock_value(customer_record, current_stock_prices):
    """Compute the customer's stock portfolio value given
       the current stock prices."""
```

```
    stocks = get_customer_stocks(customer_record)

    # Initialize portfolio value
    portfolio_value = 0.0

    # For each stock in the portfolio, add it to the total value
    for stock_ticker, num_shares in stocks.items():
        this_stock_value = get_value_of_one_stock(stock_ticker,
                                                  num_shares,
                                                  current_stock_prices)

        if this_stock_value < 0:   # Error code!
            return -1              # This function also returns Error

        # Add this to the portfolio
        portfolio_value += this_stock_value

        print(num_shares, 'shares of', stock_ticker, 'worth', this_stoc

    print('Overall portfolio value =', portfolio_value)
    return portfolio_value
```

What is the current value of a customer's stock portfolio?

Let's try this function

In [7]:
```
# Create the dictionary of the current prices
current_stock_prices = {'AAPL':110.12, 'GOOG':496.04, 'FB':76.46, 'YHOO

# Call the function
stock_value(customer_record, current_stock_prices)
```

```
50 shares of AAPL worth 5506.0
500 shares of SPY worth 100920.0
Overall portfolio value = 106426.0
```

Out[7]:
```
106426.0
```

Let's create another customer who's similar, but with different stocks.

How do we do this?

- we need a copy that we we can change
- we need a **deep** copy

In [8]:
```
import copy

# create a deep copy of this customer
customer_record_2 = copy.deepcopy(customer_record)

# Update his stock portfolio
# Access the assets (index 0), the stocks (index 3), and change it
customer_record_2[0][3]['SNE'] = 50  # 50 shares of SONY

# Call the function
stock_value(customer_record_2, current_stock_prices)
```

```
50 shares of AAPL worth 5506.0
500 shares of SPY worth 100920.0
ERROR: Stock ticker SNE not recognized
```

Out[8]:
```
-1
```

What is the customer's overall real-estate value?

We need to consider

- all real-estate properties owned under assets,
- minus all mortgages under liabilities.

In [9]:
```
def real_estate_value(customer_record):
    """Compute a customer's overall real-estate value."""

    # We need to look at both assets and liabilities
    assets, liabilities = customer_record

    # assets = (banks, real_estates, retirements, stocks, forex)
    real_estate_assets = assets[1]

    print('Real estate assets =', real_estate_assets)

    # This is a list; compute total real-estate asset value
    # Python provides a function for this: sum(real_estate_assets),
    # but let's do it directly.
    asset_value = 0
    for value in real_estate_assets:
        asset_value += value
```

```
    print('Asset value =', asset_value)

    # liabilities = (mortgages, credit_cards)
    real_estate_liabilities = liabilities[0]

    print('Real estate liabilities =', real_estate_liabilities)

    # This is again a list
    liability_value = 0
    for value in real_estate_liabilities:
        liability_value += value

    print('Liability value =', liability_value)

    overall_value = asset_value - liability_value
    print('Overall value =', overall_value)

    # Return everything
    return asset_value, liability_value, overall_value
```

In [10]:
```
real_estate_value(customer_record)
```

```
Real estate assets = [50000, 80000]
Asset value = 130000
Real estate liabilities = []
Liability value = 0
Overall value = 130000
```

Out[10]:
```
(130000, 0, 130000)
```

What would a customer's record be if he/she pays off as much credit card debt from her bank accounts?

How would we do this?

- Compute total credit card amount and bank balance
- Figure out how much money we must transfer from banks to credit cards:
  - **bank balance >= debt**: zero out the debt, or
  - **debt > bank balance**: zero out the bank accounts

    - In both cases, we need to reduce amounts from a list.

- Write a function that reduces amounts from a list

- Put it all together using a "glue" function

Step 1: Compute total credit card and bank balance

In [11]:
```
def compute_bank_balance_and_debt(customer_record):

    # assets = (banks, real_estates, retirements, stocks, forex)
    banks = customer_record[0][0]
    total_bank_balance = sum(banks) # sum() is a special function that

    # liabilities = (mortgages, credit_cards)
    credit_cards = customer_record[1][1]
    total_credit_card_debt = sum(credit_cards)

    # Return a tuple
    return total_bank_balance, total_credit_card_debt

# Test
compute_bank_balance_and_debt(customer_record)
```

Out[11]:
```
(30000, 6350)
```

Step 2: What is the amount we can transfer from banks to credit cards?

In [12]:
```
def amount_to_transfer(total_bank_balance, total_credit_card_debt):

    # The amount we can transfer is the minimum of the two
    # We can use the built-in min() function, but let's do it the hard

    if total_bank_balance >= total_credit_card_debt:
        return total_credit_card_debt
    else:
        return total_bank_balance

# Test
total_bank, total_credit = compute_bank_balance_and_debt(customer_recor
print('Total bank =', total_bank)
print('Total credit =', total_credit)
print('Amount to transfer =', amount_to_transfer(total_bank, total_crec
```

```
Total bank = 30000
Total credit = 6350
Amount to transfer = 6350
```

Step 3: Define a function that reduces a given amount from a list (whether banks or credit cards).

In [13]:

```
def reduce_list_amounts(either_list, amount):
    """either_list is either the bank balance list,
       or the credit card list. Reduce the given amount
       from this list."""

    amount_still_left = amount

    while amount_still_left > 0:
        # Work from the end of the list
        last_item = either_list[-1]

        if last_item <= amount_still_left:
            # We can remove this item from the list
            either_list.pop()
            amount_still_left -= last_item

        else:
            # We need to only remove a partial amount from the last ite
            either_list[-1] -= amount_still_left
            amount_still_left = 0

# Test
amount_list = [50, 40, 10, 20, 30]
print('List before update =', amount_list)
reduce_list_amounts(amount_list, 55)
print('List after update =', amount_list)
reduce_list_amounts(amount_list, 95)
print('List after second update =', amount_list)
```

```
List before update = [50, 40, 10, 20, 30]
List after update = [50, 40, 5]
List after second update = []
```

Step 4: The "glue" function that calls all the previous functions.

In [14]:

```
def pay_off_credit_cards(customer_record):
    """Pay off as much credit card debt as possible
       from bank accounts."""

    total_bank, total_credit = compute_bank_balance_and_debt(customer_r
```

```
    to_transfer = amount_to_transfer(total_bank, total_credit)

    # Create a new record
    resulting_customer_record = copy.deepcopy(customer_record)

    # Reduce the transferred amount from the bank balances
    banks = resulting_customer_record[0][0]
    reduce_list_amounts(banks, to_transfer)

    # Reduce the transferred amounts from the credit cards
    credit_cards = resulting_customer_record[1][1]
    reduce_list_amounts(credit_cards, to_transfer)

    return resulting_customer_record

print('customer_record: banks =', customer_record[0][0], 'credit cards
resulting_customer_record = pay_off_credit_cards(customer_record)
print('After paying off credit cards:', resulting_customer_record[0][0]
print('credit cards =', resulting_customer_record[1][1])
```

```
customer_record: banks = [10000, 20000] credit cards = [235
0, 4000]
After paying off credit cards: [10000, 13650]
credit cards = []
```

What will the customer's real-estate portfolio look like in 5 years?

We'll assume:

- Each year, property values either increase by 10% or remain the same, both being equally likely
- Mortgage interest grows at 5% each year

Simulate the change in value of the portfolio over time a 10,000 times, and return:

- the average value after 10 years, and
- the *median* value (half the simulations are lower than the median, and the remaining half higher)

How do we do this?

- Calculate the total property value and total mortgage now.
- Define a function for one simulation:
  - For 10 years:
  - Grow property values by either 10% or 0% (using Python's random number generator)
  - Grow mortgage by 5%
- Run the simulation 10,000 times, and return the average and the median.

Step 1: Calculate starting property value and mortgage.

We already did this earlier...

In [15]:
```
# Create a new customer record
customer_record_3 = copy.deepcopy(customer_record)

# Let's say this customer has just bought properties.
# Set mortgage = property_value at starting point.
mortgage_list = customer_record_3[1][0]
mortgage_list.append(50000)
mortgage_list.append(80000)

# See the real estate record
properties, mortgages, overall = real_estate_value(customer_record_3)
```

```
Real estate assets = [50000, 80000]
Asset value = 130000
Real estate liabilities = [50000, 80000]
Liability value = 130000
Overall value = 0
```

Step 2: Run a simulation.

In [16]:
```
import random
def simulate(property_value, mortgage):
    """Simulate property value either increasing by 10% or
    staying the same each year, and mortgage increasing
    by 4% per year. Do this for 10 years."""

    for year in range(10):
        # random.random() means the random() function
```

```
        # of the package named random
        # This gives a random number between 0 and 1
        if random.random() <= 0.5:
            growth = 1.1  # 10% increase
        else:
            growth = 1.0  # no change
        property_value *= growth

        mortgage *= 1.05  # always 5% increase

    return property_value, mortgage

# Test; different simulations can give different results for property_v
print(simulate(1000, 1000))
print(simulate(1000, 1000))
print(simulate(1000, 1000))
```

```
(1948.7171000000005, 1628.8946267774422)
(1948.7171000000005, 1628.8946267774422)
(1331.0, 1628.8946267774422)
```

Step 3: Run the simulation 10,000 times, and find the average and median.

In [17]:

```
def run_multiple_simulations(property_value, mortgage, num_simulations)
    """Run num_simulations of the growth of property_values
    and mortgages."""

    overall_values = []
    for i in range(num_simulations):
        property_after_10, mortgage_after_10 = simulate(property_value,
        overall_after_10 = property_after_10 - mortgage_after_10
        overall_values.append(overall_after_10)

    # Now print statistics about overall_values
    print
    print('Average =', sum(overall_values) * 1.0 / num_simulations)
    overall_values.sort()
    print('Median =', overall_values[int(num_simulations / 2)])

# Test
run_multiple_simulations(property_value=1000, mortgage=1000, num_simula
```

```
Average = -5.577182451772244
Median = -18.38462677744201
```

In [18]:

```
property_value_at_start, mortgage_at_start, overall_at_start = real_est
```

```
run_multiple_simulations(property_value_at_start, mortgage_at_start, 10
```

```
Real estate assets = [50000, 80000]
Asset value = 130000
Real estate liabilities = [50000, 80000]
Liability value = 130000
Overall value = 0
Average = 130.67181827682447
Median = -2390.0014810673892
```

Note that even though the "average" property value increase was 5%, the same as the mortgage increase, we still tend to end up with a loss.

Can you guess the reason for this?