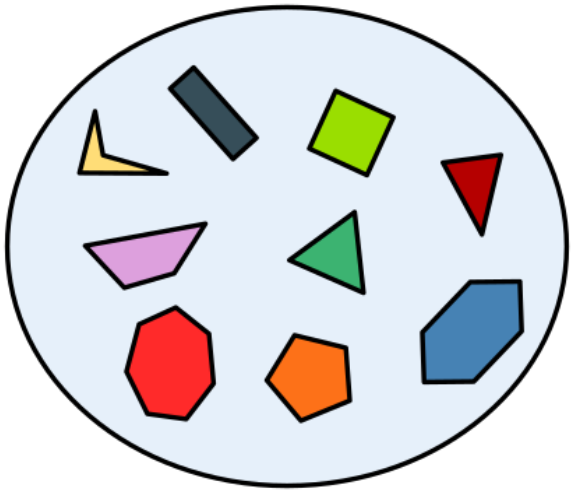
ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

@ZTUEDUUA

АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРИЗОВАНИХ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНИХ СИСТЕМ

Лекція 10
Тема: Множини

1. Створення множини.
2. Дублювання елементів у множині.
3. Додавання, оновлення та видалення елементів множини.
4. Операція з множинами в Python.
5. Перевірка, чи є дві множини рівними.



1. Створення множини.

В Python для створення множини всі елементи поміщають усередині фігурних дужок {}, розділених комами.

Множина може містити будь-яку кількість елементів, і вони можуть бути різних типів **(int, float, кортеж, рядки** тощо). Але множина не може мати змінювані елементи, такі як списки, словники або інші множини.

Розглянемо приклад:

```python
# Множина цілочисленного типу
student_id = {112, 114, 116, 118, 115}
print('Student ID:', student_id)

# Множина рядкового типу
vowel_letters = {'a', 'e', 'i', 'o', 'u'}
print('Vowel Letters:', vowel_letters)

# Множина змішаного типу
mixed_set = {'Hello', 101, -2, 'Bye'}
print('Set of mixed data types:', mixed_set)
```

Результат:

Student ID: {112, 114, 115, 116, 118}
Vowel Letters: {'u', 'a', 'e', 'i', 'o'}
Set of mixed data types: {'Hello', 'Bye', 101, -2}

Тут створено різні типи множин, помістивши всі елементи усередині фігурних дужок {}.

Створення порожньої множини

Порожні фігурні дужки створюють порожній словник у Python. А для створення порожньої множини потрібно використати функцію set() без жодних аргументів. Наприклад:

```
# Створення порожньої множини
empty_set = set()

# Створення порожнього словника
empty_dictionary = { }

# Перевірка типу даних empty_set
print('Data type of empty_set:', type(empty_set))

# Перевірка типу даних dictionary_set
print('Data type of empty_dictionary', type(empty_dictionary))
```

Результат:

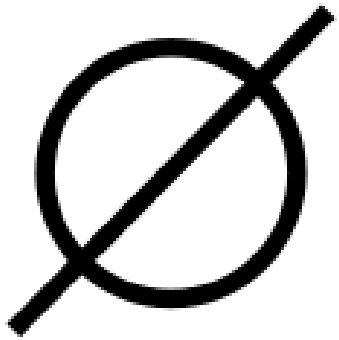
Data type of empty_set: <class 'set'> Data type of empty_dictionary <class 'dict'>

Тут:

empty_set — порожня множина, створена за допомогою set();

empty_dictionary — порожній словник, створений за допомогою {}.

Використаши функцію type() для визначення типу даних empty_set та empty_dictionary.



2.Дублювання елементів у множині

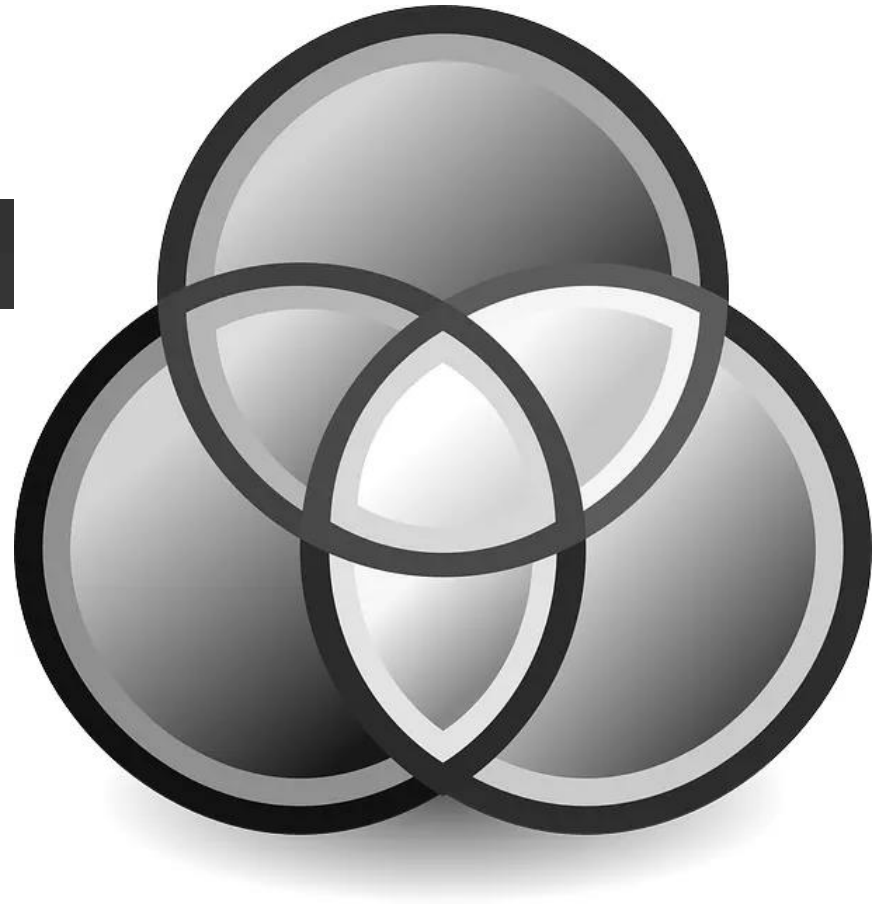
Подивимося, що станеться, якщо ми спробуємо включити в набір елементи, що дублюються.

```
1 numbers = {2, 4, 6, 6, 2, 8}
2 print(numbers)
```

Результат:

```
{8, 2, 4, 6}
```

Тут бачимо, що у множині немає однакових елементів, оскільки вона може містити дублі.



3.Додавання, оновлення та видалення елементів множини в Python

Множини змінювані. Однак, оскільки вони невпорядковані, індексування не має сенсу. Ми не можемо отримати доступ або змінити елемент множини за допомогою індексації або зрізу. Тип даних set не підтримує цього.

Додавання елемента до множини
В Python метод add() використовується для додавання елемента до множини. Наприклад:

Тут add() додає значення 32 до нашої множини.

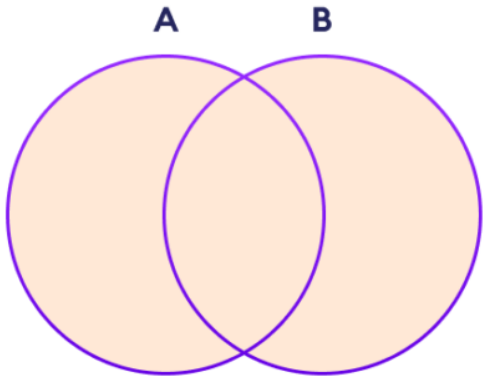
```
numbers.add(32)
```

Тут ми створили множину з ім’ям numbers. Зверніть увагу на рядок:

```
numbers = {21, 34, 54, 12}

print('Initial Set:',numbers)

# Використання методу add()
numbers.add(32)

print('Updated Set:', numbers)
```

Результат:

```
Initial Set: {34, 12, 21, 54}
Updated Set: {32, 34, 12, 21, 54}
```



Оновлення множини

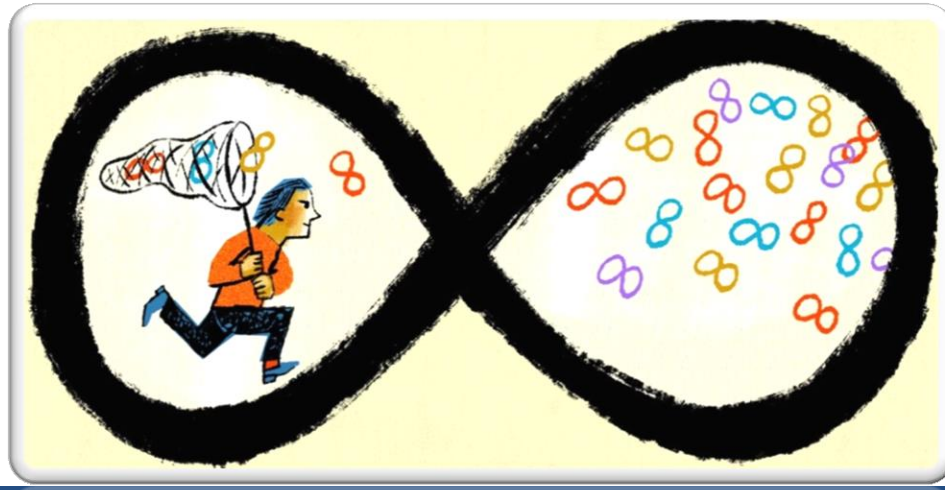
В Python метод update() використовується для оновлення множини елементів інших типів даних (списки, кортежі тощо). Наприклад:

```
1 companies = {'Lacoste', 'Ralph Lauren'}
2 tech_companies = ['apple', 'google', 'apple']
3
4 companies.update(tech_companies)
5
6 print(companies)
```

Результат:

```
{'Lacoste', 'Ralph Lauren', 'apple', 'google'}
```

Тут всі унікальні елементи **tech_companies** додаються до множини companies.



Видалення елемента з множини

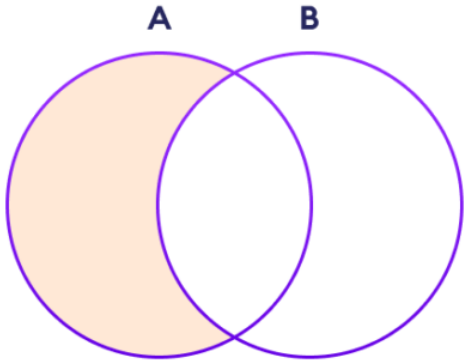
В Python метод discard() використовується для видалення зазначеного елемента з множини. Наприклад:

```
languages = {'Swift', 'Java', 'Python'}

print('Initial Set:',languages)

# Видалення 'Java' з множини
removedValue = languages.discard('Java')

print('Set after remove():', languages)
```

Результат:

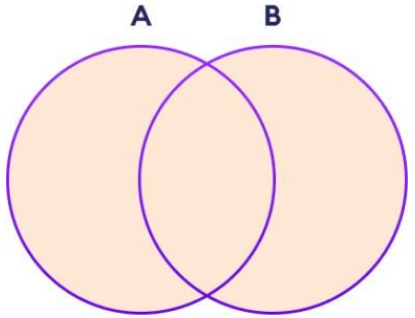
```
Initial Set: {'Python', 'Swift', 'Java'}
Set after remove(): {'Python', 'Swift'}
```

Тут ми використали метод discard(), щоб видалити 'Java' з множини language.



4.Операції з множинами в Python

Python надає різні вбудовані методи для виконання математичних операцій з множинами, таких як об’єднання, перетин, різниця та симетрична різниця.



Об’єднання множин

Об’єднання двох множин A та B включає всі елементи множин A та B.

Для виконання операції об’єднання множин використовується оператор або метод union().
Наприклад:

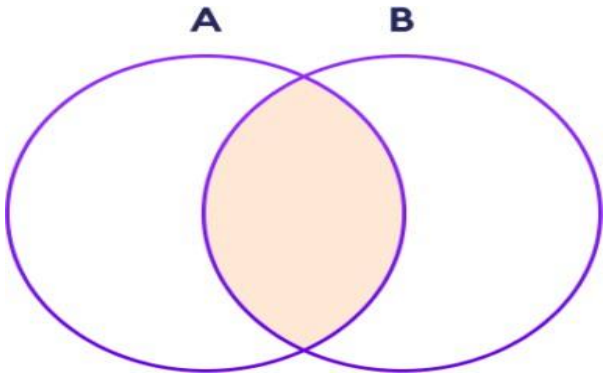
```
# Перша множина
A = {1, 3, 5}

# Друга множина
B = {0, 2, 4}

# Виконання операції об'єднання за допомогою |
print('Union using |:', A | B)

# Виконання операції об'єднання за допомогою union()
print('Union using union():', A.union(B))
```

Результат:

```
Union using |: {0, 1, 2, 3, 4, 5}
Union using union(): {0, 1, 2, 3, 4, 5}
```

Перетин множин

Перетин двох множин A і B включає загальні елементи між множинами A та B.



В Python для виконання операції перетину множин використовується оператор & або метод intersection(). Наприклад:

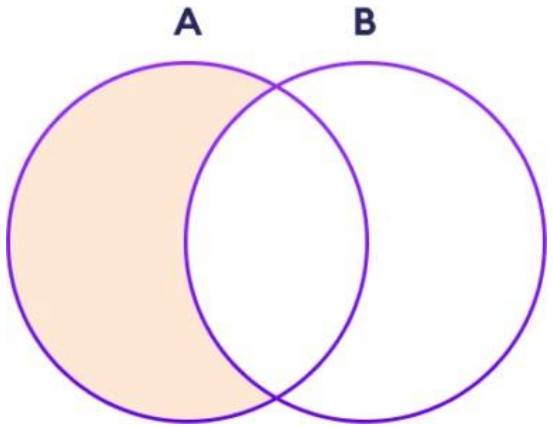
```python
# Перша множина
A = {1, 3, 5}

# Друга множина
B = {1, 2, 3}

# Виконання операції перетину за допомогою &
print('Intersection using &:', A & B)

# Виконання операції перетину за допомогою intersection()
print('Intersection using intersection():', A.intersection(B))
```

Результат:

```
Intersection using &: {1, 3}
Intersection using intersection(): {1, 3}
```

Різниця множин

Різниця між двома множинами A і B включає елементи множини A, яких немає в множині B.



В Python для виконання операції різниці між двома множинами використовується оператор - або метод difference(). Наприклад:

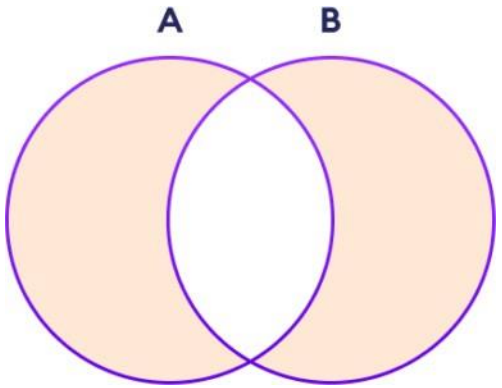
```
# Перша множина
A = {2, 3, 5}

# Друга множина
B = {1, 2, 6}

# Виконання операції різниці за допомогою -
print('Difference using -:', A - B)

# Виконання операції різниці за допомогою difference()
print('Difference using difference():', A.difference(B))
```

Результат:

```
Difference using -: {3, 5}
Difference using difference(): {3, 5}
```

Симетрична різниця множин

Симетрична різниця двох множин A і B включає всі елементи A і B без спільних елементів.



В Python для виконання операції симетричної різниці між двома множинами використовується оператор ^ або метод symmetric_difference().

Наприклад:

```
1  # Перша множина
2  A = {2, 3, 5}
3
4  # Друга множина
5  B = {1, 2, 6}
6
7  # Виконання операції симетричної різниці за допомогою &
8  print('using ^:', A ^ B)
9
10 # Виконання операції симетричної різниці за допомогою symmetric_difference()
11 print('using symmetric_difference():', A.symmetric_difference(B))
```

Результат:

```
using ^: {1, 3, 5, 6}
using symmetric_difference(): {1, 3, 5, 6}
```

5. Перевірка, чи є дві множини рівними

В Python оператор == використовується, щоб перевірити, чи рівні дві множини. Наприклад:

```
# Перша множина
A = {1, 3, 5}

# Друга множина
B = {3, 5, 1}

# Перевіряємо, чи рівні дві множини
if A == B:
    print('Set A and Set B are equal')
else:
    print('Set A and Set B are not equal')
```

Результат:

```
Set A and Set B are equal
```

Тут A та B мають однакові елементи, тому умова

```
if A == B
```

обчислюється в True. Отже, виконується print('Set A and Set B are equal') всередині if.

@ZTUEDUUA

ЖИТОМИРСЬКА ПОЛІТЕХНІКА

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

- **Розвиваємо лідерів**
- **Створюємо інновації**
- **Змінюємо світ на краще**