Python

v. 3.10

Краткий справочник для обучающихся старших классов

Автор-составитель: Фомин А. Т. учитель информатики и физики
МБОУ гимназии №14 г. Ейска МО Ейский район

Сайт автора: http://inf-w.ru/
последнее изменение 06.01.22 г.
v. 1.2

Оглавление

Таблица 1. Программа на языке python

```
'''
Это многострочный комментарий; или """ использовать двойные кавычки """
'''
# Это однострочный комментарий. Ниже a и b – это строки
a = input('a = ')                # Строка (здесь подсказка) заключается
b = input("b = ")                # в одинарные или двойные кавычки
c = int(input('c = '))           # ввод с преобразованием к типу int
d, e = map(int, input().split()) # ввод в одной строке с разделителем
# «пробел» и преобразованием к типу int.
# Перенос на новую строку с помощью символа '\'. После этого символа
# ничего не должно быть - только перевод на новую строку (нажать Enter)
f = c * (d + e) – d // c \
              * (10 – e)
# Неформатированный вывод:
print("Строка a =>", a, "\nСтрока b =>", b)
# Форматированный вывод:
print('c ={:>5}'.format(c),   # Можно переносить в пределах
      'd ={:>5}'.format(d),   # (), [] и {} скобок
      'e ={:>5}'.format(e),   # sep() - аргумент функции print(),
      sep='\n')               # символ-разделитель (здесь – новая строка)
a = 2; b = 5                  # Объединение нескольких логических строк
# Отступы в программе определяют блок; они имеют синтаксическое значение!
if a < 7 or b > 10:
    a += b                    # блоки должны быть одинаковой символьной
else:                         # ширины (например, однократная табуляция)
    b *= a
for j in range(10):           # вывод в одну строку; end() - аргумент
    print(a, end=' ')         # print(), завершающий символ (отмена '\n')
```

Таблица 2. Ключевые слова

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Таблица 3. Операции

Приоритет*	Операция	Описание
1	`(), [], {}`	Связывание или выражение в скобках; отображение списка, словаря, множества
2	`x[], x[:], x(), a.attr`	Индексация, срез, вызов и ссылка на атрибут
3	`await x`	Await expression
4	`**`	Возведение в степень
5	`*, @, /, //, %`	Умножение, умножение матриц, деление, деление целочисленное, остаток от деления
6	`+,-`	Сложение и вычитание
7	`<<, >>`	Побитовый сдвиг влево и сдвиг вправо
8	`&`	Побитовое И
9	`^`	Побитовый XOR (исключающее или)
10	`|`	Побитовое ИЛИ (inclusive or)
11	`in, not in, is, is not, <, <=, >, >=, !=, ==`	Проверка на вхождение, идентичность, операции сравнения
12	`not x`	Логическое НЕ
13	`and`	Логическое И
14	`or`	Логическое ИЛИ
15	`if-else`	Условное выражение
16	`lambda`	Лямбда-выражение
17	`:=`	Выражение присваивания

*** По убыванию старшинства**

Таблица 4. Литералы

Литералы	Формат	Примеры
Целые	Двоичный	`0b1010, 0B111100111`
	Восьмеричный	`0o1, 0O20, 0o7155`
	Десятичный	`0, 1992211, 33`
	Шестнадцатеричный	`0xA, 0x1B8, 0X00FF`
Логические		`True, False`
Вещественные		`3.14, 10., .001, 1e100, 3.14e-10, 0e0, 3.14_15_93`
Строковые	Последовательность символов (один или более), заключенная в одинарные или двойные кавычки	`'Здесь был Vasia', "\tЗначение r = 0xF5\n"`
Форматированной строки		`f"{today:%B %d, %Y}"`
Мнимые	Комплексные числа	`3.14j, 10.j, 10j, .001j, 1e100j, 3.14e-10j, 3.14_15_93j`

Таблица 5. Фундаментальные типы python

Название		Функция (конструктор)
Числовые типы		
Целый (Integers)		`int`
Булевский (Booleans)		`bool`
Действительный		`float`
Комплексное число		`complex`
Последовательности		
Неизменяемые (Immutable sequences)	Строки (String)	`str`
	Кортежи (Tuples)	`tuple`
	Байты (Bytes)	`bytes`

Изменяемые (Mutable sequences)	Списки (Lists)	list
	Byte Arrays	bytearray
Множества (set)	Изменяемые множества	set
	Неизменяемые множества	frozenset
Словарь (dict)		dict

Таблица 6. Стандартные функции

Название*	Возвращает
abs(x)	Абсолютное значение числа x
aiter(obj_it)	Асинхронный итератор для obj_it
all(it)	True, если все элементы итерации (it) имеют значение True
any(it)	True, если какой-либо элемент итерации (it) имеет значение True
anext(it)	Следующий элемент асинхронной итерации (it) или по умолчанию
ascii(obj)	Строку в которой не ascii-символ будет заменен escape-последовательностью
bin(x)	Строковое представление целого числа x в виде двоичного числа с префиксом 0b
bool([x])	Логическое значение**
breakpoint(*args, **kws)	Режим отладки
bytearray([source[, encoding[,errors]]])	Массив байтов (изменяемую последовательность целых чисел в диапазоне 0 <= x < 256)
bytes([source[, encoding[,errors]]])	Неизменяемую последовательность байтов (целых чисел в диапазоне 0 <= x < 256)
callable(obj)	True, если аргумент (obj) является вызываемым
chr(i)	Строку в виде символа. i - кодовая точка Юникода
@classmethod	Метод класса (является декоратором)
compile(source, filename, mode,	Код или объект AST (абстрактные синтаксические деревья)

Название*	Возвращает
`flags=0, dont_inherit=False, optimize=-1)`	
`complex([real[,imag]])`	Комплексное число
`delattr(obj,name)`	Удаление атрибута name объекта obj (если позволяет объект)
`dict(it,**kwarg)`	Словарь
`dir([obj])`	Список имен (всех / obj)
`divmod(a, b)`	Кортеж из частного и остатка от деления (a//b, a%b)
`enumerate(it, start=0)`	Кортеж в виде пронумерованной последовательности it (начиная с start)
`eval(expr[,globals[, locals]])`	Результат вычисления выражения expr (в виде строки)
`exec(obj[,globals[, locals]])`	Динамическое выполнение кода
`filter(fun, it)`	Итератор из элементов последовательности (it) для которых функция (fun) возвращает значение True
`float([x])`	Число с плавающей точкой сконструированной из числа или строки x
`format(val[, format_spec])`	Форматированное представление val
`frozenset([it])`	Неизменяемое множество
`getattr(obj, name[, default])`	Значение названного атрибута obj
`globals()`	Словарь имен текущего пространства имен модуля
`hasattr(obj, name)`	True, если строка является именем name одного из атрибутов объекта obj
`hash(obj)`	Хэш-значение объекта obj
`help([obj])`	Вызов встроенной справочной системы
`hex(x)`	Результат преобразования целого числа x в шестнадцатеричное строковое представление в нижнем регистре с префиксом 0x
`id(obj)`	Идентификатор объекта obj
`input([promt])`	Строку подсказки promt (если есть) выводит

Название*	Возвращает
	в стандартный поток без символа новой строки. Затем считывает и возвращает строку из ввода
`int([x]) или int(x, base=10)`	Целое число сконструированное из числа или строки x (или 0, если нет аргументов)
`isinstance(obj, cls)`	True, если объект obj является экземпляром класса cls
`issubclass(class, cls)`	True, если class является подклассом класса cls
`iter(obj[,sentinel])`	Объект-итератор
`len(s)`	Длину объекта s (количество элементов в коллекции)
`list([it])`	Список
`locals()`	Словарь (таблицу) локальных имен
`map(fun, it, ...)`	Итератор, который применяет функцию fun к каждому элементу итерируемого объекта it
`max(it, *[,key, default]) или max(args[, key])`	Наибольший элемент аргумента (-ов)
`memoryview(obj)`	Объект «представления памяти», созданного из объекта obj
`min(it, *[,key, default]) или min(args[, key])`	Минимальный элемент аргумента (-ов)
`next(it[, default])`	Следующий элемент итерации
`object`	Объект (это класс)
`oct(x)`	Восьмеричное представление числа x в виде строки с префиксом «0o»
`open(file, [mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None])`	Открывает файл и возвращает файловый объект. file — абсолютный или относительный путь к файлу. Необязательные аргументы: mode — режим открытия; buffering — политика буферизации; encoding — кодировка (для текстовых файлов); errors — обработка ошибок кодирования и декодирования; newline — режим новой строки (None —

Название*	Возвращает
	определять автоматически); closefd — имеет значение False, если задан дескриптор файла, а не имя; opener — объект открытия файла
`ord(c)`	Целое число, код символа c
`pow(base, exp[, mod])`	Возведение base в степень exp или pow(base, exp) % mod
`print(*objects[, sep=' ', end='\n', file=sys.stdout, flush=False])`	Записывает объекты в поток вывода. sep — символ разделитель; end — завершающий символ; file — объект метода write(), используется только с текстовыми файлами; flush — политика буферизации
`property(fget=None, fset=None, fdel=None, doc=None)`	Свойства атрибута
`range(stop)` или `range(start, stop[, styep])`	Неизменяемый тип последовательности (арифметическая прогрессия)
`repr(obj)`	Представление объекта в виде строки
`reversed(seq)`	Реверсивный итератор объекта seq
`round(d[, n])`	Округление d с указанной точностью n. Если точность не указана, то до ближайшего целого
`set([it])`	Множество из итерируемого объекта it
`setattr(obj, name, val)`	Устанавливает значения val атрибута name объекта obj
`slice(stop)` или `slice(start,stop[,step])`	Объект среза
`sorted(it, key=None, reverse=False)`	Новый отсортированный список из итерации it. key — ключ сортировки, reverse — изменяет направление сортировки
`@staticmethod`	Преобразует метод в статический (является декоратором)
`str(obj='')` или `str(obj=b'', encoding='utf-8', errors='strict')`	Строковую версию объекта obj
`sum(it[, start=0])`	Сумму элементов (числовой)

Название*	Возвращает
	последовательности it, начиная с указанной позиции start
`super([type[, obj-or-type]]`	Объект, который делегирует вызовы метода родительскому или родственному классу type. obj-or-type — определяет порядок поиска
`tuple([it])`	Кортеж
`type(obj) или type(name, bases, dict, **kwds)`	Тип объекта
`vars([obj])`	Словарь локальных имен
`zip(*iterables, strict=False)`	Итератор кортежей

*** Доступны в любом месте программы**

**** Здесь и далее, квадратные скобки означают необязательную часть**

Таблица 7. Метод format

Спецификация формата								
[[fill]	align]	[sign]	[#]	[0]	[with]	[group]	[.prec]	[type]

`fill`	Символ-заполнитель	Любой символ
`align`	Выравнивание	"<" | ">" | "=" | "^"
`sign`	Знак	"+" | "-" | ""
`with`	Ширина	Целое число
`group`	Группировка	"_" | ","
`prec`	Дробные знаки	Целое число
`type`	Тип	"b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"

Тип	Описание
	String
s	По умолчанию
None	Тоже, что и «s»

Тип	Описание
	Integer
b	Двоичное представление целого числа. Если указан «#», то с префиксом «0b»
c	Преобразование в символ Юникода
d	Десятичное представление
o	Восьмеричное представление. Если указан «#», то с префиксом «0o»
x	Шестнадцатеричное представление. Если указан «#», то с префиксом «0x»
X	Тоже, но в верхнем регистре
n	Тоже, что и «d», но с учетом локали (не работает для русской локали)
None	Тоже, что и «d»
	Float, Decimal
e	Научная нотация (формат с плавающей точкой). Если не указан «#», то, в случае отсутствия дробных знаков, точка удаляется
E	Тоже, но в верхнем регистре
f	Формат с фиксированной точкой. Если не указан «#», то, в случае отсутствия дробных знаков, точка удаляется
F	Тоже, что и f, но обозначения NAN и INF в верхнем регистре
g	Обший формат. Для заданной точности округляет число до значащих цифр, а затем форматирует результат либо в формате с фиксированной точкой, либо в экспоненциальном представлении, в зависимости от его величины. Если точность не указана, то используется 6 значащих цифр
G	Тоже, что и «g», но переключается на «E», если число слишком большое
n	Тоже, что и «g», но с учетом локали (не работает с русской локалью)
%	Представление в процентах (умножает на 100 и отображает в формате «f», за которым следует знак «%»)
None	Тоже, что и «g»

Таблица 8. Основные конструкции языка python

Модули	Строки программы
`# Импорт модуля без включения имен:`	`Переносить логические строки программы`

```
import turtle
import math, decimal
# вызов метода с помощью dot-нотации:
x = math.sqrt(2)
# Импорт всех имен модуля:
from turtle import *
# вызов метода по имени
reset()
# Импорт некоторых имен модуля:
from math import sqrt, pi
x = sqrt(pi)
# Подмена имени:
import turtle as t
t.reset()
```

```
можно только после символа '\'
if a < b / c and\
    a < 0.5  and\
    b > 3.0:
# После символа '\' не должно быть
# никаких других
# Допускается переносить внутри (), {} и
# [] скобок:
print('a =', a, # Можно
      'b =', b, # использовать
      'c =', c) # комментарии
L = [[1, 2, 3],
     [4, 5, 6],
     [7, 8 ,9]]
```

Ввод

```
# X – это тип str:
X = input() # без подсказки
# вывод строки-подсказки
X = input('X = ')
# Для преобразования входных данных в
# другие типы используются
# соответствующие конструкторы классов:
X = int(input()) # целое
X = float(input()) # действительное
# Ввод строки с разделителем.
# Разделитель – пробел; x и y – тип float
x, y = map(float, input().split())
# Разделитель - ':'; h, m и s – тип int
h, m, s = map(int, input().split(':'))
```

Вывод

```
print(*objects, sep=' ', end='\n',
file=sys.stdout, flush=False)
Аргументы:
*objects – список объектов вывода
разделенных запятой;
sep – символ-разделитель (по умолчанию –
пробел);
end – завершающий символ (по умолчанию
перевод на новую строку);
file – файловый объект используемый для
стандартного вывода (по умолчанию –
системный);
flush – буферизация вывода (True –
принудительный сброс)
```

Инструкция if- elif - else*

```
if выражение:
    Инструции1
[elif выражение:
    Инструкции2
else:
    Инструкции3]
```

Условное выражение if - else

```
Тернарная операция
# выражение1 будет выполняться, если
# выражение_условие будет == True,
# иначе выражение2:
выражение1 if выражение_условие else
выражение2
```

Инструкция цикла for по коллекции

```
for переменная in последовательность:
    Инструкции
[else:
    Инструкции] # Будут выполнены
# Заполнение и вывод элементов списка
L = []
```

Инструкция цикла while

```
while условие:
    Инструкции
[else:
    Инструкции]
# Инструкции необязательного блока будут
# выполнены, если выполнялись шаги цикла
# (если условие было == True). Условие –
```

```
# Заполнение числами от 0 до 99
for j in range(100):
    L.append(j)
for j in L: # Вывод элментов
    print(j)
```

```
# это любое выражение скалярного типа.
# Бесконечный цикл:
while True:
    Инструкции # или
while 1:
    Инструкции
```

Функции

```
# Описание функции
def имя(Список формальных параметров):
    '''Документирование функции'''
    Инструкции
    return выражение
# Список формальных параметров – это
# локальные переменные функции
# Необязательная строка документирования
# выражение – выражение возвращаемого
# значения. Вызов функции:
x = имя(Список аргументов)
# global – объявление глобальной
# видимости в пределах модуля
# nonlocal – позволяет ссылаться на любую
# перменную, внешнюю, по отношению к
# данной функции
```

Локальная и глобальная видимость

```
a = 1 # a – глобальная переменная
def fun1():
    a = 10 # a – локальная перменная
fun1()
print(a) # 1
def fun2():
    global a # теперь a - глобальная
    a = 10
fun2()
print(a) # 10
def fun3():
    b = 3 # b – локальная переменная
    def fun4():
        nonlocal b # b – глобальная для
        b = 6      # fun4()
        return b
    return fun4()
print(fun3()) # 6
```

Исключения

```
try:
    # Здесь располагается код, который
    # может вызвать исключение
    [raise Exception("message")]
    # Инструкция raise возбуждает
    # исключение. Exception, это один из
    # типов исключения м. б. стандартный
    # или пользовательский
except (Тип исключения1, Тип исключения2,
    ...) [as Переменная]:
    # Код в блоке выполняется, если тип
    # исключения совпадает с одним из
    # типов исключений или является
    # наследником одного из этих типов.
    # Полученное исключение доступно в
    # необязательной переменной.
[except (Тип исключения3,
         Тип исключения4, ...):]
    # Количество блоков except не
```

Классы

```
# Определение класса
class ИмяКласса1:
    pass # ничего не содержит
# Поля класса не обязательно объявлять в
# блоке класса, а можно создавать вне
# блока. Экземпляр (объект) класса:
Объект1 = ИмяКласса1()
# определяем поля класса (как структура в
# языках C/C++):
Объект1.поле1 = 10
Объект1.поле2 = [] # Список
# Заполняем список:
for j in range(10):
    Объект1.поле2.append(j)
# Автоматическая инициализация полей:
class ИмяКласса2:
    # Метод инициализации
    def __init__(self):
        self.поле1 = 20
```

```
    # ограничено
    [raise] # Сгенерировать исключение
    # "поверх" полученного; без
    # параметров - повторно сгенерировать
[except:] # Будет выполнено при любом
    # исключении, не обработанном
    # типизированными блоками except
[else:]
    # Выполняется код блока, если не было
    # перехвачено исключений.
[finally:]
    # Будет исполнено в любом случае,
    # после соответствующего
    # блока except или else
```

```
    # Метод (функция) класса
    def Имя_метода(self):
        return 5 * self.поле1
Объект2 = ИмяКласса2() # Создание объекта
print(Объект2.поле1) # 20
print(Объект2.Имя_метода()) # 100
# self – первый обязательный аргумент,
# которому передается экземпляр класса.
class ИмяКласса3: # Передача аргументов
    def __init__(self, val1, val2):
        self.поле1 = val1
        self.поле2 = val2
    def Swap(self): # возвращаем кортеж
        return self.поле2, self.поле1
    def Sum(self): # возвращаем сумму
        return self.поле1 + self.поле2
x, y = 20, 30 # x = 20 y = 30
Объект3 = ИмяКласса3(x, y)
x, y = Объект3.Swap() # Обмен значениями
print(Объект3.Sum()) # 50
```

Таблица 9. Методы модуля math

Функция	Описание
Округление, комбинаторика, теория чисел	
`ceil(x)`	Округляет число x вверх (“потолок”), при этом ceil(1.5) == 2, ceil(-1.5) == -1. Возвращает целый тип
`comb(n, k)`	Возвращает количество способов выбрать k элементов из n элементов без повторений и без учета порядка (число сочетаний из n элементов по k элементов) $C_n^k = \frac{A_n^k}{k!} = \frac{n!}{k!(n-k)!}$
`copysign(x, y)`	Возвращает абсолютное значение x (тип float) со знаком y
`fabs(x)`	Абсолютное значение вещественного числа x
`factiorial(x)`	Возвращает целое – факториал x. Если аргумент действительное или отрицательное число возбуждается исключение ValueError
`floor(x)`	Округляет число x вниз (“пол”), при этом floor(1.5) == 1, floor(-1.5) == -2
`fmod(x, y)`	Деление по модулю действительных чисел x и y

Функция	Описание
frexp(x)	Возвращает мантиссу и порядок числа как кортеж
fsum(it)	Возвращает точную сумму действительных элементов в итерации
gcd(*integers)	Наибольший общий делитель произвольного числа элементов
isclose(a, b, rel_tol = 1e-09, abs_tol = 0.0)	Возвращает True, если a и b близки друг к другу. rel_tol – макс. допустимая разница. abs_tol – мин. абс. допуск, полезен для сравнения с 0.
isqrt(n)	Возвращает целочисленный квадратный корень целоги числа n (наиб. целое a такое, что $a^2 \leqslant n$)
lcm(*integers)	Наименьшее общее кратное произвольного числа элементов
modf(x)	Разлагает на целую и дробную части. Оба действительных числа имеют знак x
perm(n, k=None)	Возвращает количество способов выбрать k элементов из n элементов без повторений, с учетом порядка число размещений из n элементов по k) $A_n^k = n(n-1)\cdot\ldots\cdot(n-k+1) = \frac{n!}{(n-k)!}$, если k=None, то функция возвращает n!
trunc(x)	Округление x в сторону нуля (отбрасывание дробной части). Возвращает целое
Корни, степени и логарифмы	
exp(x)	Экспонента, возвращает e^x
log(x[, base])	С одним аргументом возвращает натуральный логарифм
log2(x)	Точное значение логарифма x по основанию 2
log10(x)	Аналогично, по основанию 10
pow(x, y)	Возвращает x^y, оба аргумента преобразуются к float. Стандартную pow() и операцию ** используйте для целых чисел
sqrt(x)	Квадратный корень из x
Тригонометрия	
sin(x)	Синус угла, задаваемого в радианах
cos(x)	Косинус угла, задаваемого в радианах
tan(x)	Тангенс угла, задаваемого в радианах
asin(x)	Арксинус, возвращает значение в радианах
acos(x)	Арккосинус, возвращает значение в радианах

Функция	Описание
atan(x)	Арктангенс, возвращает значение в радианах
atan2(x, y)	Вычисляет арктангенс (x, y) с использованием знаков аргументов, для правильного определения квадранта (угол ф в полярных координатах)
hypot(*coord)	Возвращает евклидову норму (длина вектора от начала координат до точки, заданной координатами coord; радиус-вектор r в полярных координатах). Для двух координат равносильно вычислению гипотенузы: sqrt(x*x + y*y)
degrees(x)	Преобразует угол из радианов в градусную меру
radians(x)	Обратные преобразования мер угла
Константы	
pi	Константа π = 3,141592...
e	Число Эйлера e = 2,718281...
τ	Тау (постоянная окружности, 2π), τ = 6,283185...

Таблица 10. Общие методы последовательностей

Метод	Описание
x in s	Возвращает True, если элемент x есть в последовательности s
x not in s	Возвращает True, если элемента x нет в последовательности s
s + t	Объединение двух последовательностей s и t
s * n	Добавление к s копии s, n раз
s[i]	Индексирование элементов, начиная с 0
s[i:j:k]	Срез от i до j с шагом k
len(s)	Длина последовательности s
min(s)	Минимальный элемент последовательности s
max(s)	Максимальный элемент последовательности s
s.index(x[, i[, j]])	позиция первого вхождения x в s (начиная с i (входит) и заканчивая j (не входит)
s.count(x)	Количество вхождений x в s

Таблица 11. Методы list

Метод	Описание
`L.append(x)`	Добавляет элемент в конец списка. Эквивалентно `L[len(L):] = [x]`
`L.extend(it)`	Расширяет список, добавляя все элементы из коллекции it. Эквивалентно `L[len(L):] = iterable`
`L.insert(i, x)`	Вставляет элемент в заданной позиции. Первый аргумент является индексом элемента перед которым осуществляется вставка
`L.remove(x)`	Удаляет из списка первый элемент, значение которого равно x. Возвращается ошибка, если такой элемент не будет найден
`L.pop([i])`	Удаляет элемент в данной позиции i и возвращает его. Если не указан индекс, `L.pop()` удаляет и возвращает последний элемент из списка. (Квадратные скобки означают, что параметр является необязательным)
`L.clear()`	Удаляет из списка все его элементы. Эквивалентно `del L[:]`
`L.sort(key=None, reverse=False)`	Сортирует список на месте (в отличие от sorted). Гарантированно стабильна (порядок элементов сохраняется)
`L.reverse()`	Реверс элементов списка (расположение элементов меняется на обратный порядок)
`L.copy()`	Возвращает неполную копию списка. Эквивалентно `L[:]`

Таблица 12. Escape-последовательности

Символ	Описание
`\newline`	backslash, игнорирует переход на новую строку
`\\`	backslash
`\'`	одинарная кавычка
`\"`	двойная кавычка
`\?`	вопросительный знак
`\\`	обратный слеш
`\a`	звуковой сигнал (ASCII Bell (BEL))
`\b`	забой (ASCII Backspace (BS))

Символ	Описание
\f	новая страница (ASCII Formfeed (FF))
\n	новая строка (ASCII Linefeed (LF))
\r	возврат каретки (ASCII Carriage Return (CR))
\t	горизонтальная табуляция (ASCII Horizontal Tab (TAB))
\v	вертикальная табуляция (ASCII Vertical Tab (VT))
\ooo	символ с восьмеричным значением
\xhh	символ с шестнадцатеричным значением
\N{name}	Имя символа в базе данных Unicode
\uxxxx	Символ с 16-битным шестнадцатеричным значением
\Uxxxxxxxx	Символ с 32-битным шестнадцатеричным значением

Таблица 13. Методы str

Метод	Описание
S.find(S1[, start[, end]])	Поиск подстроки S1 в строке S. Возвращает позицию первого вхождения или -1. Необязательные аргументы **start** и **end** интерпретируются как срез
S.rfind(S1[, start[, end]])	Поиск подстроки S1 в строке S. Возвращает позицию последнего вхождения или -1. Необязательные аргументы **start** и **end** интерпретируются как срез
S.index(S1[, start[, end]])	Аналогичен **find()**, но вызывает ошибку **ValueError**, если подстрока не найдена.
S.replace(S1, S2[, count])	Заменяет подстроку S1 подстрокой S2 не больше, чем первые count вхождений
S.split(char)	Получение списка по разделителю char
S.splitlines()	Получение списка строк разбитых символом разрыва строки (не включается в результирующий список)
S.isdigit()	Состоит ли строка из цифр (**True**, если да)
S.isalpha()	Состоит ли строка из букв
S.isalnum()	Состоит ли строка из цифр или букв

Метод	Описание
S.islower()	Состоит ли строка из символов в нижнем регистре
S.isupper()	Состоит ли строка из символов в верхнем регистре
S.isspace()	Состоит ли строка из пробельных символов
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
S.join(list)	Сборка строки из списка с разделителем S
S.center(width[, fill])	Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию)
S.count(S1[, start[, end]])	Возвращает количество непересекающихся вхождений подстроки S1
S.strip()	Удаление пробельных символов в начале и в конце строки
S.format(*args, **kwargs)	Форматирование строки

Таблица 14. Операции set и frozenset

Метод	Описание
len(Set)	Мощность множества (количество элементов)
x in Set x not in Set	Тест на членство (вхождение)
Set.isdisjoint(other)	Возвращает True, если нет общих членов с other
Set.issubset(other)	Проверяет, все ли элементы находятся в other
Set <= other	
Set < other	Проверяет, что Set является подмножеством other
Set.issuperset(other)	Проверяет, все ли элементы множества other есть в Set
Set >= other	
Set > other	Проверяет, является ли Set надмножеством
Set.union(*others)	Объединение множеств (возвращает новое множество)
Set | other | …	

Метод	Описание
Set.intersection(*others)	Пересечение множеств (возвращает множество в котором только элементы общие для Set и other(s))
Set & other & …	
Set.difference(*others)	Возвращает множество элементов, которых нет в other(s)
Set - other - …	
Set.symmetric_difference(other)	Возвращает элементы, которые есть либо в Set, либо в other, но не в обоих одновременно (исключающее ИЛИ)
Set ^ other	
Set.copy()	Возвращает неполную копию множества

Таблица 15. Модификаторы set

Метод	Описание
Set.update(*others)	Обновляет множество элементами других множеств
Set |= other | …	
Set.intersection_update (*others)	Обновляет множество, сохранив только те элементы, которые являются общими с другими множествами (пересечение)
Set &= other & …	
Set.difference_update(*others)	Обновляет множество, удалив элементы, найденные в других множествах
Set -= other | …	
Set.symmetric_difference_update (other)	Обновляет множество, сохранив только элементы, которые найдены в любом другом множестве, но не являются общими (исключающее или)
Set ^= other	
Set.add(elem)	Добавляет элемент elem в множество
Set.remove(elem)	Удаляет элемент elem из множества. Возбуждается исключение KeyError, если такого элемента в множестве нет
Set.discard(elem)	Удаляет elem, если он присутствует в множестве
Set.pop()	Удаляет и возвращает произвольный элемент из множества. Возбуждается исключение KeyError, если набор пуст
Set.clear()	Удаляет все элементы из множества