

# Chương 6: Strings.

Slice là gì?

- Một đoạn của chuỗi gọi là slice.
- Cách chọn slice giống như chọn ký tự nhưng dùng dấu hai chấm ::

```
s = "Monty Python"
```

```
s[0:5] # 'Monty' → từ index 0 đến 4
```

```
s[6:12] # 'Python' → từ index 6 đến 11
```

Slice bao gồm ký tự đầu tiên, nhưng không bao gồm ký tự cuối.

String slices

Khái niệm

- String = dãy ký tự (sequence of characters).
- Truy cập từng ký tự bằng [].

Index (chỉ số)

- Python đánh số từ 0.
  - fruit[0] → ký tự đầu
  - fruit[1] → ký tự thứ hai
- Index phải là số nguyên, nếu không sẽ bị TypeError.

Ghi nhớ nhanh

- "banana":
  - b → index 0
  - a → index 1
  - n → index 2
- fruit[1] = 'a'.

A string is a sequence

Hàm len()

- len(fruit) → số ký tự của chuỗi.
- "banana" → 6 ký tự.

Lỗi IndexError

- fruit[len(fruit)] → lỗi (vượt phạm vi).
- Vì index hợp lệ là 0 → 5.

Lấy ký tự cuối

- Cách chuẩn: fruit[len(fruit) - 1].
- Cách nhanh: fruit[-1] (index âm).

Index âm

- fruit[-1] → ký tự cuối
- fruit[-2] → ký tự kế cuối

Getting the length of a string using len

Traversal là gì?

- Là quá trình xử lý chuỗi từng ký tự một, từ đầu đến cuối.
- Mỗi lần lặp → lấy 1 ký tự → làm gì đó với nó → lặp tiếp cho đến hết chuỗi.

Duyệt chuỗi bằng while

```
index = 0
```

```
while index < len(fruit):
```

```
    letter = fruit[index]
```

```
    print(letter)
```

```
    index = index + 1
```

Cách hoạt động:

- Bắt đầu tại index = 0.
- Lặp đến khi index == len(fruit) thì dừng lại.
- Ký tự cuối cùng được truy cập là fruit[len(fruit) - 1].

Duyệt chuỗi bằng for (cách đơn giản hơn).

- Mỗi vòng lặp, biến char tự động nhận ký tự tiếp theo.
- Lặp đến khi hết ký tự trong chuỗi.

a Traversal through a string with a loop

Immutable là gì?

- Chuỗi trong Python không thể thay đổi sau khi tạo.
- Không thể gán giá trị trực tiếp cho ký tự trong chuỗi:

```
greeting = "Hello, world!"
```

```
greeting[0] = "J"
```

"Object" = chuỗi, "item" = ký tự trong chuỗi.

Cách thay đổi chuỗi

- Bạn không thể sửa trực tiếp, nhưng có thể tạo chuỗi mới dựa trên chuỗi cũ:

```
greeting = "Hello, world!"
```

```
new_greeting = "J" + greeting[1:]
```

```
print(new_greeting) # 'Jello, world!'
```

- Đây là kết hợp ký tự mới + slice của chuỗi cũ.
- Chuỗi gốc greeting không bị thay đổi.

Strings are immutable

# Chương 6: Strings.

khái niệm

- Chuỗi = object, có methods = hàm đi kèm.
- `dir(obj)` → xem danh sách method

Cách gọi

```
word = "banana"
word.upper() # 'BANANA'
word.find("a") # 1
word.strip() # bỏ khoảng trắng đầu/cuối
word.startswith("H") # True/False
word.lower() # 'banana'
word.count("a") # 3
```

string methods

Mục tiêu

- Kiểm tra xem một chuỗi có xuất hiện trong chuỗi khác hay không.
- Kết quả là Boolean: True hoặc False.

Cú pháp

```
"a" in "banana" # True
"seed" in "banana" # False
```

Cách hoạt động:

- Nếu chuỗi bên trái xuất hiện bất kỳ đâu trong chuỗi bên phải → True.
- Nếu không → False.

The in operator

Mục tiêu

- Đếm số lần ký tự xuất hiện trong chuỗi.
- Đây là mẫu tính toán kiểu counter.

Ví dụ:

```
word = "banana"
count = 0
for letter in word:
    if letter == "a":
        count = count + 1
print(count)
```

Cách hoạt động:

- Khởi tạo count = 0.
- Duyệt từng ký tự trong chuỗi.
- Nếu ký tự bằng "a" → tăng biến count lên 1.
- Khi kết thúc vòng lặp → count chứa tổng số ký tự "a".

Looping and counting

- So sánh chuỗi: ==, <, >
- Ví dụ:

```
word = "banana"
if word == "banana":
    print("All right, bananas.")
elif word < "banana":
    print("Your word, " + word + ", comes before banana.")
else:
    print("Your word, " + word + ", comes after banana.")
```

- Lưu ý: chữ hoa < chữ thường → "Pineapple" < "banana" là True.
- Giải pháp: chuẩn hóa chữ thường trước khi so sánh:

```
word.lower() < "banana"
```

String comparison