Tuple dan Set Struktur data tetap dan data unik

Tuple: Struktur Data Tetap (Immutable)

- **Tuple**
  - **Pengertian: Struktur data yang digunakan untuk menyimpan beberapa item dalam satu variabel. Berbeda dengan list, tuple bersifat immutable, artinya setelah tuple dibuat,tidak bisa mengubah elemen di dalamnya**

Tuple: Struktur Data Tetap (Immutable)

Method pada Tuple

Method	Deskripsi
count()	Mengembalikan jumlah kemunculan nilai tertentu dalam tuple
index()	Mencari nilai dalam tuple dan mengembalikan posisi (indeks) tempat nilai tersebut ditemukan

Penjelasan dan Contoh Setiap Metode

- count()
  - Mengembalikan jumlah kemunculan nilai tertentu dalam tuple

```
x = (1, 2, 3, 2, 4, 2, 5)
result_count = x.count(2)
print("1. count(2):", result_count)
```

Output :

```
count(2): 3
```

Penjelasan dan Contoh Setiap Metode

- index()
  - Mencari nilai dalam tuple dan mengembalikan posisi (indeks) tempat nilai tersebut ditemukan

```
tuple_x = (10, 20, 30, 40, 50)
print("index(20):", tuple_x.index(30))
```

Output :

index(20): 2

Set: Struktur Data dengan Data Unik

- **Set**
  - **Pengertian: Koleksi yang tidak terurut dan tidak dapat memiliki duplikat. Set berfungsi untuk menyimpan elemen yang unik, dan berbeda dengan tuple dan list, set tidak menjaga urutan elemen**

Set: Struktur Data dengan Data Unik

❖ Method pada Set

Method	Shortcut	Deskripsi
add()		Menambahkan elemen ke dalam set
clear()		Menghapus semua elemen dalam set
copy()		Mengembalikan salinan dari set
difference()	-	Mengembalikan set yang berisi perbedaan antara dua set atau lebih
difference_update()	-=	Menghapus item dalam set yang juga ada pada set lain yang ditentukan
discard()		Menghapus item tertentu jika ada dalam set (tidak menimbulkan error jika item tidak ada)
intersection()	&	Mengembalikan set yang berisi irisan (intersection) antara dua set
intersection_update()	&=	Menghapus item dalam set yang tidak ada pada set lain yang ditentukan
isdisjoint()		Mengembalikan True jika dua set tidak memiliki irisan (intersection)

Set: Struktur Data dengan Data Unik

❖ Method pada Set

Method	Shortcut	Deskripsi
issubset()	<= <	Mengembalikan True jika set ini adalah subset dari set lain (semua elemen set ini ada pada set lain)
issuperset()	>= >	Mengembalikan True jika set ini adalah superset dari set lain (set ini mengandung semua elemen set lain).
pop()		Menghapus elemen acak dari set dan mengembalikannya
remove()		Menghapus elemen yang ditentukan dari set (akan menghasilkan error jika elemen tidak ada)
symmetric_difference()	^	Mengembalikan set yang berisi selisih simetris antara dua set
symmetric_difference_update()	^=	Memperbarui set dengan selisih simetris antara set ini dan set lainnya
union()	|	Mengembalikan sebuah **set** yang berisi **gabungan** dari set-set
update()	|=	Memperbarui **set** dengan **gabungan** dari set ini dan set-set lainnya

Penjelasan dan Contoh Setiap Metode

- add()
  - Menambahkan elemen ke dalam set

```
set_a = {1, 2, 3}
set_a.add(5)
print("add(5):", set_a)
```

Output :

```
add(5): {1, 2, 3, 5}
```

Penjelasan dan Contoh Setiap Metode

- clear()
  - Menghapus semua elemen dalam set

```
set_b = {1, 2, 3}
set_b.clear()
print("clear():", set_b)
```

Output :

clear(): set()

Penjelasan dan Contoh Setiap Metode

- copy()
  - Mengembalikan salinan dari set

```
set_c = {1, 2, 3, 4}
result_copy = set_c.copy()
print("copy():", result_copy)
```

Output :

```
copy(): {1, 2, 3, 4}
```

Penjelasan dan Contoh Setiap Metode

- difference()
  - Mengembalikan set yang berisi perbedaan antara dua set atau lebih

```
set_d1 = {1, 2, 3, 4}
set_d2 = {3, 4, 5, 6}
print("difference(set_d2):", set_d1.difference(set_d2))
```

Output :

difference(set_d2): {1, 2}

Penjelasan dan Contoh Setiap Metode

- difference_update()
  - Menghapus item dalam set yang juga ada pada set lain yang ditentukan

```
set_e1 = {1, 2, 3, 4, 7}
set_e2 = {4, 5, 6, 7, 8}
set_e1.difference_update(set_e2)
print("difference_update(set_e2):", set_e1)
```

Output :

```
ifference_update(set_e2): {1, 2, 3}
```

Penjelasan dan Contoh Setiap Metode

- discard()
  - Menghapus item tertentu jika ada dalam set (tidak menimbulkan error jika item tidak ada)

```
set_f = {1, 2, 3}
set_f.discard(2)
print("discard(2):", set_f)
```

Output :

```
discard(2): {1, 3}
```

Penjelasan dan Contoh Setiap Metode

- intersection()
  - Mengembalikan set yang berisi irisan (intersection) antara dua set

```
set_g1 = {1, 2, 3, 4}
set_g2 = {3, 4, 5, 6}
print("intersection(set_g2):", set_g1.intersection(set_g2))
```

Output :

intersection(set_g2): {3, 4}

Penjelasan dan Contoh Setiap Metode

- intersection_update()
  - Menghapus item dalam set yang tidak ada pada set lain yang ditentukan

```
set_h1 = {3, 4, 7}
set_h2 = {3, 4, 5, 6}
set_h1.intersection_update(set_h2)
print("intersection_update(set_h2):", set_h1)
```

Output :

intersection_update(set_h2): {3, 4}

Penjelasan dan Contoh Setiap Metode

- isdisjoint()
  - Mengembalikan True jika dua set tidak memiliki irisan (intersection)

```
set_i1 = {1, 2, 3}
set_i2 = {4, 5, 6}
print("isdisjoint(set_i2):", set_i1.isdisjoint(set_i2))
```

Output :

```
isdisjoint(set_i2): True
```

Penjelasan dan Contoh Setiap Metode

- issubset()
  - Mengembalikan True jika set ini adalah subset dari set lain (semua elemen set ini ada pada set lain)

```
set_j1 = {3, 4}
set_j2 = {1, 2, 3, 4}
print("issubset(set_j2):", set_j1.issubset(set_j2))
```

Output :

issubset(set_j2): True

Penjelasan dan Contoh Setiap Metode

- issuperset()
  - Mengembalikan True jika set ini adalah superset dari set lain (set ini mengandung semua elemen set lain).

```
set_k1 = {1, 2, 3, 4}
set_k2 = {3, 4}
print("issuperset(set_k2):", set_k1.issuperset(set_k2))
```

Output :

issuperset(set_k2): True

Penjelasan dan Contoh Setiap Metode

- pop()
  - Menghapus elemen acak dari set dan mengembalikannya

```
set_1 = {1, 2, 3}
popped = set_1.pop()
print("pop():", popped)
print("Set setelah pop:", set_1)
```

Output :

pop(): 1

Set setelah pop: {2, 3}

Penjelasan dan Contoh Setiap Metode

- remove()
  - Menghapus elemen yang ditentukan dari set (akan menghasilkan error jika elemen tidak ada)

```
set_m = {1, 2, 3}
set_m.remove(2)
print("remove(2):", set_m)
```

Output :

```
remove(2): {1, 3}
```

Penjelasan dan Contoh Setiap Metode

- symmetric_difference()
  - Mengembalikan set yang berisi selisih simetris antara dua set

```
set_n1 = {1, 2, 3, 4}
set_n2 = {3, 4, 5, 6}
print("symmetric_difference(set_n2):",set_n1.symmetric_difference(set_n2))
```

Output :
symmetric_difference(set_n2): {1, 2, 5, 6}

Penjelasan dan Contoh Setiap Metode

- symmetric_difference_update()
  - Memperbarui set dengan selisih simetris antara set ini dan set lainnya

```
set_o1 = {1, 2, 3}
set_o2 = {3, 4, 5}
set_o1.symmetric_difference_update(set_o2)
print("symmetric_difference_update(set_o2):", set_o1)
```

Output :

```
symmetric_difference_update(set_o2): {1, 2, 4, 5}
```

Penjelasan dan Contoh Setiap Metode

- union()
  - Mengembalikan sebuah **set** yang berisi **gabungan** dari set-set

```
set_p1 = {1, 2, 3}
set_p2 = {3, 4, 5}
print("union(set_p2):", set_p1.union(set_p2))
```

Output :

```
union(set_p2): {1, 2, 3, 4, 5}
```

Penjelasan dan Contoh Setiap Metode

- update()
  - Memperbarui **set** dengan **gabungan** dari set ini dan set-set lainnya

```
set_q1 = {1, 2}
set_q2 = {3, 4}
set_q1.update(set_q2)
print("update(set_q2):", set_q1
```

Output :

update(set_q2): {1, 2, 3, 4}

Kesimpulan penggunaan Tuple dan Set dalam algoritma

- **Tuple : Cocok untuk *menyimpan pasangan data tetap* seperti (kata, jumlah).**

- **Set: Cocok untuk *operasi himpunan* seperti mencari kesamaan dan perbedaan data**