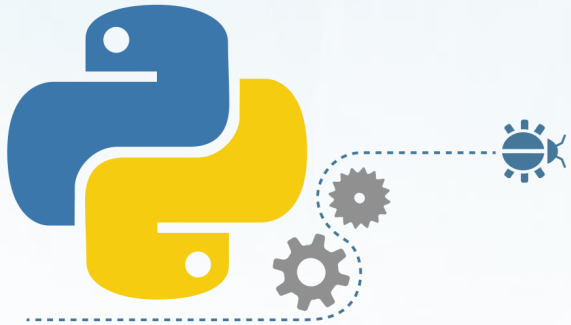
TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education

KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN HỆ THỐNG THÔNG TIN

NHẬP MÔN LẬP TRÌNH PYTHON (IPPA233277)

CÁC KIỂU DỮ LIỆU PHỨC HỢP (tuple – set – dictionary)



GV. Trần Quang Khải

1. Hiểu được cơ chế vận hành của Tuple, Set, Dictionary
2. Khai báo và sử dụng được Tuple, Set, Dictionary
3. Thực hiện được các hàm: duyệt, gán, chèn, nối đuôi, xóa, đảo ngược, sắp xếp, slicing

1. Giới thiệu về tuple
2. Thao tác với tuple
3. Giới thiệu về set
4. Thao tác với set
5. Giới thiệu về dictionary
6. Thao tác với dictionary

- 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

	Tính thứ tự	Thay đổi	Lặp giá trị
LIST	x	x	x
TUPLE	x		x
SET		*	
DICTIONARY	**		

* - 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

- Là kiểu dữ liệu chứa các đối tượng không thể thay đổi được trong python
- Gần giống với list nhưng khác ở 02 điểm:
  - ✓ Tuple không thể thay đổi
  - ✓ Sử dụng cặp dấu ngoặc tròn ( ) thay cho ngoặc vuông [ ]
- Các phần tử cách nhau dấu phẩy và có một chỉ số vị trí (index) bắt đầu từ 0

Ví dụ:

```
tuple1 = (1, 2, 3, 4, 5, 6, 7)
tuple2 = tuple((1, 2, 3, 5, 3, 4, 2, 6, 7))
print(len(tuple1)) # 7
print(len(tuple2)) # 9
```

- Các đối tượng trong tuple có thể ở nhiều kiểu dữ liệu khác nhau
- Tuple hiệu quả và truy cập nhanh hơn List
- Tuple có thể chuyển đổi thành List, và ngược lại
- Tuple không thể thay đổi sau khi đã được tạo ra
- Ta không thể thực hiện các thao tác add, delete và search trên tuple
- Giá trị của các phần tử trong tuple không thể thay đổi
- Không thể xóa từng phần tử riêng lẻ của tuple
- Lệnh del sẽ xóa toàn bộ Tuple
- Tạo ra một tuple rỗng, cú pháp: tup1 = ();
- Tạo ra một tuple chỉ chứa một giá trị đơn, thêm dấu phẩy: tup1 = (50,);

- **Các thao tác truy xuất tuple giống với list**
- Khi tuple được tạo ra, không thể thay đổi giá trị. Tuple thì không thay đổi (**unchangeable**) và bất biến (**immutable**)
- Ta có thể chuyển đổi tuple → list → thay đổi giá trị của list → tuple

Ví dụ:

```
tuple1 = (1, 2, 3, 4, 5, 6, 7)
list1 = list(tuple1)
list1[0] = 8
tuple1 = tuple(list1)
print(tuple1) # (8, 2, 3, 4, 5, 6, 7)
```

- Cách 1: Chuyển đổi qua lại giữa tuple – list
- Thêm tuples vào tuples

Ví dụ:

```
tuple1 = (1, 2, 3, 4, 5, 6, 7)
tuple2 = (8, )
tuple1 += tuple2
print(tuple1) # (1, 2, 3, 4, 5, 6, 7, 8)
```

- Xóa phần tử khỏi tuples: cách 1
- Xóa tuple: **del <tuple>**

- Khi khởi tạo một tuple, các giá trị được gán cho tuple → packing (đóng gói)
- Trong python cho phép trích xuất các giá trị từ tuple → unpacking (mở gói)
- Có thể sử dụng **một dấu * duy nhất** để lấy các giá trị còn lại trong tuple khi unpacking

Ví dụ:

```
friends = ('Tí', 'Tèo', 'Tũn', 'Tĩn')
(Ti, Teo, Tun, Tin) = friends
(Ti, Teo, *AnhEm) = friends
print(Ti)     # Tí
print(Tun)    # Tũn
print(AnhEm) # ['Tũn', 'Tĩn']
```


- Đếm phần tử xuất hiện trong tuple : **count(<value>)**
- Vị trí đầu tiên của giá trị cần tìm : **index(<value>)**

Ví dụ:

```
tuple1 = (1, 2, 3, 2, 5, 4, 2)
print(tuple1.count(2)) # 3
print(tuple1.count(8)) # 0
print(tuple1.index(2)) # 1
print(tuple1.index(8)) # ... ?
```

- Tuple hỗ trợ các toán tử + (Nối tuple) và * (Nhân bản tuple)
- Bảng dưới đây minh họa cú pháp và ý nghĩa của các toán tử trong Tuple

Python Expression	Results	Descriptions
`len((1, 2, 3))`	3	Length
`(1, 2, 3) + (4, 5, 6)`	(1, 2, 3, 4, 5, 6)	Concatenation
`('Hi!', ) * 4`	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
`3 in (1, 2, 3)`	TRUE	Membership
`for x in (1, 2, 3): print(x)`	123	Iteration

- Được sử dụng để lưu tập dữ liệu không thứ tự, không thể thay đổi* và không được lập chỉ mục, không trùng lặp
- Khai báo set sử dụng dấu ngoặc nhọn { } hoặc hàm set( )
- ***Chú ý:*** 1 – True hoặc 0 – False được xem là cùng một giá trị
- Lấy số lượng phần tử trong set: len(<set_name>)

Ví dụ:

```
myset = {1, 2, 3, 4, 5}
myset = set({"1, 2, 3, 4, 5"})
```

(*) Nhưng có thể xóa hoặc thêm mới vào set

- Không thể truy xuất set thông qua index hoặc key
- Sử dụng vòng lặp để lấy đối tượng

Ví dụ:

```
myset = {1, 2, 3, 4, 5}
for x in myset:
    print(x)
```

- Sử dụng phương thức **add()** để thêm đối tượng mới vào trong set
- Sử dụng phương thức **update()** để thêm đối tượng từ một đối tượng khác như: set, list, tuple, dictionary,…

Ví dụ:

```
myset1  = {1, 2, 3, 4, 5}
myset2  = {6, 7}
mylist  = [6, 7]
mytyple = (6, 7)
myset1.add(6)          ; print(myset1) # … ?
myset1.update(myset2)  ; print(myset1) # … ?
myset1.update(mylist)  ; print(myset1) # … ?
myset1.update(mytyple) ; print(myset1) # … ?
```

- Sử dụng phương thức **remove(<value>)** để xóa đối tượng cụ thể (**phát sinh lỗi** nếu đối tượng không tồn tại)
- Sử dụng phương thức **discard(<value>)** để xóa đối tượng (**không phát sinh lỗi** nếu đối tượng không tồn tại)
- Sử dụng phương thức **pop( )** để xóa đối tượng ngẫu nhiên
- Sử dụng phương thức **clear( )** để xóa toàn bộ đối tượng có trong set
- Sử dụng từ khóa **del** để xóa đối tượng set

Ví dụ:

```
myset = {1, 2, 3, 4, 5}
myset.discard(3)          ; print(myset)   # … ?
myset.discard(7)          ; print(myset)   # … ?
myset.remove(2)           ; print(myset)   # … ?
myset.remove(2)           ; print(myset)   # … ?
```

- Sử dụng phương thức **union()** hoặc **toán tử |** để nối **hai set** hoặc giữa **set và tuple**

Ví dụ:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.union(y) # hoặc x | y
print(z) # …?
```

- Sử dụng phương thức **intersection()** hoặc **toán tử &** giữ lại đối tượng chung giữa **hai set** (giao)

Ví dụ:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.intersection(y) # hoặc x & y
print(z) # …?
```

- Sử dụng phương thức **intersection_update()** giống với intersection() và gán (đè) vào set hiện tại

Ví dụ:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.intersection_update(y) # hoặc x &= y
print(x) # …?
```

- Sử dụng phương thức **difference(<set>)** hoặc **toán tử -** để giữ lại đối tượng không xuất hiện trong <set>

Ví dụ:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.difference(y) # hoặc z = x - y
print(z) # …?
```

- Sử dụng phương thức **difference_update(<set>)** giống với difference() và cập nhật vào set hiện tại

Ví dụ:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.difference_update(y) # hoặc x -= y
print(x) # …?
```


- Sử dụng phương thức **symmetric_difference(<set>)** hoặc **toán tử ^** không xuất hiện đồng thời ở các sets

Ví dụ:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.symmetric_difference(y) # hoặc z = x ^ y
print(z) # …?
```

- Sử dụng phương thức **symmetric_difference_update(<set>)** giống symmetric_difference(), gán (đè) set hiện tại

Ví dụ:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.symmetric_difference_update(y) # hoặc x ^= y
print(x) # …?
```

- Dictionary là danh sách có chứa tập hợp các phần tử có tính thứ tự *, có thể thay đổi và không được trùng lặp
- Mỗi phần tử của dictionary được thể hiện dưới dạng một cặp (key : value)
  - ✓ Một cặp (key : value) trong dictionary còn được gọi là một **item**
  - ✓ Mỗi key được phân tách với value tương ứng bằng một dấu :
  - ✓ Các item được phân tách bởi dấu phẩy ,
  - ✓ Toàn bộ dictionary được bọc bởi một cặp dấu { }
  - ✓ Trong dictionary, tất cả các key phải là duy nhất (unique)
  - ✓ Các giá trị ở trong dictionary có thể được truy cập bởi cặp dấu [ ]
  - ✓ Giá trị của một đối tượng trong dictionary có thể là bất kỳ kiểu dữ liệu nào

Ví dụ:

```
myinfo =  { "name": "Tran", "age": 37, "address" : "Vietnam" }
```

(*) Áp dụng cho phiên bản python từ 3.7, trước đó dictionary không có tính thứ tự

- Khai báo dictionary:

```
mydict = { "name": "Tran", "age": 37, "country" : "Vietnam" }
mydict = dict( name = "Tran", age = 37, country = "Vietnam" )
```

- Số lượng đối tượng có trong dictionary: `len()`

- Sử dụng **key_name**
- Sử dụng phương thức **get(<key_name>)**
- Lấy danh sách keys sử dụng phương thức **keys()**
- Lấy danh sách giá trị sử dụng phương thức **values()**
- Lấy danh sách đối tượng theo dạng tuples sử dụng phương thức **items()**
- Kiểm tra key có tồn tại trong dictionary: **if key_name in <dictionary>:**

Ví dụ:

```
print(mydict['name’]) # Tran
print(mydict.get('age')) # 37
print(mydict.keys())  # dict_keys(['name', 'age', 'country'])
print(mydict.values())# dict_values(['Tran', 37, 'Vietnam’])
print(mydict.items()) # dict_items([('name','Tran'), ('age',37), ('country','Vietnam')])
```


- Sử dụng **key_name**
- Sử dụng phương thức **update({'key_name':new_value})**

Ví dụ:

```
mydict['name'] = 'Quang’
mydict.update({'name':'Quang'})
```

- Sử dụng phương thức **pop('key_name')**
- Sử dụng phương thức **popitem()** để xóa đối tượng cuối được thêm vào (*)
- Sử dụng phương thức **clear()** để làm trống dictionary
- Sử dụng từ khóa **del** để xóa đối tượng dictionary

Ví dụ:

```
mydict.pop('name')
mydict.popitem()
mydict.clear()
del mydict
```

(*) Phiên bản python trước 3.7 sẽ xóa đối tượng ngẫu nhiên

- Lấy **key_name**:
  - `for key in mydict: print(key)`
  - `for key in mydict.keys(): print(key)`
- Lấy **value**:
  - `for key in mydict: print(mydict[key])`
  - `for x in mydict.values(): print(x)`
- Lấy giá trị **(key : value)**:
  - `for key, value in mydict.items(): print(key, value)`

Phương thức	Mô tả	Cú pháp
copy()	Sao chép toàn bộ từ điển sang từ điển mới	dict.copy ()
update()	Cập nhật một từ điển bằng cách thêm một mục mới hoặc một cặp khóa-giá trị vào một mục hiện có hoặc bằng cách xóa mục hiện có.	dict.update ([other])
items()	Trả về danh sách các cặp tuple (Khóa, Giá trị) trong từ điển.	dictionary.items ()
sort()	Sắp xếp các phần tử trong từ điển.	dictionary.sort ()
len()	Trả về số lượng cặp trong từ điển	len (dict)
cmp()	So sánh giá trị và khóa của hai từ điển	cmp (dict1, dict2)
str()	Chuyển đổi từ điển thành định dạng chuỗi có thể in	str(dict)

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