Множини та відображення

№ уроку: 7 **Курс:** Python Starter

Засоби навчання: PyCharm

Огляд, мета та призначення уроку

Після завершення уроку учні матимуть уявлення про множини та відображення в Python та основні стандартні класи множин і відображень, їх призначення та використання.

Вивчивши матеріал даного заняття, учень зможе:

- Мати уявлення про множини
- Використовувати класи set та frozenset
- Мати уявлення про відображення та словники
- Використовувати клас dict та інші класи з модуля collections
- Використовувати представлення словників
- Створювати функції із довільною кількістю іменованих параметрів
- Розпаковувати словники та інші відображення в іменовані параметри функції

Зміст уроку

1. Що таке множини?
2. Створення множин
3. Зміна множин
4. Видалення елементів з множини
5. Операції з множинами Python
6. Що таке відображення?
7. Сортування словників

Резюме

Множина – це невпорядкований набір елементів. Кожен елемент множини має бути унікальним і незмінюваним.
Однак сама множина змінювана. Ми можемо додавати або видаляти елементи з неї. Множини також можуть використовуватися для виконання математичних операцій, таких як **об'єднання**, **перетин**, **симетрична різниця** і так далі.

Створення множин
Множини створюються шляхом поміщення всіх елементів у фігурні дужки {}, розділених комами, або за допомогою вбудованого класу set(). Зверніть увагу, що порядок елементів у множинах не зберігається.
Вона може мати будь-яку кількість елементів, і вони можуть бути різних типів (**цілі числа, числа з плаваючою комою, кортеж, рядок**). Множина **не може** мати в якості своїх елементів змінювані елементи, такі як списки або словники.

```
my_set = {1, 2, 3}
print(my_set)
# {1, 2, 3}

my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
# {1.0, (1, 2, 3), 'Hello'}
```

Наступний приклад добре відображає головну характеристику множин:

```
my_set = {1, 2, 3, 4, 3, 2}
```


CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
print(my_set)

# множина не може зберігати дублікати (дублікати видаляються автоматично)
# Виведення: {1, 2, 3, 4}
```

Множини можна створити з будь-якого ітерованого об'єкта, наприклад, списку:

```
my_set = set([1, 2, 3, 2])
print(my_set)

# Виведення: {1, 2, 3}
```

Ми **не можемо** створити множину зі змінюваних об'єктів::

```
my_set = {1, 2, [3, 4]}

# Traceback (most recent call last):
#   File "<string>", line 15, in <module>
#     my_set = {1, 2, [3, 4]}
# TypeError: unhashable type: 'list'
```

Створення порожньої множини трохи складніше, ніж порожній список. Справа в тому, що синтаксис створення словника та множини дуже схожий. Тому запис my_var = {} створить порожній словник, а не множину. Щоб створити пусту множину, потрібно написати: my_var = set().

Зміна множини

Множини змінювані. Але оскільки вони неупорядковані, індексація не має значення.

Ми не можемо отримати доступ до елемента множини або змінити його за допомогою індексації чи зрізу.

Ми можемо додати один елемент за допомогою методу add() та кількох елементів за допомогою методу update(). Метод update() може приймати в якості аргументів **кортежі, списки, рядки чи інші набори**. У всіх випадках дублікати буде видалено.

```
# створення множини
my_set = {1, 3}
print(my_set)

# Такий запис не спрацює
# my_set[0]

# Додавання елементу
my_set.add(2)
print(my_set)
# Виведення: {1, 2, 3}

# додавання кількох елементів
my_set.update([2, 3, 4])
print(my_set)
# Виведення: {1, 2, 3, 4}

# додавання списку та множини одночасно
my_set.update([4, 5], {1, 6, 8})
print(my_set)
# Виведення: {1, 2, 3, 4, 5, 6, 8}
```

Видалення елементів з множини

Окремий елемент можна видалити із набору за допомогою методів discard() та remove().

Єдина різниця між ними полягає в тому, що функція discard() залишає множину без змін, якщо елемент відсутній у наборі. А функція remove() викликає помилку, якщо елемент відсутній у множині.

Наступний приклад ілюструє це:

```
# Різниця між discard() та remove()
```


CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
# Створення множини
my_set = {1, 3, 4, 5, 6}
print(my_set)

# Виключення елементу
# Виведення: {1, 3, 5, 6}
my_set.discard(4)
print(my_set)

# Видалення елементу
# Виведення: {1, 3, 5}
my_set.remove(6)
print(my_set)

# Виключення елементу,
# якого немає у множині
# Виведення: {1, 3, 5}
my_set.discard(2)
print(my_set)

# Видалення элементу
# якого немає у множині
# Виведення: KeyError
my_set.remove(2)
```

Так само ми можемо видалити і отримати елемент за допомогою методу pop().
Оскільки set - це **невпорядкований** тип даних, неможливо визначити, який елемент "виштовхуватиметься".
Ми також можемо видалити всі елементи з множини за допомогою методу clear().

Операції з множинами Python

Множини можуть використовуватися для виконання математичних операцій над наборами, таких як об'єднання, перетин, різниця і симетрична різниця. Ми можемо зробити це за допомогою операторів чи методів.
Розглянемо наступні дві множини для наступних операцій.

```
>>> A = {1, 2, 3, 4, 5}
>>> B = {4, 5, 6, 7, 8}
```

Об'єднання множин

Об'єднання A і B - це набір всіх елементів з обох множин.
Об'єднання здійснюється за допомогою оператора |. Те саме можна зробити за допомогою методу union().

```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# Використання оператора |
# Виведення: {1, 2, 3, 4, 5, 6, 7, 8}
print(A | B)
```

Спробуйте такі приклади в оболонці Python:

```
>>> A.union(B)
{1, 2, 3, 4, 5, 6, 7, 8}

>>> B.union(A)
{1, 2, 3, 4, 5, 6, 7, 8}
```

Перетин множин

Перетин A і B - це набір елементів, загальних в обох множинах.
Перетин здійснюється за допомогою оператора &. Те саме можна зробити за допомогою методу crossection().

```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
```


CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
# Використання оператора &
# Виведення: {4, 5}
print(A & B)
```

Спробуйте наступні приклади в оболонці Python:

```
>>> A.intersection(B)
{4, 5}

>>> B.intersection(A)
{4, 5}
```

Різниця множин

Відмінність множини B від множини A (**A - B**) - це набір елементів, які знаходяться тільки в A, але не в B.
Так само **B - A** - це набір елементів у B, але не в A.
Різниця виконується за допомогою оператора -. Те саме можна зробити за допомогою методу difference().

```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# Використання оператора - на A
# Виведення: {1, 2, 3}
print(A - B)
```

Спробуйте наступні приклади в оболонці Python:

```
>>> A.difference(B)
{1, 2, 3}

>>> B - A
{8, 6, 7}

>>> B.difference(A)
{8, 6, 7}
```

Симетрична різниця множин

Симетрична різниця A і B - це набір елементів A і B, але не в обох (за винятком перетину).
Симетрична різниця виконується за допомогою оператора ^. Те саме можна зробити за допомогою методу symmetric_difference().

```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# Використання опертора ^
# Виведення: {1, 2, 3, 6, 7, 8}
print(A ^ B)
```

Спробуйте наступні приклади в оболонці Python:

```
>>> A.symmetric_difference(B)
{1, 2, 3, 6, 7, 8}

>>> B.symmetric_difference(A)
{1, 2, 3, 6, 7, 8}
```

Інші методи множин

Метод	Опис
add()	Додає елемент до множини
clear()	Видаляє всі елементи з множини
copy()	Повертає копію множини
difference()	Повертає різницю між двома множинами
difference_update()	Приділяє всі елементи, що присутні в іншій множині

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

discard()	Видаляє елемент із множини, якщо він є членом. (Нічого не робить, якщо елемент не встановлений)
intersection()	Повертає перетин двох множин як нову множину.
intersection_update()	Оновлює множину з перетином себе та іншого
isdisjoint()	Повертає True, якщо дві множини мають нульовий перетин
issubset()	Повертає True, якщо інша множина містить цю множину
issuperset()	Повертає True, якщо ця множина містить іншу множину
pop()	Видаляє та повертає довільний елемент множини. Викликає помилку KeyError, якщо множина порожня
remove()	Видаляє елемент із множини. Якщо елемента немає, викликає помилку KeyError
symmetric_difference()	Повертає симетричну різницю двох множин як нову множину
symmetric_difference_update()	Оновлює множину симетричної різниці між собою та іншим
union()	Повертає об'єднання множин у нову множину
update()	Оновлює множину з об'єднанням себе та інших

Що таке відображення?
Відображення – це контейнер із невпорядкованою колекцією пар елементів "ключ-значення". У різних мовах синонімом відображень є терміни словник, хеш-таблиця чи асоціативний масив.
Відображення в Python представлені єдиним типом dict (**словник**), в якому як ключ може виступати будь-який об'єкт, що **хешується**, а в якості значення - довільний об'єкт.
Створити словник можна кількома способами:

```
# Усі ці приклади створюють однакові словники a = dict(one=1, two=2, three=3)
b = {'one': 1, 'two': 2, 'three': 3}
c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
d = dict([('two', 2), ('one', 1), ('three', 3)])
e = dict({'three': 3, 'one': 1, 'two': 2}) print(a == b == c == d == e)
print(a)

# Використання включень словників (аналогічно до спискових включень) print({string: string.upper() for string in ('one', 'two', 'three')})
```

Ті, хто мав досвід програмування Сі-подібними мовами, можуть подумати, що в цьому рядку помилка (бо в них операції порівняння зв'язувалися б зліва направо):

```
print(a == b == c == d == e)
```

Однак у Python таке порівняння є абсолютно коректним і дійсно перевіряє всі значення на рівність один одному. Усі послідовні операції порівняння та перевірки рівності об’єднуються за допомогою операції and.
У функції можна передавати довільну кількість позиційних аргументів, які зберігаються в кортежі. Також можна передавати довільну кількість іменованих аргументів, які зберігаються у словнику. Для цього перед іменем даного словника у списку формальних параметрів ставиться два символи **. Якщо використовуються обидва способи передачі довільної кількості аргументів, параметр у формі **kwargs у сигнатурі функції повинен йти після параметра у формі *args.

```
def function(**kwargs): print(kwargs)
```


CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
function(arg1='value1', arg2='value2')

# Аналогічно можна розпаковувати будь-які відображення
# в іменовані параметри при виклику функції.

options = {
'sep': ', ',
'end': ';\n'
}

print('value1', 'value2', **options)
Розглянемо деякі операції над словниками:
"""Огляд операцій зі словниками"""
phonebook = {
'Jack': '032-846',
'Guido': '917-333',
'Mario': '120-422',
'Mary': '890-532',
# остання кома ігнорується
}

# len(d) – кількість елементів. print(len(phonebook), 'entries found')

print()

# d[key] - Отримання значення з ключем key. Якщо такий ключ не існує і відображення реалізує спеціальний метод __missing (self, key), то він # викликається. Якщо ключ не існує і метод __missing не визначений,
# викидається виключення KeyError. try:
print('Mary:', phonebook['Mary']) print('Lumberjack:', phonebook['Lumberjack'])
except KeyError as e:
print('No entry for', *e.args) print()
# d[key] = value – змінити значення або створити нову пару ключ-значення, якщо ключ не існує.
phonebook['Lumberjack'] = '000-777'

# key in d, key not in d – перевірка наявності ключа у відображенні. for person in ('Guido', 'Mary', 'Ahmed'):
if person in phonebook:
print(person, 'is in the phonebook') else:
print('No entry found for', person)

print()

# iter(d) – те саме, що iter(d.keys()). print('People in the phonebook:')
for person in phonebook: print(person)

print()

# copy() – створити неповну копію словника. phonebook_copy = phonebook.copy() print('Phonebook:', phonebook) print('Phonebook copy:', phonebook_copy)

print()

# clear()– видалити всі елементи словника. phonebook_copy.clear()
```


CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
print('Phonebook:', phonebook) print('Phonebook copy:', phonebook_copy)

print()

# (метод класу) dict.fromkeys(sequence[, value]) – створює новий словник з
# ключами з послідовності sequence та заданим значенням (за замовчуванням – None).
numbers_dict = dict.fromkeys(range(3), 42) print(numbers_dict)

print()

# d.get(key[, default]) – безпечне отримання значення за ключем (ніколи не викидає KeyError).
Якщо ключ не знайдено, повертається значення default (за замовчуванням – None).
for key in range(5):
print('{}:'.format(key), numbers_dict.get(key, 0)) print()
# d.items() – в Python 3 повертає об'єкт представлення словника,
# відповідний парам (двохелементним кортежам) виду (ключ, значення). В Python 2 повертає
# відповідний список, а метод iteritems() повертає ітератор. Аналогічний метод у Python 2.7 – viewitems().
print('Items:', phonebook.items())

# d.keys() – в Python 3 повертає об'єкт представлення словника,
# відповідний ключам словника. У Python 2 повертає відповідний
# список, а метод iterkeys() повертає ітератор. Аналогічний метод у Python 2.7 – viewkeys().
print('Keys:', phonebook.keys())

# d.values() – в Python 3 повертає об'єкт представлення словника,
# відповідний значенням. В Python 2 повертає відповідний список, а
# метод itervalues() повертає ітератор. Аналогічний метод у Python 2.7 – viewvalues().
print('Values:', phonebook.values()) print()
# d.pop(key[, default]) – якщо ключ key існує, видаляє елемент із словника
# та повертає його значення. Якщо ключ не існує і встановлено значення
# default, повертається це значення, інакше викидається виняток
# KeyError.

number = phonebook.pop('Lumberjack') print('Deleted Lumberjack (was ' + number + ')') print(phonebook)

print()

# d.popitem() – видаляє довільну пару ключ-значення та повертає її. Якщо # словник порожній, виникає виняток KeyError. Метод корисний для алгоритмів, які обходять словник, видаляючи вже оброблені значення (наприклад, певні алгоритми, пов'язані з теорією графів).

person = phonebook.popitem()
print('Popped {} (phone: {})'.format(*person)) print()

# d.setdefault(key[, default]) – якщо ключ key існує, повертає
відповідне значення. Інакше створює елемент із ключем key та значенням # default. default за замовчуванням дорівнює None.
for person in ('Jack', 'Liz'):
phone = phonebook.setdefault(person, '000-000') print('{}: {}'.format(person, phone))

print(phonebook)
```


CyberBionic systematics

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
print()

# d.update(mapping) – приймає або інший словник чи відображення, або
# ітерабельний об'єкт, що складається з ітерабельних об'єктів - пар ключ-значення,
# або іменовані аргументи. Додає відповідні елементи до словника,
# перезаписуючи елементи з існуючими ключами.
phonebook.update({'Alex': '832-438', 'Alice': '231-987'})phonebook.update([('Joe', '217-531'), ('James', '783-428')]) phonebook.update(Carl='783-923', Victoria='386-486') print(phonebook)
```

Сортування словника за допомогою циклу for

Ми можемо відсортувати словник за допомогою циклу for. Спочатку ми використовуємо функцію sorted() для впорядкування значень словника. Потім ми перебираємо відсортовані значення, знаходячи ключі для кожного значення. Ми додаємо ці пари ключ-значення у відсортованому порядку до нового словника.

Примітка. Сортування не дозволяє змінювати порядок словника на місці. Записуємо впорядковані пари в абсолютно новий порожній словник

```
dict1 = {1: 1, 5: 9, 3: 4}
# Sort the keys
sorted_keys = sorted(dict1.keys())
print(sorted_keys)
# Sort the values
sorted_values = sorted(dict1.values())
print(sorted_values)
sorted_dict_k, sorted_dict_val = {}, {}

# Sort the keys
for i in sorted_values:
    for k in dict1.keys():
        if dict1[k] == i:
            sorted_dict_k[k] = dict1[k]
            break

print(sorted_dict_k)
```

Додаткові завдання

Завдання 1

Є рядок, в якому зберігаються 1000 слів. Створіть словник із ключами - унікальними словами та значеннями - кількістю повторів кожного слова у послідовності.

Завдання 2

Створіть прототип програми «Бібліотека», де є можливість перегляду та внесення змін за структурою: автор: твір. Передбачте можливість виведення на екран сортування за автором та твором.

Завдання 3

Створіть прототип програми «Облік кадрів», в якій є можливість перегляду та внесення змін до структури(реалізуйте інтерфейс(меню), за допомогою якого можна робити маніпуляції з даними):

прізвище:
- посада: ...
- досвід роботи: …
- портфоліо: …
- коефіцієнт ефективності: …
- стек технологій: …
- зарплата: …

Передбачте можливість виведення на екран сортування за прізвищем та найефективнішим співробітником.

CyberBionic systematics

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Самостійна діяльність учня

Завдання 1
Дано два рядки. Виведіть на екран символи, які є в обох рядках.

Завдання 2
Створіть програму, яка емулює роботу сервісу зі скорочення посилань. Повинна бути реалізована можливість введення початкового посилання та короткої назви і отримання початкового посилання за її назвою.

Завдання 3
Створіть програму, яка має 2 списки цілочисельних значень та друкує список унікальних значень без повтору, які є в 1 списку (немає в другому) і навпаки.

Завдання 4
Ознайомтеся за допомогою документації з класами OrderedDict, defaultdict та ChainMap модуля collections.

Рекомендовані ресурси

Документація Python
https://docs.python.org/3/tutorial/datastructures.html#sets
https://docs.python.org/3/tutorial/datastructures.html#dictionaries
https://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset
https://docs.python.org/3/library/stdtypes.html#mapping-types-dict
https://docs.python.org/3/library/stdtypes.html#dictionary-view-objects
https://docs.python.org/3/library/collections.html

Статті у Вікіпедії про ключові поняття, розглянуті на цьому уроці
https://uk.wikipedia.org/wiki/Множина
https://uk.wikipedia.org/wiki/Множина_(тип_даних)
https://en.wikipedia.org/wiki/Map_(mathematics)
https://uk.wikipedia.org/wiki/Асоціативний_масив
https://uk.wikipedia.org/wiki/Мультимножина

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com