



# PERL & PYTHON

**Course Code: CS466**

**No. of Credit Hours: 2 credits (LEC)**

Lecturer: Dang, Tran Huu Minh

Mobile/ Zalo: 0918.763.367

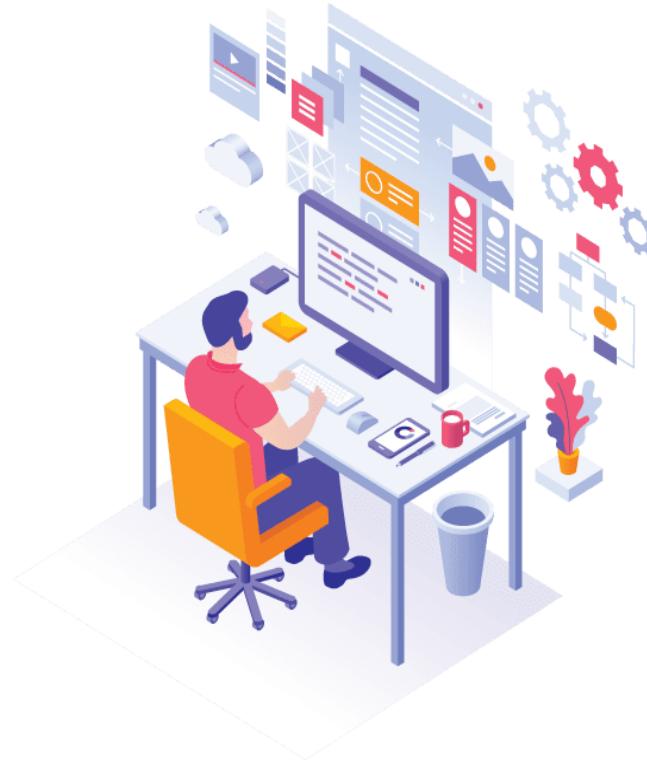
Email: [tranhminhdang@dtu.edu.vn](mailto:tranhminhdang@dtu.edu.vn)



## GIỚI THIỆU THÔNG TIN SLIDE

Chủ đề:

# Data Structures in Python



Thời lượng: 120 phút

Lecturer: Dang, Tran Huu Minh

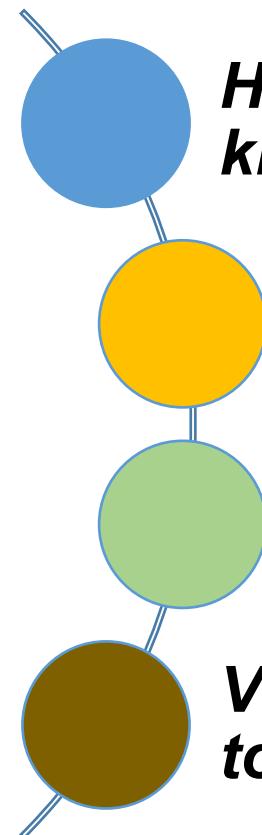
Mobile/ Zalo: 0918.763.367

Email: [tranhminhdang@dtu.edu.vn](mailto:tranhminhdang@dtu.edu.vn)



Sau khi học xong, sinh viên sẽ có được những khả năng sau:

## MỤC TIÊU BÀI HỌC



**Hiểu và áp dụng một cách chính xác các kiểu dữ liệu trong Python**

**Hiểu các hàm và phương thức liên quan đến các kiểu dữ liệu.**

**Hiểu được cách chuyển đổi giữa chúng, tìm kiếm, sắp xếp và lọc dữ liệu.**

**Vận dụng kiến thức để giải quyết các bài toán lập trình cụ thể.**



# Reference materials

## Textbooks:

- 1.Randal L. Schwartz, Tom Phoenix, Brian D Foy (2021). *Learning Perl: Making Easy Things Easy and Hard Things Possible, 8th Edition*. O'Reilly Media.
- 2.Eric Chou (2023). *Mastering Python Networking, Fourth Edition*. Packt Publishing.

## Reference Materials:

- 1.Florian Dedov (2020). *The Python Bible 7 in 1: Volumes One To Seven (Beginner, Intermediate, Data Science, Machine Learning, Finance, Neural Networks, Computer Vision)*. Independently published.



# Data Structures in Python

- LIST, SET, TUPLE, DICTIONARY, ARRAY
- Regular Expressions And Strings
- Files
- Conclusion

NỘI  
DUNG





## NỘI DUNG TÌM HIỂU VỀ:

1. LIST
2. SET
3. TUPLE
4. DICT

## YÊU CẦU:

Nhóm lẻ: Tìm hiểu về 1, 3

Nhóm Chẵn: Tìm hiểu về 2, 4

Nội dung tìm hiểu chi tiết như mẫu → (VD)

Thời gian: 45 phút

Trình bày bằng Slide (tối thiểu 20 slide)

VD (ví dụ)

01.  
LIST

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List



## TÌM HIỂU VỀ: LIST

### 01. **LIST**

- 01** List là gì trong Python?
- 02** Một số ưu điểm của List
- 03** Tạo một List
- 04** Truy cập, thay đổi, xóa các phần tử
- 05** Phương thức trong List
- 06** Kiểm tra, lặp qua các phần tử trong List



## 01. **LIST**

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

List chứa những phần tử được ngăn cách nhau bởi dấu "," và nằm trong cặp dấu ngoặc vuông ([ ]).

### 01 List là gì trong Python?

List dùng để lưu trữ một dãy các phần tử có thể thay đổi (mutable), có thứ tự (ordered) và cho phép các phần tử trùng lặp.

Mỗi phần tử trong list có thể là bất kỳ kiểu dữ liệu nào như số, chuỗi, hoặc thậm chí là một list khác.

Ví dụ:

# Ví dụ. Danh sách với một số phần tử:

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

# Ví dụ. Danh sách chứa các kiểu dữ liệu khác nhau:

```
mixed_list = [1, "Hello", 3.14, True]
```



## 01. LIST

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

## 02 Một số ưu điểm của List

- **Có thứ tự:** Các phần tử trong list được lưu trữ theo một thứ tự nhất định.
- **Dễ dàng thay đổi:** Bạn có thể thay đổi giá trị của các phần tử trong list sau khi đã tạo ra list.
- **Hỗ trợ các phương thức mạnh mẽ:** Python cung cấp nhiều phương thức hữu ích để thao tác với list.
- **Có thể chứa các phần tử trùng lặp:** Không giống như set, list cho phép các phần tử trùng lặp.



# 01. **LIST**

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

## 03 Tạo một List

Tạo một list bằng cách sử dụng dấu ngoặc vuông [] hoặc hàm list()

```
# Tạo list trống  
empty_list = []
```

```
# Tạo list với các phần tử ban đầu  
monhoc = ['cs201', 'cs252', 'cs466']
```



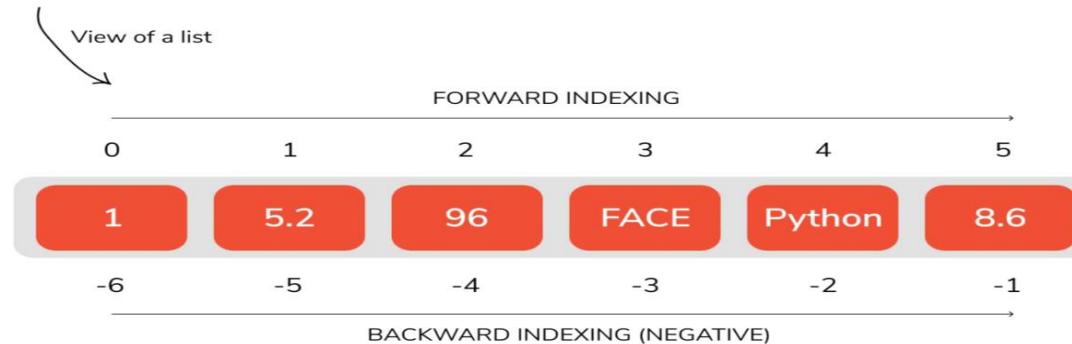
## 01. **LIST**

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

List: [a, b, c, d]  
index: 0 1 2 3  
Reverse index: -4 -3 -2 -1

### Ví dụ:

List = [ 1, 5.2, 96, "FACE", "Python", 8.6]



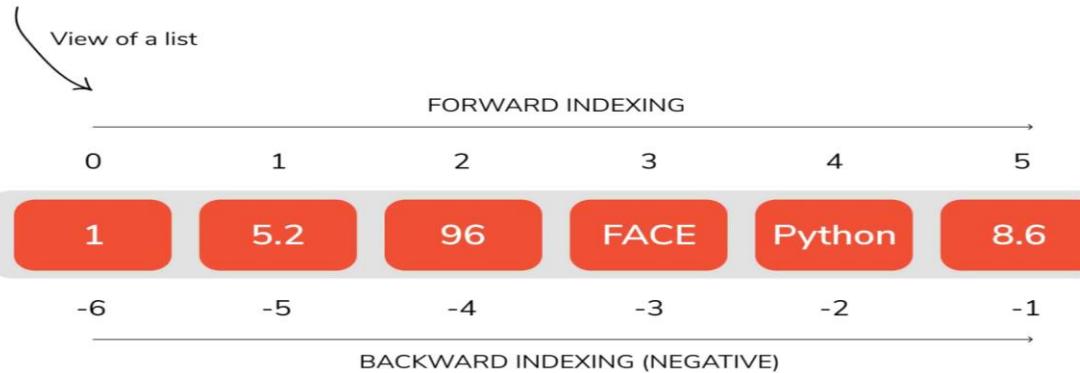


## 01. LIST

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

### 04 Truy cập, thay đổi, xóa các phần tử

List = [ 1, 5.2, 96, "FACE", "Python", 8.6]



**Truy cập phần tử:** Sử dụng chỉ số (index).

# Truy cập phần tử đầu tiên

**first\_list = list[0] # return: 1**

# Truy cập phần tử cuối cùng

**last\_list = fruits[-1] # 8.6**

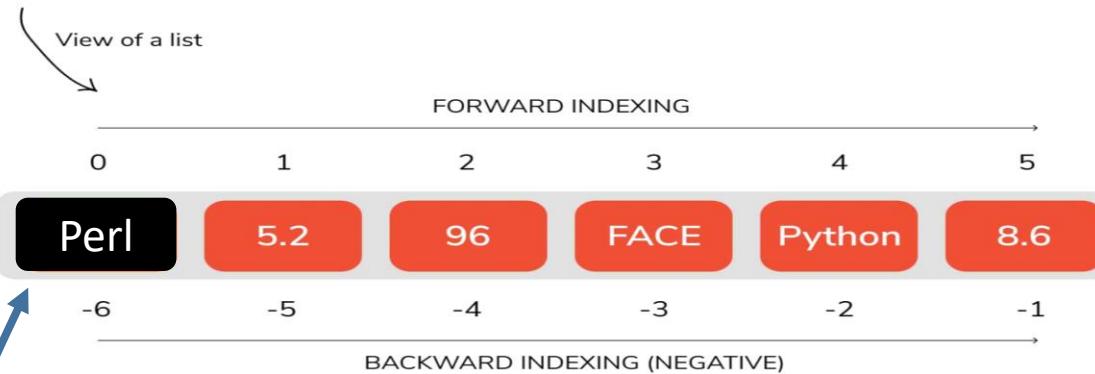


## 01. LIST

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

### 04 Truy cập, thay đổi, xóa các phần tử

List = [ 1, 5.2, 96, "FACE", "Python", 8.6]



**Thay đổi phần tử:** Gán giá trị mới cho phần tử theo chỉ số.

**List[0] = "Perl"**

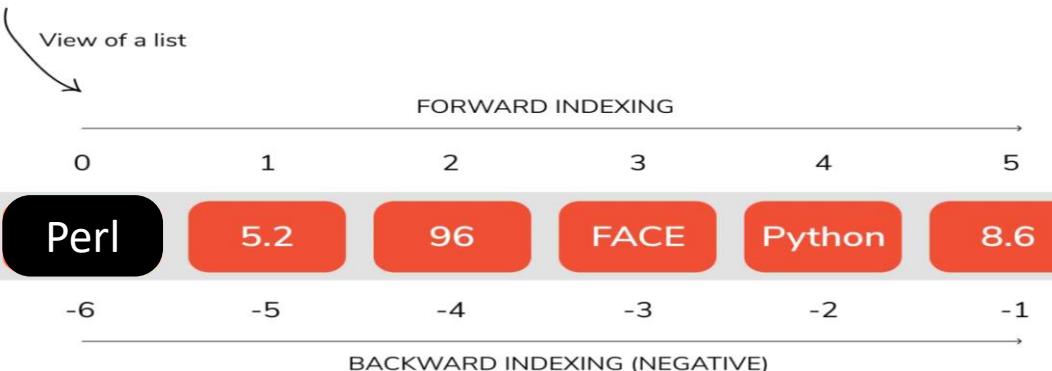


## 01. LIST

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

### 04 Truy cập, thay đổi, xóa các phần tử

List = [ 1, 5.2, 96, "FACE", "Python", 8.6]



**Xóa phần tử: del, remove(), hoặc pop().**

# Xóa phần tử theo chỉ số

**del List[1]**

# Xóa phần tử theo giá trị

**List.remove("Perl")**

# Xóa phần tử cuối cùng và trả về nó

**last\_list = List.pop()**



## 01. LIST

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

- 01 .pop()
- 02 .sort()
- 03 .count()
- 04 .clear()
- 05 .copy()
- 06 .append()
- 07 .insert()
- 08 .extend()
- 09 .remove()
- 10 .reverse()

## 05 Phương thức trong List

- **append(item)**: Thêm một phần tử vào cuối list.
- **insert(index, item)**: Thêm một phần tử vào vị trí chỉ định.
- **extend(iterable)**: Thêm các phần tử từ một iterable (như list khác) vào cuối list hiện tại.
- **remove(item)**: Xóa phần tử đầu tiên có giá trị bằng item.
- **pop([index])**: Xóa và trả về phần tử tại index (mặc định là phần tử cuối cùng).



## 01. LIST

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

### 05 Phương thức trong List

- **clear():** Xóa tất cả các phần tử trong list.
- **index(item):** Trả về chỉ số của phần tử đầu tiên có giá trị bằng item.
- **count(item):** Đếm số lần xuất hiện của item trong list.
- **sort(key=None, reverse=False):** Sắp xếp các phần tử trong list.
- **reverse():** Đảo ngược thứ tự các phần tử trong list.
- **copy():** Tạo một bản sao của list.



## 01. **LIST**

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

## 05 Phương thức trong List

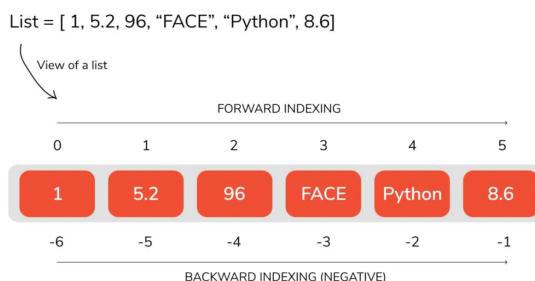
### Ví dụ:

Input	Method	Output
[10, 20, 30]	.append(40)	[10, 20, 30, 40]
[10, 20, 30]	.extend([40])	[10, 20, 30, 40]
[10, 20, 30]	.insert(2, 40)	[10, 20, 40, 30]
[10, 20, 30, 20]	.count(20)	2
[10, 20, 30]	.clear()	[]
[10, 20, 30, 40]	.index(40)	3
[10, 20, 30, 40]	.remove(10)	[20, 30, 40]
[10, 20, 30, 40]	.pop(2)	[10, 20, 40]
[30, 20, 10]	.reverse()	[10, 20, 30]
[30, 10, 40, 20]	.sort()	[10, 20, 30, 40]
[10, 20, 30]	.copy()	[10, 20, 30]



## 01. LIST

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List



### 06 Kiểm tra, lặp qua các phần tử trong List

- **Kiểm tra phần tử:** Sử dụng toán tử in.

```
if ('Python' in List ):  
    print("True")  
else:  
    print("False")
```

→ Kết quả trả về True nếu 'Python' có trong List

- **Lặp qua các phần tử:** Sử dụng for.

```
for i in List:  
    print(i)
```



## 01. **LIST**

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

## 07 Ví dụ

Các câu lệnh dưới đây cho kết quả gì?

List = [10, 20, 30 , 40, 50]

1. Print(List[1 : 4 : 1])

2. Print(List[1 : 4])

3. Print(List[4 : 1 : -1])

4. Print(List[-5 : -2 : 1])

5. Print(List[-5 : -2])

6. Print(List[-2 : -5 : -1])



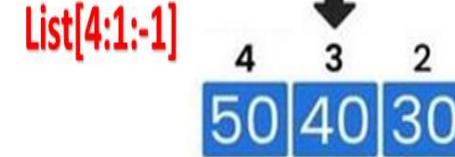
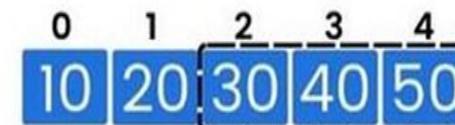
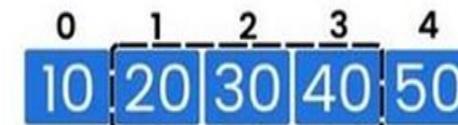
## 07 Ví dụ

### 01. LIST

- 01 List là gì trong Python?
- 02 Một số ưu điểm của List
- 03 Tạo một List
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong List
- 06 Kiểm tra, lặp qua các phần tử trong List

List = [10, 20, 30, 40, 50]  
1. Print(List[1 : 4 : 1])  
2. Print(List[1 : 4])  
3. Print(List[4 : 1 : -1])  
4. Print(List[-5 : -2 : 1])  
5. Print(List[-5 : -2])  
6. Print(List[-2 : -5 : -1])

QUESTION





## 02. **TUPLE**

- 01 Tuple là gì trong Python?**
- 02 Một số ưu điểm của Tuple**
- 03 Tạo một Tuple**
- 04 Truy cập, thay đổi, xóa các phần tử**
- 05 Phương thức trong Tuple**
- 06 Kiểm tra, lặp qua các phần tử  
trong Tuple**



## 02

Kiểu dữ liệu TUPLE

- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

### 01 Tuple là gì trong Python?

- Tuple dùng để lưu trữ một dãy các phần tử có thứ tự (ordered) và không thể thay đổi (immutable).
- Các phần tử trong tuple có thể là bất kỳ kiểu dữ liệu nào, và một tuple có thể chứa nhiều kiểu dữ liệu khác nhau.

#### Ví dụ:

- ( ) An empty tuple
- (33, 55, 77) A tuple of numbers
- (33, 3.3, (3+3j)) A tuple of mixed numbers
- (33, '33', [3, 3]) A tuple of mixed types
- (('x','y'), ('X','Y')) A tuple of tuples



- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

## 02 Một số ưu điểm của Tuple

- **Bất biến:** Một khi đã tạo, các phần tử trong tuple không thể thay đổi. Điều này làm cho tuple an toàn hơn khi bạn không muốn dữ liệu bị thay đổi ngoài ý muốn.
- **Có thứ tự:** Các phần tử trong tuple được lưu trữ theo thứ tự nhất định và có thể truy cập thông qua chỉ số.
- **Hiệu suất cao:** Vì tuple là bất biến, chúng tiêu tốn ít bộ nhớ hơn và xử lý nhanh hơn so với list.
- **Được sử dụng làm khóa trong dictionary:** Tuple có thể được sử dụng làm khóa cho dictionary vì chúng là bất biến.



## 02

Kiểu dữ liệu TUPLE

- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

### 03 Tạo một Tuple

Tạo một tuple bằng cách sử dụng dấu ngoặc đơn () hoặc hàm tuple().

# Tạo tuple trống  
**empty\_tuple = ()**

# Tạo tuple với các phần tử ban đầu  
**monhoc = ('CS201', 'CS252', 'CS466')**

# Tạo tuple từ list  
**monhoc\_list = ['CS101', 'CS226', 'CS211']**  
**monhoc\_tuple = tuple(monhoc\_list)**



## 02

Kiểu dữ liệu TUPLE

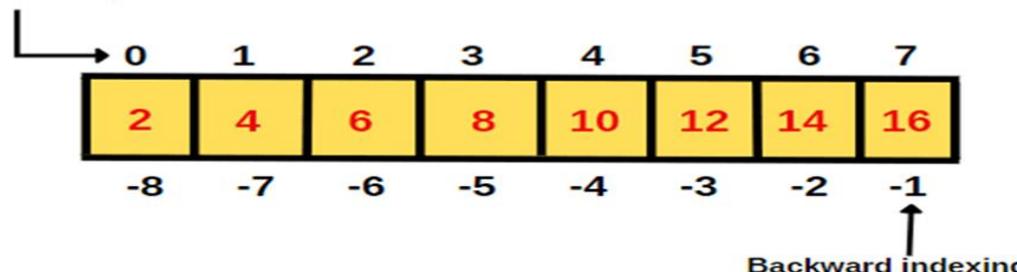
- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

### 04 Truy cập, thay đổi, xóa các phần tử

Sử dụng chỉ số (**index**).

`data = (2, 4, 6, 8, 10, 12, 14, 16)`

Forward indexing



`data[0] = 2 = data[-8], data[1] = 4 = data[-7], data[2] = 6 = data[-6],`

`data[3] = 8 = data[-5], data[4] = 10 = data[-4], data[5] = 12 = data[-3],`

`data[6] = 14 = data[-2], data[7] = 16 = data[-1]`



## 02

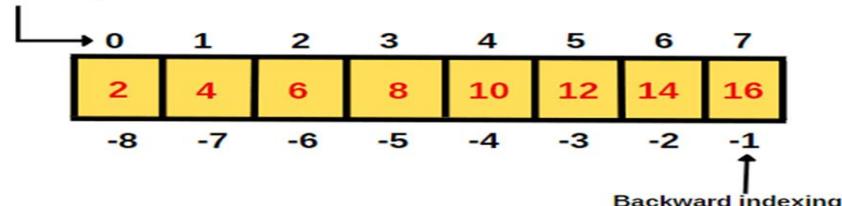
Kiểu dữ liệu TUPLE

- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

### 04 Truy cập, thay đổi, xóa các phần tử

`data = (2, 4, 6, 8, 10, 12, 14, 16)`

Forward indexing



- **Truy cập phần tử:** Sử dụng chỉ số (index).

# Truy cập phần tử đầu tiên

**`first_data = data[0] # kết quả: 2`**

# Truy cập phần tử cuối cùng

**`last_data = fruits[-1] # Kết quả: 16`**

`data[0] = 2 = data[-8], data[1] = 4 = data[-7], data[2] = 6 = data[-6],`

`data[3] = 8 = data[-5], data[4] = 10 = data[-4], data[5] = 12 = data[-3],`

`data[6] = 14 = data[-2], data[7] = 16 = data[-1]`



## 02

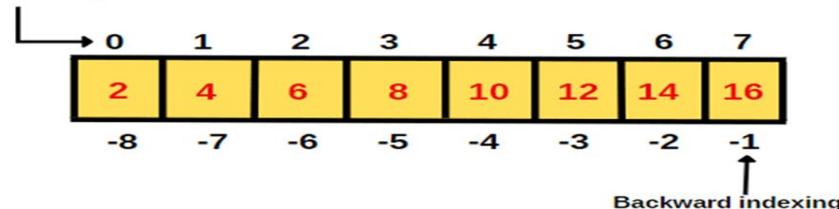
Kiểu dữ liệu TUPLE

- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

### 04 Truy cập, thay đổi, xóa các phần tử

`data = (2, 4, 6, 8, 10, 12, 14, 16)`

Forward indexing



**Thay đổi phần tử:** Vì tuple là bất biến, bạn không thể thay đổi các phần tử trực tiếp. Tuy nhiên, bạn có thể tạo một tuple mới từ các phần tử của tuple cũ.

# Tạo một tuple mới

`new_data = data[:1] + (100,) + data[2:]`

**Kết quả:** (2, 100, 6, 8, 10, 12, 14, 16)



## 02

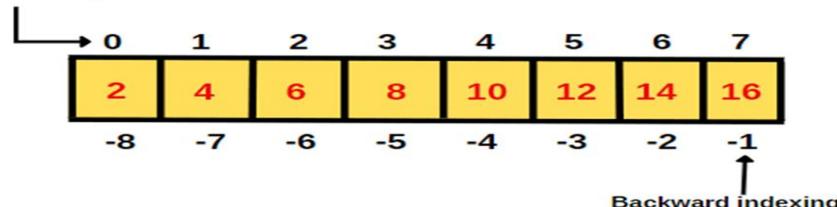
Kiểu dữ liệu TUPLE

- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

### 04 Truy cập, thay đổi, xóa các phần tử

```
data = (2, 4, 6, 8, 10, 12, 14, 16)
```

Forward indexing



**Xóa tuple:** Bạn không thể xóa một phần tử cụ thể trong tuple, nhưng bạn có thể xóa toàn bộ tuple.

```
# Xóa toàn bộ tuple  
del data
```



## 02

Kiểu dữ liệu TUPLE

- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

### 05 Phương thức trong Tuple

- **count(item)**: Đếm số lần xuất hiện của item trong tuple.
- **index(item)**: Trả về chỉ số của phần tử đầu tiên có giá trị bằng item.

```
my_tuple = (1, 2, 3, 1, 4, 1)
```

```
count = my_tuple.count(1)
```

```
# kết quả: 3
```

```
index = my_tuple.index(3)
```

```
# Kết quả: 2
```



## 02

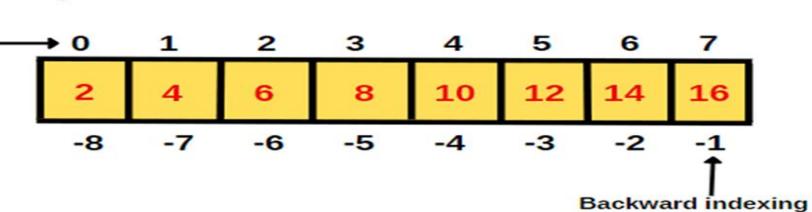
Kiểu dữ liệu TUPLE

- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

### 06 Kiểm tra, lặp qua các phần tử trong Tuple

`data = (2, 4, 6, 8, 10, 12, 14, 16)`

Forward indexing



▪ **Kiểm tra phần tử:** Sử dụng toán tử `in`.

```
if (12 in data):
    print("12 có trong data")
else:
    print("12 không có trong data")
```

# Kết quả trả về: 12 có trong data



## 02

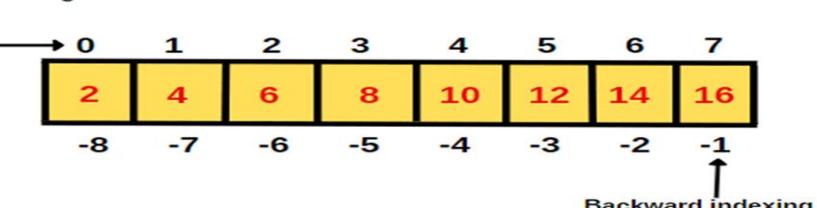
Kiểu dữ liệu TUPLE

- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

### 06 Kiểm tra, lặp qua các phần tử trong Tuple

`data = (2, 4, 6, 8, 10, 12, 14, 16)`

Forward indexing



#### ▪ Lặp qua các phần tử: Sử dụng `for`.

```
for i in data:  
    print(i)
```

Kết quả trả về:  
**2**  
**4**  
**6**  
**8**  
**10**  
**12**  
**14**  
**16**



## 02

Kiểu dữ liệu TUPLE

- 01 Tuple là gì trong Python?
- 02 Một số ưu điểm của Tuple
- 03 Tạo một Tuple
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Tuple
- 06 Kiểm tra, lặp qua các phần tử trong Tuple

## 07 Bài tập

### Slicing Tuples

`my_tuple=( 1, 2, 3, 4, 5 )`

-5    -4    -3    -2    -1

`my_tuple[-4:-2]=(2,3)`

↶ Bắt đầu tại index -4 đến trước index -2

`my_tuple[2:4]=?`

`my_tuple[1:3]=?`

`my_tuple[:4]=?`

`my_tuple[0]=?`

`my_tuple[1]=?`

`my_tuple[0:]=?`

`my_tuple[:]=?`



## 2.1 Các kiểu dữ liệu trong Python

### 03. **SET**

- 01 Set là gì trong Python?**
- 02 Một số ưu điểm của Set**
- 03 Tạo một Set**
- 04 Truy cập, thay đổi, xóa các phần tử**
- 05 Phương thức trong Set**
- 06 Kiểm tra, lặp qua các phần tử trong Set**



## 03 Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

### 01 Set là gì trong Python?

Set là một cấu trúc dữ liệu trong Python dùng để lưu trữ một tập hợp các phần tử duy nhất, không có thứ tự và không thể thay đổi (immutable).

Các phần tử trong set phải là các đối tượng không thay đổi như số, chuỗi, hoặc tuple.



## 03

Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

### 02

## Một số ưu điểm của Set

- **Duy nhất:** Mỗi phần tử trong set là duy nhất, không có phần tử trùng lặp.
- **Hiệu suất cao:** Set có hiệu suất cao khi kiểm tra sự tồn tại của một phần tử, thêm hoặc xóa phần tử.
- **Hỗ trợ các phép toán tập hợp:** Set hỗ trợ các phép toán tập hợp như union (hợp), intersection (giao), difference (hiệu), và symmetric difference (hiệu đối称).



## 03 Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

### 03 Tạo một Set

Tạo một set bằng cách sử dụng dấu ngoặc nhọn {} hoặc hàm set().

```
# Tạo set rỗng  
empty_set = set()
```

```
# Tạo set với các phần tử ban đầu  
data = {'Perl', 'Python', 'Java'}
```



## 03

Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

## 04 Truy cập, thay đổi, xóa các phần tử

- **Xóa phần tử:** Sử dụng remove(), discard(), hoặc pop().

# Xóa phần tử

**data.remove('C++')**

→ kết quả gây lỗi nếu 'C++' không có trong set

**data.discard('C++')**

→ Không gây lỗi nếu 'C++' không có trong set



## 03

Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

## 05 Phương thức trong Set

- **add(item)**: Thêm một phần tử vào set.
- **remove(item)**: Xóa phần tử khỏi set, gây lỗi nếu phần tử không tồn tại.
- **discard(item)**: Xóa phần tử khỏi set, không gây lỗi nếu phần tử không tồn tại.
- **pop()**: Xóa và trả về một phần tử ngẫu nhiên từ set.
- **union(other\_set) hoặc |**: Trả về một set mới bao gồm các phần tử của cả hai set.



## 03

Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

## 05 Phương thức trong Set

- **intersection(other\_set) hoặc &:** Trả về một set mới bao gồm các phần tử có trong cả hai set.
- **difference(other\_set) hoặc -:** Trả về một set mới bao gồm các phần tử có trong set đầu tiên nhưng không có trong set thứ hai.
- **symmetric\_difference(other\_set) hoặc ^:** Trả về một set mới bao gồm các phần tử chỉ có trong một trong hai set.



## 03

Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

### 05 Phương thức trong Set

```
set1 = {'a', 'b', 'c'}
```

```
set2 = {'b', 'c', 'd', 'e'}
```

# Hợp của hai set

```
union_set = set1.union(set2)
```

```
union_set = set1 | set2
```

# Giao của hai set

```
intersection_set = set1.intersection(set2)
```

```
intersection_set = set1 & set2
```



## 03

Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

### 05 Phương thức trong Set

```
set1 = {'a', 'b', 'c'}
```

```
set2 = {'b', 'c', 'd', 'e'}
```

# Hiệu của hai set

```
difference_set = set1.difference(set2)
```

```
difference_set = set1 - set2
```

# Hiệu đối xứng của hai set

```
sym_diff_set =  
set1.symmetric_difference(set2)  
sym_diff_set = set1 ^ set2
```



## 03

Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

### 06 Kiểm tra, lặp qua các phần tử trong Set

```
data = {'Perl', 'Python', 'Java'}
```

- **Kiểm tra phần tử:** Sử dụng toán tử **in**.

```
if ("Perl" in data):  
    print("Perl có trong data")  
else:  
    print("Perl không có trong data")
```

```
# Kết quả trả về: Perl có trong data
```



## 03

Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

### 06 Kiểm tra, lặp qua các phần tử trong Set

```
data = {'Perl', 'Python', 'Java'}
```

▪ **Lặp qua các phần tử:** Sử dụng **for**.

```
for i in data:  
    print(i)
```

Kết quả  
trả về:  
**Perl**  
**Python**  
**Java**



## 03 Kiểu dữ liệu SET

- 01 Set là gì trong Python?
- 02 Một số ưu điểm của Set
- 03 Tạo một Set
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Set
- 06 Kiểm tra, lặp qua các phần tử trong Set

## 07 Bài tập

### Sử dụng các phương thức Set

1. Tạo một set với các phần tử:  
"a", "b", "c", "d".
2. Thêm phần tử "e" vào set.
3. Xóa một phần tử ngẫu nhiên khỏi set và in phần tử đã xóa.
4. Sao chép set sang một set mới và in set mới.
5. Xóa tất cả các phần tử trong set.



## 2.1 Các kiểu dữ liệu trong Python



- 01** Dictionary là gì trong Python?
- 02** Một số ưu điểm của Dictionary
- 03** Tạo một Dictionary
- 04** Truy cập, thay đổi, xóa các phần tử
- 05** Phương thức trong Dictionary
- 06** Kiểm tra, lặp qua các phần tử trong Dictionary



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử trong Dictionary

### 01 Dictionary là gì trong Python?

Dictionary là một cấu trúc dữ liệu mạnh mẽ để lưu trữ các cặp khóa - giá trị (key-value pairs).

Mỗi khóa phải là duy nhất và không thay đổi (immutable), ví dụ như chuỗi, số hoặc tuple.

```
monhoc = {  
    'key1': 'value1',  
    'key2': 'value2',  
    'key3': 'value3'  
}
```



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử trong Dictionary

02

### Một số ưu điểm của Dictionary

- Cho phép lưu trữ dữ liệu theo cặp key-value.
- Tìm kiếm nhanh chóng dựa trên khóa.
- Dễ dàng thêm, sửa, xóa phần tử.
- Là cấu trúc dữ liệu linh hoạt và mạnh mẽ.



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử  
trong Dictionary

### 03 Tạo một Dictionary

Tạo một dictionary bằng cách đặt các cặp key-value trong dấu ngoặc nhọn {} và phân cách chúng bằng dấu phẩy “,”  
Ví dụ:

```
monhoc = {  
    "CS226": "Linux & Unix",  
    "CS252": "Network Computer",  
    "CS466": "Perl & Python"  
}
```



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử  
trong Dictionary

### 04 Truy cập, thay đổi, xóa các phần tử

```
monhoc = {  
    "CS226": "Linux & Unix",  
    "CS252": "Network Computer",  
    "CS466": "Perl & Python  
}
```

- **Truy cập phần tử:** Sử dụng key để truy cập giá trị tương ứng.

```
# Truy cập giá trị của key 'CS466'  
value = monhoc['CS466']
```



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử  
trong Dictionary

### 04 Truy cập, thay đổi, xóa các phần tử

```
monhoc = {  
    "CS226": "Linux & Unix",  
    "CS252": "Network Computer",  
    "CS466": "Perl & Python"  
}
```

- **Thay đổi giá trị:** Gán giá trị mới cho một key đã tồn tại

```
# Thay đổi giá trị của key 'CS226'  
monhoc['CS226'] = 'cs226'
```



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử trong Dictionary

### 04 Truy cập, thay đổi, xóa các phần tử

```
monhoc = {  
    "CS226": "Linux & Unix",  
    "CS252": "Network Computer",  
    "CS466": "Perl & Python"  
}
```

- **Xóa phần tử:** Sử dụng del để xóa một cặp key-value

```
# Xóa cặp key 'CS252' và giá trị tương ứng  
del monhoc['CS252']
```



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử trong Dictionary

### 05 Phương thức trong Dictionary

#### 1. clear()

Xóa tất cả các phần tử khỏi dictionary.

#### 2. copy()

Trả về một bản sao nông (shallow copy) của dictionary.

#### 3. fromkeys(iterable, value=None)

Tạo một dictionary mới với các keys từ một iterable và giá trị mặc định.



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử trong Dictionary

05

## Phương thức trong Dictionary

### 4. get(key, default=None)

Trả về giá trị của key nếu key tồn tại, nếu không thì trả về giá trị mặc định.

### 5. items()

Trả về một đối tượng view với các cặp (key, value) của dictionary.

### 6. keys()

Trả về một đối tượng view với các keys của dictionary.



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử trong Dictionary

### 05 Phương thức trong Dictionary

#### 7. pop(key, default=None)

Xóa key và trả về giá trị tương ứng. Nếu key không tồn tại, trả về giá trị mặc định.

#### 8. popitem()

Xóa và trả về một cặp (key, value) ngẫu nhiên từ dictionary.

#### 9. setdefault(key, default=None)

Nếu key tồn tại, trả về giá trị của key.

Nếu không, chèn key với giá trị mặc định và trả về giá trị mặc định đó.



## 04

Kiểu dữ liệu  
Dictionary

- 01 Dictionary là gì trong Python?
- 02 Một số ưu điểm của Dictionary
- 03 Tạo một Dictionary
- 04 Truy cập, thay đổi, xóa các phần tử
- 05 Phương thức trong Dictionary
- 06 Kiểm tra, lặp qua các phần tử  
trong Dictionary

### 05 Phương thức trong Dictionary

#### 10. update([other])

Cập nhật dictionary với các cặp key-value từ một dictionary khác hoặc từ một iterable của các cặp key-value.

#### 11. values()

Trả về một đối tượng view với các giá trị của dictionary.



## 04

Kiểu dữ liệu  
Dictionary

### Bài tập

```
monhoc = {  
    "CS226": "Linux & Unix",  
    "CS252": "Network Computer",  
    "CS466": "Perl & Python"  
}
```

Hãy cho biết kết quả trả về là gì?

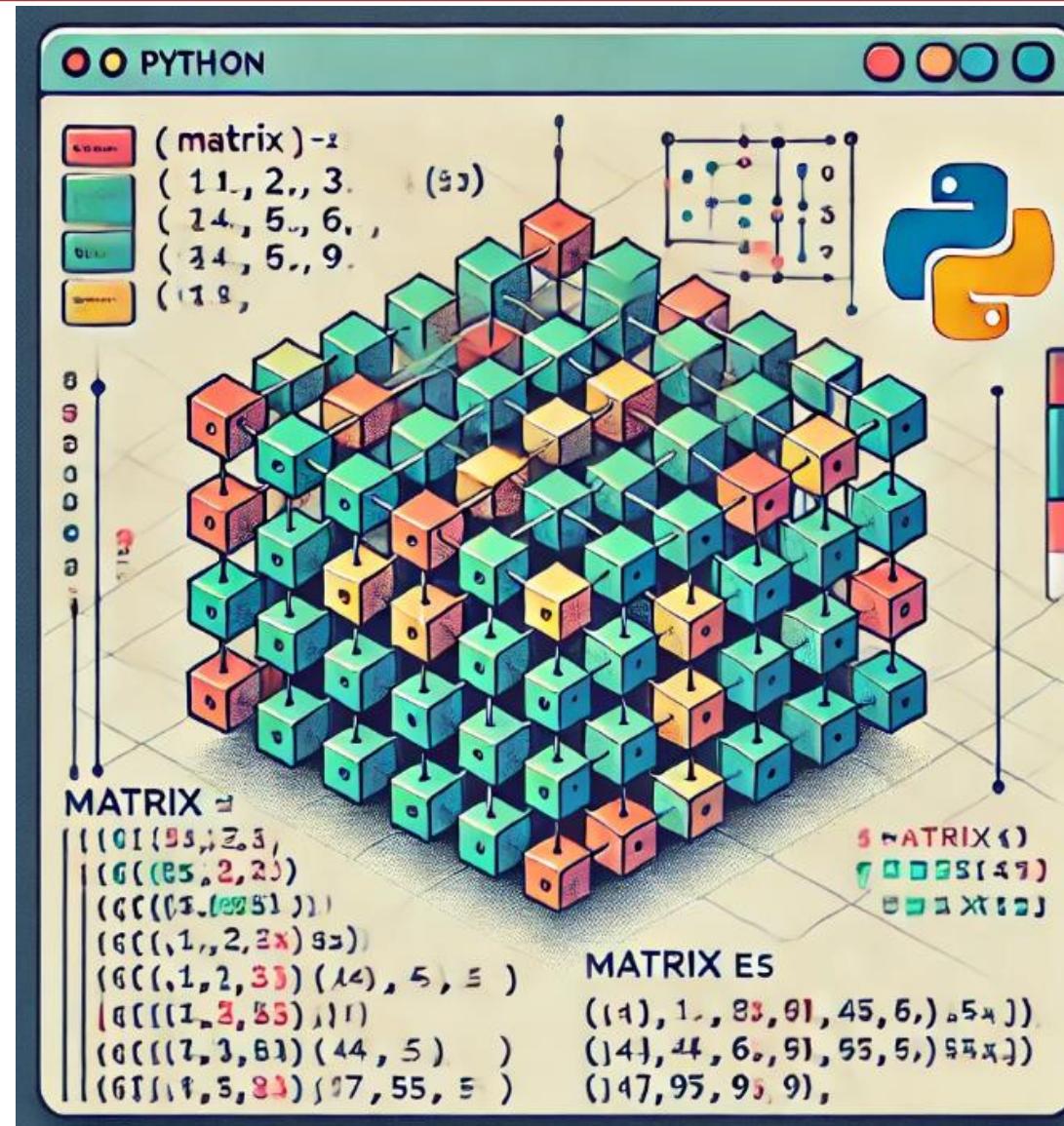
1. `>> print(monhoc.keys())`
2. `>> print(monhoc.values())`
3. `>> print(monhoc.items())`
4. `>> print(monhoc.get('CS201', 0))`
5. `>> monhoc.update({"CS201": "TUD"})  
print(monhoc)`
6. `>> monhoc.pop('CS226')  
print(monhoc)`
7. `>> monhoc.popitem()  
print(monhoc)`
8. `>> monhoc.clear()  
print(my_dict)`



# Matrix in Python

Ma trận là một cấu trúc dữ liệu toán học được sử dụng rộng rãi trong nhiều lĩnh vực, bao gồm xử lý ảnh, khoa học máy tính, và học máy.

Trong Python, chúng ta có thể sử dụng các thư viện như NumPy để tạo, thao tác và thực hiện các phép toán trên ma trận.

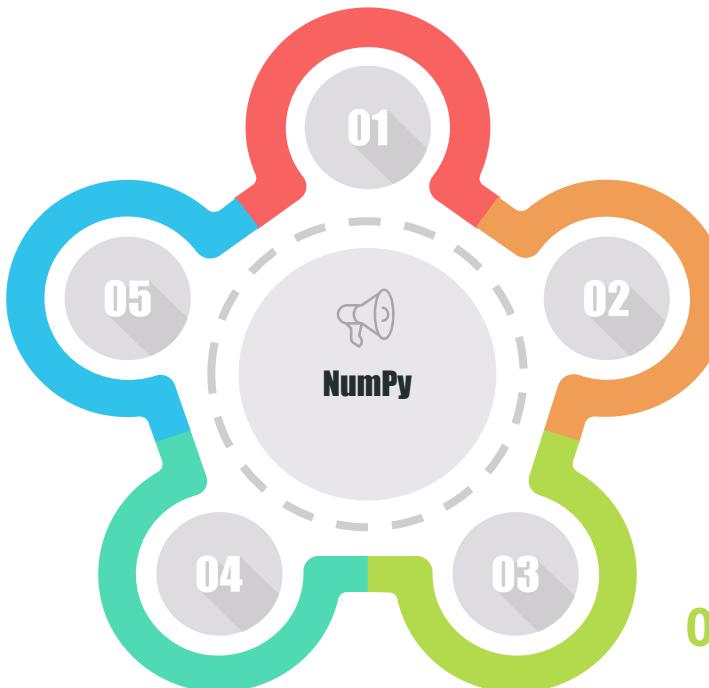




## Matrix in Python

### 05. Ví dụ

```
import numpy as np
A = np.array([[2, 4], [5, -6]])
B = np.array([[9, -3], [3, 6]])
# Phép cộng
Cong = A + B
# Phép nhân
Nhan = A.dot(B)
print(Cong)
Print(Nhan)
```



### 04. Thao tác trên ma trận

- Tạo Ma Trận
- Duyệt Qua Ma Trận
- Phép Cộng Ma Trận
- Nhân Ma Trận

### 01. Nested list

Nested list là dạng danh sách lồng ghép, nghĩa là một list xuất hiện với vai trò là phần tử của một list khác.

### 02. NumPy

NumPy là thư viện được viết bằng Python

### 03. Một số hàm trong NumPy

- np.zeros(x,y)
- np.ones(x,y)
- np.arange(x)
- np.shape()



## Tạo ma trận trong Python

### Sử dụng danh sách lồng nhau

Chúng ta có thể tạo ma trận bằng cách sử dụng danh sách lồng nhau trong Python. Mỗi danh sách con đại diện cho một hàng trong ma trận.

### Sử dụng NumPy

Thư viện NumPy cung cấp chức năng mạnh mẽ để tạo ma trận. Chúng ta có thể sử dụng hàm `np.array()` để chuyển đổi danh sách lồng nhau thành ma trận NumPy.



## Tạo ma trận trong Python

```
import numpy as np
```

```
# Tạo ma trận với numpy
```

```
A = np.array([[1, 2], [3, 4]])
```

```
B = np.array([[5, 6], [7, 8]])
```

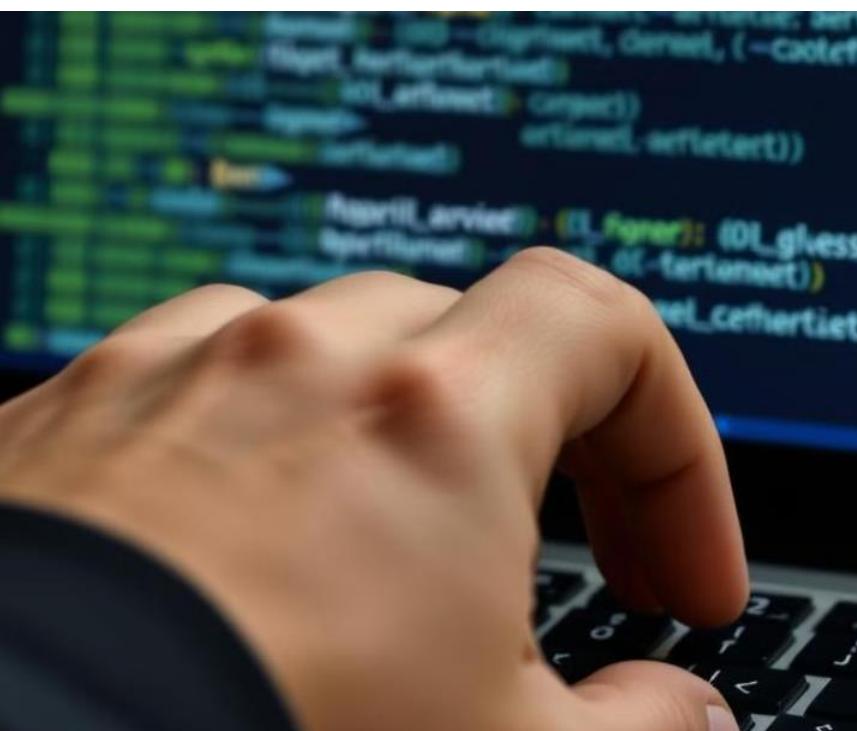
### Sử dụng NumPy

Thư viện NumPy cung cấp chức năng mạnh mẽ để tạo ma trận. Chúng ta có thể sử dụng hàm `np.array()` để chuyển đổi danh sách lồng nhau thành ma trận NumPy.

Lưu ý: cần cài đặt thư viện NumPy  
`pip install numpy`



## Truy cập và thao tác với các phần tử của ma trận



- 1
- 2
- 3

### Truy cập phần tử

Chúng ta có thể truy cập các phần tử trong ma trận bằng cách sử dụng chỉ số hàng và cột.

### Thay đổi giá trị

Chúng ta có thể thay đổi giá trị của các phần tử trong ma trận bằng cách gán giá trị mới cho chỉ số tương ứng.

### Thêm và xóa hàng/cột

Chúng ta có thể thêm hoặc xóa hàng/cột trong ma trận bằng cách sử dụng các phương thức thích hợp.



## Các phép toán cơ bản trên ma trận

### 1 Cộng và trừ

Chúng ta có thể cộng và trừ các ma trận bằng cách cộng/trừ các phần tử tương ứng.

### 2 Nhân ma trận

Phép nhân ma trận được thực hiện bằng cách nhân các phần tử của hàng trong ma trận đầu tiên với các phần tử của cột trong ma trận thứ hai.

### 3 Chia ma trận

Phép chia ma trận được thực hiện bằng cách nhân ma trận bị chia với ma trận nghịch đảo của ma trận chia.





## Ứng dụng ma trận trong xử lý dữ liệu



Phân tích  
dữ liệu

Ma trận cho phép biểu diễn dữ liệu một cách hiệu quả, giúp dễ dàng phân tích và xử lý.

Học máy

Ma trận được sử dụng rộng rãi trong các thuật toán học máy như hồi quy tuyến tính và mạng nơ-ron.

Xử lý ảnh

Ảnh có thể được biểu diễn dưới dạng ma trận, giúp chúng ta thực hiện các thao tác như xoay, phóng to/thu nhỏ, và lọc.



## Ma trận chuyển vị và ứng dụng



### Khái niệm

Ma trận chuyển vị là ma trận thu được bằng cách hoán đổi vị trí các hàng và cột của ma trận ban đầu.

### Ứng dụng

Ma trận chuyển vị được sử dụng trong các phép toán liên quan đến ma trận đối xứng, tính toán ma trận hiệp phương sai, và giải hệ phương trình tuyến tính.



## Ma trận nghịch đảo và ứng dụng

### 1 Khái niệm

Ma trận nghịch đảo của một ma trận vuông là ma trận sao cho tích của chúng là ma trận đơn vị.

### 2 Ứng dụng

Ma trận nghịch đảo được sử dụng để giải hệ phương trình tuyến tính, tính toán định thức của ma trận, và trong các phép toán liên quan đến ma trận vuông.

### 3 Điều kiện tồn tại

Không phải mọi ma trận vuông đều có ma trận nghịch đảo.

Ma trận nghịch đảo chỉ tồn tại khi định thức của ma trận khác 0.



## EXAMPLE

### 1. Tạo Ma Trận với List

```
# Tạo ma trận 3x3
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
```

```
# Truy cập phần tử tại hàng 2, cột 3
print(matrix[1][2]) # Output: 6
```

### 2. Duyệt Qua Ma Trận

```
for row in matrix:
    for element in row:
        print(element, end=" ")
    print()
```



## EXAMPLE

### 3. Phép Cộng Ma Trận

```
# Tạo hai ma trận 2x2
```

```
A = [[1, 2], [3, 4]]
```

```
B = [[5, 6], [7, 8]]
```

```
# Cộng ma trận
```

```
result = [[0, 0], [0, 0]]
```

```
for i in range(len(A)):
```

```
    for j in range(len(A[0])):
```

```
        result[i][j] = A[i][j] + B[i][j]
```

```
print("Kết quả phép cộng ma trận:")
```

```
for r in result:
```

```
    print(r)
```

### 4. Nhân Ma Trận

```
# Ma trận A là 2x3 và B là 3x2
```

```
A = [[1, 2, 3], [4, 5, 6]]
```

```
B = [[7, 8], [9, 10], [11, 12]]
```

```
# Kết quả sẽ là ma trận 2x2
```

```
result = [[0, 0], [0, 0]]
```

```
# Nhân ma trận
```

```
for i in range(len(A)):
```

```
    for j in range(len(B[0])):
```

```
        for k in range(len(B)):
```

```
            result[i][j] += A[i][k] * B[k][j]
```

```
print("Kết quả phép nhân ma trận:")
```

```
for r in result:
```

```
    print(r)
```



## EXAMPLE

### 5. Sử dụng Thư Viện numpy cho Ma Trận

Cài đặt numpy:  
pip install numpy

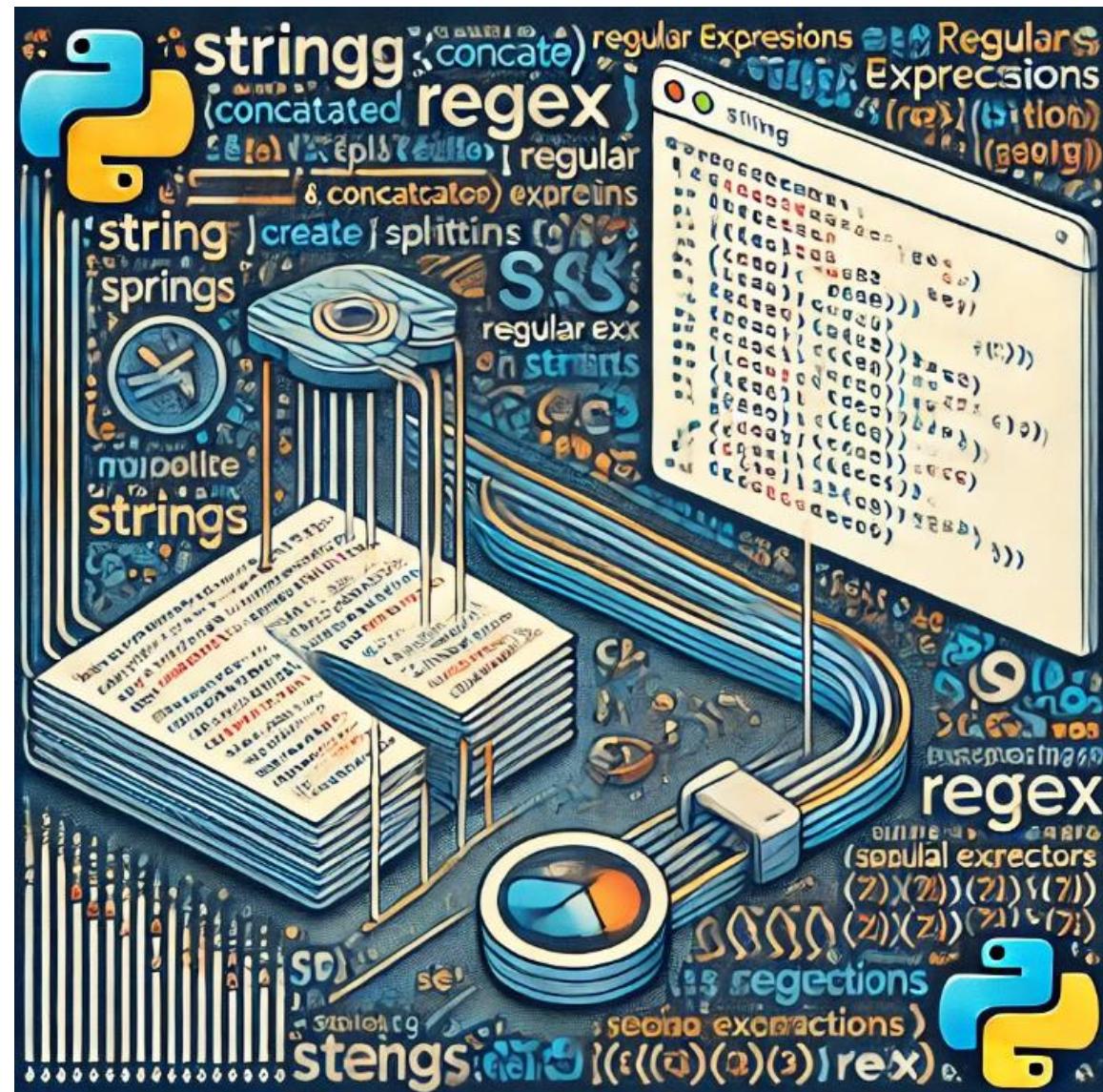
```
import numpy as np
# Tạo ma trận với numpy
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

# Cộng ma trận
print("Cộng ma trận:")
print(A + B)
# Nhân ma trận
print("Nhân ma trận:")
print(np.dot(A, B))
```



## Regular Expressions And Strings

Regular Expressions (RegEx) là một công cụ mạnh mẽ để tìm kiếm, thay thế và xử lý văn bản trong nhiều ngôn ngữ lập trình. Nó sử dụng một ngôn ngữ đặc biệt để mô tả các mẫu văn bản và cho phép bạn tìm kiếm, trích xuất hoặc sửa đổi dữ liệu dựa trên các mẫu đó.





## STRING

### Ưu điểm

Phổ biến và thường được sử dụng khi làm việc với văn bản.

### Ví dụ

```
my_string_1 = 'Hello, world!'  
my_string_2 = "Python is awesome"
```

- 1 Kiểu dữ liệu string trong Python  
Được sử dụng để lưu trữ chuỗi ký tự.
- 2 Cách khai báo biến kiểu string  
Bạn đặt chuỗi ký tự trong dấu nháy đơn hoặc dấu nháy kép.
- 3
- 4



## Element access in STRING

STRING = "AASHINA"

REVERSE INDEX	-7	-6	-5	-4	-3	-2	-1
FORWARD INDEX	0	1	2	3	4	5	6
A	A	S	H	I	N	N	A

Truy xuất phần tử:  
index.

STRING[0] = 'A'  
STRING[1] = 'A'  
STRING[2] = 'S'  
STRING[3] = 'H'  
STRING[4] = 'I'  
STRING[5] = 'N'  
STRING[6] = 'A'

STRING[-7] = 'A'  
STRING[-6] = 'A'  
STRING[-5] = 'S'  
STRING[-4] = 'H'  
STRING[-3] = 'I'  
STRING[-2] = 'N'  
STRING[-1] = 'A'

- ❑ Ví dụ về truy xuất phần tử trong string

my\_string = "Python"

first\_char = my\_string[0] # Lấy ký tự đầu tiên, 'P'

third\_char = my\_string[2] # Lấy ký tự thứ ba, 't'



## Operations on the STRING data type

- Cộng chuỗi,
- Nhân chuỗi và
- Kiểm tra xem một chuỗi có tồn tại.

□ Ví dụ về các phép toán trên kiểu dữ liệu string

```
str1 = "Hello"
```

```
str2 = "World"
```

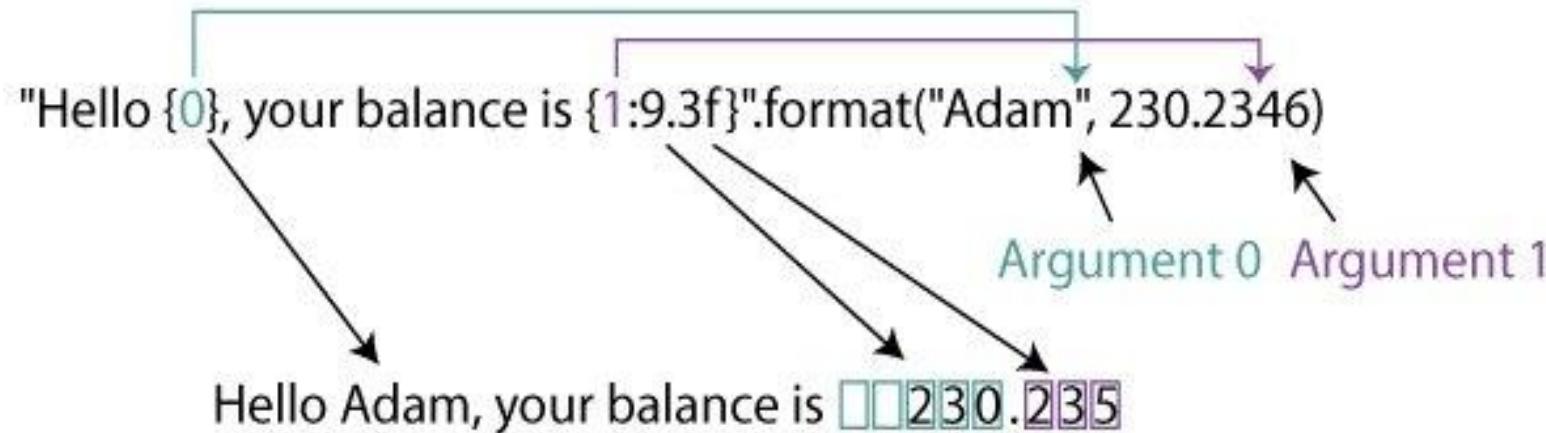
```
Cong_str = str1 + " " + str2      # Cộng chuỗi
```

```
Nhan_str = str1 * 3              # Nhân chuỗi
```

```
check_substring = "lo" in str1 # Kiểm tra chuỗi con
```



## Formatting in string



- Để định dạng chuỗi, sử dụng phương thức **.format()** hoặc **f-strings** (Python 3.6+).
- Phương thức format() cho phép bạn định dạng các phần đã chọn của chuỗi.
- Để kiểm soát các giá trị như vậy, hãy thêm chỗ dành sẵn (dấu ngoặc nhọn {})



## Formatting in string

```
price = 49  
txt = "The price is {} dollars"  
print(txt.format(price))
```

#Result: The price is 49 dollars

Thêm chỗ dành sẵn nơi  
bạn muốn hiển thị giá

Định dạng giá sẽ  
được hiển thị dưới  
dạng số có hai số  
thập phân

```
price = 49  
txt = "The price is {:.2f} dollars"  
print(txt.format(price))
```

#Result: The price is 49.00 dollars



## Formatting in string

- Thêm nhiều giá trị: để sử dụng nhiều giá trị hơn, chỉ cần thêm nhiều giá trị hơn vào phương thức format()



- str1="nội dung"
  - str2="nội dung"
  - str3="nội dung"
  - txt="hiển thị str 1 {} hiển thị nội dung str2 {} và hiển thị nội dung str3 {}"
  - print(txt.format(str1, str2, str3))
- str1 = "CS466 A"
  - str2 = "CS466"
  - str3 = "A"
  - txt = "Mã lớp {} bao gồm: Mã môn {} và Ký hiệu {}."
  - print(txt.format(str1, str2, str3))



## Formatting in string

- Số chỉ mục: Có thể sử dụng chỉ mục để điền giá trị cách linh hoạt và chính xác bằng các {index}



- str1="nội dung"
  - str2="nội dung"
  - str3="nội dung"
  - txt="hiển thị str 1 {0} hiển thị nội dung str2 {1} và hiển thị nội dung str3 {2}"
  - print(txt.format(str1, str2, str3))
- 
- str1 = "CS466 A"
  - str2 = "CS466"
  - str3 = "A"
  - txt = "Mã lớp {0} bao gồm: Mã môn {1} và Ký hiệu {2}."
  - print(txt.format(str1, str2, str3))



## Formatting in string

### Chỉ mục được đặt tên:

- Để sử dụng các chỉ mục đã đặt tên bằng cách nhập tên bên trong dấu ngoặc nhọn {Mamon}, nhưng phải sử dụng tên khi chuyển các giá trị tham số **txt.format(Mamon = “CS466”):**

```
txt = “Mã môn {0}: Tên môn học: Perl & Python
```

```
print(txt.format(Mamon=“CS466”))
```



## Python String Methods

### Python String Methods

capitaliz e  
casifold  
center  
count  
encode  
endswith  
expandtabs  
find  
format  
format\_map  
index

Input	Method	Output
'hello WORLD'	.capitalize()	Hello world
'HELLO WORLD'	.lower()	hello world
'hello world'	.upper()	HELLO WORLD
'Python'	.center(10, '*')	**Python**
'HELLO WORLD'	.count('L')	3
'HELLO WORLD'	.index('O')	4
'HELLO WORLD'	.find('OR')	7
'31/01/2022'	.replace('/', '-')	31-01-2022
'31/01/2022'	.split('/')	['31', '01', '2022']
'abc123'	.isalnum()	True
'12345'	.isnumeric()	True
'hello world'	.islower()	True
'HELLO WORLD'	.isupper()	True



## Phương thức (String) chuỗi trong Python

### 1. capitalize()

*Viết hoa ký tự đầu tiên của chuỗi.*

### 2. casefold()

*Chuyển đổi chuỗi thành chữ thường, hữu ích cho việc so sánh chuỗi không phân biệt chữ hoa chữ thường.*

### 3. center(width, fillchar= )

*Căn giữa chuỗi trong một chuỗi rộng hơn.*

### 4. count(sub, start=0, end=len(string))

*Đếm số lần xuất hiện của một chuỗi con trong chuỗi.*

### 5. encode(encoding='utf-8', errors='strict')

*Mã hóa chuỗi thành bytes.*



## Phương thức (String) chuỗi trong Python

### 6. `endswith(suffix, start=0, end=len(string))`

*Kiểm tra xem chuỗi có kết thúc bằng một chuỗi con cụ thể hay không.*

### 7. `expandtabs(tabsize=8)`

*Thay thế các ký tự tab bằng khoảng trắng.*

### 8. `find(sub, start=0, end=len(string))`

*Trả về vị trí đầu tiên của chuỗi con, nếu không tìm thấy trả về -1.*

### 9. `format(*args, **kwargs)`

*Định dạng chuỗi sử dụng các đối số.*

### 10. `index(sub, start=0, end=len(string))`

*Tương tự như `find()` nhưng sẽ gây lỗi nếu không tìm thấy.*



## Phương thức (String) chuỗi trong Python

### 11. isalnum()

*Kiểm tra xem chuỗi có phải chỉ chứa chữ và số hay không.*

### 12. isalpha()

*Kiểm tra xem chuỗi có phải chỉ chứa chữ cái hay không.*

### 13. isdigit()

*Kiểm tra xem chuỗi có phải chỉ chứa chữ số hay không.*

### 14. islower()

*Kiểm tra xem tất cả các ký tự trong chuỗi có phải là chữ thường hay không.*

### 15. isspace()

*Kiểm tra xem chuỗi có phải chỉ chứa khoảng trắng hay không.*



## Phương thức (String) chuỗi trong Python

### 16. **istitle()**

*Kiểm tra xem chuỗi có phải là tiêu đề (mỗi từ viết hoa chữ cái đầu tiên) hay không.*

### 17. **isupper()**

*Kiểm tra xem tất cả các ký tự trong chuỗi có phải là chữ hoa hay không.*

### 18. **join(iterable)**

*Tham gia các phần tử của iterable thành một chuỗi, sử dụng chuỗi hiện tại làm dấu phân cách.*

### 19. **ljust(width, fillchar= )**

*Căn trái chuỗi trong một chuỗi rộng hơn.*

### 20. **lower()**

*Chuyển đổi chuỗi thành chữ thường.*



## Phương thức (String) chuỗi trong Python

### 21. **lstrip(chars=None)**

Xóa các ký tự từ đầu chuỗi.

### 22. **partition(sep)**

Chia chuỗi thành ba phần: phần trước separator, separator, và phần sau separator.

### 23. **replace(old, new, count=-1)**

Thay thế các chuỗi con bằng một chuỗi con khác.

### 24. **rfind(sub, start=0, end=len(string))**

Tìm kiếm chuỗi con từ phải sang trái và trả về vị trí xuất hiện cuối cùng, nếu không tìm thấy trả về -1.

### 25. **rindex(sub, start=0, end=len(string))**

Tương tự như rfind() nhưng sẽ gây lỗi nếu không tìm thấy.



## Phương thức (String) chuỗi trong Python

### 26. rjust(width, fillchar= )

*Căn phải chuỗi trong một chuỗi rộng hơn.*

### 27. rpartition(sep)

*Tương tự như partition() nhưng tìm kiếm từ phải sang trái.*

### 28. rsplit(sep=None, maxsplit=-1)

*Chia chuỗi từ phải sang trái.*

### 29. rstrip(chars=None)

*Xóa các ký tự từ cuối chuỗi.*

### 30. split(sep=None, maxsplit=-1)

*Chia chuỗi thành danh sách các chuỗi con.*

### 31. splitlines(keepends=False)

*Chia chuỗi thành danh sách các dòng.*



## Phương thức (String) chuỗi trong Python

### 32. **startswith(prefix, start=0, end=len(string))**

*Kiểm tra xem chuỗi có bắt đầu bằng một chuỗi con cụ thể hay không.*

### 33. **strip(chars=None)**

*Xóa các ký tự từ đầu và cuối chuỗi.*

### 34. **swapcase()**

*Đổi chữ hoa thành chữ thường và ngược lại.*

### 35. **title()**

*Chuyển chuỗi thành tiêu đề (mỗi từ viết hoa chữ cái đầu tiên).*

### 36. **upper()**

*Chuyển chuỗi thành chữ hoa.*

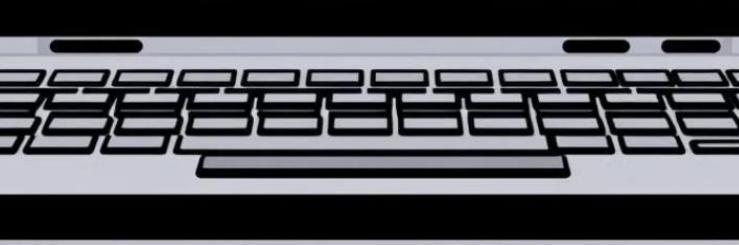
### 37. **zfill(width)**

*Thêm các ký tự 0 vào đầu chuỗi để đạt được độ dài cụ thể.*



## Cú pháp cơ bản của Regular Expressions

Ditelfg rrcerpen tist:  
Lesticaletlme.(! ? , lef )  
Lestting the irterpen is, (t" "s, ".., ":-  
vetfel that lasttetlnns,  
thatls whidking sectphorrrh.is,(ty")  
teatclet libslle, trketmm,(t;  
Lprooltl; the thy docift;  
tastife (s, ttstF,  
epfectfilt, teriflfipat(t );  
tasder



### Ký tự thông thường

Các ký tự thông thường khớp với chính chúng trong văn bản. Ví dụ, "a" khớp với chữ "a" trong văn bản.

### Ký tự đặc biệt

Các ký tự đặc biệt có ý nghĩa đặc biệt trong RegEx. Ví dụ, "." khớp với bất kỳ ký tự nào.

### Biểu thức nhóm

Biểu thức nhóm được đặt trong dấu ngoặc đơn "(...)" và cho phép bạn nhóm các phần tử cùng nhau để khớp với chúng như một đơn vị.

### Biểu thức lặp lại

Biểu thức lặp lại xác định số lần lặp lại của một mẫu. Ví dụ, "\*" khớp với 0 hoặc nhiều lần lặp lại.

## 1 Ký tự chấm "."

Khớp với bất kỳ ký  
tự nào trừ ký tự  
xuống dòng.

## 3 Ký tự dấu gạch ngang "-"

Sử dụng trong dấu ngoặc vuông để chỉ ra một phạm vi ký tự. Ví dụ, "a-z" khớp với bất kỳ chữ cái thường nào từ "a" đến "z".

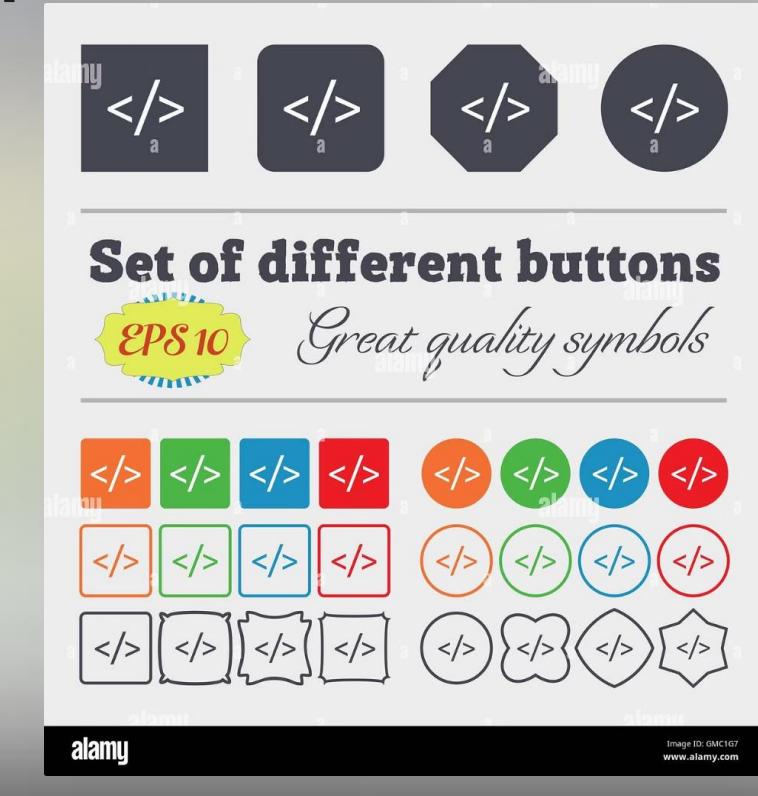
## 2 Ký tự dấu ngoặc vuông "[]"

Khớp với bất kỳ ký tự  
nào trong tập hợp  
được định nghĩa trong  
dấu ngoặc vuông.

## 4 Ký tự dấu mũ "<sup>^</sup>"

Trong dấu ngoặc vuông, nó phủ định tập hợp ký tự. Ví dụ, "[^a-z]" khớp với bất kỳ ký tự nào không phải chữ cái thường.

# Các loại ký tự đặc biệt trong Regular Expressions





## Tìm kiếm và thay thế văn bản bằng Regular Expressions

### Tìm kiếm

Bạn có thể sử dụng RegEx để tìm kiếm các mẫu văn bản cụ thể trong một chuỗi.

1. Các số điện thoại.
2. Các địa chỉ email.
3. Các thẻ HTML.

### Thay thế

Bạn có thể sử dụng RegEx để thay thế các mẫu văn bản đã tìm thấy bằng văn bản mới.

1. các số điện thoại bằng dấu hoa thị.
2. các địa chỉ email bằng "\*\*\*\*@\*\*\*\*".
3. các thẻ HTML bằng văn bản thuần túy.

### Xử lý chuỗi

Bạn có thể sử dụng RegEx để trích xuất thông tin từ văn bản.

1. Trích xuất số điện thoại từ một văn bản.
2. Trích xuất địa chỉ email từ một văn bản.
3. Trích xuất tiêu đề từ một trang web.



## Ứng dụng của Regular Expressions trong Python



- 1 **Kiểm tra khớp**  
Sử dụng hàm `re.match()` hoặc `re.search()` để kiểm tra xem một chuỗi có khớp với một biểu thức chính quy hay không.
- 2 **Tìm kiếm và thay thế**  
Sử dụng hàm `re.sub()` để thay thế tất cả các khớp tìm thấy trong một chuỗi bằng một chuỗi khác.
- 3 **Trích xuất dữ liệu**  
Sử dụng hàm `re.findall()` để trích xuất tất cả các khớp tìm thấy trong một chuỗi thành một danh sách.
- 4 **Phân tích cú pháp**  
Sử dụng các hàm `re.split()` hoặc `re.compile()` để chia tách chuỗi theo một mẫu cụ thể.



## Phân tích và xử lý chuỗi ký tự bằng Regular Expressions

### Phân tích cú pháp

RegEx cho phép bạn xác định các cấu trúc và mẫu trong văn bản, giúp bạn phân tích cú pháp và hiểu ý nghĩa của dữ liệu.

### Lọc dữ liệu

Bạn có thể sử dụng RegEx để loại bỏ các phần không mong muốn hoặc trích xuất các phần có liên quan từ một chuỗi văn bản.

### Chuyển đổi dữ liệu

RegEx cung cấp các công cụ để thay đổi định dạng, cấu trúc hoặc nội dung của chuỗi văn bản theo ý muốn.

### Kiểm tra dữ liệu

RegEx là một công cụ hiệu quả để xác thực dữ liệu, đảm bảo rằng dữ liệu đáp ứng các yêu cầu mong đợi.



## FILE IN PYTHON

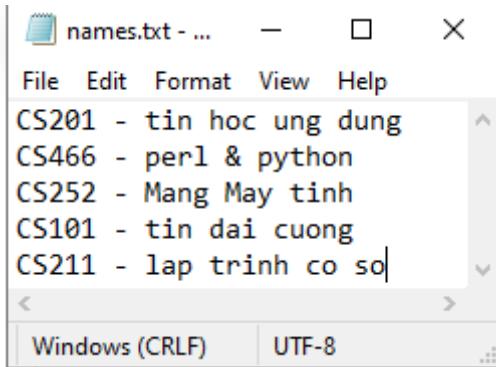


The slide features a central illustration of a person sitting at a desk, viewed from behind, working on a computer. There are four monitors on the desk, each displaying a different screen, possibly representing different windows or applications. The background is light gray with some subtle patterns.

- 01 What is a File?
- 02 Opening a File
- 03 Access Modes
- 04 Closing a File
- 05 Reading data from a file
- 06 Writing data into a file
- 07 File pointer position
- 08 Python File Methods



## Working with Files



`open(<file>, <mode>)`

Relative or absolute path to the file (including the extension).

A string (character) that indicates what you want to do with the file.

### Example:

code	5/1/2020 10:57 AM	Python File	1 KB
names	5/1/2020 10:57 AM	Text Document	1 KB

→ `Open("names.txt") # Đường dẫn tương đối là "name.txt"`



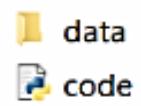
## Working with Files

```
open(<file>, <mode>)
```

Relative or absolute path to the file (including the extension).

A string (character) that indicates what you want to do with the file.

Example: Tệp "name.txt" nằm trong thư mục dữ liệu



←

5/1/2020 11:00 AM

File folder

5/1/2020 10:57 AM

Python File

1 KB



```
Open("data/names.txt")
```



## How to Read a File

```
<var> = open(<file>, "r")
```

Variable that will  
store the file object

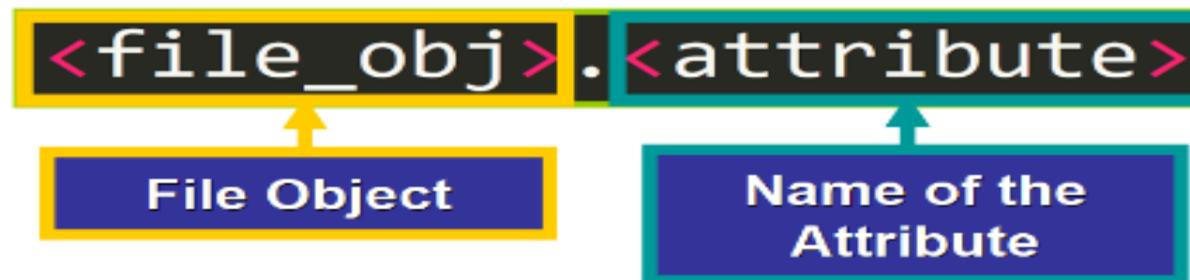
- Đọc ( "r" ).
- Nối ( "a" )
- Viết ( "w" )
- Tạo nén ( "x" )

Example: Mở tệp Tên và đọc

```
names_file = open("data/names.txt", "r")
```



## File Objects



File objects have attributes, such as:

- name: the name of the file.
- closed: True if the file is closed. False otherwise.
- mode: the mode used to open the file

### For Example

```
f = open("data/names.txt", "a")
Print(f.mode)
# output: "a"
```



## Methods to Read a File

### **Readline()**

Readline() đọc một dòng của tệp cho đến khi nó đến cuối dòng đó.

#### **Read()**

Phương thức đầu tiên mà bạn cần tìm hiểu là read(), trả về toàn bộ nội dung của tệp dưới dạng chuỗi.

#### **Readlines()**

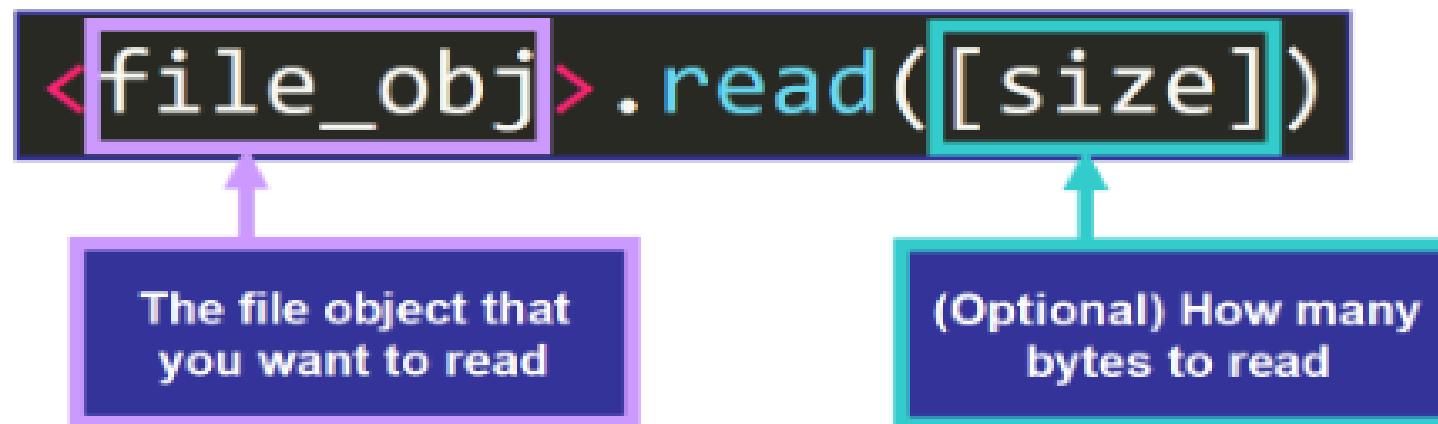
Readlines() trả về một danh sách với tất cả các dòng của tệp dưới dạng các phần tử (chuỗi) riêng lẻ.



## Methods to Read a File

- **Read()**

Phương thức đầu tiên mà bạn cần tìm hiểu là read(), trả về toàn bộ nội dung của tệp dưới dạng chuỗi.





## Methods to Read a File



### Read()

#### For Example:

```
f = open("data/names.txt")
Print(f.read())
```

#### # Ouput

```
CS201 - tin hoc ung dung
CS466 - perl & python
CS252 - Mang May tinh
CS101 - tin dai cuong
CS211 - lap trinh co so
```

# input

The screenshot shows a Windows Notepad window titled "names.txt - ...". The window contains the following text:  
CS201 - tin hoc ung dung  
CS466 - perl & python  
CS252 - Mang May tinh  
CS101 - tin dai cuong  
CS211 - lap trinh co so

```
f = open("data/names.txt")
Print(f.read(5))
```

#### # Ouput

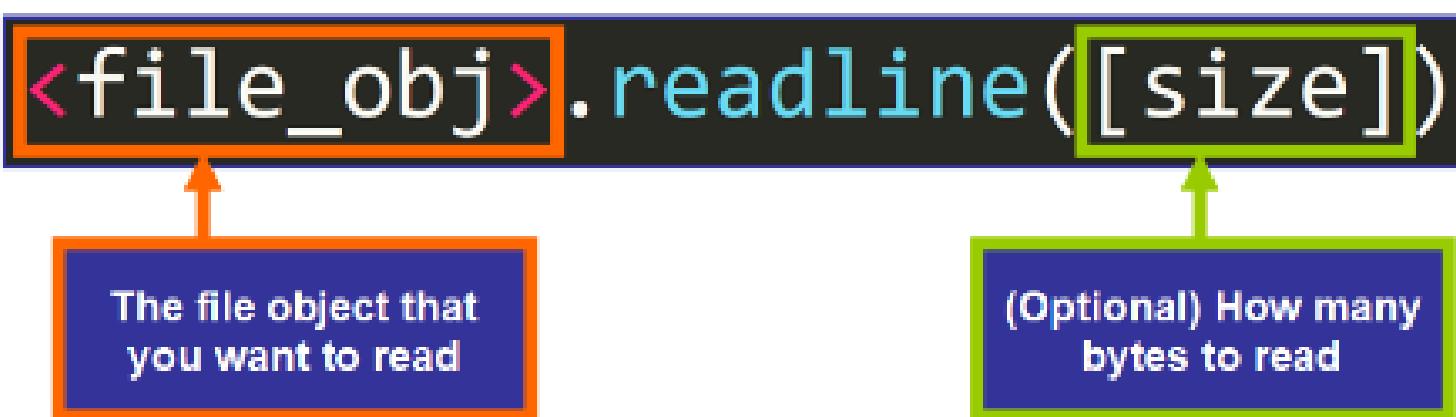
```
CS201
```



## Methods to Read a File

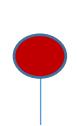
- **Readline()**

Readline() đọc một dòng của tệp cho đến khi nó đến cuối dòng đó.





## Methods to Read a File



### Readline()

For Example:

```
f = open("data/names.txt")
Print(f.readline())
f.close() # View Slide
```

# Ouput

CS201 - tin hoc ung dung

# input

The screenshot shows a Windows Notepad window titled "names.txt - ...". The window contains the following text:  
CS201 - tin hoc ung dung  
CS466 - perl & python  
CS252 - Mang May tinh  
CS101 - tin dai cuong  
CS211 - lap trinh co so

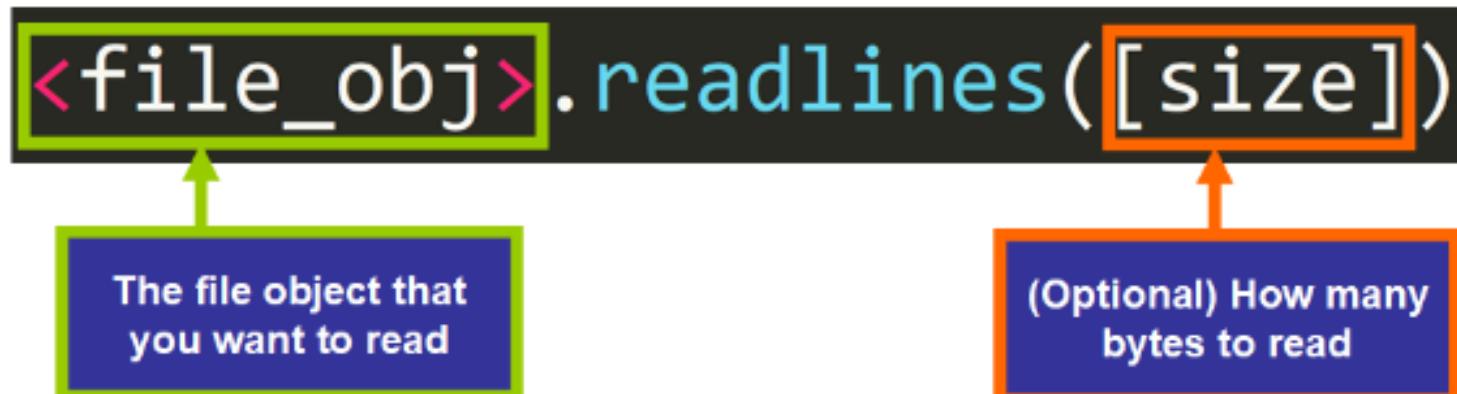
At the bottom of the window, there are two buttons: "Windows (CRLF)" and "UTF-8".



## Methods to Read a File

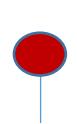
- **Readlines()**

Readlines() trả về một danh sách với tất cả các dòng của tệp dưới dạng các phần tử (chuỗi) riêng lẻ. Lưu ý: \n





## Methods to Read a File



### Readlines()

For Example:

```
f = open("data/names.txt")
Print(f.readlines())
f.close() # View Slide
```

# Ouput

```
['CS201 - tin hoc ung dung\n', 'CS466 - perl & python\n',
'CS252 - Mang May tinh\n', 'CS101 - tin dai cuong\n',
'CS211 - lap trinh co so']
```

# input

The screenshot shows a Windows Notepad window titled "names.txt - ...". The window contains the following text:

```
File Edit Format View Help
CS201 - tin hoc ung dung
CS466 - perl & python
CS252 - Mang May tinh
CS101 - tin dai cuong
CS211 - lap trinh co so
```

At the bottom of the window, there are two encoding options: "Windows (CRLF)" and "UTF-8".



## Methods to Read a File

- **Close()**

**! Important:** Bạn cần đóng tệp sau khi hoàn thành tác vụ để giải phóng các tài nguyên liên quan đến tệp. Để làm điều này, bạn cần gọi phương thức close(), như thế này:

The diagram shows the Python code `<file_obj>.close()` enclosed in a black box with a yellow border. An orange arrow points from a blue box below it to the `<file_obj>` placeholder. The blue box contains the text **The file object that you want to close**.

```
f = open("data/names.txt")
Print(f.readlines())
f.close()
```



**Tip:** Bạn có thể nhận được cùng một danh sách với list(f).

Use: **For loop**

## Ví dụ 1

```
f = open("data/names.txt")
for line in f.readlines():
    # Do something with each line
f.close()
```

## Ví dụ 2

```
f = open("data/names.txt", "r")
for line in f:
    # Do something with each line
f.close()
```



## How to Create a File

- **New File:**

```
<var> = open("<file_name>.<extension>", "x")
```

Create the file

For Example

```
f = open("new_file.txt", "x")
```



## How to Create a File

### Append

"Nối thêm" có nghĩa là thêm một cái gì đó vào cuối một thứ khác. Chế độ "a" cho phép bạn mở một tệp để thêm một số nội dung vào đó.



### Write

Đôi khi, bạn có thể muốn xóa nội dung của tệp và thay thế hoàn toàn bằng nội dung mới. Bạn có thể làm điều này với phương thức write() nếu bạn mở tệp bằng chế độ "w".



## How to Modify a File

### Append

"Nối thêm" có nghĩa là thêm một cái gì đó vào cuối một thứ khác. Chế độ "a" cho phép bạn mở một tệp để thêm một số nội dung vào nó.

names.txt

```
CS201 - tin hoc ung dung  
CS466 - perl & python  
CS252 - Mang May tinh  
CS101 - tin dai cuong  
CS211 - lap trinh co so
```



```
<file_obj>.write(<string>)
```

Content that you want  
to append (String)

### For example

```
f = open("data/names.txt", "a")  
f.write("\nCS226")  
f.close()
```

Result

names.txt

```
CS201 - tin hoc ung dung  
CS466 - perl & python  
CS252 - Mang May tinh  
CS101 - tin dai cuong  
CS211 - lap trinh co so  
CS226
```



## How to Modify a File

Ví dụ:

```
f = open("data/names.txt", "w")
f.write("CS226")
f.close()
```

Result

names.txt

CS266



### Write

Đôi khi, bạn có thể muốn xóa nội dung của tệp và thay thế hoàn toàn bằng nội dung mới. Bạn có thể làm điều này với phương thức write() nếu bạn mở tệp bằng chế độ "w".

names.txt

CS201 - tin hoc ung dung  
CS466 - perl & python  
CS252 - Mang May tinh  
CS101 - tin dai cuong  
CS211 - lap trinh co so



## FILE IN PYTHON

### ❖ How to Modify a File

Ví dụ:

```
f = open("data/names.txt", "w")
f.writelines(["\nCS201","\nCS466"])
f.close()
```

Result

names.txt

CS266
CS201
CS466



### Writelines()

Nếu bạn muốn viết nhiều dòng cùng một lúc, bạn có thể sử dụng phương thức writelines(), phương thức này có một danh sách các chuỗi. Mỗi chuỗi đại diện cho một dòng sẽ được thêm vào tệp.

names.txt

CS266
-------



## Open File For Multiple Operations

error

Nếu bạn mở một tệp ở chế độ "r" (đọc), sau đó thử ghi vào tệp đó:

```
f = open("data/names.txt")
f.write("CS266")
f.close()
```

Bạn sẽ gặp lỗi này

```
Traceback (most recent call last):
  File "<path>", line 9, in <module>
    f.write("C266")
io.UnsupportedOperation: not writable
```

Tương tự, nếu bạn mở một tệp ở chế độ "w" (ghi), sau đó thử đọc nó:

```
f = open("data/names.txt", "w")
print(f.readlines())
f.write("CS266")
f.close()
```

error

```
Traceback (most recent call last):
  File "<path>", line 14, in <module>
    print(f.readlines())
io.UnsupportedOperation: not readable
```



## Open File For Multiple Operations

Làm thế nào chúng ta có thể giải quyết vấn đề này? Để có thể đọc tệp và thực hiện một thao tác khác trong cùng một chương trình, bạn cần thêm biểu tượng "+" vào chế độ, như thế này:



```
f = open("data/names.txt", "w+") # Read + Write  
f = open("data/names.txt", "a+") # Read + Append  
f = open("data/names.txt", "r+") # Read + Write
```

You will get this error

```
f = open("data/names.txt")  
f.write("CS266")  
f.close()
```

```
Traceback (most recent call last):  
File "<path>", line 9, in <module>  
    f.write("C266")  
io.UnsupportedOperation: not writable
```

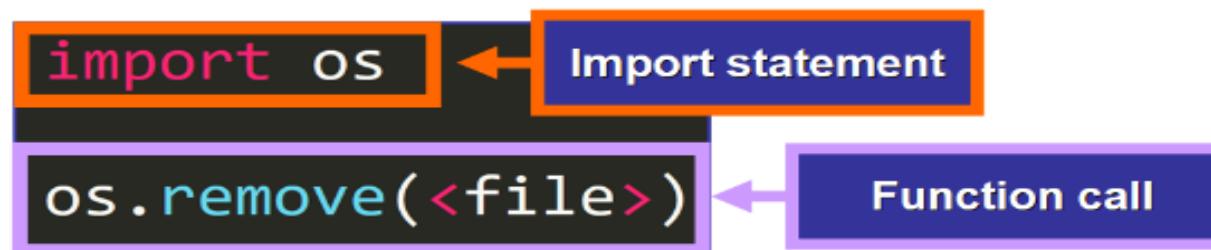
```
f = open("data/names.txt", "w")  
print(f.readlines())  
f.write("CS266")  
f.close()
```

```
Traceback (most recent call last):  
File "<path>", line 14, in <module>  
    print(f.readlines())  
io.UnsupportedOperation: not readable
```



## How to Delete File

Để xóa tệp bằng Python, bạn cần nhập một mô-đun gọi là hệ điều hành có chứa các chức năng tương tác với hệ điều hành của bạn.



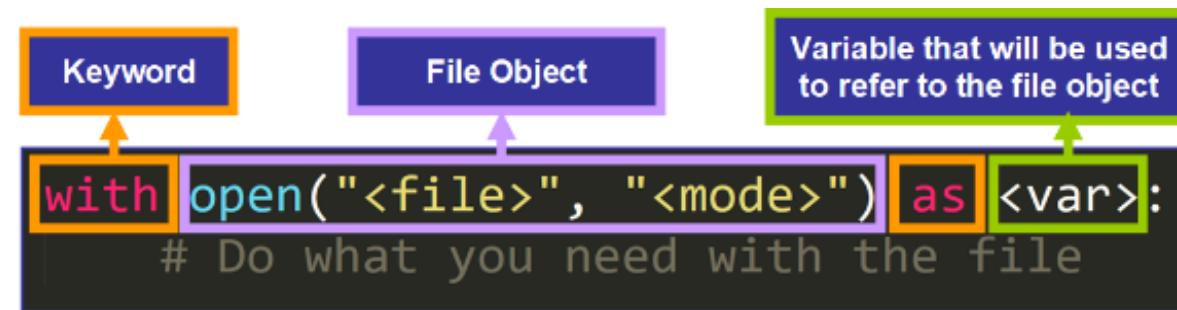
- **Ví dụ:**

```
import os
os.remove("sample_file.txt")
```



## Meet Context Managers

Trình quản lý ngữ cảnh là các cấu trúc Python sẽ giúp cuộc sống của bạn dễ dàng hơn nhiều. Bằng cách sử dụng chúng, bạn không cần phải nhớ đóng tệp ở cuối chương trình.



```
with open("<path>", "<mode>") as <var>:  
    # Working with the file...  
    # The file is closed here!
```



## Meet Context Managers

```
with open("<path>", "<mode>") as <var>:  
    # Working with the file...  
    # The file is closed here!
```

Ví dụ:

```
with open("data/names.txt", "r+") as f:  
    print(f.readlines())
```



# Q&A

## Data Structures in Python



## Conclusion

... ftte attel  
atteel werituarrenn.

**Câu hỏi thảo luận 1:**

**Cách sử dụng các cấu trúc dữ liệu trong Python để xây dựng hệ thống mạng?**

Bạn sẽ sử dụng cấu trúc dữ liệu nào để lưu trữ thông tin về các thiết bị mạng (routers, switches, servers), cũng như thông tin về các kết nối giữa chúng?





## Conclusion

... ftte attel  
atteel werituarrenn.

### Câu hỏi thảo luận 2:

Làm thế nào để sử dụng Regular Expressions trong việc kiểm tra và xác thực các cấu hình mạng trên router hoặc server?





## References

- [1] Lê Văn Cường, *Python cơ bản và ứng dụng*, NXB Đại học Quốc gia Hà Nội, 2019
- [2] Randal L. Schwartz, Tom Phoenix & Brian D Foy (2008). *Learning Perl, 5<sup>th</sup> Edition*. O'Reilly Media.
- [3] Mark Lutz (2013). *Learning Python, 5<sup>th</sup> Edition*. O'Reilly Media. Stein,
- [4] Lincoln D. (2001). *Network Programming with Perl*. Addison-Wesley.
- [5] Rhodes, Brandon & Goerzen, John (2014). *Foundations of Python Network Programming, 3rd Edition*. U.S.: Apress.

## Website & YouTube

[6] [www.python.org](http://www.python.org)

[7] [www.perl.org](http://www.perl.org)

[8] <https://www.youtube.com/c/TheNetNinja/playlists>

[9] <https://www.youtube.com/watch?v=T11QYVfZoD0>

[10] [https://www.youtube.com/watch?v=xbT7Pvh\\_5LQ](https://www.youtube.com/watch?v=xbT7Pvh_5LQ)



# THANK YOU!