

Алгоритмы и структуры данных в языке Python

Лекция 2

Строки в Python

Строка – набор символов, заключенный в кавычки:

```
"Пример", 'Пример', 'тема "Строки"'
'''Пример
строки'''
```

В строках можно использовать специальные символы (экранированные последовательности):

\n – перевод строки, \t – знак табуляции, \\ – обратный слэш, \' – апостроф, \" – кавычки и другие.

```
print("C:\\\\Документы\\\\Задачи")
print("\nСтрока 1\nСтрока 2")
print(r"\\\\Строка 1\nСтрока 2")
```

Если перед открывающей кавычкой стоит символ 'r' (в любом регистре), то механизм экранирования отключается. Такая строка не может заканчиваться символом обратного слэша.

Длина строки может быть любой.

Операции над строками

Строки являются неизменяемыми объектами.

Символы строки нумеруются с нуля. Если номер отрицательный, то нумерация выполняется с конца строки.

0	1	2	3	4	5
П	р	и	м	е	р
-6	-5	-4	-3	-2	-1

Операции:

- конкатенация: `s + '1'`, `'ab' + 'cde'`
- повторение: `'Кря' * 3`
- проверка на вхождение: `'рим' in 'Пример'`
- доступ по индексу: `s[1]`, `s[-1]`
- получение среза

Операция получения среза

Срез – фрагмент строки. Формат операции:

[начало : конец : шаг]

Все параметры являются необязательными. Параметры могут быть отрицательными.

Если параметр «начало» не указан, то используется 0.

Символ с номером «конец» не входит в фрагмент. Если параметр «конец» не указан, то возвращается фрагмент до конца строки.

Если «шаг» не указан, то используется 1.

```
s = "Пример"
print( s[1:3])      ри
print( s[:2])       Пие
print( s[:-1])      ремирП
print( s[0:1])      П
print( s[-1:])       р
print( s[:-1])       Приме
```

Встроенные функции, полезные при работе со строками

`int()`, `float()` –

преобразование строки
в целое или
вещественное число;

`str()` – преобразование
объекта в строковое
представление;

`len()` – возвращает число
символов в строке;

`chr()` – возвращает строку,
состоящую из символа
с заданным кодом;

`ord()` – для заданного
символа возвращает
его код.

```
>>> int("123")
```

```
123
```

```
>>> float("-12e-5")
```

```
-0.00012
```

```
>>> str(34.56)
```

```
'34.56'
```

```
>>> len("Пример")
```

```
6
```

```
>>> chr(57), chr(255)
```

```
('9', 'ÿ')
```

```
>>> ord("9"), ord("я")
```

```
(57, 1103)
```

Методы преобразования строк

`lstrip()`, `rstrip()`, `strip()` –
создают копию строки, в
которой удалены пробелы
в начале, в конце или и в
начале и в конце строки;
`upper()` – делает все символы
строки прописными;
`lower()` – делает все символы
строки строчными;
`title()` – делает первую букву
каждого слова
прописной;
`capitalize()` – делает первую
букву прописной.

```
>>> s="  строка  "  
>>> s.lstrip()  
'строка '  
>>> s.rstrip()  
'  строка'  
>>> s.strip()  
'строка'  
>>> s.upper()  
'  СТРОКА '  
>>> "ПРИМЕР".lower()  
'пример'  
>>> "это дом.".title()  
'Это Дом.'  
>>> "это дом.".capitalize()  
'Это дом.'
```


Методы поиска и замены в строке

`find(s1 [, i [, j]])` – возвращает позицию первого вхождения фрагмента `s1` в строку. Параметры `i, j` определяют срез `[i:j]` для поиска. Если `s1` в строку не входит, то возвращает `-1`.

```
>>> s="12345 3451"  
>>> s.find("34"), s.find("34", 4), s.find("6")  
(2, 6, -1)
```

`count(s1 [, i [, j]])` – возвращает число вхождений фрагмента `s1` в строку. Если `s1` в строку не входит, то возвращает `0`.

```
>>> s.count("34"), s.count("34",4), s.count("6")  
(2, 1, 0)
```

`replace(s1, s2 [, n])` – создает новую строку, в которой фрагмент `s1` исходной строки заменяется на `s2`. `n` определяет количество замен.

```
>>> s.replace("34", "***"), s.replace("34", "***", 1)  
( '12**5 **51', '12**5 3451' )
```

Понятие списка в Python

```
>>> A = [2, 3, 78]
>>> B = ['yes', 'no', 'ok']
>>> C = [[1,2,3], [4,5,6]]
>>> D = [2, 'no', [1,3,5,7]]
>>> A[0]
2
>>> D[2]
[1, 3, 5, 7]
>>> D[1]=55
>>> D
[2, 55, [1, 3, 5, 7]]
>>> D[2][3]=10
>>> D
[2, 55, [1, 3, 5, 10]]
>>>
```


Операции над списками

Списки являются изменяемыми объектами.

Элементы списка нумеруются с нуля. Можно использовать отрицательные номера.

Операции:

- конкатенация: $[1, 2] + x + [5, 6, 7]$
- повторение: $[1, 2] * 3$, $[0] * n$
- проверка на вхождение: $2 \text{ in } [1, 2, 3]$
- доступ по индексу: $L[1]$, $L[-1]$
- получение среза: $L[1:]$, $L[::-1]$, $L[:]$
 - $L[1:5] = []$
 - $L[1:5] = [10, 20]$

Создание списка (метод append)

```
x=[]  
x.append(1)  
x.append(4)  
print(x)
```

```
y=[]  
n = int(input("n="))  
for i in range(n):  
    y.append(int(input("=> ")))  
print(y)
```

```
[1, 4]  
n=3  
=> 11  
=> 222  
=> 3333  
[11, 222, 3333]
```

Создание списка (функция list и метод split)

```
x=list("Пример 1")
print(x)
y=list(range(10,0,-1))
print(y)
s="f1 f2 f3"
print( s.split() )
print( s.split(None, 1) )
s1="f1, f2,, f3"
print( s1.split(',') )
print( s1.split(',', 1) )
```

```
['П', 'р', 'и', 'м', 'е', 'р', ' ', ' ', '1']
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
['f1', 'f2', 'f3']
['f1', 'f2 f3']
['f1', ' f2', '', ' f3']
['f1', ' f2,, f3']
```

Методы split и join

`split()` – разделяет строку на подстроки по указанному разделителю и добавляет их в список.

`rsplit()` аналогичен, но поиск символа-разделителя выполняется справа налево.

`join()` – обратная функция: преобразует последовательность в строку.

```
s="f1 f2 f3"
print( s.split() )
print( s.split(None, 1) )
s1="f1, f2,, f3"
print( s1.split(',') )
print( s1.split(',', 1) )
```

```
['f1', 'f2', 'f3']
['f1', 'f2 f3']
['f1', ' f2', '', ' f3']
['f1', ' f2,, f3']
```

```
L = ["123", "45", "67", "89"]
s1 = "-".join(L)
print(s1)
```

```
123-45-67-89
```

Создание списка (генераторы списков)

```
A = [ i*3 for i in "abc"]  
print(A)  
n = 3  
B = [ i ** 2 for i in range(1, n + 1)]  
print(B)  
from random import randint  
C = [ randint(1, 5) for i in range(n)]  
print(C)  
D = [ i*j for i in range(1,n) for j in range(1,n) if i!=j]  
print (D)
```

```
['aaa', 'bbb', 'ccc']  
[1, 4, 9]  
[1, 3, 5]  
[2, 2]
```

```
D = []  
for i in range(1,n):  
    for j in range(1,n):  
        if i!=j:  
            D.append(i*j)
```

Создание копии списка

```
A = [1, 2, [3]]
B = A[:]
C = list(A)
D = A
from copy import deepcopy
E = deepcopy(A)
A[0] = 4
A[2][0] = 0
print('A: ', A)
print('B: ', B)
print('C: ', C)
print('D: ', D)
print('E: ', E)
```

A:	[4, 2, [0]]
B:	[1, 2, [0]]
C:	[1, 2, [0]]
D:	[4, 2, [0]]
E:	[1, 2, [3]]

Перебор элементов списка

```
A = [0] * int(input("n = "))  
for i in range(len(A)):  
    A[i] = int(input("=>"))
```

```
for el in A:  
    el = el * 2  
    print(el, end = ' ')  
print('')
```

```
i = 0  
n = len(A)  
while i < n:  
    A[i] = A[i] + 1 # A[i] += 1  
    print(A[i], end = ' ')  
    i += 1  
print('')
```

```
n = 3  
=>10  
=>20  
=>30  
20 40 60  
11 21 31  
>>>
```


Многомерные списки

```
A = [[1,2,3], [4,5,6], [7,8,9]]
```

```
A = [[1,2,3],  
      [4,5,6],  
      [7,8,9]]
```

```
B = [['a11', 'a12'],  
      ['a21'],  
      ['a31', 'a32', 'a33']]
```

```
A[0][1] = 100
```

Словари

Словарь (dict) – это неупорядоченный набор из нуля или более пар «ключ: значение».

Доступ к элементу словаря осуществляется по ключу, а не по индексу.

В качестве ключа можно использовать неизменяемые объекты: числа, строки, кортежи. Значения могут быть любого типа.

Словари – изменяемый тип данных, поэтому в них можно добавлять и удалять элементы.

```
>>> D = {"yes":1, "no":0}; D  
{'no': 0, 'yes': 1}
```

Создание словаря

#указываем элементы в фигурных скобках

```
d1 = {'L': 'large', 'S': 'small'}
```

```
print(d1)
```

```
print (d1['L'])
```

#заполняем поэлементно

```
d2 = {}
```

```
d2[2] = "неуд."
```

```
d2[3] = "удовл."
```

```
d2[4] = "хор."
```

```
d2[5] = "отл."
```

```
print(d2)
```

```
print(d2[4], d2[5])
```

```
{'L': 'large', 'S': 'small'}
```

```
large
```

```
{2: 'неуд.', 3: 'удовл.', 4: 'хор.', 5: 'отл.'}
```

```
хор. отл.
```

Создание словаря (функция dict)

```
#пары ключ = значение
d3 = dict(a=1, b=2, c=3)
print(d3)
#используем словарь
d4 = dict(d3)
print(d4)
#список кортежей (ключ, значение)
d5 = dict([("Иванов", 1352), ("Петров", 2156)])
print(d5)
#список списков [ключ, значение]
d6 = dict([[1, "белый"], [4, "красный"]])
print(d6)
```

```
{'b': 2, 'c': 3, 'a': 1}
{'b': 2, 'c': 3, 'a': 1}
{'Петров': 2156, 'Иванов': 1352}
{1: 'белый', 4: 'красный'}
```

Создание словаря (другие способы)

```
#функция zip
id = ["a", "b", "c"]
val = [2, 8, 16]
L = list(zip(id, val))
print(L)
D1 = dict(zip(id, val))
print(D1)
#генераторы словарей
D2 = {i:v for (i,v) in zip(id, val)}
print(D2)
D3 = {i:0 for i in id}
print(D3)
D4 = {k:input("=>") for k in range(1,3)}
print(D4)
```

```
[('a', 2), ('b', 8), ('c', 16)]
{'b': 8, 'a': 2, 'c': 16}
{'b': 8, 'a': 2, 'c': 16}
{'b': 0, 'a': 0, 'c': 0}
=>Иванов
=>Петров
{1: 'Иванов', 2: 'Петров'}
```

Операции над словарями

Доступ к элементу []. Можно изменить или добавить элемент;

Метод get. Если ключ есть в словаре, то возвращает значение. Если ключа нет, то возвращает None или значение второго параметра;

Оператор in проверяет наличие ключа. Результат – True или False;

Функция len() возвращает количество ключей в словаре;

Оператор del удаляет элемент.

```
>>> d={'a':21, 'b':47}
>>> d['a']=25
>>> d['c']=87
>>> d
{'b': 47, 'a': 25, 'c': 87}
>>> d.get('a'), d.get('f',100)
(25, 100)
>>> 'a' in d, 'm' in d
(True, False)
>>> len(d)
3
>>> del d['a']; d
{'b': 47, 'c': 87}
```

Перебор элементов словаря

```
d={'t':84, 'b':47}
for key in d:
    print(key, d[key])

for key in d.keys():
    print(key, d[key])

for key in sorted(d.keys()):
    print(key, d[key])

for val in d.values():
    print(val)
```

t	84
b	47
t	84
b	47
b	47
t	84
	84
	47

Методы для работы со словарями

`clear()` – удаляет все элементы;
`copy()` – возвращает поверхностную копию словаря;
`keys()` – возвращает объект, содержащий все ключи;
`values()` – возвращает объект, содержащий все значения;
`items()` – возвращает объект, содержащий все ключи и значения в виде кортежей;
`pop()` – удаляет элемент с указанным ключом и возвращает его значение. Если ключ отсутствует, то возвращает значение 2 параметра;
`update()` – добавляет элемент в словарь.

```
>>> d={'t':84, 'b':47}
>>> d1=d.copy(); d1
{'t': 84, 'b': 47}
>>> d1.clear(); d1
{}
>>> d.keys()
dict_keys(['t', 'b'])
>>> d.values()
dict_values([84, 47])
>>> d.items()
dict_items([('t', 84), ('b', 47)])
>>> d.pop('t');d
84
{'b': 47}
>>> d.pop('t', None);d
{'b': 47}
>>> d.update(c=5, d=8)
>>> d
{'b': 47, 'd': 8, 'c': 5}
```

Пример использования словаря


Для введенных слов (каждое слово вводится отдельно, признак конца ввода – пустая строка) вычислить, сколько раз вводилось каждое слово.

Вывести на экран слова в порядке частоты встречаемости слов.

```
D={}
S=input("").strip()
while S!="":
    D[S] = D.get( S, 0 ) + 1
    S=input("").strip()
print(D)

T = tuple((D[k], k) for k in D)
print(T)

T1=sorted(T)
for n, w in T1:
    print(w, '-', n)
```



```
if S in D:
    D[S] = D[S]+1
else:
    D[S] = 1
```

```
aa
abc
aa
bac
aa
abc

{'aa': 3, 'abc': 2, 'bac': 1}
((3, 'aa'), (2, 'abc'), (1, 'bac'))
bac - 1
abc - 2
aa - 3
>>>
```

Множества

Множество (set) – неупорядоченный набор уникальных элементов.

Множество может содержать только элементы неизменяемых типов (числа, строки, кортежи).
Элементы могут быть разных типов.

```
s = {1, 2, 3}
s1 = set()
s2 = set("abcd")
s3 = set([1,2,2,1])
s4 = set((1,1,2,5))
print(s,s1,s2,s3,s4, sep="\n")
```

```
{1, 2, 3}
set()
{'c', 'a', 'd', 'b'}
{1, 2}
{1, 2, 5}
```

Примеры работы с множествами

```
A = {7, 4, 3, 5, 7, 4}
print('A =', A)
```

```
B = sorted(A)
print ('B =', B)
```

```
C = {t for t in range ( 1 , 11 )}
print ('C =', C)
```

```
D = {t for t in range ( 1 , 11 ) if t%3==0}
print('D =', D)
```

```
L = ['Иванов', 'Петров', 'Сидоров', 'Петров', 'Иванов']
S = set(L); print ('S =', S)
L1 = list(S); print ('L1 =', L1)
```

```
L = list( set(L) )
print ('L =', L)
```

```
A = {3, 4, 5, 7}
B = [3, 4, 5, 7]
C = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
D = {9, 3, 6}
S = {'Петров', 'Иванов', 'Сидоров'}
L1 = ['Петров', 'Иванов', 'Сидоров']
L = ['Петров', 'Иванов', 'Сидоров']
```