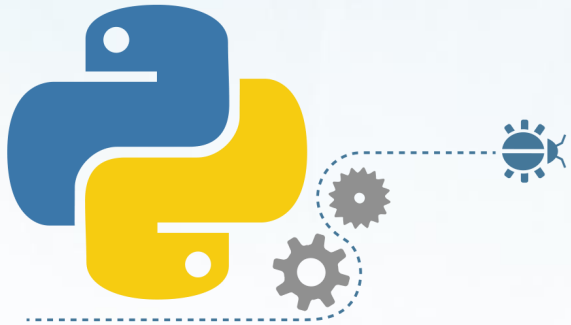
TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education

KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN HỆ THỐNG THÔNG TIN

NHẬP MÔN LẬP TRÌNH PYTHON (IPPA233277)

CÁC KIỂU DỮ LIỆU PHỨC HỢP (List)



GV. Trần Quang Khải

1. Hiểu và nắm được kiểu danh sách (list)
2. Vận dụng các thao tác trên kiểu danh sách
3. Nắm và vận dụng được khái niệm danh sách đa chiều

1. Collection
2. Cách khai báo và sử dụng List
3. Duyệt List
4. Gán giá trị cho các phần tử trong List
5. Phương thức thêm đối tượng (insert, append, extend)
6. Phương thức xóa đối tượng (remove, pop, clear, del)
7. Phương thức xử lý list (reverse, sort, copy, join)
8. List đa chiều

- Một collection cho phép đặt nhiều giá trị vào một "biến" duy nhất
- Dùng để lưu trữ các tập dữ liệu. Trong python có 04 loại dữ liệu dạng tập hợp:
  - **List** : tập các giá trị có tính thứ tự và có thể thay đổi, phần tử có thể trùng lặp về giá trị
  - **Tuple** : tập các giá trị có tính thứ tự và không thể thay đổi, phần tử có thể trùng lặp về giá trị
  - **Set** : tập các giá trị không có tính thứ tự và không thay đổi *, không chỉ mục, phần tử không trùng lặp
  - **Dictionary** : tập các giá trị có tính thứ tự ** và không thể thay đổi, phần tử không trùng lặp về giá trị

* - Thiết lập không thể thay đổi nhưng có thể xóa hoặc thêm phần tử

** - Kể từ phiên bản Python 3.7, dictionaries có thể được sắp xếp, còn các phiên bản trước thì không có tính thứ tự

Ví dụ:

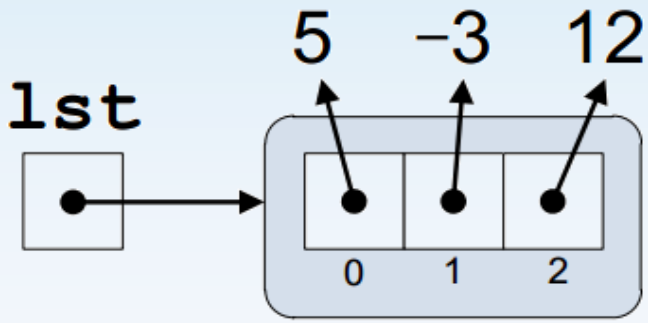
```
friends = ['Tí', 'Tèo', 'Tũn', 'Tĩn']
locations = ['Tân An', 'Thủ Thừa', 'Kiến Tường', 'Bến Lức', 'Đức Hòa']
```


- List trong python là một đối tượng dùng để lưu trữ các đối tượng khác
- List có thể chứa bất kỳ một kiểu dữ liệu nào, tuy nhiên nên thống nhất một kiểu để dễ dàng trong quá trình xử lý

✓ **Khai báo list rỗng**	`lst = []`
✓ **Khai báo list có các giá trị**	`lst = [2, 5, -1, 5, 8]`
✓ **Khai báo list có 10 phần tử với giá trị mặc định là 0.5**	`lst = [0.5] * 10`
✓ **Lấy số lượng phần tử của list**	`len(list)`
✓ **Kiểm tra phần tử tồn tại trong list sử dụng từ khóa "in"**	`if value in list:`

Ví dụ: Cho `lst = [5, -3 , 12]`

```
print(lst[0])
print(lst)
print(len(lst))
```



- Với Python, ta có thể duyệt list theo nhiều cách:
  - ✓ Duyệt theo collection
  - ✓ Duyệt theo chỉ số index

```
lst = [5, 7, 2, 9, 6, 3, 10, 17, 16]
```

- Phần tử đầu tiên có chỉ số index = 0 `# lst[1] = 7`
- Chỉ số âm bắt đầu từ cuối list `# lst[-1] = 16`
- Lấy danh sách các phần tử liên tục **[[start]:[end][:step]]** lấy từ start → end - 1 với bước nhảy step

```
# lst[2:5] = …
# lst[ :5] = …
# lst[2: ] = …
# lst[ : ] = …
```

```
# lst[-4:  ] = …
# lst[-4:  ] = …
# lst[  :-1] = …
# lst[-4:-1] = …
```

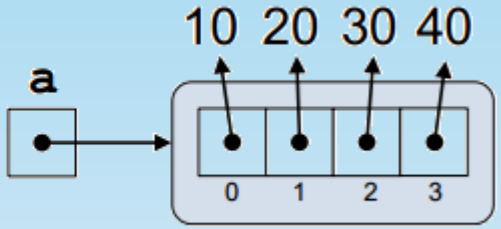
Ví dụ:

```
print('*' * 35)
lst = [5, 7, 2, 9, 6, 3, 10, 17, 16]
for x in lst:
    print(x, end = '\t')
print()
print('*' * 35)
for i in range(len(lst)):
    x = lst[i]
    print(x, end = '\t')
print()
print('*' * 35)
for i in range(len(lst) - 1, -1, -1):
    x = lst[i]
    print(x, end = '\t')
```

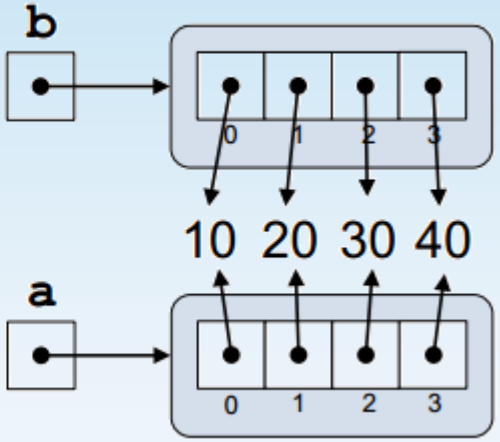
```
***********************************
5	7	2	9	6	3	10	17	16
***********************************
5	7	2	9	6	3	10	17	16
***********************************
16	17	10	3	6	9	2	7	5
```

- `lst = [2, 4, 6, 8]` tham chiếu tới List
- `lst[2]` tham chiếu tới phần tử thứ 2 có vị trí thứ 3 trong list (giá trị 6)

```
a = [10, 20, 30, 40]
```




```
b = [10, 20, 30, 40]
```



`a[0]` → ?

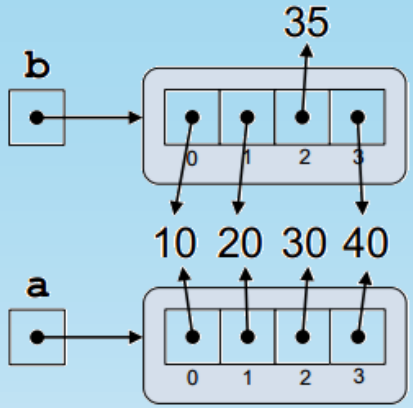
`b[3]` → ?

`a[4]` → ?

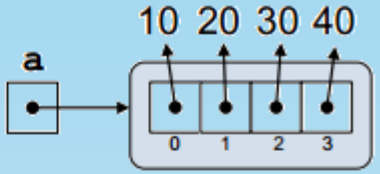
```
b[2] = 35
```

a[2] → ?

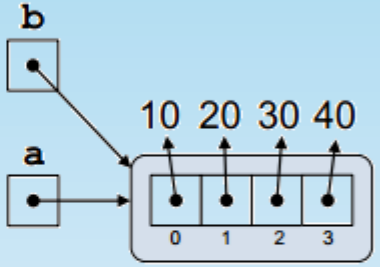
b[2] → ?



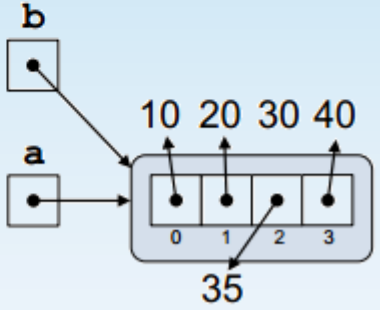

```
a = [10, 20, 30, 40]
```




```
b = a
```




```
b[2] = 35
```



a[2] → ?

b[2] → ?

Ví dụ:

```
lst = [5, 7, 2, 9, 6, 3, 10, 17, 16]                          # len(lst) = 9
➔ lst[1:2] = [6, 8]   # [5, 6, 8, 2, 9, 6, 3, 10, 17, 16] # len(lst) = 10
➔ lst[1:3] = [7]      # [5, 7, 9, 6, 3, 10, 17, 16]       # len(lst) = 8
```

- **Sử dụng vòng lặp for để duyệt các phần tử:**

```
for x in lst:
        print(x)
```

- **Sử dụng vòng lặp for kết hợp index:**

```
for i in range(len(lst)):
        print(lst[i])
```

- **Sử dụng vòng lặp while:**

```
        i = 0
        while i < len(lst):
                print(lst[i]);
                i = i + 1
```

- **Sử dụng List Comprehension:**

```
    [expression for item in iterable [if condition]]
```

Ví dụ:

```
                [print(x) for x in lst]
                [print(x) for x in lst if x % 2 == 0]
new_list = [x if x % 2 == 0 for x in lst]
```


- Python hỗ trợ hàm insert trong list, và tự động chèn vào vị trí thích hợp

Cú pháp: **insert(vị trí, giá trị)**

Ví dụ:

```
lst = [0, 1, 2, 3]
print(lst) # [0, 1, 2, 3]
lst.insert(2, 9)
print(lst) # [0, 1, 9, 2, 3]
lst.insert(0, 17)
print(lst) # [17, 0, 1, 9, 2, 3]
lst.insert(10, 5)
print(lst) # ... ?
```

- Python hỗ trợ hàm append trong list, và chèn giá trị mới vào cuối list

Ví dụ:

```
lst1 = [0, 1, 2, 3]
lst2 = [5, 6]
print(lst1) # [0, 1, 2, 3]
lst1.append(5)
print(lst1) # [0, 1, 2, 3, 5]
lst1.append(-6)
print(lst1) # ... ?
lst1.append(lst2)
print(lst1) # ... ?
```

- Python hỗ trợ hàm extend trong list, để thêm vào cuối list hiện tại các giá trị từ một đối tượng lặp (iterable object) dạng danh sách khác (tuples, sets, dictionaries,…)

Ví dụ:

```
lst1 = [0, 1, 2, 3]
lst2 = [4, 5]
print(lst1) # [0, 1, 2, 3]
lst1.extend(list2)
print(lst1) # [0, 1, 2, 3, 4, 5]
```

- Python hỗ trợ hàm remove trong list, và **xóa phần tử cụ thể đầu tiên có trong list**

Cú pháp: **remove(giá trị)**

Ví dụ:

```
lst = [0, 1, 2, 3, 4, 5, 3]
print(lst) # [0, 1, 2, 3, 4, 5, 3]
lst.remove(1)
print(lst) # [0, 2, 3, 4, 5, 3]
lst.remove(2)
print(lst) # ... ?
lst.remove(3)
print(lst) # ... ?
lst.remove(6)
print(lst) # ... ?
```

- Python hỗ trợ hàm pop trong list, và xóa phần tử cụ thể thông qua vị trí phần tử đó có trong list
- Nếu không chỉ định vị trí, hàm pop sẽ xóa phần tử cuối cùng có trong list

Cú pháp: **pop([vị trí])**

Ví dụ:

```
lst = [0, 1, 2, 3]
print(lst) # [0, 1, 2, 3]
lst.pop(1)
print(lst) # [0, 2, 3]
lst.pop()
print(lst) # ... ?
lst.pop(-1)
print(lst) # ... ?
lst.pop(-9)
print(lst) # ... ?
```

- Python hỗ trợ phương thức clear để xóa toàn bộ các đối tượng có trong list.

Cú pháp: **clear()**

Ví dụ:

```
lst = [0, 1, 2, 3]
print(lst) # [0, 1, 2, 3]
lst.clear()
print(lst) # []
```

- Python hỗ trợ từ khóa del trong list, và xóa phần tử cụ thể có trong list hoặc xóa đối tượng list.

Cú pháp: **del lst[vị trí] / del lst**

Ví dụ:

```
lst = [0, 1, 2, 3]
print(lst) # [0, 1, 2, 3]
del lst[1]
print(lst) # [0, 2, 3]
del lst[-1]
print(lst) # ... ?
del lst[-5]
print(lst) # ... ?
del lst
print(lst) # ... ?
```

- Sắp xếp các đối tượng trong list theo thứ tự tự chữ và số - alphanumerically (mặc định tăng dần)

Cú pháp: **sort([reverse = True])**

sort([key = function]) # sắp xếp theo hàm

sorted(iterable, key = key, reverse = reverse)

Ví dụ:

```
lst = ["Orange", "mango", "Kiwi", "pineapple", "banana"]
lst.sort() # sắp xếp mặc định phân biệt hoa thường (chữ hoa đứng trước)
print(lst) # ['Kiwi', 'Orange', 'banana', 'mango', 'pineapple']
lst.sort(key = str.lower) # sắp xếp không phân biệt hoa thường
print(lst) # ['banana', 'Kiwi', 'mango', 'Orange', 'pineapple’]
```


- Nghịch đảo thứ tự hiện tại của các phần tử trong list

Cú pháp: **reverse()**

Ví dụ:

```
lst = ["Orange", "mango", "Kiwi", "pineapple", "banana"]
lst.reverse()
print(lst) # ['banana', 'pineapple', 'Kiwi', 'mango', 'Orange']
```

- Khi sử dụng lệnh gán list với nhau, chương trình sẽ hiểu theo kiểu tham chiếu, tức là khi thay đổi trên list này sẽ tự động thau đổi trên các list tham chiếu đến.
- Để tránh trường hợp này, ta sử dụng phương thức copy() hoặc list()

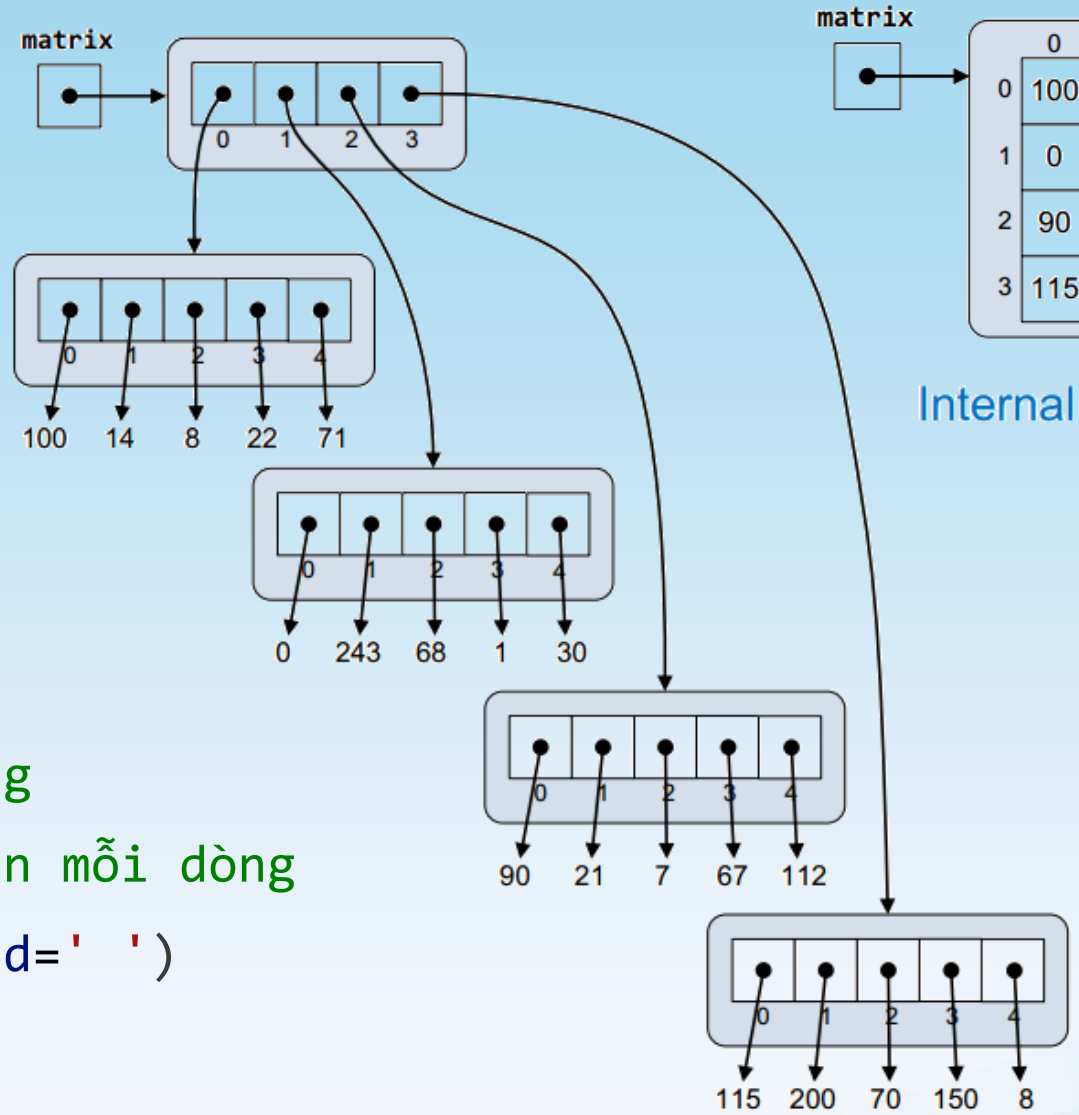
Ví dụ:

```
lst1 = [5, 7, 2, 9, 6, 3, 10, 17, 16]
lst2 = lst1.copy()
lst3 = list(lst1)
```

- Có nhiều cách để nhập, nối nhiều list lại với nhau trong python
  - ✓ Sử dụng toán tử +
  - ✓ Sử dụng phương thức append
  - ✓ Sử dụng phương thức extend

- Python hỗ trợ List đa chiều

```
matrix = [
    [100,  14,  8,  22,  71],
    [0  , 243, 68,   1,  30],
    [90 ,  21,  7,  67, 112],
    [115, 200, 70, 150,   8]
]
print(matrix)
for row in matrix: # duyệt từng dòng
    for x in row: # lấy phần tử trên mỗi dòng
        print('{:>4}'.format(x), end=' ')
    print()
```



- Khởi tạo list đa chiều có row dòng column cột

```
row = 5
column = 3
lst = [[0] * column] * row
print(lst)
# [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```


- ✓ Họ tên : Trần Quang Khải
- ✓ Email : khaitq@hcmute.edu.vn
- ✓ Zalo (mã Qr)