Алексей Васильев

Язык программирования Python

Киев 2019

Лекция 4. Множества и словари

- Знакомство с множествами
- Операции с множествами
- Знакомство со словарями
- Операции со словарями
- Примеры использования множеств и словарей

Создание множества

Множество (тип set) — это неупорядоченный набор уникальных элементов

Создание множества:

- **Элементы перечисляются (через запятую) в фигурных скобках: A={1,2,3} B={1,3,7,9,3}**
- **С помощью функции set():**
  A=set([1,3,5]) B=set((1,2,3)) C=set("Hello")
- **Размер множества - с помощью функции len()**

Если вызвать функцию set() без аргументов, будет создано пустое множество. А если использовать пустые фигурные скобки, то будет создан пустой словарь.

- **Проверка на принадлежность множеству:**
  элемент in множество
- **Проверка на непринадлежность множеству:**
  элемент not in множество

Создание множества

Программа (CreatingSet.py)

```
# Создание множеств:
A={10,50,20,10,50}
B=set([6,2,6,0,4,0])
C=set("Hello World")
# Проверка содержимого множеств:
print("A:",A)
print("Элементов:",len(A))
print("B:",B)
print("Элементов:",len(B))
print("C:",C)
print("Элементов:",len(C))
```

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

Случайные числа

Программа (RandNums.py)

```
# Подключение функций для генерирования случайных чисел:
from random import *
# Инициализация генератора случайных чисел:
seed(2019)
# Количество разных случайных чисел:
count=10
# Создание пустого множества:
nums=set()
# Генерирование случайных чисел:
while len(nums)<count:
    nums.add(randint(1,count+5))
# Отображение результата:
print(nums)
```

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

Операции с множествами

Основные операции

- Добавление элемента: метод add()
- Удаление элемента: remove() (генерирует ошибку если удаляемого элемента нет) и discard()
- Очистка множества: метод clean()
- Объединение множеств: A|B и A.union(B)
- Пересечение множеств: A&B и A.intersection(B)
- Разность множеств: A-B и A.difference(B)
- Симметрическая разность:
  A^B и A.symmetric_difference(B)

Исходные множества (A и B) при этом не меняются

Операции с множествами

Операции с изменением множества A:

- Объединение множеств: A.update(B)
- Пересечение множеств: A.intersection_update(B)
- Разность множеств: A.difference_update(B)
- Симметрическая разность:
  A.symmetric_difference_update(B)

Основные операции

Программа (UsingSets.py)

```
Созданы множества:
A = {1, 3, 5, 7, 9}
B = {0, 2, 4, 6, 8}
C = {7, 9, 11, 13, 15}
Объединение множеств:
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}
Пересечение множеств:
A & B = set()
A & C = {9, 7}
Разность множеств:
A - C = {1, 3, 5}
C - A = {11, 13, 15}
Симметрическая разность множеств:
A ^ C = {1, 3, 5, 11, 13, 15}
C ^ A = {1, 3, 5, 11, 13, 15}
Исходные множества:
A = {1, 3, 5, 7, 9}
B = {0, 2, 4, 6, 8}
C = {7, 9, 11, 13, 15}
```

```
# Создание множеств:
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)}
# Содержимое множеств:
print("Созданы множества:")
print("A =",A)
print("B =",B)
print("C =",C)
# Объединение множеств:
print("Объединение множеств:")
print("A | B =",A.union(B))
print("B | A =",B.union(A))
print("A | C =",A|C)
# Пересечение множеств:
print("Пересечение множеств:")
print("A & B =",A.intersection(B))
print("A & C =",A&C)
# Разность множеств:
print("Разность множеств:")
print("A - C =",A-C)
print("C - A =",C.difference(A))
# Симметрическая разность множеств:
print("Симметрическая разность множеств:")
print("A ^ C =",A^C)
print("C ^ A =",C.symmetric_difference(A))
# Исходные множества:
print("Исходные множества:")
print("A =",A)
print("B =",B)
print("C =",C)
```

Основные операции

```
# Создание множеств:
A={0,5,10,15,20}
B={10,15,20,25,30}
# Содержимое множеств:
print("Созданы множества:")
print("A =",A)
print("B =",B)
# Пересечение множеств:
print("Пересечение множеств A и B:")
A.intersection_update(B)
print("A =",A)
# Объединение множеств:
print("Объединение с множеством {1,20,100}:")
A.update({1,20,100})
print("A =",A)
# Разность множеств:
print("Разность множеств B и A:")
B.difference_update(A)
print("B =",B)
# Симметрическая разность множеств:
print("Симметрическая разность множеств B и {30,35}:")
B.symmetric_difference_update({30,35})
print("B =",B)
```

Программа (UsingSets_2.py)

```
Созданы множества:
A = {0, 5, 10, 15, 20}
B = {10, 15, 20, 25, 30}
Пересечение множеств A и B:
A = {10, 20, 15}
Объединение с множеством {1,20,100}:
A = {1, 20, 100, 10, 15}
Разность множеств B и A:
B = {25, 30}
Симметрическая разность множеств B и {30,35}:
B = {35, 25}
```

Основные операции

Программа (UsingSets_3.py)

```
A={0,5,10,15,20}
B={10,15,20,25,30}
print("Созданы множества:")
print("A =",A)
print("B =",B)
print("Пересечение множеств A и B:")
A&=B
print("A =",A)
print("Объединение с множеством {1,20,100}:")
A|={1,20,100}
print("A =",A)
print("Разность множеств B и A:")
B-=A
print("B =",B)
print("Симметрическая разность множеств B и {30,35}:")
B^={30,35}
print("B =",B)
```

```
Созданы множества:
A = {0, 5, 10, 15, 20}
B = {10, 15, 20, 25, 30}
Пересечение множеств A и B:
A = {10, 20, 15}
Объединение с множеством {1,20,100}:
A = {1, 20, 100, 10, 15}
Разность множеств B и A:
B = {25, 30}
Симметрическая разность множеств B и {30,35}:
B = {35, 25}
```

Сравнение множеств

Программа (UsingSets_4.py)

```
# Исходные множества:
A={1,2}
B={1,2,3}
C={3,2,1}
# Содержимое множеств:
print("A =",A)
print("B =",B)
print("C =",C)
# Сравнение множеств:
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
```

Пример: состав числа

Программа (Didgits.py)

```
# Считывание числа:
number=int(input("Введите число: "))
# Если число отрицательное:
if number<0:
    number*=-1
# Создается пустое множество:
digits=set()
# Если был введен ноль:
if number==0:
    digits.add(0)
# Если число не равно нулю:
else:
    # Перебор цифр в представлении числа:
    while number!=0:
        # Последняя цифра в представлении числа:
        digits.add(number%10)
        # В представлении числа отбрасывается
        # последняя цифра:
        number//=10
print("Число состоит из таких цифр:")
# Отображение результата:
for n in digits:
    print(n,end=" ")
# Переход к новой строке:
print()
```

```
Введите число: 2501175
Число состоит из таких цифр:
0 1 2 5 7
```

Пример: общие буквы

Программа (Symbols.py)

```
# Считывание текстовых значений:
text=input("Первый текст: ")
# Первое множество:
A=set(text)
text=input("Второй текст: ")
# Второе множество:
B=set(text)
# Общие буквы:
C=A&B
# Результат:
print("Буквы из первого текста: ",A)
print("Буквы из второго текста: ",B)
print("Общие буквы: ",C)
```

```
Первый текст: программа
Второй текст: абракадабра
Буквы из первого текста:  {'г', 'о', 'р', 'п', 'а', 'м'}
Буквы из второго текста:  {'а', 'р', 'к', 'б', 'д'}
Общие буквы:  {'а', 'р'}
```

Задание - 1

Напишите программу, в которой генерируется 15 случайных целых чисел: 5 чисел попадают в диапазон значений от 1 до 10, и 10 чисел попадают в диапазон от 10 до 30

Задание - 2

Напишите программу, в которой пользователь вводит два целых числа, а программой определяются цифры, которые есть в представлении обоих чисел

Задание - 3

Напишите программу для вычисления множества чисел (в пределах первой полусотни) таких, что они делятся или на 3, или на 4, но при этом не длятся одновременно на 3 и 4

Задание - 4

Напишите программу для создания множества, элементами которого являются кортежи (по два элемента в каждом) с нечетными числами: (1,3), (3,5), (5,7) и так далее

Создание словаря

Словарь (тип **dict**) — это набор элементов, доступ к которым выполняется по ключу

Создание словаря:

- В блоке из фигурных скобок указываются (через запятую) выражения вида ***ключ:значение***
- С помощью функции **dict()**
- Размер словаря - с помощью функции **len()**

Если ключами словаря являются текстовые значения, то аргументами функции **dict()** могут передаваться конструкции вида **ключ=значение**

Вызов функции **dict()** без аргументов приводит к созданию пустого словаря. Пустой блок из фигурных скобок **{}** также соответствует пустому словарю (не множеству)

Доступ к элементам словаря: **словарь[ключ]** (возможна ошибка класса **KeyError** если элемента нет) или метод **get()** (значение **None** если элемента нет)

Метод **keys()** возвращает итерируемый объект класса **dict_keys** со значениями ключей. Метод **values()** возвращает итерируемый объект класса **dict_values** со значениями элементов словаря

Создание словаря

Программа (Dictionary.py)

```
# Первый словарь:
nums={100:"сотня",1:"единица",10:"десятка"}
# Содержимое словаря:
print(nums)
print("1:   ",nums[1])
print("10:  ",nums[10])
print("100:",nums[100])
# Операции со словарем:
nums[3]="тройка"
nums[10]="десять"
nums.pop(100)
print(nums)
# Второй словарь:
order=dict(Первый=1,Третий=3,Последний=10)
# Содержимое словаря:
print(order)
print("Первый:    ",order["Первый"])
print("Третий:    ",order["Третий"])
print("Последний:",order["Последний"])
# Операции со словарем:
order["Последний"]=12
del order["Третий"]
order["Пятый"]=5
print(order)
```

```
{100: 'сотня', 1: 'единица', 10: 'десятка'}
1:    единица
10:   десятка
100:  сотня
{1: 'единица', 10: 'десять', 3: 'тройка'}
{'Первый': 1, 'Третий': 3, 'Последний': 10}
Первый:     1
Третий:     3
Последний: 10
{'Первый': 1, 'Последний': 12, 'Пятый': 5}
```

Создание словаря - 2

Программа (Dictionary_2.py)

```
# Создание словаря:
age=dict([["Кот  Матроскин",5],["Пес  Шарик",7],["Дядя Федор",12]])
# Перебор ключей:
for s in age.keys():
    print(s+":",age[s])
# Перебор значений:
for v in age.values():
    print(v,end=" ")
print()
# Создание словаря:
color=dict([[(255,0,0),"Красный"],[(0,255,0),"Зеленый"],[(0,0,255),"Синий"]])
# Обращение к элементам словаря:
color[(255,255,0)]="Желтый"
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),"Белый"))
print("(255,255,255):",color.get((255,255,255),"Белый"))
```

```
Кот Матроскин: 5
Пес Шарик: 7
Дядя Федор: 12
5 7 12
(255,0,0): Красный
(255,255,0): Желтый
(0,255,0): Зеленый
(0,0,255): Синий
(255,255,255): Белый
```

Создание словаря - 3

Программа (Dictionary_3.py)

```
# Список со значениями ключей:
days=["Пн","Вт","Ср","Чт","Пт","Сб","Вс"]
# Создание словарей:
week={days[s]:s for s in range(len(days))}
myweek={d:days.index(d) for d in days}
# Проверка результата:
print(week)
print(myweek)
# Создание еще одного словаря:
sqrs={k:k**2 for k in range(1,11) if k%2!=0}
# Проверка результата:
print(sqrs)
```

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

Копия словаря

```
# Исходный словарь:
A={"Начальный":1,"Средний":2,"Последний":3}
# Создание копии словаря:
B=dict(A)
C=A.copy()
# Создание словаря на основе словаря:
D={k:v*10 for k,v in A.items()}
# Отображение результата:
print("Созданы словари:")
print("A =",A)
print("B =",B)
print("C =",C)
print("D =",D)
# Изменение исходного словаря:
for k in A:
    A[k]*=100
# Отображение результата:
print("После изменения оригинала:")
print("A =",A)
print("B =",B)
print("C =",C)
print("D =",D)
```

Программа (DictCopy.py)

```
Созданы словари:
A = {'Начальный': 1,'Средний': 2,'Последний': 3}
B = {'Начальный': 1,'Средний': 2,'Последний': 3}
C = {'Начальный': 1,'Средний': 2,'Последний': 3}
D = {'Начальный': 10,'Средний': 20,'Последний': 30}
После изменения оригинала:
A = {'Начальный': 100,'Средний': 200,'Последний': 300}
B = {'Начальный': 1,'Средний': 2,'Последний': 3}
C = {'Начальный': 1,'Средний': 2,'Последний': 3}
D = {'Начальный': 10,'Средний': 20,'Последний': 30}
```

Копия словаря - 2

Программа (DictCopy_2.py)

```
# Импорт функции для создания полной копии:
from copy import deepcopy
# Исходный словарь:
A={"один":1,"два":"двойка","три":[3,4]}
# Поверхностная копия словаря:
B=dict(A)
C=A.copy()
# Полная копия словаря:
D=deepcopy(A)
# Отображение результата:
print("A =",A)
print("B =",B)
print("C =",C)
print("D =",D)
# Изменение значений элементов исходного словаря:
A["два"]=2
A["три"][1]=5
# Проверка результата:
print("A =",A)
print("B =",B)
print("C =",C)
print("D =",D)
```

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

Работа со словарями

```
A=dict(zip([1,2,3],['K','L','M']))
B=dict.fromkeys([10,20,30],'Z')
print("A =",A)
print("B =",B)
# Сравнение словарей:
print("Сравнение словарей:")
print("A==B:",A==B)
print("A!=B:",A!=B)
# Добавление одного словаря к другому:
A.update(B)
# Проверка результата:
print("Объединение словарей:")
print("A =",A)
# Проверка содержимого словаря:
print("Проверка содержимого словаря:")
print((20,'Z') in A.items())
print(20 in A)
print('Z' in A)
print(5 not in A)
# Очистка словаря:
A.clear()
# Проверка результата:
print("Словарь после очистки:")
print("A =",A)
```

Программа (UsingDict.py)

Метод **items()** возвращает объект класса **dict_items** с элементами словаря (список с кортежами по два элемента вида **(ключ,словарь)**)

```
A = {1: 'K', 2: 'L', 3: 'M'}
B = {10: 'Z', 20: 'Z', 30: 'Z'}
Сравнение словарей:
A==B: False
A!=B: True
Объединение словарей:
A = {1: 'K', 2: 'L', 3: 'M', 10: 'Z', 20: 'Z', 30: 'Z'}
Проверка содержимого словаря:
True
True
False
True
Словарь после очистки:
A = {}
```

Задание - 5

Напишите программу, которая выполняется следующим образом. Пользователь вводит текст. На основе этого текста создается словарь. Ключами словаря служат символы из текста, а значениями элементов словаря являются количества вхождений соответствующих символов в текст. Например, если пользователь вводит текст "ABBCAB", то словарь будет состоять из трех элементов с ключами "A", "B" и "C", а значения элементов соответственно равны 2 (в тексте 2 буквы "A"), 3 (в тексте 3 буквы "B") и 1 (в тексте 1 буква "C").

Домашнее задание

[1] Напишите программу, в которой пользователь вводит текстовое значение, и для этого текстового значения определяются гласные буквы, которые представлены во введенном тексте

[2] Напишите программу, в которой на основе двух словарей создается новый словарь. В этот новый словарь включатся те элементы, которые представлены в каждом из исходных словарей (имеются в виду ключи элементов). Значениями элементов в создаваемом словаре являются множества из значений соответствующих элементов в исходных словарях