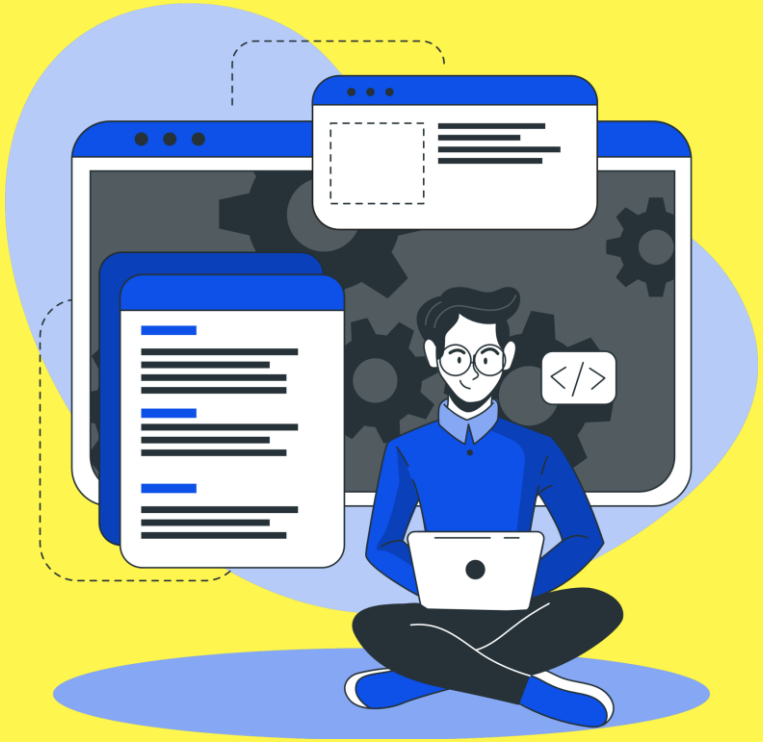
SET



Set trong Python là một tập hợp các phần duy nhất không có thứ tự. Chúng thường được sử dụng để tính toán các phép toán liên quan tới tập hợp.

Các tính chất của Set

- **Sets are unordered:** Các phần tử trong set không có thứ tự nhất định nào cả.
- **Set items are unique:** Các phần tử trong set là độc nhất, không có 2 phần tử nào trong set trùng nhau.
- **Sets are unindexed:** Set không thể truy cập được thông qua chỉ số.
- **Sets are changeable (mutable):** Set có thể thay đổi.

1. Tạo set:

Để tạo set bạn để các phần tử trong set phân cách nhau bởi dấu phẩy và đặt trong đóng mở ngoặc tròn.

Set không lưu những giá trị trùng nhau vì thế nếu set của bạn có các phần tử giống nhau nó sẽ tự bị loại bỏ.


```
s = {"28tech", "python", "28tech", "java", "set", "set"}
print(s)
t = {1, 2, 3, 1, 2, "abcd"}
print(t)
```

OUTPUT

```
{'python', 'java', 'set', '28tech'}
{1, 2, 3, 'abcd'}
```

1. Tạo set:

CHÚ Ý

- Set không thể lưu được những phần tử có thể thay đổi được giá trị (mutable) ví dụ như list.
- Các object không thể thay đổi như tuple, str, int… thì có thể làm phần tử của list được vì chúng không thể thay đổi.

EXAMPLE

```
s = {[1, 2], "python", "28tech"}
print(s)
```

OUTPUT

```
TypeError: unhashable type: 'list'
```

2. Set constructor:

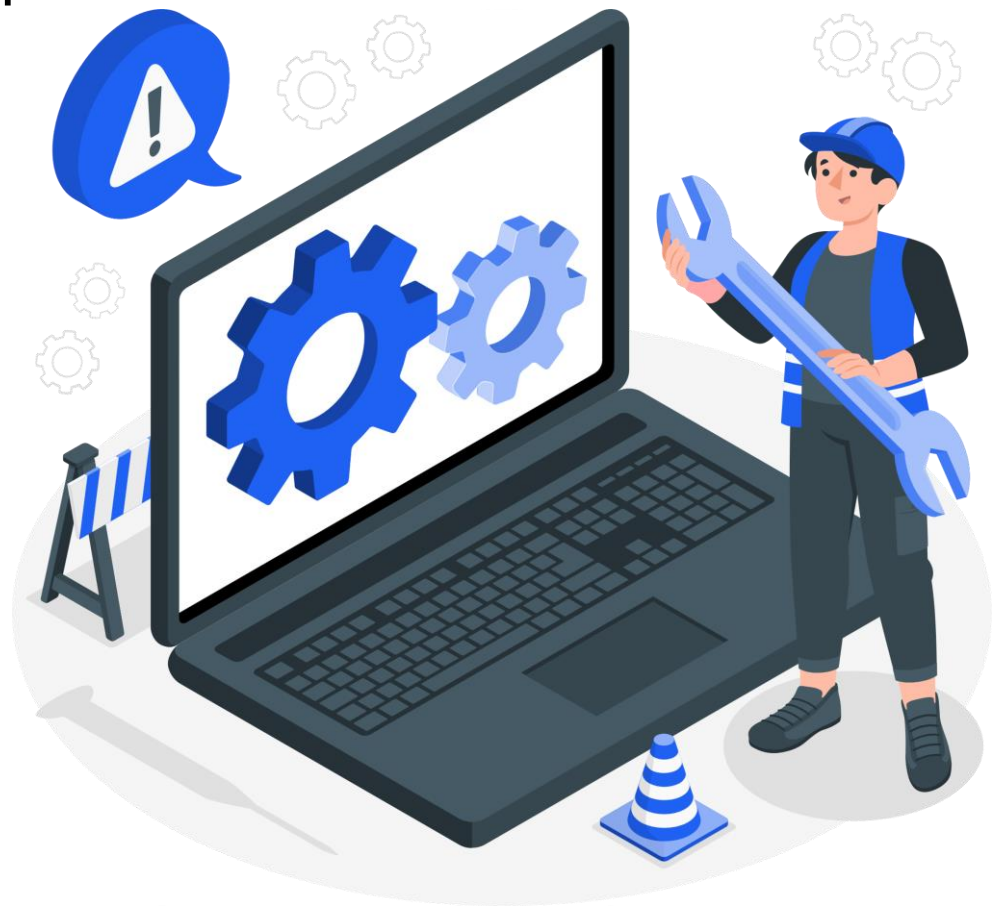
Bạn có thể tạo set bằng set constructor từ các object khác như list, str, range…

EXAMPLE

```
s = "28techtech"
t = set(s)
print(t)
a = [1, 2, 3, 1, 2]
b = set(a)
print(b)
```

OUTPUT

```
{'2', 't', '8', 'c', 'e', 'h'}
{1, 2, 3}
```



3. Thêm phần tử vào set:

Để thêm phần tử vào set ta sử dụng **hàm add()**, nếu bạn thêm 1 phần tử đã có mặt trong set thì phần tử này sẽ không được thêm vào vì set không lưu giá trị trùng.

EXAMPLE

```
s = {"28tech", "abc", "python"}
s.add("abc")
s.add(1)
print(s)
```

OUTPUT

{'abc', 1, '28tech', 'python'}

Để thêm nhiều phần tử vào set bạn có thể sử dụng **hàm update().**

EXAMPLE

```
s = {"28tech", "abc", "python"}
s.update([100, "abc", "28tech", 113])
print(s)
```

OUTPUT

{'python', 100, 'abc', 113, '28tech'}

4. Xóa phần tử khỏi set:

Để xóa phần tử khỏi set ta sử dụng **hàm remove(), discard(), pop(), clear.**

remove() và discard()


```
s = {"28tech", "abc", "python", "C++"}
s.remove("28tech")
print(s)
s.discard("abc")
print(s)
```

OUTPUT

```
{'abc', 'python', 'C++'}
{'python', 'C++'}
```

CHÚ Ý

2 hàm remove và discard đều có chức năng xóa phần tử khỏi set nhưng remove() sẽ sinh ra lỗi Keyerror nếu bạn cố tình xóa một phần tử trong set, ngược lại discar() thì không.

4. Xóa phần tử khỏi set:

pop() và clear()

pop(): Xóa một phần tử ngẫu nhiên trong set.

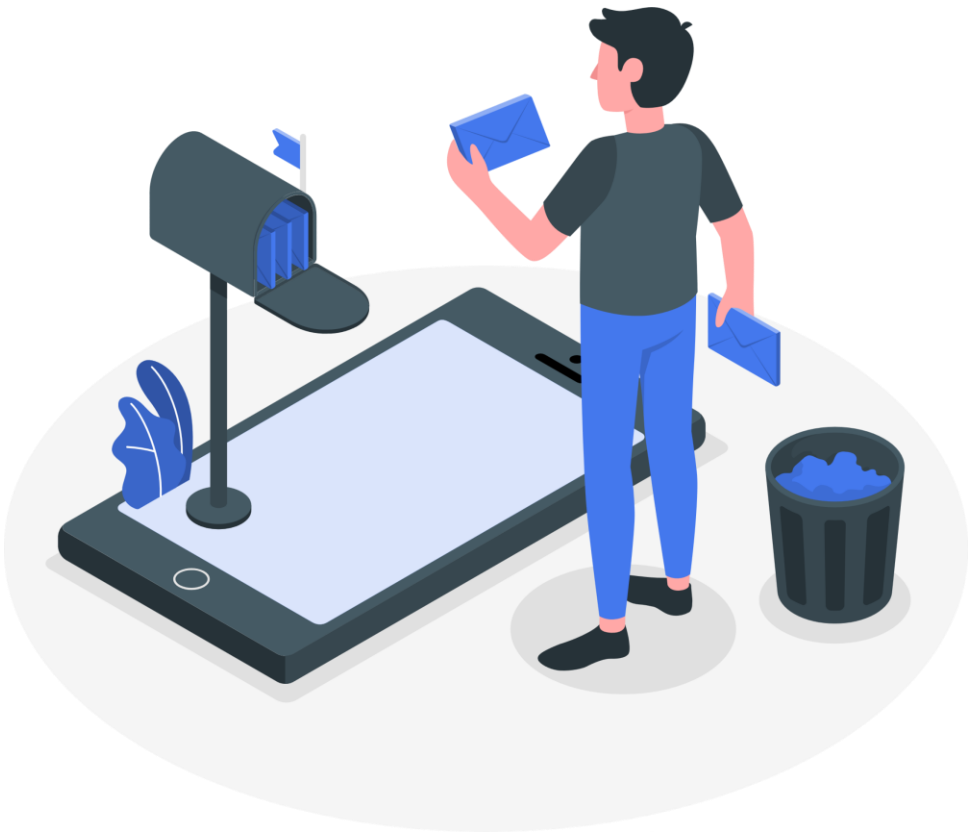
clear(): Xóa toàn bộ các phần tử trong set.

EXAMPLE

```
s = {"28tech", "abc", "python", "C++"}
s.pop()
print(s)
s.clear()
print(s)
```

OUTPUT

```
{'python', 'C++', '28tech'}
set()
```



5. Hàm len(), duyệt set và toán tử in:

Để tìm số lượng phần tử trong set ta sử dụng **hàm len().**


```
s = {"28tech", "abc", "python", "C++"}
print(len(s))
for x in s:
    print(x, end = ' ')
if "28tech" in s :
    print("FOUND")
else:
    print("NOT FOUND")
```

OUTPUT

```
4
abc python C++ 28tech
FOUND
```

6.Các phép toán trên tập hợp:

Union: Phép hợp lấy ra các phần tử thuộc một trong 2 tập hợp, sử dụng **hàm union()** hoặc **phép toán** | để tìm hợp của 2 tập.


```
s = {"28tech", "abc", "python"}
t = {"28tech", "amazon", "python"}
u = s.union(t)
v = s | t
print(s)
print(t)
print(u)
print(v)
```

OUTPUT

```
{'abc', 'python', '28tech'}
{'python', 'amazon', '28tech'}
{'abc', 'python', '28tech', 'amazon'}
{'abc', 'python', '28tech', 'amazon'}
```

6.Các phép toán trên tập hợp:

Intersection: Phép giao lấy ra các phần tử thuộc cả 2 tập, sử dụng **hàm intersection()** hoặc **toán tử &** để tìm giao của 2 tập


```
s = {"28tech", "abc", "python"}
t = {"28tech", "amazon", "python"}
u = s.intersection(t)
v = s & t
print(s)
print(t)
print(u)
print(v)
```

OUTPUT

```
{'abc', 'python', '28tech'}
{'python', 'amazon', '28tech'}
{'python', '28tech'}
{'python', '28tech'}
```

6.Các phép toán trên tập hợp:

Difference: Phép difference của tập A, B lấy ra các phần tử thuộc tập A nhưng không thuộc tập B, bạn có thể sử dụng **hàm difference** hoặc **toán tử -** để tìm difference của 2 tập.


```
s = {"28tech", "abc", "python"}
t = {"28tech", "amazon", "python"}
u = s.difference(t)
v = s - t
print(s)
print(t)
print(u)
print(v)
```

OUTPUT

```
{'abc', 'python', '28tech'}
{'python', 'amazon', '28tech'}
{'abc'}
{'abc'}
```

6.Các phép toán trên tập hợp:

Symmetric Difference: Phép Symmetric Difference của 2 tập A và B lấy ra các phần tử thuộc 1 trong 2 tập nhưng bỏ đi những phần tử thuộc cả 2 tập, bạn có thể sử dụng **hàm symmetric_difference** hoặc **toán tử ^.**


```
s = {"28tech", "abc", "python"}
t = {"28tech", "amazon", "python"}
u = s.symmetric_difference(t)
v = s ^ t
print(s)
print(t)
print(u)
print(v)
```

OUTPUT

```
{'abc', 'python', '28tech'}
{'python', 'amazon', '28tech'}
{'abc', 'amazon'}
{'abc', 'amazon'}
```

7.Các hàm phổ biến của set:

isdisjoint(): Xác định xem 2 tập có phần tử chung nào không ?


```
s = {"28tech", "abc", "python"}
t = {"28tech", "amazon", "python"}
u = {"BKA", "UET", "HCMUS"}
print(s.isdisjoint(t))
print(s.isdisjoint(u))
```

OUTPUT

```
False
True
```

7.Các hàm phổ biến của set:

issubset(): Kiểm tra xem tập này có phải là tập con của tập khác không?


```
s = {"28tech", "abc", "python"}
t = {"28tech", "python"}
u = {"BKA", "UET", "28tech"}
print(t.issubset(s))
print(u.issubset(s))
```

OUTPUT

```
True
False
```

7.Các hàm phổ biến của set:

issuperset(): Kiểm tra xem tập này có phải là cha của tập khác không?


```
s = {"28tech", "abc", "python"}
t = {"28tech", "python"}
u = {"BKA", "UET", "28tech"}
print(s.issuperset(t))
print(u.issuperset(t))
```

OUTPUT

```
True
False
```