

✓ List trong python

✓ khái niệm

- ✓ List là một cấu trúc dữ liệu có thứ tự, có thể chứa nhiều phần tử

Mỗi phần tử trong List có một chỉ số, được sử dụng để truy cập phần tử đó

Các phần tử trong List có thể là bất kỳ kiểu dữ liệu nào

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
print(my_list)
```

```
danh_sach = [1, 2, 3, 4.5, "huytv", 2 + 2j, [4, 5]]
print(danh_sach)
print(danh_sach[3])
print(danh_sach[-4])
```

```
[1, 2, 3, 4.5, 'huytv', (2+2j), [4, 5]]
4.5
4.5
```

```
my_list_complex = [[1, 2], (3, 4), {"a": 1}, {1, 2, 3}]
print(my_list_complex)
```

```
[[1, 2], (3, 4), {'a': 1}, {1, 2, 3}]
```

```
def greet():
    return "Hi!"

class Person:
    pass

items = [greet, Person(), len]
```

```
[<function greet at 0x0000024CAD2BD760>, <__main__.Person object at 0x0000024CAD1078C0>, <built-in function len>]
```

✓ Cấu trúc list trong python

Index (Chỉ số):

Index để cập đến vị trí của một phần tử trong list

Index:

Thường bắt đầu từ số 0 cho phần tử đầu tiên (tăng 1 đơn vị cho các phần tử tiếp theo)

Hoặc phần tử cuối cùng -1 (giảm 1 đơn vị cho các phần tử từ cuối đến phần tử đầu tiên)

Value (Giá trị):

Value là giá trị thực sự của phần tử đó trong list

✓ Truy cập vào List

```
# khởi tạo danh sách rỗng
my_list = []
print(my_list)
```

```
[]
```

```
my_list = [1, 2, 5, "hello", [4, 5]]
# print(my_list[3])
# print(my_list[-2])
for phantu in my_list:
    print(phantu)
```

```
1
2
5
hello
[4, 5]
```

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
print(my_list)
print(type(my_list))
print(len(my_list))
my_list[0] = 100
print(my_list)
my_list[4][0] = 300
print(my_list)
# tạo danh sách với hàm list()
```

```
[1, 2, 4.5, 'python', [3, 4], (2+3j)]
<class 'list'>
6
[100, 2, 4.5, 'python', [3, 4], (2+3j)]
[100, 2, 4.5, 'python', [300, 4], (2+3j)]
```

✓ Duyệt qua các phần tử list

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
# for item in my_list:
#     print(item)
for phan_tu in enumerate(my_list):
    print(phan_tu)
```

```
(0, 1)
(1, 2)
(2, 4.5)
(3, 'python')
(4, [3, 4])
(5, (2+3j))
```

✓ Dùng hàm enumerate

```
danh_sach = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
# for items in enumerate(my_list):
#     print(items)
danh_sach_con = danh_sach[2:4]
print(danh_sach_con)
```

```
[4.5, 'python']
```

✓ Slicing trong list

✓ mylist[start:stop:step]

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
sub_list = my_list[1:4]
print(sub_list)
```

```
[2, 4.5, 'python']
```

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
sub_list = my_list[:4]
```

```
print(sub_list)
```

```
[1, 2, 4.5, 'python']
```

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
sub_list = my_list[1:5:2]
print(sub_list)
```

```
ds_so =[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# ds_so_chan =ds_so[1:8:2]
# print(ds_so_chan)
# print(len(ds_so))
# ds_so.append("good bye")
# ds_so.append(9)
# ds_so.append([4,7])
# ds_so.insert(3,"Hello")
# print(ds_so)
ds_so1 =[10, 20, 40]
ds_so.extend(ds_so1)
print(ds_so)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 20, 40]
```

✓ Các phương thức trong list

✓ Chiều dài list

len()

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
len_list = len(my_list)
print(len_list)
```

```
6
```

✓ append(): thêm phần tử cuối list

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list.append("Data Science")
print(my_list)
my_list.append([7, 8, 9])
print(my_list)
```

```
my_list = [1, 2, "hello", [4, 5]]
print(my_list)
my_list.insert(2, "goodbye")
print(my_list)
```

```
[1, 2, 'hello', [4, 5]]
[1, 2, 'goodbye', 'hello', [4, 5]]
```

✓ insert("index,Giá trị cần thêm"): thêm phần tử vào index bất kỳ

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list.insert(2, "Data Science")
print(my_list)
my_list.insert(0, [10, 20, 30])
print(my_list)
```

```
[1, 2, 'Data Science', 4.5, 'python', [3, 4], (2+3j)]
[[10, 20, 30], 1, 2, 'Data Science', 4.5, 'python', [3, 4], (2+3j)]
```

✓ extend(): nối 1 list vào cuối 1 list khác

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list2 = ["a", "b", "c"]
my_list.extend(my_list2)
print(my_list)
```

```
my_list = [1, 2, 3]
my_list1 = ["a", "b", "c"]
my_list.extend(my_list1)
print(my_list)
```

```
[1, 2, 3, 'a', 'b', 'c']
```

✓ xóa 1 phần tử dùng remove(value)

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list.remove([3, 4])
print(my_list)
```

```
[1, 2, 4.5, 'python', (2+3j)]
```

✓ Xóa 1 phần tử dùng pop(index): giữ lại giá trị xóa

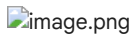
```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
so_thuc = my_list.pop(2)
print(my_list)
print(so_thuc)
```

```
[1, 2, 'python', [3, 4], (2+3j)]
4.5
```

✓ xóa 1 phần tử hoặc nhiều phần tử dùng del()

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
# del my_list[2]
# print(my_list)
del my_list[1:4]
print(my_list)
```

```
[1, [3, 4], (2+3j)]
```



✓ clear: xóa tất cả phần tử khỏi list

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list.clear()
print(my_list)
```

✓ copy(): tạo bản sao cho list

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list1 = my_list.copy()
print(my_list1)
```

```
[1, 2, 4.5, 'python', [3, 4], (2+3j)]
```

✓ sort(): sắp xếp các phần tử trong danh sách

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list.sort()
print(my_list)
```

```
my_list1 = [3, 1, 4.5, 2, 5]
my_list1.sort()
print(my_list1)
```

```
[1, 2, 3, 4.5, 5]
```

- ✓ `sort(reverse=True)`: sắp xếp danh sách giảm dần theo giá trị

```
my_list1 = [3, 1, 4, 2, 5]
my_list1.sort(reverse = True)
print(my_list1)
```

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

- ✓ `Reverse()`: đảo ngược danh sách

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list.reverse()
print(my_list)
```

```
[(2+3j), [3, 4], 'python', 4.5, 2, 1]
```

- ✓ trong list có thể dùng `dir` để liệt kê các thuộc tính và phương thức trong list

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
print(dir(my_list))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__form
```

- ✓ Toán tử + trong list

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list1 = ["a", "b", "c"]
my_list3 = my_list + my_list2
print(my_list3)
```

```
[1, 2, 4.5, 'python', [3, 4], (2+3j), 'a', 'b']
```

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
my_list1 = ["a", "b", "c"]
my_list += my_list1
print(my_list)
```

```
[1, 2, 4.5, 'python', [3, 4], (2+3j), 'a', 'b', 'c']
```

- ✓ Toán tử `in` và `not in`

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
ket_qua = 2 in my_list
print(ket_qua)
```

```
True
```

```
my_list = [1, 2, 4.5, "python", [3, 4], 2 + 3j]
ket_qua = 2 not in my_list
print(ket_qua)
```

```
False
```

- ✓ List comprehensions



```
# tạo list có các số chẵn từ 0 đến 20
new_list = []
for i in range(21):
    if i % 2 == 0:
        new_list.append(i)
print(new_list)
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
new_list = [i for i in range(21)]
print(new_list)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

```
# kết hợp điều kiện if
new_list = [i for i in range(21) if i % 2 == 0]
print(new_list)
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
new_list = [x if x % 2 == 0 else -x for x in range(21)]
print(new_list)
```

```
[0, -1, 2, -3, 4, -5, 6, -7, 8, -9, 10, -11, 12, -13, 14, -15, 16, -17, 18, -19, 20]
```

```
new_list = [x + y for x in [1, 2, 3] for y in [10, 20, 30]]
print(new_list)
```

```
[11, 21, 31, 12, 22, 32, 13, 23, 33]
```

✓ Kiểu dữ liệu tuple

✓ Tuple là một cấu trúc dữ liệu giống list, nhưng:

Không thay đổi được (immutable) sau khi tạo.

Dùng dấu ngoặc tròn () thay vì [].

```
tuple1 = (1, 2, 3, 4.5, "python", (3, 4), 2 + 3j)
print(tuple1)
tuple2 = (100, )
print(tuple2)
```

```
(1, 2, 3, 4.5, 'python', (3, 4), (2+3j))
(100,)
```

✓ truy cập phần tử trong tuple

```
tuple = (1, 2, 4.5, "python", [3, 4], 2 + 3j)
print(tuple)
print(tuple[1])
print(type(tuple[1:4]))
```

```
(1, 2, 4.5, 'python', [3, 4], (2+3j))
2
<class 'tuple'>
```


✓ Tuple không thể thay đổi

```
tuple = (1, 2, 4.5, "python", [3, 4], 2 + 3j)
tuple[0] = 100
print(tuple)
```

✓ Các phương thức sử dụng tuple

```
tuple = (1, 2, 4.5, "python", [3, 4], 2 + 3j)
print(len(tuple))
print(tuple.count(2))
print(tuple.index("python"))
```

✓ Kiểu dữ liệu tập hợp

 image.png

```
my_set = {1, 2, 3, 4, 5}
print(my_set)
my_set1 = set([1, 2, 2, 3, 6, 7, 8])
print(my_set1)
```

```
{1, 2, 3, 4, 5}
{1, 2, 3, 6, 7, 8}
```

```
invalid_set = {set([1, 2, 3]), [4, 5, 6], {"a": 1, "b": 2}}
print(invalid_set)
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[67], line 1
----> 1 invalid_set = {set([1, 2, 3]), [4, 5, 6], {"a": 1, "b": 2}}
      2 print(invalid_set)

TypeError: unhashable type: 'set'
```

✓ Các phương thức set

✓ chiều dài set: len()

```
my_set = {1, 2, 3, 4, 5}
print(my_set)
print(len(my_set))
```

```
{1, 2, 3, 4, 5}
5
```

✓ kiểm tra 1 phần tử có trong set: in

```
my_set = {1, 2, 3, 4, 5}
if 3 in my_set:
    print("3 có trong set")
else:
    print("3 không có trong set")
```

```
3 có trong set
```

✓ thêm 1 phần tử vào set: add()

```
my_set = {1, 2, 3, 4, 5}
my_set.add(6)
print(my_set)
```

```
{1, 2, 3, 4, 5, 6}
```

✓ Thêm nhiều hơn 1 phần tử vào set: update()

```
my_set = {1, 2, 3, 4, 5}
my_set.update([6, 7, 8])
print(my_set)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

✎ Xóa 1 phần tử khỏi set: remove(), discard(), pop()

```
my_set = {1, 2, 3, 4, 5}
my_set.remove(3)
print(my_set)
```

```
{1, 2, 4, 5}
```

```
my_set = {1, 2, 4, 5}
my_set.remove(3)
print(my_set)
```

```
-----
KeyError                                Traceback (most recent call last)
Cell In[75], line 2
      1 my_set = {1,2,4,5}
----> 2 my_set.remove(3)
      3 print(my_set)

KeyError: 3
```

```
my_set = {1, 2, 4, 5}
my_set.discard(3)
print(my_set)
```

```
{1, 2, 4, 5}
```

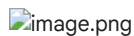
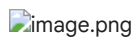
```
my_set = {1, 2, 3, 4, 5}
my_set.pop()
print(my_set)
```

```
{2, 3, 4, 5}
```

```
my_set = {1, 2, 3, 4, 5}
my_set.clear()
print(my_set)
```

```
set()
```

✎ Các phép toán trong set



✎ union

```
my_set1 = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}
my_set3 = my_set1.union(my_set2)
print(my_set3)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

```
my_set1 = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}
my_set3 = my_set1 | my_set2
print(my_set3)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```


▼ intersection

```
my_set1 = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}
# intersection
my_set3 = my_set1.intersection(my_set2)
print(my_set3)
```

```
{4, 5}
```

```
my_set1 = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}
# intersection
my_set3 = my_set1 & my_set2
print(my_set3)
```

```
{4, 5}
```

▼ Symmetric Difference

```
my_set1 = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}
# symmetric difference
my_set3 = my_set1.symmetric_difference(my_set2)
print(my_set3)
```

```
{1, 2, 3, 6, 7, 8}
```

```
my_set1 = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}
# symmetric difference
my_set3 = my_set1 ^ my_set2
print(my_set3)
```

```
{1, 2, 3, 6, 7, 8}
```

▼ Difference

```
my_set1 = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}
# Difference
my_set3 = my_set1.difference(my_set2)
print(my_set3)
```

```
{1, 2, 3}
```

```
my_set1 = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}
# symmetric difference
my_set3 = my_set1 - my_set2
print(my_set3)
```

```
{1, 2, 3}
```

▼ set trong xử lý dữ liệu

```
## loại bỏ các phần tử trùng lặp trong list sử dụng set
my_list = [1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 8, 8, 9]
print("danh sách ban đầu:", my_list)
my_set = set(my_list)
print("sau khi chuyển sang set:", my_set)
```

```
danh sách ban đầu: [1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 8, 8, 9]
sau khi chuyển sang set: {1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
emails = ["a@gmail.com", "b@gmail.com", "a@gmail.com", "c@gmail.com"]
```

```
unique_emails = set(emails)
print(unique_emails)
print(f"Số email duy nhất: {len(unique_emails)}")
```

```
{'a@gmail.com', 'b@gmail.com', 'c@gmail.com'}
Số email duy nhất: 3
```

✓ tìm kiếm giá trị tương tự

```
my_set1 = {1, 2, 3, 4, 5}
my_set2 = {4, 5, 6, 7, 8}
# union
my_set3 = my_set1 & my_set2
print(my_set3)
```

```
{4, 5}
```

```
python_students = {"Huy", "Lan", "Minh", "Nam"}
ds_students = {"Lan", "Nam", "Phuong"}
```

```
common = python_students & ds_students
print(common)
# {'Lan', 'Nam'}
all_students = python_students | ds_students
print(all_students)
```

```
{'Lan', 'Nam'}
{'Minh', 'Phuong', 'Lan', 'Huy', 'Nam'}
```

✓ Chuẩn hóa dữ liệu

```
my_set = {1, 2, 3, 4, 5, -999, 1000, 5000}
filtered_set = {x for x in my_set if 0 < x < 1000}
print(filtered_set)
```

```
{1, 2, 3, 4, 5}
```

✓ List comprehension trong set



```
squares = {x**2 for x in range(6)}
print(squares) # {0, 1, 4, 9, 16, 25}
```

```
{0, 1, 4, 9, 16, 25}
```

```
even_numbers = {x for x in range(10) if x % 2 == 0}
print(even_numbers) # {0, 2, 4, 6, 8}
```

```
{0, 2, 4, 6, 8}
```

```
Set_so_chinh_phuong={x ** 2 for x in range(1, 11)}
print(Set_so_chinh_phuong)
```

```
{64, 1, 4, 36, 100, 9, 16, 49, 81, 25}
```

```
letters = {ch for ch in "hello world" if ch.isalpha()}
print(letters) # {'h', 'e', 'l', 'o', 'w', 'r', 'd'}
```

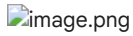
```
{'o', 'r', 'l', 'h', 'w', 'd', 'e'}
```

```
pairs = {(x, y) for x in range(3) for y in range(3) if x != y}
print(pairs)
```

```
# {(0, 1), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1)}
```

```
{(0, 1), (1, 2), (2, 1), (2, 0), (0, 2), (1, 0)}
```

Dictionary



```
# ví dụ:
# Dictionary
my_dict = {"name": "Alice", "age": 25, "city": "New York"}
print(my_dict)
```

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

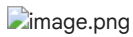
```
my_dict = {"name": "Alice",
           "age": 25,
           "city": "New York"}
print(my_dict)
```

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

```
my_dict = {"Ten": "Alice",
           "NamSinh": 2000,
           "MonHoc": ["Toan", "Ly", "Hoa"]}
print(my_dict)
```

```
{'Ten': 'Alice', 'NamSinh': 2000, 'MonHoc': ['Toan', 'Ly', 'Hoa']}
```

Key trong dictionary



```
# tạo menu trong chương trình máy tính bỏ túi
menu = {
    "1": "Cộng",
    "2": "Trừ",
    "3": "Nhân",
    "4": "Chia",
    "5": "Lũy thừa",
    "6": "Giải Phương trình bậc nhất",
    "7": "Giải Phương trình bậc 2",
    "0": "Thoát",
}

print("==== * 10 + \"MENU\" + \"==== * 10")
for k, v in menu.items():
    print(f"{k}. {v}")
print("====*10 + \"====\" + \"====*10")
```

```
=====MENU=====
1. Cộng
2. Trừ
3. Nhân
4. Chia
5. Lũy thừa
6. Giải Phương trình bậc nhất
7. Giải Phương trình bậc 2
0. Thoát
=====
```

Value trong dictionary



✓ quản lý trong dictionary

✓ khởi tạo

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
print(my_dict)
```

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

✓ truy cập

```
print(my_dict['name'])
```

```
Alice
```

✓ xóa

```
del my_dict["name"]  
print(my_dict)
```

```
{'age': 25, 'city': 'New York'}
```

✓ thay đổi giá trị

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
my_dict["name"] = "Huy"  
print(my_dict)
```

```
{'name': 'Huy', 'age': 25, 'city': 'New York'}
```

✓ thêm mới giá trị

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
print(my_dict)  
my_dict["job"] = "Speaker"  
print(my_dict)
```

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}  
{'name': 'Alice', 'age': 25, 'city': 'New York', 'job': 'Speaker'}
```

✓ Duyệt và lặp của dictionary

✓ Duyệt qua các key và in các key

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
for x in my_dict:  
    print(x)
```

```
name  
age  
city
```

✓ Duyệt qua các key và in ra các value

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
for x in my_dict:
```

```
print(my_dict[x])
```

```
Alice  
25  
New York
```

các phương thức trong dictionary

✓ copy()

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
print(my_dict)  
my_dict1 = my_dict.copy()  
print(my_dict1)
```

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}  
{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

✓ key()

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
for key in my_dict.keys():  
    print(key)
```

```
name  
age  
city
```

✓ value()

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
for value in my_dict.values():  
    print(value)
```

```
Alice  
25  
New York
```

✓ items()

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
for key, value in my_dict.items():  
    print(key, ":", value)
```

```
name : Alice  
age : 25  
city : New York
```

✓ get()

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
print(my_dict.get("name"))
```

```
Alice
```

✓ update()

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}  
my_dict.update({"name": "Huy"})  
print(my_dict)
```

```
{'name': 'Huy', 'age': 25, 'city': 'New York'}
```

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}
my_dict.update({"Job": "Interviewer"})
print(my_dict)
```

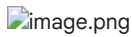
```
{'name': 'Alice', 'age': 25, 'city': 'New York', 'Job': 'Interviewer'}
```

✓ pop(): xóa cặp key ,value ra khỏi dict

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}
my_dict.pop("age")
print(my_dict)
```

```
{'name': 'Alice', 'city': 'New York'}
```

✓ Dict: comprehensions



```
# Ví dụ: Tạo dict mới giá trị là bình phương nếu các giá trị ban đầu là số chẵn, ngược lại giữ
# nguyên giá trị hiện tại
```

```
my_dict = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
my_dict1 = {k: v**2 if v % 2 == 0 else v for k, v in my_dict.items()}
print(my_dict1)
```

```
{'a': 1, 'b': 4, 'c': 3, 'd': 16}
```

```
my_list1 = [1,2,3]
my_list2=['a','b','c']
# my_dict = {x:y for x in my_list1 for y in my_list2}
# print(my_dict)
my_dict1 = {x:y for x,y in zip(my_list1, my_list2)}
print(my_dict1)
```

```
{1: 'c', 2: 'c', 3: 'c'}
{1: 'a', 2: 'b', 3: 'c'}
```

```
# chuyển danh sách cột trong google sheet thành dictionary
sinhvien_id = [1, 2, 3]
sinhvien_ten = ["Nguyễn Văn Nam", "Le Hoa", "Tuấn"]
sinhvien_email = ["nam@mail.com", "hoa@mail.com", "tuan@mail.com"]
```

```
sinhvien = [
    {"id": i, "name": n, "email": e}
    for i, n, e in zip(sinhvien_id, sinhvien_ten, sinhvien_email)
]
print(sinhvien)
# [{'id': 1, 'name': 'Nam', 'email': 'nam@mail.com'}, ...]
```

```
[{'id': 1, 'name': 'Nguyễn Văn Nam', 'email': 'nam@mail.com'}, {'id': 2, 'name': 'Le Hoa', 'email': 'hoa@mail.com'}, {'id': 3, 'name': 'Tuấn', 'email': 'tuan@mail.com'}]
```

```
my_list1 = [1, 2, 3]
my_list2 = ['a', 'b', 'c']

my_dict = {x : y for x in my_list1 for y in my_list2}
print(my_dict)
```

```
{1: 'c', 2: 'c', 3: 'c'}
```

