SẮP XẾP TRONG PYTHON

1. Hàm sort() của list:

Để sắp xếp các phần tử trong list theo thứ tự tăng dần hoặc giảm dần, các bạn có thể sử dụng hàm sort của list. Mặc định thì sort sẽ được sắp xếp theo thứ tự tăng dần về số và tăng dần về thứ tự từ điển với xâu kí tự.

CÚ PHÁP

```
sort(key = …., reverse = …)
```

- **Key:** Đây là hàm được sử dụng để làm tiêu chí sắp xếp.
- **Reverse:** Nếu reverse = True thì sẽ sắp xếp theo thứ tự ngược.

Hàm sort không trả về giá trị nào cả mà thay đổi trực tiếp list.

1. Hàm sort() của list:

Khi hàm sort thiếu các tham số này nó hoạt động bình thường.

EXAMPLE

```
a = [5, 1, 3, 2, 4]
a.sort()
print(a)
b = ["python", "28tech", "c++", "java"]
b.sort()
print(b)
```

OUTPUT

```
[1, 2, 3, 4, 5]
['28tech', 'c++', 'java', 'python']
```

Sort list theo thứ tự giảm dần.

EXAMPLE

```
a = [5, 1, 3, 2, 4]
a.sort(reverse = True)
print(a)
b = ["python", "28tech", "c++", "java"]
b.sort(reverse = True)
print(b)
```

OUTPUT

```
[5, 4, 3, 2, 1]
['python', 'java', 'c++', '28tech']
```

1. Hàm sort() của list:

Custom sort với tham số key

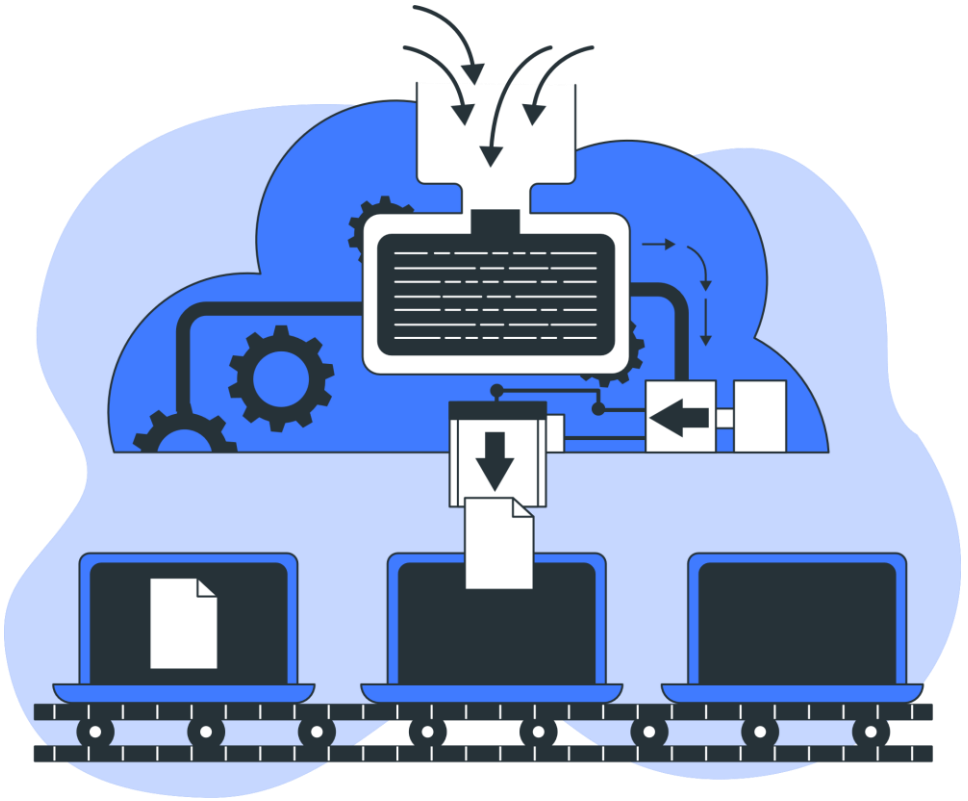
Ví dụ: Sắp xếp theo tổng chữ số tăng dần.

```
def sumDigit(n):
    res = 0
    while n != 0 :
        res += n % 10
        n //= 10
    return res

if __name__ == '__main__':
    a = [3, 24, 111, 30, 12, 21, 99, 32]
    a.sort(key = sumDigit)
    print(a)
```

OUTPUT

[3, 111, 30, 12, 21, 32, 24, 99]



1. Hàm sort() của list:

CHÚ Ý

Hàm sort trong Python là Tim sort, có tính chất stable vì thế những phần tử trong list có cùng tổng chữ số sẽ được giữ nguyên thứ tự ban đầu.

EXAMPLE

Sắp xếp các từ theo chiều dài :

```
a = ["28tech", "python", "java", "c", "c#"]
a.sort(key = len)
print(a)
```

OUTPUT

['c', 'c#', 'java', '28tech', 'python']

EXAMPLE

Sắp xếp nested list theo thành phần thứ 2 tăng dần, sau đó tới thành phần thứ 1 tăng dần:

```
def getItem(n):
    return n[1], n[0]

a = [[3, 2], [4, 1], [5, 3], [3, 1], [4, 4]]
a.sort(key = getItem)
print(a)
```

OUTPUT

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

2. Hàm sort() với lambda:

Tham số key của hàm sort có thể sử dụng biểu thức lambda để thay thế.

Sort theo trị tuyệt đối:

```
a = [-10, 230, 5, -4, -3, 10]
a.sort(key = lambda x : abs(x))
print(a)
a.sort(key = lambda x : abs(x), reverse = True)
print(a)
```

OUTPUT

```
[-3, -4, 5, -10, 10, 230]
[230, -10, 10, 5, -4, -3]
```

Sort nested list:

```
a = [[1, 2], [3, 1], [3, 4], [0, 5], [1, 1]]
a.sort(key = lambda x : x[0])
print(a)
```

OUTPUT

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

2. Hàm sort() với lambda:

Tham số key của hàm sort có thể sử dụng biểu thức lambda để thay thế.

Sort nested list theo thứ tự tăng dần của thành phần thứ nhất và giảm dần theo thành phần thứ 2:

```
a = [[1, 2], [3, 1], [3, 4], [0, 5], [1, 3]]
a.sort(key = lambda x : (x[0], -x[1]))
print(a)
```

OUTPUT

```
[[0, 5], [1, 3], [1, 2], [3, 4], [3, 1]]
```

2. Hàm sort() với lambda:

Tham số key của hàm sort có thể sử dụng biểu thức lambda để thay thế.

Sort một list các dict:

```
a = [
    {'name' : 'Tran Xuan Loc', 'job' : 'Dev', 'salary' : 500},
    {'name' : 'Thieu Ngoc Tuan', 'job' : 'Dev', 'salary' : 1500},
    {'name' : 'Phung Duc Kien', 'job' : 'BA', 'salary' : 5000},
    {'name' : 'Huynh Manh Tuong', 'job' : 'Tester', 'salary' : 2000}
]
a.sort(key = lambda x : x.get('salary'))
for x in a :
    print(x)
```

OUTPUT

```
{'name': 'Tran Xuan Loc', 'job': 'Dev', 'salary': 500}
{'name': 'Thieu Ngoc Tuan', 'job': 'Dev', 'salary': 1500}
{'name': 'Huynh Manh Tuong', 'job': 'Tester', 'salary': 2000}
{'name': 'Phung Duc Kien', 'job': 'BA', 'salary': 5000}
```

2. Hàm sort() với lambda:

Sort theo lương tăng dần, nếu 2 người cùng lương thì sort theo tên tăng dần theo thứ tự từ điển:

```
a = [
    {'name' : 'Tran Xuan Loc', 'job' : 'Dev', 'salary' : 500},
    {'name' : 'Thieu Ngoc Tuan', 'job' : 'Dev', 'salary' : 1500},
    {'name' : 'Phung Duc Kien', 'job' : 'BA', 'salary' : 500},
    {'name' : 'Huynh Manh Tuong', 'job' : 'Tester', 'salary' : 2000}
]
a.sort(key = lambda x : (x.get('salary'), x.get('name')))
for x in a :
    print(x)
```

OUTPUT

```
{'name': 'Phung Duc Kien', 'job': 'BA', 'salary': 500}
{'name': 'Tran Xuan Loc', 'job': 'Dev', 'salary': 500}
{'name': 'Thieu Ngoc Tuan', 'job': 'Dev', 'salary': 1500}
{'name': 'Huynh Manh Tuong', 'job': 'Tester', 'salary': 2000}
```

3. Kết hợp hàm sort với itemgetter() và attrgetter():

Bạn có thể sử dụng **itemgetter()** và **attrgetter()** từ module operator.


```
from operator import itemgetter
friends = [("Huynh Manh Tuong", 30),
           ("Tran Nhat Phi", 28),
           ("Tuan Binh", 21),
           ("Thieu Tuan", 35)]
friends.sort(key = itemgetter(1))
for x in friends:
    print(x)
```

OUTPUT

```
('Tuan Binh', 21)
('Tran Nhat Phi', 28)
('Huynh Manh Tuong', 30)
('Thieu Tuan', 35)
```

Sort theo nhiều tiêu chí.


```
from operator import itemgetter
friends = [("Huynh Manh Tuong", 30, 'B'),
           ("Tran Nhat Phi", 28, 'A'),
           ("Tuan Binh", 28, 'C'),
           ("Thieu Tuan", 30, 'Z')]
friends.sort(key = itemgetter(1, 2))
for x in friends:
    print(x)
```

OUTPUT

```
('Tran Nhat Phi', 28, 'A')
('Tuan Binh', 28, 'C')
('Huynh Manh Tuong', 30, 'B')
('Thieu Tuan', 30, 'Z')
```

3. Kết hợp hàm sort với itemgetter() và attrgetter():

Bạn có thể sort list, tuple,.. bằng hàm sorted, hàm này không thay đổi iterable mà bạn truyền cho nó mà nó sẽ trả về một list đã được sắp xếp tương ứng với iterable của bạn truyền vào. Cách sử dụng không có gì khác hàm sort ở tham số key và reverse.

EXAMPLE

```
a = (5, 4, 1, 2, 3)
b = sorted(a)
print(a)
print(b)
```

OUTPUT

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

EXAMPLE

```
s = "abczza28tech"
t = sorted(s)
print(s)
print(t)
```

OUTPUT

```
abczza28tech
['2', '8', 'a', 'a', 'b', 'c', 'c', 'e', 'h', 't', 'z', 'z']
```