PYTHON CHEAT SHEET

Chuỗi ký tự (Strings): Cơ bản

5.1. Bản chất String

- **Immutable:** Không thể thay đổi nội dung sau khi tạo.
- **Unicode:** Python 3 dùng Unicode mặc định.
- **Bytes:** Dùng `bytes` cho dữ liệu nhị phân.

5.2. Khai báo và Escape Sequences

Các cách khai báo:

```
# 1. Single/Double quotes
s = "Hello"

# 2. Multi-line (Triple quotes)
m = '''Line 1
Line 2'''

# 3. Raw String (r'...') - Keeps backslash
path = r'C:\Users\Admin'
# Useful for Regex: r'\d+'
```

5.3. Chỉ mục (Indexing) và Cắt chuỗi

Cú pháp: `s[start : end : step]`

- Index âm: Tính từ cuối (-1 là ký tự cuối).

```
s = "PYTHON"
# Get char
first = s[0]     # 'P'
last  = s[-1]    # 'N'

# Slicing
sub = s[0:2]     # 'PY'
rev = s[::-1]    # 'NOHTYP' (Reverse)
```

5.4. Tìm kiếm và Kiểm tra

- Dùng `in` để kiểm tra sự tồn tại.

```
s = "banana"

# Check existence
if "nan" in s: print("Yes")

# Find index
i = s.find("na")    # Returns 2
j = s.find("z")     # Returns -1 (Safe)
# s.index("z")      # Raises ValueError if not found

# Check start/end
s.startswith("ba") # True
```

PYTHON CHEAT SHEET

Định dạng, Hiệu năng và Nâng cao

5.5. Định dạng (Formatting)

1. f-strings (Python 3.6+) - Khuyên dùng:

```
name = "AI"; v = 2.0
# Embed variables directly
msg = f"Model {name} v{v}"

# Number formatting
pi = 3.14159
print(f"Pi = {pi:.2f}") # Output: 3.14
```

2. Cách cũ (Legacy):

```
"Hello {0}".format(name) # .format()
"Age: %d" % 25           # % operator
```

5.6. Hiệu năng (Performance)

Lưu ý: Tránh cộng chuỗi (+) trong vòng lặp vì rất chậm ($O(n^2)$).
Giải pháp: Dùng List Append + Join.

```
items = ["A", "B", "C"]

# BAD way:
# s = ""
# for i in items: s += i

# GOOD way:
s = "".join(items)
# Output: "ABC"
```

5.7. Các hàm phổ biến (Methods)

Các hàm này trả về chuỗi mới, không đổi chuỗi gốc.

- `.strip()`: Xóa khoảng trắng 2 đầu.
- `.upper()` / `.lower()`: Viết hoa / thường.
- `.split(sep)`: Tách chuỗi thành List.
- `.join(iter)`: Nối List thành chuỗi.
- `.replace(old, new)`: Thay thế.
- `.count(sub)`: Đếm số lần xuất hiện.

5.8. Regex Parsing

Dùng module `re` để trích xuất dữ liệu phức tạp.

```
import re
line = 'From stephen.marquard@uct.ac.za Sat'

# Extract email domain after '@'
# Regex: @ followed by non-space chars
match = re.search(r'@([^ ]+)', line)

if match:
    domain = match.group(1)
    # Output: uct.ac.za
```