

PYTHON CHEAT SHEET

Danh sách (Lists): Khai báo, Phương thức & Slicing

6.1 - 6.2. Khai báo và Aliasing (Cẩn thận)

- **List:** Là tập hợp thay đổi được (mutable).
- **Aliasing:** Gán biến list ($b = a$) chỉ sao chép tham chiếu, không sao chép dữ liệu.

```
# 1. Create list
nums = [1, 2, 3]
seq = list(range(5)) # [0, 1, 2, 3, 4]

# 2. Aliasing (Reference issue)
a = [1, 2, 3]
b = a          # b points to the same object
b[0] = 99      # a is now [99, 2, 3] too!

# 3. Copying (Safe way)
c = a.copy()   # Shallow copy
d = a[:]       # Slicing copy
```

6.3. Thêm và Xóa (Phương thức)

Các hàm này thay đổi trực tiếp list gốc (In-place).

```
lst = [10, 20, 30]

# ADD items
lst.append(40)      # [10, 20, 30, 40]
lst.insert(1, 15)   # Insert 15 at index 1
lst.extend([50, 60]) # Add multiple items

# REMOVE items
x = lst.pop()       # Remove & return last item
lst.remove(20)       # Remove first value 20
lst.clear()          # Delete all items
```

6.3. Sắp xếp (Sorting)

- `sort()`: Sắp xếp list hiện tại.
- `key`: Hàm tùy chỉnh tiêu chí sắp xếp.

```
words = ["apple", "bat", "cherry"]

# Sort by length (len function)
words.sort(key=len)
# Result: ['bat', 'apple', 'cherry']

# Reverse sort
words.sort(reverse=True)
```

6.4. Slicing Nâng cao

Cú pháp: `lst[start : end : step]`

```
a = [0, 1, 2, 3, 4, 5, 6]

# Step slicing
evens = a[::2]   # [0, 2, 4, 6]
odds  = a[1::2]  # [1, 3, 5]

# Slice Assignment (Modify list size)
a[1:3] = ['x', 'y', 'z']
# Replaces items at index 1,2 with 3 new items
```

PYTHON CHEAT SHEET

List Comprehensions, Ma trận & Xử lý dữ liệu

6.5. List Comprehensions

Cách viết tắt (Pythonic) thay thế vòng lặp for để tạo list mới. Nhanh và gọn.

```
# Syntax: [expr for item in iterable if cond]
```

```
nums = [1, 2, 3, 4, 5]
```

```
# Square of even numbers
```

```
sq = [x*x for x in nums if x % 2 == 0]
```

```
# Result: [4, 16]
```

```
# Generator Expression (Saves memory)
```

```
gen = (x*x for x in nums)
```

```
# Use 'gen' for large data streams
```

6.6. Nested Lists (Ma trận) - QUAN TRỌNG

Cách tạo ma trận đúng và sai (tránh lỗi tham chiếu).

```
rows = 3; cols = 2
```

```
# 1. WRONG WAY (Dangerous Aliasing):
```

```
mat = [[0] * cols] * rows
```

```
mat[0][0] = 1
```

```
# ERROR: All rows become [1, 0]!
```

```
# 2. CORRECT WAY (Independent rows):
```

```
mat = [[0]*cols for _ in range(rows)]
```

```
mat[0][0] = 1
```

```
# OK: Only first row changes.
```

6.7. Sắp xếp dữ liệu phức tạp

Sử dụng lambda để sắp xếp list chứa tuples hoặc dicts.

```
data = [('A', 5), ('B', 2), ('C', 8)]
```

```
# Sort by 2nd element (the number)
```

```
data.sort(key=lambda x: x[1])
```

```
# Result: [('B', 2), ('A', 5), ('C', 8)]
```

```
# Reverse iterator
```

```
rev = list(reversed(data))
```

6.8. Ví dụ: Đếm tần suất từ

Dùng Counter để đếm nhanh hơn tự viết vòng lặp.

```
from collections import Counter
```

```
text = "apple banana apple orange banana"
```

```
words = text.split()
```

```
# Count frequencies
```

```
counts = Counter(words)
```

```
# Result: {'apple': 2, 'banana': 2, 'orange': 1}
```

```
# Get Top 2 most common
```

```
top = counts.most_common(2)
```