

# Програмування

---

## ТЕМА 11. МНОЖИНИ

# Множини

---

Тип множини використовують у задачах, у яких має значення тільки належність або неналежність елемента деякій множині.

В основному реалізація множин у програмуванні повторює відомі операції та відношення для множин у математиці.

Множини, є такими, що змінюються (mutable).

# Носій типу множина

---

Множина позначається включенням її елементів у фігурні дужки через кому.

$\{x_1, \dots, x_n\}$ .

Порожня множина позначається `set()`, щоб уникнути плутанини зі словниками.

Нехай множина  $M$  є носієм типу елементів  $t$ .

Тоді носієм типу множини елементів типу  $t$  буде

$M_s = 2^M$  – множина всіх підмножин множини  $M$ .

# Основні операції для МНОЖИН

Операція	Опис
$\{x_1, \dots, x_n\}$	Створити множину з елементів $x_1, \dots, x_n$
set()	Порожня множина
set(x)	Перетворення x у множину (x повинно належати типу, що ітерується)
$a \mid b$	Об'єднання множин $a \cup b$
$a \& b$	Перетин множин $a \cap b$
$a - b$	Різниця множин $a \setminus b$
$a \wedge b$	Симетрична різниця множин a та b
len(a)	Довжина a – кількість елементів у множині
min(a)	Найменший елемент множини a
max(a)	Найбільший елемент множини a
a.copy()	Повертає копію множини a

# Основні відношення для МНОЖИН

---

Для множин визначено відношення з  $Rel = \{==, !=, >, <, >=, <= \}$  а також  $in, not\ in$ .

Відношення  $a == b$  означає рівність множин.

Відношення  $a != b \equiv \text{not } (a == b)$ .

Відношення  $a < b$  означає включення множини  $a$  у множини  $b$ .

Відношення  $a <= b$  означає нестроге включення множини  $a$  у множини  $b$ .

Відношення  $a > b \equiv b < a$ .

Відношення  $a >= b \equiv b <= a$ .

$x \text{ in } a == \text{True}$ , коли  $x$  входить у  $a$

$x \text{ not in } a == \text{True}$ , коли  $x$  не входить у  $a$

# Основні інструкції для МНОЖИН

---

Для множин визначено присвоєння та виведення.

**a = b, print(a)**

Введення не визначено, тому треба вводити множину поелементно.

Визначено також цикл по всіх елементах множини

**for x in a:**

***P***

# Приклад

---

Перевірити, чи складаються 2 рядки  $s_1$  та  $s_2$  з одних і тих же символів. Тобто, чи справедливе твердження, що кожний символ  $s_1$  входить у  $s_2$  та кожний символ  $s_2$  входить у  $s_1$ .

# Додаткові операції для МНОЖИН

---

Операція	Опис
<code>a.union(b)</code>	Об'єднання множин $a \cup b$ , те ж саме, що й $a   b$
<code>a.intersection(b)</code>	Перетин множин $a \cap b$ , те ж саме, що й $a \& b$
<code>a.difference(b)</code>	Різниця множин $a \setminus b$ , те ж саме, що й $a - b$
<code>a.symmetric_difference(b)</code>	Симетрична різниця множин $a$ та $b$ , те ж саме, що й $a \wedge b$



# Додаткові відношення для МНОЖИН

Відношення	Опис
<code>a.isdisjoint(b)</code>	True, коли перетин $a$ та $b$ – порожня множина, те ж саме, що й $a \& b == \text{set}()$
<code>a.issubset(b)</code>	Чи є $a$ підмножиною $b$ , те ж саме, що й $a \leq b$
<code>a.issuperset(b)</code>	Чи є $b$ підмножиною $a$ , те ж саме, що й $b \leq a$

# Додаткові інструкції для МНОЖИН

Інструкція	Опис
<code>a.add(x)</code>	Додає елемент <code>x</code> до множини <code>a</code>
<code>a.remove(x)</code>	Видаляє елемент <code>x</code> з множини <code>a</code> . Якщо елемента немає у множині, - дає помилку
<code>a.discard(x)</code>	Видаляє елемент <code>x</code> з множини <code>a</code> , якщо цей елемент є у множині
<code>a.pop()</code>	Видаляє довільний елемент з множини <code>a</code> . Якщо множина порожня та в ній немає жодного елемента, - дає помилку
<code>a.clear()</code>	Видаляє всі елементи з множини <code>a</code>
<code>a.update(b)</code>	Оновити <code>a</code> значенням об'єднання <code>a</code> та <code>b</code> . Те ж саме, що й <code>a = a   b</code>

# Додаткові інструкції для множин.2

---

Інструкція	Опис
<code>a.intersection_update(b)</code>	Оновити $a$ значенням перетину $a$ та $b$ . Те ж саме, що й $a = a \& b$
<code>a.difference_update(b)</code>	Оновити $a$ значенням різниці $a$ та $b$ . Те ж саме, що й $a = a - b$
<code>a.symmetric_difference_update(b)</code>	Оновити $a$ значенням симетричної різниці $a$ та $b$ . Те ж саме, що й $a = a \wedge b$

# Незмінні множини frozenset

---

У Python окрім звичайних множин є також множини, що не змінюються (immutable), frozenset.

Для створення такої множини треба писати

`frozenset(t)`, де `t` – вираз типу, що ітерується.

Для `frozenset` визначено всі ті ж основні операції, відношення та інструкції, що й для звичайних множин `set`.

Також визначені додаткові операції та відношення, наведені вище.

Не визначені тільки додаткові інструкції.

`frozenset` можуть фігурувати у виразах разом із звичайними множинами. При цьому результат виразу буде того типу, до якого належить перший операнд операції (`set` або `frozenset`).

`frozenset` використовують тоді, коли множина після створення не змінюється і потрібна більша швидкодія у порівнянні з використанням звичайних множин `set`.

# Множиноутворення

---

Множиноутворення (set comprehension) – це вираз, результатом якого є множина.

Множиноутворення схоже на спискоутворення та словникоутворення.

Вираз має такий синтаксис:

**`{e(x) for x in t if F}`**

- де  $e(x)$  – вираз,  $t$  – вираз типу, що ітерується,  $F$  – умова.

Python вибирає всі  $x$  з  $t$ , які задовольняють умову  $F$ , додає у множину  $e(x)$  та повертає отриману множину.

Якщо умова  $F$  відсутня, то  $if F$  опускають.

# Приклад «Тур коня»

---

«Тур коня». Знайти шлях шахової фігури «кінь» з поля шахової дошки  $(x, m)$  на поле  $(y, k)$ , де  $x, y$  - вертикалі (позначаються літерами від а до h),  $m, k$  - горизонталі (позначаються цифрами від 1 до 8).

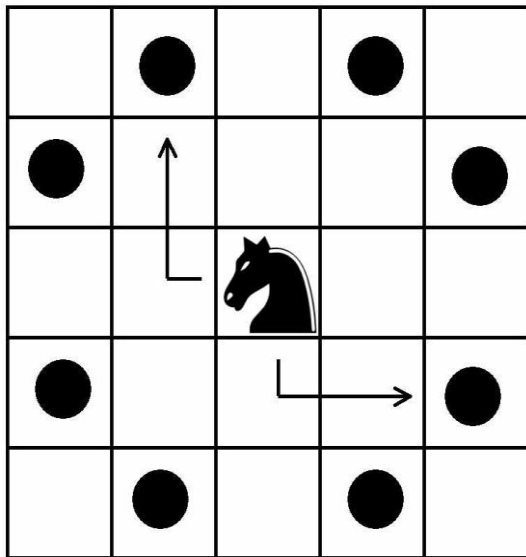
Програма, яку ми розбирали у попередній темі, знаходила шлях коня, але цей шлях був неоптимальний через вибір наступного ходу, починаючи з початку списку ходів.


Побудуємо програму, яка буде знаходити більш оптимальний шлях.

# Приклад «Тур коня».2

На  $i$ -му кроці будуємо граничну множину полів шахової дошки, яких кінь може досягти з початкової позиції за  $i$  кроків.

Коли у множину потрапляє кінцеве поле, йде повернення до початкової позиції з побудовою шляху.



2	3	4	1	2	1	4	3
3	2	1	2	3	2	1	2
2	3	2	3		3	2	3
3	2	1	2	3	2	1	2
2	3	4	1	2	1	4	3
3	2	3	2	3	2	3	2
4	3	2	3	2	3	2	3
3	4	3	4	3	4	3	4

# Резюме

---

Ми розглянули:

1. Множини: носій, основні операції, відношення та інструкції для множин.
2. Додаткові операції, відношення та інструкції для множин.
3. Множини, що не змінюються.
4. Множиноутворення.



# Де прочитати

---

1. Обвінцев О.В. Інформатика та програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. A Byte of Python (Russian) Версія 2.01 Swaroop С Н (Translated by Vladimir Smolyar),  
<http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf>
3. Марк Лутц, Изучаем Python, 4-е издание, 2010, Символ-Плюс
4. Python 3.4.3 documentation
5. Бублик В.В., Личман В.В., Обвінцев О.В.. Інформатика та програмування. Електронний конспект лекцій, 2003 р.,
6. [http://www.python-course.eu/python3\\_sets\\_frozensets.php](http://www.python-course.eu/python3_sets_frozensets.php)