LAB: SỬ DỤNG CÁC LOẠI CONTAINER TRONG PYTHON

Tác giả: Đặng Kim Thi

1. Mục tiêu

- Hiểu cách sử dụng các loại container (`list`, `tuple`, `set`, `dict`).
- Biết khi nào nên sử dụng từng loại container.
- Thực hành giải quyết các bài toán liên quan đến quản lý dữ liệu bằng container.

---

2. Các ví dụ mẫu

Ví dụ 1: Quản lý danh sách học sinh (List)

Mô tả: Ta cần lưu danh sách học sinh trong một lớp học và có thể thay đổi danh sách này.

```
hoc_sinh = ["Minh", "Linh", "An"]

# Thêm học sinh
hoc_sinh.append("Nam")

# Xóa học sinh
hoc_sinh.remove("Linh")

# Hiển thị danh sách
print(hoc_sinh)  # Output: ['Minh', 'An', 'Nam']
```

Tại sao dùng List? Vì danh sách học sinh có thể thay đổi, ta cần một cấu trúc dữ liệu có thể thêm, xóa, và truy cập nhanh.

Ví dụ 2: Quản lý thông tin cá nhân cố định (Tuple)

Mô tả: Ta cần lưu thông tin cá nhân của một người, dữ liệu này không thay đổi sau khi khai báo.

```
thong_tin = ("Minh", 20, "Hà Nội")
print(thong_tin[0])  # Output: Minh
```

Tại sao dùng Tuple? Vì dữ liệu cá nhân này không thay đổi, dùng `tuple` giúp bảo vệ dữ liệu và tối ưu hiệu suất.

Ví dụ 3: Quản lý danh sách môn học không trùng lặp (Set)

Mô tả: Ta muốn lưu danh sách các môn học mà một học sinh đăng ký, không cho phép trùng môn.

```
mon_hoc = {"Toán", "Vật lý", "Hóa học"}

# Thêm môn học
mon_hoc.add("Sinh học")

# Thêm môn học trùng sẽ không ảnh hưởng
mon_hoc.add("Toán")

print(mon_hoc)  # Output: {'Toán', 'Vật lý', 'Hóa học', 'Sinh học'}
```

Tại sao dùng Set? Vì tập hợp môn học không cần trật tự và không cho phép trùng lặp.

Ví dụ 4: Quản lý điểm học sinh (Dictionary)

Mô tả: Ta cần lưu điểm số của học sinh theo từng môn học.

```
điểm = {"Minh": 8.5, "Linh": 9.0, "An": 7.8}

# Truy xuất điểm số
print(điểm["Minh"])  # Output: 8.5

# Cập nhật điểm số
điểm["Minh"] = 8.8

print(điểm)  # Output: {'Minh': 8.8, 'Linh': 9.0, 'An': 7.8}
```

Tại sao dùng Dictionary? Vì dữ liệu cần lưu theo cặp (tên - điểm), giúp truy xuất nhanh.

Ví dụ 5: Quản lý hàng đợi khách hàng (List + Queue)

Mô tả: Ta muốn quản lý thứ tự phục vụ khách hàng.

```
from collections import deque

hang_doi = deque(["Minh", "Linh", "An"])

# Phục vụ khách hàng đầu tiên
hang_doi.popleft()

print(hang_doi)  # Output: deque(['Linh', 'An'])
```

Tại sao dùng deque? Vì hàng đợi yêu cầu truy xuất nhanh ở cả hai đầu, `deque` tối ưu hơn `list`.

---

3. Bài tập thực hành

Hãy viết chương trình sử dụng container phù hợp để giải quyết các bài toán sau:

Bài tập 1: Quản lý danh sách nhân viên

- Viết chương trình lưu danh sách nhân viên và cho phép thêm/xóa nhân viên.

Bài tập 2: Danh sách số nguyên tố (Tuple)

- Viết chương trình lưu danh sách các số nguyên tố từ 1 đến 50 bằng `tuple`.

Bài tập 3: Quản lý các sản phẩm không trùng lặp (Set)

- Viết chương trình lưu danh sách các sản phẩm mà cửa hàng bán, đảm bảo không có sản phẩm trùng lặp.

Bài tập 4: Sắp xếp danh sách học sinh theo điểm

- Nhập danh sách học sinh và điểm số, sau đó sắp xếp theo điểm từ cao xuống thấp.

Bài tập 5: Đếm số lần xuất hiện của từ trong văn bản (Dictionary)

- Nhập một đoạn văn, đếm số lần xuất hiện của từng từ.

Bài tập 6: Xử lý dữ liệu log truy cập (Set)

- Cho danh sách IP truy cập vào hệ thống, lọc ra danh sách IP duy nhất.

Bài tập 7: Hệ thống hàng đợi khách hàng (Queue)

- Viết chương trình mô phỏng hệ thống hàng đợi khách hàng, phục vụ lần lượt.

Bài tập 8: Danh sách công việc ưu tiên (List + Sort)

- Nhập danh sách công việc và mức độ ưu tiên, sắp xếp theo mức độ ưu tiên.

Bài tập 9: Chuyển đổi danh sách thành từ điển

- Nhập danh sách cặp (tên, điểm), chuyển thành dictionary.

Bài tập 10: Thống kê số lượng sản phẩm bán ra

- Nhập danh sách sản phẩm đã bán, đếm số lượng mỗi sản phẩm bán ra bằng dictionary.

Hãy chọn container phù hợp cho từng bài toán!