Setler

Pythonda set matematikteki kümeleri karşılar. Set sırasız, değiştirilebilir ve aynı elemandan birden fazla bulundurmayan bir iterabledır. Setler sırasız olduğu için listeler ve tuplelarda yapılan indexleme yapılamaz.

3.1. SET OLUSTURMAK

```
Python tupleları farklı değişken türlerini bir arada tutabilir. Parantez zorunlu
değildir. tuple(iterable) fonksiyonu ile de tuple yapılabilir. Setler {} kullanılarak ya
da set(iterable) fonksiyonu ile oluşturulabilir.
set1 = {'a', 'b', 'c'}
set2 = set(['a', 'b', 'c'])# {'c', 'a', 'b'} belli bir sırası yoktur, yerleri değişebilir
set3 = {42, 'a', 3.1415, None}  # farklı tipte elemanları olabilir
newset = {1, 2, 3, 3, 3}  # aynı elemandan birden fazla yazsak bile sadece bir kez alır
print(newset)  # {1, 2, 3}

Set elemanları immutable(değişmez) olmak zorundadır. Listeler ya da dictionaryler
mutable(değişebilir) oldukları için setlere {} içinde eleman olarak atayamayız.
set1 = {['a', 'b', 'c']}
print(set1)  # TypeError: unhashable type: 'list'
set2 = {{'a': 1, 'b': 2}}
print(set2)  # TypeError: unhashable type: 'dict'

Setler listenin aksine aynı elemandan birden fazla bulunduramaz.
s = "aabbccdd"
# string bir iterable olduğu için fonksiyonlarla liste ya da set yapabiliriz.
print(list(s)) # ['a', 'a', 'b', 'b', 'c', 'c', 'd', 'd']
print(set(s))  # {'a', 'b', 'd', 'c'}

Setler boş olabilir fakat Python boş {}leri dictionary olarak görür.
set1 = set()
print(type(set1))  # <class 'set'>
set2 = {}
print(type(set2))  # <class 'dict'>
```

3.2. SET ELEMAN SAYISI VE KONTROLU

```
Liste ve tuple'lar gibi len() fonksiyonu kullanılabilir.
set1 = {'a', 'b', 'c'}
print(len(set1))   # 3

in / not in operatorları
print('a' in set1) # True
print('k' not in set1) # True

if ('b' in set1): # True olduğu için if'in içine uğrar
  print('b')      # 'b' yazdırır

if ('l' in set1): # False olduğundan if'in içine uğramaz
  print('l')      # böylece 'l' yazdırmaz
```

3.3. SETTEKI ELEMANLARA ULASMAK

```
days=set(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
for d in days:
  print(d)
```

```
Wed
Sun
Fri
Tue
Mon
Thu
Sat
```

3.4. SETLER ILE OPERASYONLAR

```
Birden fazla set belirtilmişse operasyonlar soldan sağa doğrudur. Örneğin a - b - c ise
ilk a - b bulunur sonra o setin c'den farkı bulunur.
# Aşağıdaki tüm operasyonlar için bu kümeleri kullanacağız
set1 = {'a', 'b', 'c'}
set2 = {'c', 'd', 'e'}
set3 = {'d', 'c'}

Union (birleşim)
x1.union(x2[, x3 ...])
x1 | x2 [| x3 ...]
set1.union(set2)  # {'b', 'd', 'c', 'e', 'a'}
set1 | set2       # {'b', 'd', 'c', 'e', 'a'}
print(set1)    # {'c', 'b', 'a'} aynı kalır
print(set2)    # {'e', 'c', 'd'} aynı kalır

Intersection (kesişim)
x1.intersection(x2[, x3 ...])
x1 & x2 [& x3 ...]
set1.intersection(set2, set3)  # {'c'}
set1 & set2 & set3             # {'c'}

Difference (fark)
x1.difference(x2[, x3 ...])
x1 - x2 [- x3 ...]
set1.difference(set2) # {'a', 'b'}
set1 - set2           # {'a', 'b'}

Symmetric difference (ortak bulunmayan elemaları geri döndürür)
x1.symmetric_difference(x2)
x1 ^ x2 [^ x3 ...]
set1.symmetric_difference(set2)  # {'b', 'a', 'd', 'e'}
set1 ^ set2  # {'b', 'a', 'd', 'e'}

Isdisjoint (setler ayrık küme mi diye bakar)
x1.isdisjoint(x2)
set1.isdisjoint(set2)  # False (çünkü 'c' ortak)
set3.isdisjoint(set1)  # False (çünkü 'c' ortak)

Issubset (setler birbirinin alt kümesi mi diye bakar)
x1.issubset(x2)
x1 <= x2
set2 <= set1         # False
set1.issubset(set1)  # True (Her küme kendisinin alt kümesi)
x1 < x2 (proper subset)
set1 < set1 # False (Bu operator alt kümenin o kümeye eşit olmasına False verir)

Issuperset (Bir küme diğer kümeyi kapsıyor mu diye bakar)
x1.issuperset(x2)
x1 >= x2
set3.issuperset(set1) # False
set1 >= set1          # True
x1 > x2 (proper superset)
set1 > set1  # False
```

3.5. SETLERI OPERASYONLAR ILE DÜZENLEMEK (AUGMENTED ASSIGNMENT OPERATORS)

```
Update (seti küme birleşimleri olarak değiştirir)
x1.update(x2[, x3 ...])
x1 |= x2 [| x3 ...]
set1.update(set2)
print(set1)  # {'c', 'a', 'd', 'e', 'b'} set1'i değiştirdik ama set2 aynı kalır.

Intersection update (seti küme kesişimleri olarak değiştirir)
x1.intersection_update(x2[, x3 ...])
x1 &= x2 [& x3 ...]
```

```
set2.intersection_update(set3)
print(set2)  # {'c', 'd'} set2 değişir, set3 aynı kalır

Aynı şekilde fark ve ortak bulunmayan elemanlar için asıl seti değiştiren methodlar:
x1.difference_update(x2[, x3 ...])
x1 -= x2 [| x3 ...]

x1.symmetric_difference_update(x2)
x1 ^= x2
```

set_ismi.add(eleman) methodu sete yeni eleman ekler
set_ismi.clear() methodu seti temizler, temizlenen seti yazdırırsak set() yazdırır
set_ismi.remove(eleman) methodu setten o elemanı siler, eğer o eleman sette yoksa hata mesajı verir.
set_ismi.discard(eleman) methodu setten o elemanı siler, eğer o eleman sette yoksa hiçbir şey yapmaz.
set_ismi.pop() methodu setin herhangi bir elemanını siler(random). Eğer set boşsa hata mesajı verir.

3.6. FROZEN SETLER

```
Frozen setler, setlerden sadece bir yönü ile ayrılır. Frozen setler
immutable(değişmez)dir. Setlerde uygulanabilen ve seti değiştirmeyen tüm operasyonları
frozen setlerde de uygulayabiliriz.
f_set = frozenset([1,2,3]) # frozen set oluşturmak

Frozen setler değiştirilmez olmasına rağmen augmented assignment operatorlerini(3.5.)
kullanabiliriz.
f_set = frozenset(['a', 'b', 'c'])
set1 = {'a'}
f_set &= set1
print(f_set)  # frozenset({'a'})

Neden? Çünkü Python augmented assignmentları frozen setlerde uygulamıyor. f_set &= set1
aslında f_set = f_set & set1 'e eşit. Orijinal f_set'i değiştirmiyor, f_set'e yeni
değerler atıyor.

Frozen setler ne için kullanılıyor? (3.1.)'de de gördüğümüz gibi setlere eleman olarak
mutable(değişebilir) elemanlar veremiyoruz. Eğer sete eleman olarak başka setler(mutable)
tanımlamak istersek bunu frozen setler(immutable) ile yapmamız gerekir.
f_set1 = frozenset([1, 2])
f_set2 = frozenset([3, 4])
set1 = {1, 2}
set2 = {3, 4}
yeniset = {set1, set2}       #  TypeError: unhashable type: 'set'
yeniset = {f_set1, f_set2}   #  {frozenset({3, 4}), frozenset({1, 2})}
```

EXTRA KAYNAKLAR:

(EN) SETS IN PYTHON:

1. https://www.geeksforgeeks.org/sets-in-python/
2. https://access2learn.com/tutorial/python/pythons-set-methods-and-functions/

(FR) LES ENSEMBLES:

1. https://www.developpement-informatique.com/article/226/les-ensembles-en-python
2. http://python-simple.com/python-langage/set.php

(TR) SETLER & KISITLANMIS SETLER:

1. https://www.mobilhanem.com/python-set-kume-ve-fronzsetkisitlanmis-kume/