



основы проектной деятельности

Занятие 3



Set

Множество в python - "контейнер",
содержащий не повторяющиеся элементы в
случайном порядке.

Пример:

```
a = set()
```



Set

Множество в python - "контейнер",
содержащий не повторяющиеся элементы в
случайном порядке.

Пример:

```
a = set('hello')  # {'h', 'o', 'l', 'e'}
```



Set

Множество в python - "контейнер",
содержащий не повторяющиеся элементы в
случайном порядке.

Пример:

```
a = {'a', 'b', 'c', 'd'} # {'b', 'c', 'a', 'd'}
```



Set

Множество в python - "контейнер",
содержащий не повторяющиеся элементы в
случайном порядке.


Пример:

`a = {}` # А так нельзя!



Set


Создание пустого множества подразумевает определенную хитрость. Если вы используете пустые фигурные скобки `{ }` в Python, вы скорее создадите пустой словарь, а не множество.



```
x = {}
```

```
print(type(x))    #<class 'dict'>
```

Как показано в выдаче, тип переменной x является словарем.



Чтобы создать пустое множество в Python, мы должны использовать функцию `set()` без передачи какого-либо значения в параметрах, как показано ниже:

```
x = set()
```

```
print(type(x))    #<class 'set'>
```





Доступ к элементам

Python не предоставляет прямой способ получения значения к отдельным элементам множества. Однако, мы можем использовать цикл для итерации через все элементы множества. Например:

```
months = set(["Jan", "Feb", "March", "Apr", "May", "June", "July",  
"Aug", "Sep", "Oct", "Nov", "Dec"])
```

```
for m in months:
```

```
    print(m)
```



Проверка на наличие элемента

```
months = set(["Jan", "Feb", "March", "Apr", "May",  
"June", "July", "Aug", "Sep", "Oct", "Nov", "Dec"])  
  
print("May" in months)    #TRUE
```



Проверка на наличие элемента

Код возвращает «True», а это означает, что элемент был найден во множестве. Аналогичным образом, при поиске элемента, который отсутствует во множестве, мы получим «False», как показано ниже:

```
months = set(["Jan", "Feb", "March", "Apr", "May", "June",  
"July", "Aug", "Sep", "Oct", "Nov", "Dec"])
```

```
print("Nicholas" in months) # False
```



Добавление элементов во множество

Python позволяет нам вносить новые элементы во множество при помощи функции `add()`. Например:

```
months = set(["Jan", "March", "Apr", "May", "June", "July", "Aug",  
"Sep", "Oct", "Nov", "Dec"])
```

```
months.add("Feb")
```

```
print(months)
```



Удаление элемента из множеств

Python позволяет нам удалять элемент из множества, но не используя индекс, так как множество элементов не индексированы. Элементы могут быть удалены при помощи обоих методов `discard()` и `remove()`.



Удаление элемента из множеств

Помните, что метод `discard()` не будет выдавать ошибку, если элемент не был найден во множестве. Однако, если метод `remove()` используется и элемент не был найден, возникнет ошибка.



Удаление элемента из множеств

Давайте продемонстрируем как удалять элемент при помощи метода `discard()`:

```
num_set = {1, 2, 3, 4, 5, 6}
```

```
num_set.discard(3)
```

```
print(num_set)           #{1, 2, 4, 5, 6}
```



Удаление элемента из множеств

Давайте продемонстрируем как удалять элемент при помощи метода `discard()`:

```
num_set = {1, 2, 3, 4, 5, 6}
```

```
num_set.discard(3)
```

```
print(num_set)           #{1, 2, 4, 5, 6}
```




Удаление элемента из множеств

Аналогично, метод `remove()` может использоваться следующим образом:

```
num_set = {1, 2, 3, 4, 5, 6}
```

```
num_set.remove(3)
```

```
print(num_set)           #{1, 2, 4, 5, 6}
```



Удаление элемента из множеств

Теперь попробуем удалить элемент, которого нет во множестве. Сначала используем метод `discard()`:

```
num_set = {1, 2, 3, 4, 5, 6}
```

```
num_set.discard(7)
```

```
print(num_set)
```

```
#{1, 2, 3, 4, 5, 6}
```



Удаление элемента из множеств

Выдача выше показывает, что никакого воздействия на множество не было оказано. Теперь посмотрим, что выйдет из использования метода `remove()` по аналогичному сценарию:

```
num_set = {1, 2, 3, 4, 5, 6}
```

```
num_set.remove(7)
```

```
print(num_set)
```



Удаление элемента из множеств

```
num_set = {1, 2, 3, 4, 5, 6}
num_set.remove(7)
print(num_set)
```

Выдача показывает, что метод выдал ошибку KeyError, так как мы пытались удалить элемент, которого нет во множестве.

Traceback (most recent call last):

File "C:\Users\admin\sets.py", line 2, in <module>

```
num_set.remove(7)
```

KeyError: 7



Удаление элемента из множеств

С методом `pop()`, мы можем удалить и вернуть элемент. Так как элементы находятся в произвольном порядке, мы не можем утверждать или предсказать, какой элемент будет удален.

```
num_set = {1, 2, 3, 4, 5, 6}
```

```
print(num_set.pop())
```

```
#1
```



Удаление элемента из множеств

Вы можете использовать тот же метод при удалении элемента и возврате элементов, которые остаются во множестве. Например:

```
num_set = {1, 2, 3, 4, 5, 6}
```

```
num_set.pop()
```

```
print(num_set)
```

```
#{2, 3, 4, 5, 6}
```



Удаление элемента из множеств

Метод Python под названием `clear()` поможет удалить все элементы во множестве.

Например:

```
num_set = {1, 2, 3, 4, 5, 6}
```

```
num_set.clear()
```

```
print(num_set)
```



Объединение множеств

Предположим, у нас есть два множества, А и В. Объединение этих двух множеств — это множество со всеми элементами обеих множеств. Такая операция выполняется при помощи функции Python под названием `union()`.



Объединение множеств

```
months_a = set(["Jan", "Feb", "March", "Apr",  
"May", "June"])
```

```
months_b = set(["July", "Aug", "Sep", "Oct", "Nov",  
"Dec"])
```

```
all_months = months_a.union(months_b)
```

```
print(all_months)
```

```
#{'Oct', 'Jan', 'Nov', 'May', 'Aug', 'Feb', 'Sep',  
'March', 'Apr', 'Dec', 'June', 'July'}
```



Объединение множеств

Объединение может состоять из более чем двух множеств, и все их элементы сложатся в одно большое множество. Например:

$x = \{1, 2, 3\}$

$y = \{4, 5, 6\}$

$z = \{7, 8, 9\}$

```
output = x.union(y, z)
```

```
print(output)
```

При выполнении операции объединения, дубликаты игнорируются, так что только один из двух элементов дубликатов будет отображаться.



Пересечение множеств

Предположим, у вас есть два множества: А и В. Их пересечение представляет собой множество элементов, которые являются общими для А и для В.

$x = \{1, 2, 3\}$

$y = \{4, 3, 6\}$

`z = x.intersection(y)`

`print(z) # Результат: 3`



Разница между множествами

Предположим, у вас есть два множества: A и B . Разница между A и B ($A - B$) — это множество со всеми элементами, которые содержатся в A , но не в B . Соответственно, ($B - A$) — это множество со всеми элементами в B , но не в A .

Для определения разницы между множествами в Python, мы можем использовать как функцию `difference()`



Разница между множествами

```
set_a = {1, 2, 3, 4, 5}
```

```
set_b = {4, 5, 6, 7, 8}
```

```
diff_set = set_a.difference(set_b)
```

```
print(diff_set)          #{1, 2, 3}
```



Set. Задача 1

Дан список из элементов: 'hello', 'daddy',
'hello', 'mum'

Удалить из него повторяющиеся элементы.



Set. Задача 1

```
words = ['hello', 'daddy', 'hello', 'mum']  
res = set(words)  
print (res)
```