ĐẠI NAM
UNIVERSITY

LẬP TRÌNH PYTHON

Chương 8 – Tìm hiểu thêm về Chuỗi

NỘI DUNG

- Truy cập các phần tử trong chuỗi
- Nối và cắt chuỗi
- Kiểm tra, tìm kiếm và xử lý chuỗi

Truy cập các ký tự riêng lẻ trong một chuỗi

Để truy cập một ký tự riêng lẻ trong chuỗi, có thể sử dụng:

➢ Vòng lặp for

- Cú pháp: *for character in string*:

```
str1 = 'Tu hoc Python'
for char in str1:
    print(char)
```

→

```
T
u

h
o
c

P
y
t
h
o
n
```

Truy cập các ký tự riêng lẻ trong một chuỗi

Để truy cập một ký tự riêng lẻ trong chuỗi, có thể sử dụng:

- Chỉ số (Index)
  - Mỗi ký tự trong chuỗi có một chỉ số xác định vị trí của nó trong chuỗi, bắt đầu từ 0.
  - Cú pháp: *character* = *my_string[i]*

	T	u		h	o	c		P	y	t	h	o	n
Chỉ số	0	1	2	3	4	5	6	7	8	9	10	11	12
	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
str1 = 'Tu hoc Python'
print(str1[0])
```

Truy cập các ký tự riêng lẻ trong một chuỗi

Ngoại lệ **IndexError** sẽ xảy ra nếu:

- Cố gắng sử dụng một chỉ mục (index) nằm ngoài phạm vi của chuỗi.
- Điều này có khả năng xảy ra khi vòng lặp lặp qua phạm vi vượt quá độ dài của chuỗi.

```
str1 = 'Tu hoc Python'
print(str1[13])
```

```
print(str1[13])
IndexError: string index out of range
```

=> Để tránh lỗi này, bạn nên sử dụng hàm len(string) để lấy độ dài của chuỗi

Chỉ số ở vị trí cuối cùng = độ dài string - 1

Truy cập các ký tự riêng lẻ trong một chuỗi

- Truy cập các phần tử riêng lẻ của chuỗi bằng cách sử dụng vòng lặp **for** và hàm **range** khi quan tâm vị trị

```
name = 'Phong'
for i in range(len(name)):
    print(name[i], type(name[i]))
```

```
P <class 'str'>
h <class 'str'>
o <class 'str'>
n <class 'str'>
g <class 'str'>
```

Nối chuỗi

Nối chuỗi: thêm một chuỗi vào cuối một chuỗi khác

- Sử dụng toán tử **+**
- Toán tử gán tăng cường **+=**

```
str1 = 'Đại học'
str2 = 'Đại Nam'
str3 = str1 + str2
str1 += str2
print(str3)
print(str1)
```

```
Đại họcĐại Nam
Đại họcĐại Nam
```

Chuỗi là bất biến

- **Chuỗi là bất biến:**
  - Khi chúng đã được tạo ra, chúng không thể bị thay đổi.
  - Việc nối chuỗi không thực sự thay đổi chuỗi hiện có, mà thay vào đó tạo ra một chuỗi mới và gán chuỗi mới đó cho biến đã sử dụng trước đó.
- **Không thể sử dụng biểu thức theo dạng:**
  - ***string[index] = new_character***
  - Câu lệnh kiểu này sẽ gây ra một ngoại lệ.

```
>>> name = 'Phong'
>>> name[2] = 'a'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

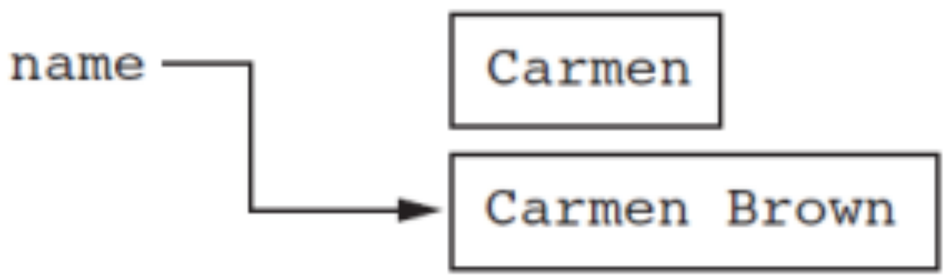
Chuỗi là bất biến, Biến thì không

The string 'Carmen' assigned to `name`

```
name = 'Carmen'
```

The string 'Carmen Brown' assigned to `name`

```
name = name + ' Brown'
```



Cắt chuỗi

Cú pháp cơ bản của cắt chuỗi là:

string[start:stop:step]

- start: Chỉ số bắt đầu của đoạn cắt (bao gồm).
- stop: Chỉ số kết thúc của đoạn cắt (không bao gồm).
- step: Bước nhảy giữa các chỉ số (tuỳ chọn).

```
text = "Hello, World!"
print(text[0:5])  # Kết quả: Hello
```

Kiểm tra, tìm kiếm và xử lý chuỗi

- Có thể sử dụng toán tử in để xác định xem một chuỗi có nằm trong chuỗi khác không.

  Cú pháp tổng quát: **string1 in string2**

- Sử dụng toán tử **not in** để xác định xem một chuỗi không nằm trong chuỗi khác.

```
string1 = 'Nam'
string2 = 'Đại Nam'
print(string1 in string2)
=> True
```

Các phương thức cơ bản

Phương thức	Mô tả
len()	Trả về độ dài của chuỗi
str()	Chuyển một giá trị thành chuỗi
upper()	Chuyển chuỗi thành chữ in hoa
lower()	Chuyển chuỗi thành chữ thường
strip()	Xóa khoảng trắng ở đầu và cuối chuỗi
replace()	Thay thế một phần của chuỗi bằng chuỗi khác
split()	Tách chuỗi thành danh sách các phần tử dựa trên ký tự phân cách
join()	Ghép danh sách các chuỗi thành một chuỗi với ký tự nối

Các phương thức cơ bản

Phương thức	Mô tả
find()	Tìm vị trí xuất hiện đầu tiên của chuỗi con
startswith()	Kiểm tra xem chuỗi có bắt đầu bằng chuỗi con cụ thể không
endswith()	Kiểm tra xem chuỗi có kết thúc bằng chuỗi con cụ thể không
isnumeric()	Kiểm tra xem chuỗi có chỉ bao gồm các ký tự số không
capitalize()	Viết hoa chữ cái đầu tiên của chuỗi
count()	Đếm số lần xuất hiện của chuỗi con
format()	Định dạng chuỗi với các giá trị thay thế

Các phương thức tìm kiếm

- **endswith(substring)**: kiểm tra xem chuỗi có kết thúc bằng chuỗi con không. Trả về **True** hoặc **False**.
- **startswith(substring)**: kiểm tra xem chuỗi có bắt đầu bằng chuỗi con không. Trả về **True** hoặc **False**.

```
# Kiểm tra với endswith()
s = "Hello, Python!"
result_ends = s.endswith("Python!")
print(result_ends)  # Output: True

# Kiểm tra với startswith()
result_starts = s.startswith("Hello")
print(result_starts)  # Output: True
```

Các phương thức

- **find(substring)**: tìm kiếm chuỗi con trong chuỗi gốc. Trả về chỉ số thấp nhất của chuỗi con, hoặc nếu chuỗi con không có trong chuỗi gốc, trả về -1.
- **replace(substring, new_string)**: Trả về một bản sao của chuỗi trong đó mọi lần xuất hiện của chuỗi con đều được thay thế bằng chuỗi mới.

```
# Ví dụ với find()
s = "Hello, Python!"
result_find = s.find("Python")
print(result_find)  # Output: 7

# Ví dụ với replace()
s_replace = s.replace("Python", "World")
print(s_replace)  # Output: "Hello, World!"
```

Toán tử lặp lại

Toán tử lặp lại: tạo ra nhiều bản sao của một chuỗi và nối chúng lại với nhau.

Ký hiệu * là toán tử lặp lại khi áp dụng cho một chuỗi và một số nguyên.

- Chuỗi là toán hạng bên trái; số nguyên là toán hạng bên phải.
- Cú pháp tổng quát: chuỗi_cần_sao_chép * n

```
# Ví dụ với toán tử lặp lại
s = "Hello! "
result = s * 3
print(result)  # Output: "Hello! Hello! Hello! "
```

Tách chuỗi

Phương thức **split**: trả về một danh sách chứa các từ trong chuỗi.

- Mặc định, phương thức sử dụng khoảng trắng làm ký tự phân tách.
- Bạn có thể chỉ định một ký tự phân tách khác bằng cách truyền nó làm tham số cho phương thức **split**.

```
# Ví dụ với split mặc định (sử dụng khoảng trắng làm phân tách)
s = "Hello Python World"
result = s.split()
print(result)  # Output: ['Hello', 'Python', 'World']

# Ví dụ với split có phân tách khác (dấu phẩy)
s2 = "apple,banana,orange"
result2 = s2.split(',')
print(result2)  # Output: ['apple', 'banana', 'orange']
```

Tổng kết

- Truy cập được các phần tử trong chuỗi
- Thực hiện nối và cắt chuỗi
- Sử dụng được một số phương thức cơ bản trong chuỗi
- Kiểm tra, tìm kiếm và xử lý chuỗi

HỎI ĐÁP

ĐẠI NAM
UNIVERSITY

Trân trọng cảm ơn!