Porównanie typów danych

| ⇣ ⇢ | **Elementy**  **unikalne** | **Kolejność** | **Zmiana konkretnego elementu** | **Nowe elementy** |
| --- | --- | --- | --- | --- |
| **LISTY** | NIE | TAK | TAK | TAK |
| **KROTKI** | NIE | TAK | NIE | NIE |
| **SŁOWNIKI** | TAK | NIE | TAK | TAK |
| **ZBIORY** | TAK | NIE | NIE | TAK |

ZBIORY BONUS W POSTACI  **| - ^ &**

**Zbiór -** pojemnik do przechowywania danych, tworzymy go za pomocą { } zbiory nie mają kluczy, tylko wartości. W zbiorach elementy muszą być unikalne tzn. że nie mogą się powtarzać. Dane pobierane z bazy danych w zbiorach zmieniają kolejność. Nie jesteśmy w stanie zmienić konkretnego elementu, natomiast do zbioru można dodać element.

Przykładowy zbiór

A = {1, 4, 20, -4, 6}

Aby dodać element do zbioru:

add - (z ang.dodać)

A.add(7)

print(A)

Aby przetworzyć listę A = [1, 4, 20, -4, 20]

w zbiór o unikalnych wartościach i usunąć duplikaty używamy funkcji set

print(set(A))

Wykonywanie dodatkowych operacji na zbiorach:

A = {1, 4, 20, -4, 6}

B = {2, 1, 25, 40, 20}

* sprawdzanie wspólnych elementów w zbiorach (koniunkcja)

| print(A&B) |
| --- |

* jakie są wszystkie elementy w obu zbiorach (suma)

| print(A|B) |
| --- |

* zbiór elementów, które były w A, ale nie ma ich w B

| print(A-B) |
| --- |

* alternatywa wykluczająca tzn. wyklucza wspólne wartości

print(A^B)

* discard - usunięcie elementu ze zbioru np. aby usunąć ze zbioru A element o wartości: 20

| A.discard(20)  print(A) |
| --- |