Discussion 4: Multiple Classes, Dictionaries, and Tuples

Reminders

- Submit your work to **Canvas Assignment / Discussion 4**
  - Commit **at least 4 times** and push to GitHub
  - Submit the **repository link** to Canvas by the end of discussion
- **Homework 4** due this **Friday, February 6th @11:59pm**
- **Spaced Practice Tool** (5 questions max/day)

Please remember:

- Commit **<u>at least 4 times</u>** to get full credit on all HW assignments and projects.
- **Your file has to run for us to grade it** – please double-check that the final version you hand in runs successfully, including all tests and any necessary interactivity in main().
- **Use meaningful commit messages!**
  - i.e. “Completed __init__ method”

Multiple Classes

- Each class has a single responsibility
- Code becomes:
  - Easier to understand
  - Easier to change
  - Easier to reuse

Example:

- Item: represents what is being sold
- Warehouse: manages many items
- Order: represents what a customer wants

Association vs. Inheritance

Association (“has a”)

- One class *uses* another class

Examples:

- A deck has many card objects
- A warehouse has many items

Inheritance (“is a”)

- One class *extends* another class

Examples:

- A dog is an animal
- An employee is a person

Dictionaries

- Store key and value pairs
- Use “square-bracket” notation to put a value into a dictionary:

  mydict[“my_key”] = my_value
- Can initialize with a dictionary literal:

  mydict = {“my_key”: “my_value”}

Example:

```
{"Water": 3, "CocaCola": 2}
```

Dictionaries

Dictionary

```
dog_dic = {
    'name': 'tofu',
    'color': 'white',
    'age': '2',
}
```

Access: `print(dog_dic['name']) # 'tofu'`

Change: `dog_dic['age'] = int(dog_dic['age']) # convert value types`

Add: `dog_dic['breed'] = 'husky' # add a new key-value pair`

Delete: `del dog_dic['color'] # delete the key 'color' and value 'white'`

Dictionaries *vs.* Lists

```
lst = ['ele_1', 'ele_2', 'ele_3']
```

```
dict = {
    'key_1': 'value_1',
    'key_2': 'value_2',
    'key_3': 'value_3'
}
```

```
lst[ <INDEX> ]
```

```
dict[ <KEY> ]
```

Dictionary methods

```
dog_dic = {
    'name': 'tofu',
    'color': 'white',
    'age': '2',
}
```

```
print(dog_dic.get('name'))
```

```
tofu
```

```
print(dog_dic.keys())
```

```
dict_keys(['name', 'color', 'age'])
```

```
print(dog_dic.values())
```

```
dict_values(['tofu', 'white', '2'])
```

```
print(dog_dic.items())
```

```
dict_items([('name', 'tofu'),
('color', 'white'), ('age', '2')])
```

Tuples

- Hold items in order separated by commas
- Immutable (cannot be changed)
- Parenthesis are optional
- You **must** add a comma after the item, even if there is only one item

Examples:

- (price, stock)
- (5,)

Discussion 4 Assignment

- Go to **Canvas Assignments > Discussion 4**
- Accept the GitHub Classroom assignment and clone the repo:

  [https://classroom.github.com/a/P2mZRliJ]

- If you are having issues:
  - Canvas Files > Discussions > Discussion 4 > discussion_4.py
- **Commit at least 4 times and push to GitHub**

Typical Git Workflow

1. Clone the repository: **git clone <link>** (from GitHub Classroom)
2. Add file to staging area: **git add <file1> (<file2>) / git add .**
3. Make snapshot of current change: **git commit -m “<message>”**
4. Upload to cloud server (GitHub): **git push**

Use **git status** to check current changes. **Make sure you are in the same directory/folder of the .py file you are working on.**

Task 1: Warehouse.add_item()

Task:

- Adds an Item to the warehouse
- If an item with the same name already exists:
- Merge stock
- Do NOT create a duplicate

Discuss with your group:

- How do we check if an item already exists?

Task 2: Order.add_line (Dictionaries)

Task:

- Store order items in a dictionary
- Update quantities if the item already exists

Discuss with your group:

- Why does the key need to be the item name?

Task 3: Warehouse.fulfill_order (Multiple Classes)

Task:

- Warehouse receives an `Order`
  - Finds matching `Item` objects
  - Sells stock if possible
  - Tracks backorders in a dictionary

Discuss with your group:

- Which class is responsible for changing stock?
- What information should be returned?

Task 4 Continued [optional]: Warehouse.inventory_report()

Task

- Return a dictionary of:

  item name → (price, stock)