Олексій Васильєв

Мова програмування Python

Київ 2020

Лекція 4. Множини і словники

- Знайомство з множинами
- Операції з множинами
- Знайомство зі словниками
- Операції зі словниками
- Приклади використання множин і словників

Створення множини

Множина (тип set) — це невпорядкований набір унікальних елементів

Створення множин:

- **Елементи перераховуються (через кому) в фігурних дужках: A={1,2,3} B={1,3,7,9,3}**
- **За допомогою функції set():**
  A=set([1,3,5]) B=set((1,2,3)) C=set("Hello")
- **Розмір множини - за допомогою функції len()**

Якщо викликати функцію set() без аргументів, буде створена пуста множина. А якщо використати пусті фігурні дужки, то буде створено пустий словник.

- **Перевірка на приналежність множині:**
  елемент in множина
- **Перевірка на неприналежність множині:**
  елемент not in множина

Створення множини

Програма (CreatingSet.py)

```
# Створення множини:
A={10,50,20,10,50}
B=set([6,2,6,0,4,0])
C=set("Hello World")
# Перевірка вмісту множин:
print("A:",A)
print("Елементів:",len(A))
print("B:",B)
print("Елементів:",len(B))
print("C:",C)
print("Елементів:",len(C))
```

```
A: {10, 50, 20}
Елементів: 3
B: {0, 2, 4, 6}
Елементів: 4
C: {'W', ' ', 'e', 'd', 'o', 'H', 'l', 'r'}
Елементів: 8
```

Випадкові числа

Програма (RandNums.py)

```
# Підключення функцій для генерування випадкових чисел:
from random import *
# Ініціалізація генератора випадкових чисел:
seed(2019)
# Кількість різних випадкових чисел:
count=10
# Створення пустої множини:
nums=set()
# Генерування випадкових чисел:
while len(nums)<count:
    nums.add(randint(1,count+5))
# Відображення результату:
print(nums)
```

```
{3, 4, 5, 6, 8, 10, 11, 13, 14, 15}
```

Операції з множинами

Основні операції

- Додавання елементу: метод add()
- Видалення елементу: remove() (генерує помилку якщо елементу що видаляється немає) і discard()
- Очистка множини: метод clean()
- Об'єднання множин: A|B і A.union(B)
- Перетин множин: A&B і A.intersection(B)
- Різниця множин: A-B і A.difference(B)
- Симетрична різниця: A^B і A.symmetric_difference(B)

Вихідні множини (A і B) при цьому не змінюються

Операції з множинами

Операції зі зміною множини A:

- Об'єднання множин: A.update(B)
- Перетин множин: A.intersection_update(B)
- Різниця множин: A.difference_update(B)
- Симетрична різниця: A.symmetric_difference_update(B)

Основні операції

Програма (UsingSets.py)

```
Створені множини:
A = {1, 3, 5, 7, 9}
B = {0, 2, 4, 6, 8}
C = {7, 9, 11, 13, 15}
Об'єднання множин:
A | B = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
B | A = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
A | C = {1, 3, 5, 7, 9, 11, 13, 15}
Перетин множин:
A & B = set()
A & C = {9, 7}
Різниця множин:
A - C = {1, 3, 5}
C - A = {11, 13, 15}
Симетрична різниця множин:
A ^ C = {1, 3, 5, 11, 13, 15}
C ^ A = {1, 3, 5, 11, 13, 15}
Вихідні множини:
A = {1, 3, 5, 7, 9}
B = {0, 2, 4, 6, 8}
C = {7, 9, 11, 13, 15}
```

```
# Створення множин:
A={2*k+1 for k in range(5)}
B={2*k for k in range(5)}
C={2*k+1 for k in range(3,8)}
# Вміст множин:
print("Створені множини:")
print("A =",A)
print("B =",B)
print("C =",C)
# Об'єднання множин:
print("Об'єднання множин:")
print("A | B =",A.union(B))
print("B | A =",B.union(A))
print("A | C =",A|C)
# Перетин множин:
print("Перетин множин:")
print("A & B =",A.intersection(B))
print("A & C =",A&C)
# Різниця множин:
print("Різниця множин:")
print("A - C =",A-C)
print("C - A =",C.difference(A))
# Симетрична різниця множин:
print("Симетрична різниця множин:")
print("A ^ C =",A^C)
print("C ^ A =",C.symmetric_difference(A))
# Вихідні множини:
print("Вихідні множини:")
print("A =",A)
print("B =",B)
print("C =",C)
```

Основні операції

Програма (UsingSets_2.py)

```
# Створення множин:
A={0,5,10,15,20}
B={10,15,20,25,30}
# Вміст множин:
print("Створені множини:")
print("A =",A)
print("B =",B)
# Перетин множин:
print("Перетин множин A і B:")
A.intersection_update(B)
print("A =",A)
# Об'єднання множин:
print("Об'єднання з множиною {1,20,100}:")
A.update({1,20,100})
print("A =",A)
# Різниця множин:
print("Різниця множин B і A:")
B.difference_update(A)
print("B =",B)
# Симетрична різниця множин:
print("Симетрична різниця множин B і {30,35}:")
B.symmetric_difference_update({30,35})
print("B =",B)
```

```
Створені множини:
A = {0, 5, 10, 15, 20}
B = {10, 15, 20, 25, 30}
Перетин множин A і B:
A = {10, 20, 15}
Об'єднання з множиною {1,20,100}:
A = {1, 20, 100, 10, 15}
Різниця множин B і A:
B = {25, 30}
Симетрична різниця множин B і {30,35}:
B = {35, 25}
```

Основні операції

Програма (UsingSets_3.py)

```
A={0,5,10,15,20}
B={10,15,20,25,30}
print("Створені множини:")
print("A =",A)
print("B =",B)
print("Перетин множин A і B:")
A&=B
print("A =",A)
print("Об'єднання з множиною {1,20,100}:")
A|={1,20,100}
print("A =",A)
print("Різниця множин B і A:")
B-=A
print("B =",B)
print("Симетрична різниця множин B і {30,35}:")
B^={30,35}
print("B =",B)
```

```
Створені множини:
A = {0, 5, 10, 15, 20}
B = {10, 15, 20, 25, 30}
Перетин множин A і B:
A = {10, 20, 15}
Об'єднання з множиною {1,20,100}:
A = {1, 20, 100, 10, 15}
Різниця множин B і A:
B = {25, 30}
Симетрична різниця множин B і {30,35}:
B = {35, 25}
```

Порівняння множин

Програма (UsingSets_4.py)

```
# Вихідні множини:
A={1,2}
B={1,2,3}
C={3,2,1}
# Вміст множин:
print("A =",A)
print("B =",B)
print("C =",C)
# Порівняння множин:
print("A==B:",A==B)
print("A!=B:",A!=B)
print("B==C:",B==C)
print("B!=C:",B!=C)
print("A<B:",A<B)
print("A>B:",A>B)
print("B<C:",B<C)
print("B<=C:",B<=C)
print("B>=C:",B>=C)
print("B>=A:",B>=A)
```

```
A = {1, 2}
B = {1, 2, 3}
C = {1, 2, 3}
A==B: False
A!=B: True
B==C: True
B!=C: False
A<B: True
A>B: False
B<C: False
B<=C: True
B>=C: True
B>=A: True
```

Приклад: склад числа

Програма (Didgits.py)

```
# Зчитування числа:
number=int(input("Уведіть число: "))
# Якщо число від'ємне:
if number<0:
    number*=-1
# Створюється пуста множина:
digits=set()
# Якщо був уведений нуль:
if number==0:
    digits.add(0)
# Якщо число не дорівнює нулю:
else:
    # Перебір цифр в представленні числа:
    while number!=0:
        # Остання цифра в представленні числа:
        digits.add(number%10)
        # В представленні числа відкидається
        # остання цифра:
        number//=10
print("Число містить такі цифри:")
# Відображення результату:
for n in digits:
    print(n,end=" ")
# Перехід до нового рядка:
print()
```

```
Уведіть число: 2501175
Число містить такі цифри:
0 1 2 5 7
```

Приклад: спільні букви

Програма (Symbols.py)

```
# Зчитування текстових значень:
text=input("Перший текст: ")
# Перша множина:
A=set(text)
text=input("Другий текст: ")
# Друга множина:
B=set(text)
# Спільні букви:
C=A&B
# Результат:
print("Букви з першого тексту: ",A)
print("Букви з другого тексту: ",B)
print("Спільні букви: ",C)
```

```
Перший текст: програма
Другий текст: абракадабра
Букви з першого тексту:  {'м', 'а', 'о', 'г', 'п', 'р'}
Букви з другого тексту:  {'а', 'б', 'д', 'р', 'к'}
Спільні букви:  {'а', 'р'}
```

Завдання - 1

Напишіть програму, в якій генерується 15 випадкових цілих чисел: 5 чисел належать до діапазону значень від 1 до 10, і 10 чисел належать до діапазону значеньвід 10 до 30

Завдання - 2

Напишіть програму, в якій користувач вводить два цілих числа, а програмою визначаються цифри, котрі є в представленні обох чисел

Завдання - 3

Напишіть програму для розрахунку множини чисел (в межах першої півсотні) таких, що вони діляться або на 3, або на 4, але при цьому не діляться одночасно на 3 і 4

Завдання - 4

Напишіть програму для створення множини, елементами якої є кортежі (по два елементи в кожному) з непарними числами: (1,3), (3,5), (5,7) і так далі

Створення словника

Словник (тип **dict**) — це набір елементів, доступ до яких виконується по ключу

Створення словника:

- В блоці з фігурних дужок вказуються (через кому) вирази виду *ключ:значення*
- За допомогою функції **dict()**
- Розмір словника - за допомогою функції **len()**

Якщо ключами словника є текстові значення, то аргументами функції **dict()** можуть передаватися конструкції виду **ключ=значення**

Виклик функції **dict()** без аргументів приводить до створення пустого словника. Пустий блок з фігурних дужок **{}** також відповідає пустому словнику (не множині)

Доступ до елементів словника: **словник[ключ]** (можлива помилка класу **KeyError** якщо елементу немає) або метод **get()** (значення **None** якщо елементу немає)

Метод **keys()** повертає ітерований об'єкт класу **dict_keys** зі значеннями ключів. Метод **values()** повертає ітерований об'єкт класу **dict_values** зі значеннями елементів словника

Створення словника

Програма (Dictionary.py)

```
# Перший словник:
nums={100:"сотня",1:"одиниця",10:"десятка"}
# Вміст словника:
print(nums)
print("1:   ",nums[1])
print("10:  ",nums[10])
print("100:",nums[100])
# Операції зі словниками:
nums[3]="трійка"
nums[10]="десятка"
nums.pop(100)
print(nums)
# Другий словник:
order=dict(Перший=1,Третій=3,Останній=10)
# Вміст словника:
print(order)
print("Перший:    ",order["Перший"])
print("Третій:    ",order["Третій"])
print("Останній:",order["Останній"])
# Операції зі словниками:
order["Останній"]=12
del order["Третій"]
order["П'ятий"]=5
print(order)
```

```
{100: 'сотня', 1: 'одиниця', 10: 'десятка'}
1:    одиниця
10:   десятка
100:  сотня
{1: 'одиниця', 10: 'десятка', 3: 'трійка'}
{'Перший': 1, 'Третій': 3, 'Останній': 10}
Перший:     1
Третій:     3
Останній: 10
{'Перший': 1, 'Останній': 12, "П'ятий": 5}
```

Створення словника - 2

Програма (Dictionary_2.py)

```
# Створення словника:
age=dict([["Кіт  Мурчик",5],["Пес  Шарік",7],["Хом'як Топтунчик",12]])
# Перебір ключів:
for s in age.keys():
    print(s+":",age[s])
# Перебір значень:
for v in age.values():
    print(v,end=" ")
print()
# Створення словника:
color=dict([[(255,0,0),"Червоний"],[(0,255,0),"Зелений"],[(0,0,255),"Синій"]])
# Звернення до елементів словника:
color[(255,255,0)]="Жовтий"
print("(255,0,0):",color[(255,0,0)])
print("(255,255,0):",color[(255,255,0)])
print("(0,255,0):",color.get((0,255,0)))
print("(0,0,255):",color.get((0,0,255),"Білий"))
print("(255,255,255):",color.get((255,255,255),"Білий"))
```

```
Кіт  Мурчик: 5
Пес  Шарік: 7
Хом'як Топтунчик: 12
5 7 12
(255,0,0): Червоний
(255,255,0): Жовтий
(0,255,0): Зелений
(0,0,255): Синій
(255,255,255): Білий
```

Створення словника - 3

Програма (Dictionary_3.py)

```
# Список зі значеннями ключів:
days=["Пн","Вт","Ср","Чт","Пт","Сб","Нд"]
# Створення словників:
week={days[s]:s for s in range(len(days))}
myweek={d:days.index(d) for d in days}
# Перевірка результату:
print(week)
print(myweek)
# Створення ще одного словника:
sqrs={k:k**2 for k in range(1,11) if k%2!=0}
# Перевірка результату:
print(sqrs)
```

```
{'Пн': 0, 'Вт': 1, 'Ср': 2, 'Чт': 3, 'Пт': 4, 'Сб': 5, 'Нд': 6}
{'Пн': 0, 'Вт': 1, 'Ср': 2, 'Чт': 3, 'Пт': 4, 'Сб': 5, 'Нд': 6}
{1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
```

Копія словника

```
# Вихідний словник:
A={"Початковий":1,"Середній":2,"Останній":3}
# Створення копії словника:
B=dict(A)
C=A.copy()
# Створення словника на основі словника:
D={k:v*10 for k,v in A.items()}
# Відображення результату:
print("Створені словники:")
print("A =",A)
print("B =",B)
print("C =",C)
print("D =",D)
# Зміна вихідного словника:
for k in A:
    A[k]*=100
# Відображення результату:
print("Після зміни оригіналу:")
print("A =",A)
print("B =",B)
print("C =",C)
print("D =",D)
```

Програма (DictCopy.py)

```
Створені словники:
A = {'Початковий': 1, 'Середній': 2, 'Останній': 3}
B = {'Початковий': 1, 'Середній': 2, 'Останній': 3}
C = {'Початковий': 1, 'Середній': 2, 'Останній': 3}
D = {'Початковий': 10, 'Середній': 20, 'Останній': 30}
Після зміни оригіналу:
A = {'Початковий': 100, 'Середній': 200, 'Останній': 300}
B = {'Початковий': 1, 'Середній': 2, 'Останній': 3}
C = {'Початковий': 1, 'Середній': 2, 'Останній': 3}
D = {'Початковий': 10, 'Середній': 20, 'Останній': 30}
```

Копія словника - 2

Програма (DictCopy_2.py)

```
# Імпорт функції для створення повної копії:
from copy import deepcopy
# Вихідний словник:
A={"один":1,"два":"двійка","три":[3,4]}
# Поверхнева копія словника:
B=dict(A)
C=A.copy()
# Повна копія словника:
D=deepcopy(A)
# Відображення результату:
print("A =",A)
print("B =",B)
print("C =",C)
print("D =",D)
# Зміна значень елементів вихідного словника:
A["два"]=2
A["три"][1]=5
# Перевірка результату:
print("A =",A)
print("B =",B)
print("C =",C)
print("D =",D)
```

```
A = {'один': 1, 'два': 'двійка', 'три': [3, 4]}
B = {'один': 1, 'два': 'двійка', 'три': [3, 4]}
C = {'один': 1, 'два': 'двійка', 'три': [3, 4]}
D = {'один': 1, 'два': 'двійка', 'три': [3, 4]}
A = {'один': 1, 'два': 2, 'три': [3, 5]}
B = {'один': 1, 'два': 'двійка', 'три': [3, 5]}
C = {'один': 1, 'два': 'двійка', 'три': [3, 5]}
D = {'один': 1, 'два': 'двійка', 'три': [3, 4]}
```

Робота зі словниками

```
A=dict(zip([1,2,3],['K','L','M']))
B=dict.fromkeys([10,20,30],'Z')
print("A =",A)
print("B =",B)
# Порівняння словників:
print("Порівняння словників:")
print("A==B:",A==B)
print("A!=B:",A!=B)
# Додавання одного словника до другого:
A.update(B)
# Перевірка результату:
print("Об'єднання словників:")
print("A =",A)
# Перевірка вмісту словника:
print("Перевірка вмісту словника:")
print((20,'Z') in A.items())
print(20 in A)
print('Z' in A)
print(5 not in A)
# Очистка словника:
A.clear()
# Перевірка результату:
print("Словник після очистки:")
print("A =",A)
```

Програма (UsingDict.py)

Метод **items()** повертає об'єкт класу **dict_items** з елементами словника (список з кортежами по два елемента виду **(ключ,словник)**)

```
A = {1: 'K', 2: 'L', 3: 'M'}
B = {10: 'Z', 20: 'Z', 30: 'Z'}
Порівняння словників:
A==B: False
A!=B: True
Об'єднання словників:
A = {1: 'K', 2: 'L', 3: 'M', 10: 'Z', 20: 'Z', 30: 'Z'}
Перевірка вмісту словника:
True
True
False
True
Словник після очистки:
A = {}
```

Завдання - 5

Напишіть програму, яка виконується наступним чином. Користувач вводить текст. На основі цього тексту створюється словник. Ключами словника слугують символи з тексту, а значеннями елементів словника є кількість входжень відповідних символів у текст. Наприклад, якщо користувач вводить текст "ABBCAB", то словник буде складатися з трьох елементів з ключами "A", "B" і "C", а значення елементів відповідно дорівнюють 2 (в тексті 2 букви "A"), 3 (в тексті 3 букви "B") і 1 (в тексті 1 буква "C").

Домашнє завдання

[1] Напишіть програму, в якій користувач вводить текстове значення, і для цього текстового значення визначаються голосні букви, котрі представлені у введеному тексті

[2] Напишіть програму, в якій на основі двох словників створюється новій словник. В цей новий словник включаються ті елементи, котрі представлені в кожному з вихідних словників (маються на увазі ключі елементів). Значеннями елементів у створеному словнику є множини зі значеннями відповідних елементів вихідних словників