Строки Python

Что это?

Строка – **неизменяемая** упорядоченная последовательность символов.

S[i] – обращение к символу
S1+S2 – сложение строк
S*3 – дублирование строки

Стандартные функции

Функция	**Назначение**
len(S)	Длина строки
ord(символ)	ASCII код символа
chr(число)	Символ по коду

Срезы

Срез – извлечение фрагмента строки

S[start:stop:step]
S – строка
start – начальный символ
stop – конечный символ (символ с номером stop в срез не входит)
step - шаг

S=“Hello, World!”
S[7:12] #World
S[:5] #Hello
S[7:] #World!
S[2:-2] #llo, Worl
S[2:11:2] #lo ol
S[::3] #Hl Wl
S[::-1] - ???

Обращение к одному символу – это тоже срез

Методы для строк

Методы применяются к конкретным строкам.

Метод	**Назначение**
S.find(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или -1
S.rfind(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или -1
S.replace(from, by)	Заменить часть строки from на by
S.split(символ)	Разбиение строки по разделителю
S.join(список)	Сборка строки из списка
S.count(str, [start],[end])	Возвращает количество непересекающихся вхождений подстроки в диапазоне
S.isdigit()	Состоит ли строка из цифр
S.isalpha()	Состоит ли строка из букв
S.strip([char])	Удаление пробелов (или символов char) в начале и в конце строки

Списки Python

Что это?

Список – **изменяемая** упорядоченная последовательность данных разного типа.

```
L=[] #пустой список
L=list() #и снова пустой список
L=[1,5,”Hello, world”,42] #список со значениями
```

Также, как и для строк используются: сложение, дублирование, функция len() и срезы.

Генератор списка

```
L = [значение for параметр in диапазон]
L= [0 for i in range(5)] #список заполнят 5 элементов со значением 0
```

Методы для списков

Метод	Назначение
list.append(x)	Добавляет элемент в конец списка
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x.
list.extend(L)	Расширяет список list, добавляя в конец список L
list.insert(i, x)	Вставляет на позицию i значение x. Последующие элементы сдвигаются вправо
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)
list.count(x)	Возвращает количество элементов со значением x
list.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент.
list.sort()	Сортирует список
list.reverse()	Разворачивает список

Также для списков используются функции min(list) и max(list), возвращающие минимум и максимум из списка, а также оператор in.

Множества Python

Что это?

Множество – неупорядоченная последовательность уникальных элементов. Может содержать только данные неизменяемых типов.

A=set() #пустое множество

A= set(“hello”) #будет множество {‘h’,’o’,’l’,’e’}

Также можно использовать функции len(), max() и min().

… И генератор множества (как в списках)

```
a = {i ** 2 for i in range(10)}
# {0, 1, 4, 81, 64, 9, 16, 49, 25, 36}
```

Если использовать случайную генерацию, то значение записывается в множество только один раз и при совпадении не дублируется.

```
a = {random.randint(10) for i in range(5)}
# может быть {2, 5, 9} или {1, 2, 7, 5}
```

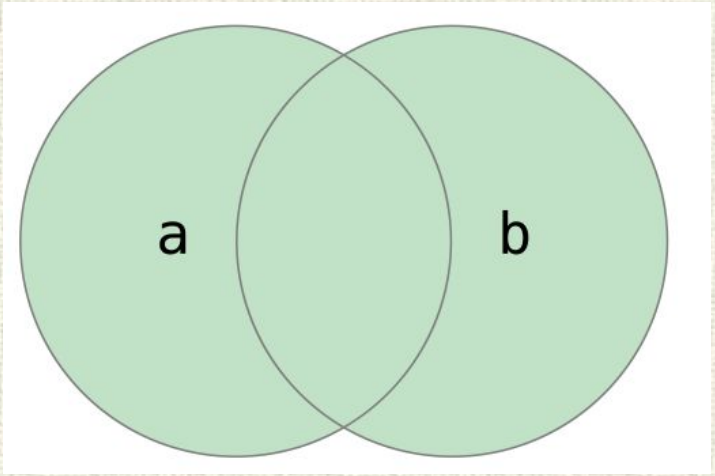
Операции с множествами

1) Объединение

Возвращает множество, содержащее неповторяющиеся элементы первого и второго множеств.

c=a | b a|=b

c=a.union(b) a.update(b)



2) Пересечение

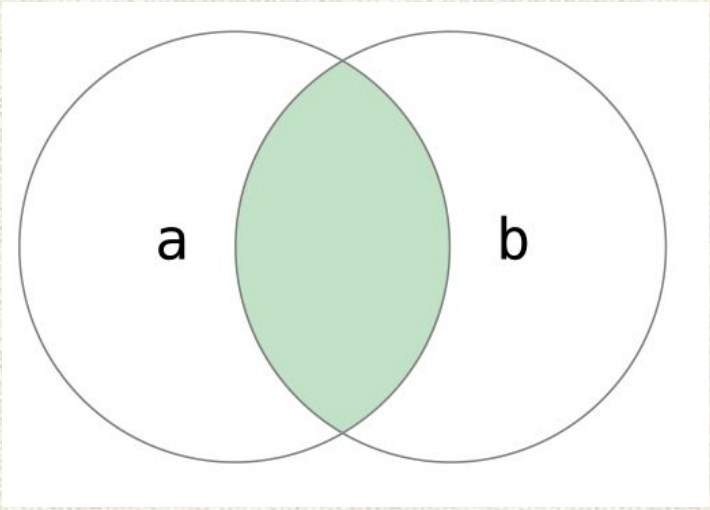
Возвращает множество, содержащее элементы, которые есть в первом и втором множествах.

c=a & b

c=a.intersection(b)

a&=b

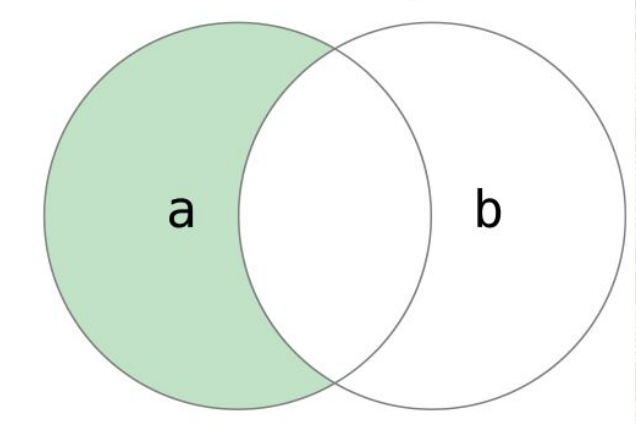
a.intersection_update(b)



Операции с множествами

3) Разность

Возвращает множество, содержащее элементы первого множества, которых нет во втором.



c=a - b

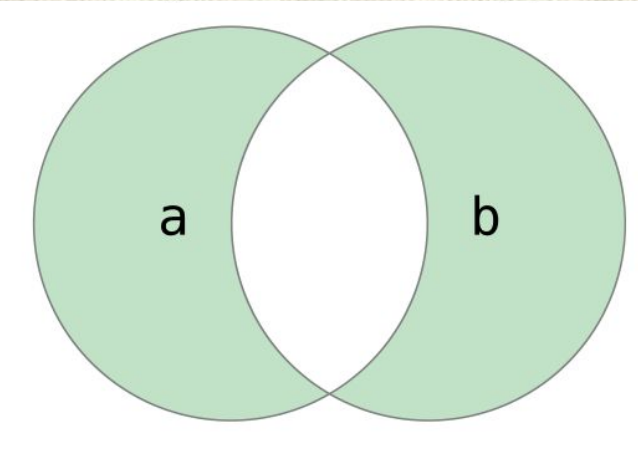
a-=b

c=a.difference(b)

a.difference_update(b)

4) Симметрическая разность

Возвращает множество, содержащее элементы, входящие в первое или второе множества, но не в оба одновременно.



c=a ^ b

c=a.symmetric_difference(b)

a^=b

a.symmetric_difference_update(b)

Методы для множеств

Метод	Назначение
set.add(elem)	добавляет элемент в множество
set.remove(elem)	удаляет элемент из множества. KeyError, если такого элемента не существует
set.discard(elem)	удаляет элемент, если он находится в множестве.
set.clear()	очистка множества

Оператор in

Возвращает значение true если элемент находится в множестве

print(3 in {3,5,7}) #true

print (‘a’ in {‘s’, ‘qwer’, ‘q’}) #false

Также существует оператор not in

Функции Python

Что это?

Функция – группа команд, объединенных одним именем, возвращающая в точку вызова какое-либо значение.

Параметры – величины, от которых зависит выполнение процедуры или функции.

Формальные – используются при описании работы функции, конкретных значений не имеют.

Фактические – передаются в функцию в точке вызова, имеют конкретные значения.

Количество и тип формальных и фактических параметров должны совпадать.

Функции

```
def Имя_Функции(параметры, через, запятую):
    тело функции
```

Переменные, объявленные внутри функции всегда (при отсутствии оператора global) являются локальными. Поэтому данные, используемые для работы функции лучше передавать через параметры, а возвращаемое значение передавать через оператор **return**.

```
def Имя_Функции(параметры, через, запятую):
    тело функции
    return значение
```

Если необходимо вернуть серию данных – используйте сложные типы данных или реализуйте алгоритм несколькими функциями.

Функции

Рекурсия – функция, вызывающая в теле сама себя.

```
def Имя_Функции(параметры):
    тело функции
    условие выхода
    вызов себя
```

```
def factorial(n):
    if n <= 0:
        return 1
    else:
        return n * factorial(n - 1)
```

Функциям можно задавать необязательные атрибуты, используя значения по умолчанию.

```
def func(a, b, c=2): # c - необязательный атрибут
    return a + b + c
func(1, 2)  # вернет 5
func(1, 2, 3)  # вернет 6
```

Анонимная – безымянная функция, вычисляющая одно выражение.

```
func = lambda x, y: x + y
func(1, 2)
(lambda x, y: x + y)(1, 2)
```