Основы Python: типы данных, операторы

Технологии и языки программирования

Юдинцев В. В.

Кафедра теоретической механики
Самарский университет

Версия 10.2016

Содержание

Кортежи

Кортеж – неизменяемый список

Кортежи (tuple) – это неизменяемые списки
Кортеж создается перечислением элементов в круглых скобках

```
1 a = (1,2,3,4,5)
```

или функцией tuple

```
1 a = tuple(range(5))
2 a
3 (0, 1, 2, 3, 4)
```

Срезы

```
a = ('Меркурий', 'Венера', 'Земля', 'Марс')
```

Первые три элемента (с 0 по 2)

```
a[0:3]
('Меркурий', 'Венера', 'Земля')
```

Каждый второй (или с шагом 2), начиная с первого

```
a[::2]
('Меркурий', 'Земля')
```

Каждый второй, начиная со второго

```
a[1::2]
('Венера', 'Марс')
```

tuple или list

- Кортеж (tuple) – это **неизменяемый** список (list): удаление и добавление элементов невозможно:
  - ~~a.append(element)~~
  - ~~a.extend((’Юпитер’, ’Сатурн’))~~
  - ~~a.insert(4,’Фаэтон’)~~
  - ~~del a[2]~~
  - ~~a.remove(’Земля’)~~
- Кортежи “работают” **быстрее** списков
- Кортежи могут использоваться как ключи словаря

```
colony = [ (1,1), (1,2), (1,3) ]

x = 2
y = 3

isOccuped = False

for cell in colony :
    if x==cell[0] and y==cell[1]:
        isOccuped = True
        break

print(isOccuped)
```

```
colony = [ (1,1), (1,2), (1,3) ]

x = 2
y = 3

isOccuped = False

for cell in colony :
    if (x, y) == cell:
        isOccuped = True
        break

print(isOccuped)
```

```
colony = [ (1,1), (1,2), (1,3) ]

x = 2
y = 3

isOccuped = (x,y) in colony

print(isOccuped)
```

или так

```
colony = [ [1,1], [1,2], [1,3] ]

x = 2
y = 3

isOccuped = [x,y] in colony

print(isOccuped)
```

Операторы и списки

Оператор +

```
1 x = (1, 2)
2
3 y = x + x
4
5 print(y)
```

Оператор +

```
1 x = (1, 2)
2
3 y = x + x
4
5 print(y)
```

```
(1, 2, 1, 2)
```

Оператор *

```
1 x = (1, 2)
2
3 y = x * 2
4
5 print(y)
```

```
(1, 2, 1, 2)
```

Строки

Строка – последовательность символов

```
message = 'Quid est veritas?'
```

Первый элемент строки

```
message[0]
'Q'
```

Отсчёт с конца строки

```
message[-3]
'a'
```

Символы с 5 по 7

```
message[5:8]
'est'
```

Движение в обратном порядке

```
message[-2:-5:-1]
'sat'
```

Многострочный текст

Для создания много длинного многострочного текста используются тройные кавычки

```
poem = '''Над небом голубым —
Есть город золотой,
С прозрачными воротами
И с яркою стеной.

А в городе том — сад:
Всё травы да цветы.
Гуляют там Животные
Невиданной красы...'''
```

Преобразование типов в строку

```
1 str(192.008)
2 '192.008'
3
4 str(192.36)*2
5 '192.008192.008'
```

Операции со строками

Сложение

```
a='Quid'
b='est'
a+' '+b
'Quid est'
```

Умножение

```
a='B'
b='z'
a+b*4
'Bzzzz'
```

Операции со строками

Разделение строки на список подстрок, разделяемых заданным символом – метод **строка.split(разделитель)**

```
a='2-12-85-06'

a.split('-')
['2','12', '85', '06']
```

Обратная операция объединения списка строк выполняется методом **разделитель.join(список)**

```
a=['2','12', '85', '06']
'*'.join(a)

'2*12*85*06'
```

Операции со строками

Разделение строки на список подстрок, разделяемых заданным символом – метод **строка.split(разделитель)**

```
a='2-12-85-06'

a.split('-')
['2','12', '85', '06']
```

Обратная операция объединения списка строк выполняется методом **разделитель.join(список)**

```
a=['2','12', '85', '06']
'*'.join(a)

'2*12*85*06'
```

Атомарные и ссылочные типы

Атомарные и ссылочные типы

Атомарные типы bool, int, float, complex копируются при присвоении

```
a = 10
b = a
b = b + 1
print(a)
10
print(b)
11
```

Переменная b содержит копию значения a

Атомарные и ссылочные типы

Ссылочные типы: списки, строки, кортежи, словари, множества

```
1 a = [1,2,3,4,5]
2 b = a
```

Атомарные и ссылочные типы

Ссылочные типы: списки, строки, кортежи, словари, множества

```
1 a = [1,2,3,4,5]
2 b = a
```

Заменяем третий элемент списка

```
1 b[2] = 0
2 print(b)
3 [1,3,0,4,5]
```

Атомарные и ссылочные типы

Ссылочные типы: списки, строки, кортежи, словари, множества

```
a = [1,2,3,4,5]
b = a
```

Заменяем третий элемент списка

```
b[2] = 0
print(b)
[1,3,0,4,5]
```

Объект (список), на который ссылается переменная **a** тоже изменился

```
print(a)
[1,3,0,4,5]
```

Переменная **b** ссылается на тот же объект, что и **a**.

Псевдоним и копия

Переменная b — это имя, которое ссылается на объект в памяти (list). На этот же объект ссылкается и имя a:

```
1 a = [1,2,3,4,5]
2 b = a
```

Для того, чтобы создать копию, необходимо явно вызвать функцию-конструктор list

```
1 a = [1,2,3,4,5]
2 b = list(a)
3 b[0] = 0
4 a
5 [1, 2, 3, 4, 5]
6 b
7 [0, 2, 3, 4, 5]
```

Изменяемые и неизменяемые типы

Список — изменяемый тип

```
a = [1,2,3,4,5]
a[0] = 0
print(a)
[1,2,0,4,5]
```

Строка – неизменяемый тип

```
word = 'мир'
print(word[1])
```

и

```
word[0] = 'i'
```

TypeError: 'str' object does not support item assignment

Множество (set)

Множества

Определение

Неупорядоченный набор неповторяющихся элементов

Создание

- При помощи функции set

```
1 s1 = set((1,2,3,4,5,6))
2 s2 = set(('d','abc',3,4,5,6))
```

- Перечисление элементов множества в {} скобках

```
1 s1 = {'d','abc',3,4,5,6}
```

Создание множества

Множество из элементов строки (букв)

```
a=set('окно')
print(a)

{'о', 'к', 'н'}
```

Множество из списка

```
a=set([1,2,3,4,6,4,6])
print(a)

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

Добавление элемента к множеству

```
1 a=set(  ['Меркурий', 'Венера', 'Земля', 'Марс']  )
2
3 a.add('Юпитер')
4
5 print(a)
```

{'Венера', 'Земля', 'Марс', 'Меркурий', 'Юпитер'}

Есть ли элемент в множестве?

```
a=set(['Меркурий','Венера','Земля','Марс'])
```

Существует ли элемент в множестве - оператор **in**

```
print('Юпитер' in a)
False
print('Марс' in a)
True
print('Юпитер' not in a)
True
```

Пересечение множеств

A & B или A.intersection(B)

```
a=set([1, 2, 3, 4])
b=set([3, 4, 5, 6])
```

Оператор **&**

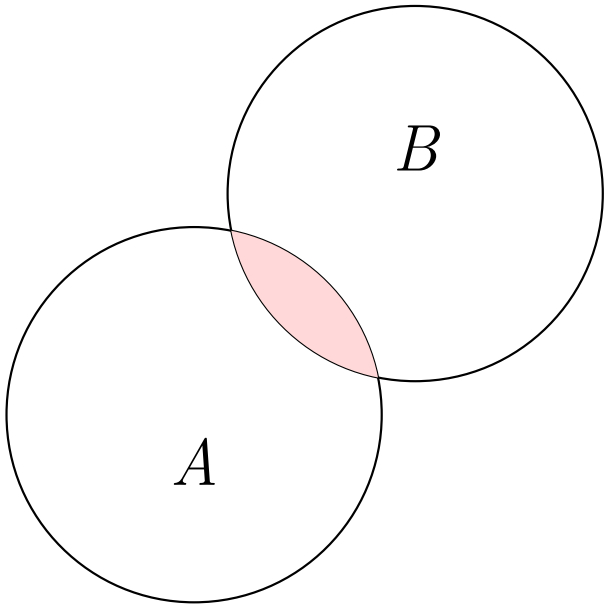
```
print(a & b)

{3, 4}
```

Метод **intersection**

```
print(a.intersection(b))

{3, 4}
```



Объединение множеств

A | B или A.union(B)

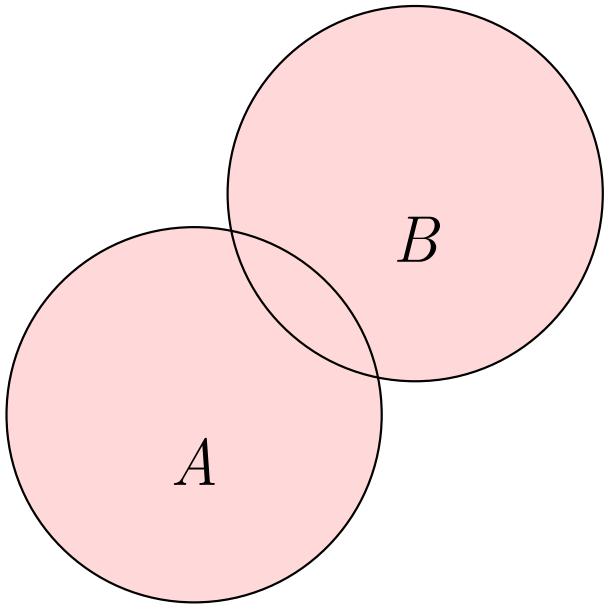
```
1 a=set([1, 2, 3, 4])
2 b=set([3, 4, 5, 6])
```

Оператор **|**

```
1 print(a  b)1, 2, 3, 4, 5, 6
```

Метод **union**

```
1 print(a.union(b))
2
3 {1, 2, 3, 4, 5, 6}
```



Разность множеств

Элементы, принадлежащие **A**, но не принадлежащие **B**

A - B или **A.difference(B)**

```
a=set([1, 2, 3, 4])
b=set([3, 4, 5, 6])
```

Оператор **-**

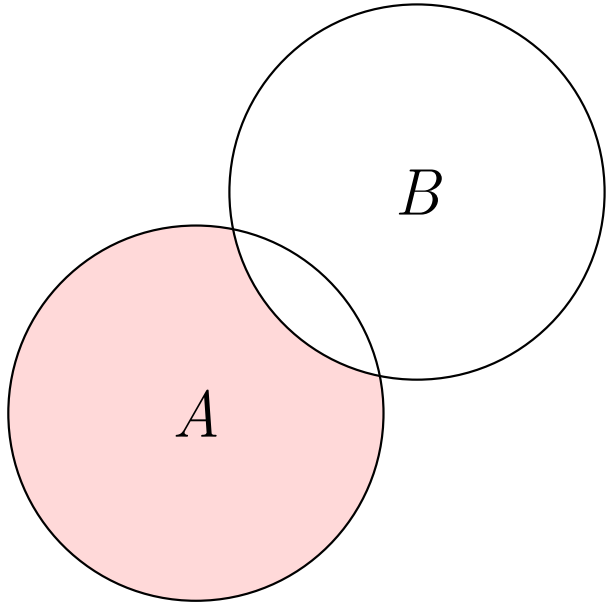
```
print(a - b)

{1, 2}
```

Метод **difference**

```
print(a.difference(b))

{1, 2}
```



Исключающее ИЛИ

Элементы, принадлежащие **A** или **B**, но не общие

A ^ B или **A.symmetric_difference(B)**

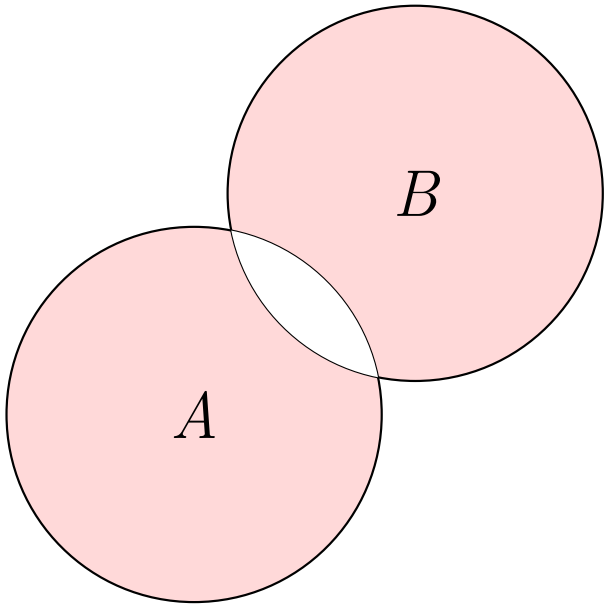
```
1 a=set([1, 2, 3, 4])
2 b=set([3, 4, 5, 6])
```

Оператор ^

```
1 print(a ^ b)
2
3 {1, 2, 5, 6}
```

Метод **symmetric_difference**

```
1 print(a.symmetric_difference(
      b))
2
3 {1, 2, 5, 6}
```



Подмножество

Все элементы, принадлежащие **A**, принадлежат **B**

A <= B или **A.issubset(B)**

```
a=set([3, 4])
b=set([3, 4, 5, 6])
```

Оператор **<=**

```
print(a <= b)

True
```

Метод **issubset**

```
print(a.issubset(b))

True
```

Подмножество

A >= B или **A.issuperset(B)**

```
a=set([1, 2, 3, 4, 5, 6])
b=set([3, 4, 5, 6])
```

Оператор **>=**

```
print(a >= b)

True
```

Метод **issuperset**

```
print(a.issuperset(b))

True
```

Задание

Координаты смежных клеток

	-1,1	0,1	1,1	
	-1,0	x,y	1,0	
	-1,-1	0,-1	1,-1	

	x-1 y+1	x y+1	x+1 y+1	
	x-1 y	x,y	x+1 y	
	x-1 y-1	x y-1	x+1 y-1	

Список координат смежных клеток

```
x, y = 2, 3

dxdy = [ ( 1, 1), ( 0, 1), (-1, 1),
         (-1, 0), (-1,-1), ( 0,-1),
         ( 1,-1), ( 1, 0) ]

nearest = list()

for (dx,dy) in dxdy:
    nearest.append((x+dx,y+dy))

print(nearest)
```

```
[(3, 4), (2, 4), (1, 4), (1, 3), (1, 2), (2, 2), (3, 2), (3, 3)]
```

Множество координат смежных клеток

```
x, y = 2, 3

dxdy = [ ( 1, 1), ( 0, 1), (-1, 1),
         (-1, 0), (-1,-1), ( 0,-1),
         ( 1,-1), ( 1, 0) ]

nearest = set()

for (dx,dy) in dxdy:
    nearest.add((x+dx,y+dy))

print(nearest)
```

{(1, 2), (3, 2), (1, 3), (3, 3), (1, 4), (2, 2), (3, 4), (2, 4)}

Координаты соседей (для списков)

```
1  colony = [ (1, 1), (1, 2), (1, 3) ]
2
3  x, y = 2, 3
4
5  neighbours = list()
6
7  for cell in nearest:
8      if cell in colony:
9          neighbours.append(cell)
10
11 print(neighbours)
```

[(1, 3), (1, 2)]

Координаты соседей (для множества)

```
1 colony = { (1, 1), (1, 2), (1, 3) }
2
3 x, y = 2, 3
4
5 neighbours = nearest & colony
6
7 print(neighbours)
```

```
{ (1, 2), (1, 3) }
```

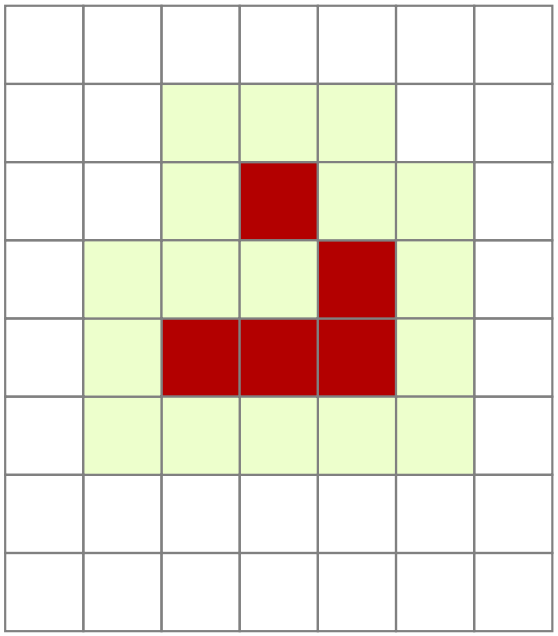
Количество соседей

Количество пар координат соседей – количество соседей

```
len(neighbours)
```

```
print("У клетки с координатами(", x, y,
      ") всего ",len(neighbours), " соседей")
```

Ареал колонии



Клетки, граничащие с колонией

Клетки, принадлежащие колонии

Ареал колонии

```
1 colony = [ (1, 1), (1, 2), (1, 3) ]
```

В ареал входят все живыве клетки

```
2 area = list(colony)
```

и все клетки, граничащие с живыми

```
3 for cell in colony:
4     for (dx,dy) in dxdy:
5         dxdy_cell = (cell[0]+dx,cell[1]+dy)
6         if dxdy_cell not in area:
7             area.append(dxdy_cell)
8
9 print(area)
```

[(1, 1), (1, 2), (1, 3), (2, 2), (0, 2), (0, 1), (0, 0), (1, 0), (2, 0), (2, 1), (2, 3), (0, 3), (2, 4), (1, 4), (0, 4)]

Ареал колонии – множество

```
1 colony = { (1, 1), (1, 2), (1, 3) }
```

В ареал входят все живыве клетки

```
2 area = set(colony)
```

и все клетки, граничащие с живыми

```
3 for cell in colony:
4     for (dx,dy) in dxdy:
5         dxdy_cell = (cell[0]+dx, cell[1]+dy)
6         area.add(dxdy_cell)
7
8 print(area)
```

{(1, 2), (0, 1), (1, 3), (0, 0), (0, 2), (1, 0), (2, 1), (1, 4), (2, 0), (2, 4), (2, 3), (0, 4), (2, 2), (0, 3), (1, 1)}

Следующее поколение

```
colony = [ (1, 1), (1, 2), (1, 3) ]

next_generation = []
for cell in area:
    cont_neighbours = 0
    for (dx,dy) in dxdy:
        nearest_cell = (cell[0]+dx, cell[1]+dy)
        if nearest_cell in colony:
            cont_neighbours+=1
    if cont_neighbours == 3 or \
    cont_neighbours == 2 and cell in colony:
        next_generation.append(cell)
print(next_generation)
```

[(1, 2), (2, 2), (0, 2)]

Список источников

1. Простой Python. Современный стиль программирования. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O’Reilly»).
2. Python 3 для начинающих `https://pythonworld.ru`
3. Python documentation `https://docs.python.org`