СПРАВОЧНИК ШКОЛЬНИКА

по языку программирования Python

Составитель: А.Г. Гильдин

СПИСКИ в языке программирования Python

Примеры интерактивной работы со списками

```
>>> s = []  # Пустой список
>>> l = ['s', 'p', ['isok'], 2]
>>> del l[1] # Удалили элемент с индексом 1
>>> s
[]
>>> l
['s', ['isok'], 2]
>>> list('список')
['с', 'п', 'и', 'с', 'о', 'к']
```

Таблица методов работы со списками. В таблице список обозначен словом li.

Метод или команда	Что делает
li.append(x)	Добавляет элемент в конец списка
li.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
li.insert(i, x)	Вставляет на i-ый элемент значение x
li.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
li.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
li.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)
li.count(x)	Возвращает количество элементов со значением x
li.sort([key=функция])	Сортирует список на основе функции
li.reverse()	Разворачивает список
li.copy()	Поверхностная копия списка
li.clear()	Очищает список
len(li)	Возвращает количество элементов списка.

СТРОКИ в языке программирования Python

Таблица функций и методов работы со строками. В таблице строка обозначена S

Метод или команда	Что делает
S = 'str'; S = "str"; S = '''str'''; S = """str"""	Литералы строк
S = "s\np\ta\nbbb"	Служебные символы (Экранированные последовательности) например, \n означает перевод строки
S = r"C:\temp\new"	Неформатированные строки (подавляют экранирование)
S = b"byte"	Строка байтов
S1 + S2	Конкатенация (сложение строк)
S1 * 3	Повторение строки
S[i]	Обращение к символу строки с номером i. Первый символ строки имеет индекс 0.
S[i:j:step]	Извлечение среза
len(S)	Длина строки
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.index(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
S.rindex(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError
S.replace(фрагмент1, фрагмент2,[count])	Заменяет в строке не более count фрагментов1 на фрагменты2. Если count не указан, заменяет все фрагменты.
S.split(символ)	Разбиение строки по разделителю. Возвращает список
S.isdigit()	Состоит ли строка из цифр
S.isalpha()	Состоит ли строка из букв
S.isalnum()	Состоит ли строка из цифр или букв
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 неотображаемых символов (пробел, символ перевода страницы ('\f'), "новая строка" ('\n'), "перевод каретки" ('\r'),

Метод или команда	Что делает
	"горизонтальная табуляция" ('\t') и "вертикальная табуляция" ('\v'))
S.istitle()	Начинаются ли слова в строке с заглавной буквы
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
S.startswith(str)	Начинается ли строка S с фрагмента str. Можно добавлять еще два параметра (beg, end): с какого символа искать и до какого.
S.endswith(str)	Заканчивается ли строка S фрагментом str. Можно добавлять еще два параметра (beg, end): с какого символа искать и до какого.
S.join(список)	Возвращает строку, собирая ее из списка. Между элементами списка вставляется разделитель равный строке S.
ord(символ)	Возвращает ASCII код символа
chr(число)	Возвращает символ по его ASCII коду
S.capitalize()	Возвращает строку, в которой первый символ в верхнем регистре, а все остальные в нижнем
S.center(width, [fill])	Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию)
S.count(str, [start],[end])	Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)
S.expandtabs([tabsize])	Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам
S.lstrip([chars])	Возвращает строку без пробельных символов в начале строки
S.rstrip([chars])	Возвращает строку без пробельных символов в конце строки
S.strip([chars])	Возвращает строку без пробельных символов в начале и в конце строки
S.partition(шаблон)	Возвращает кортеж, содержащий часть перед первым шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий саму строку, а затем две пустых строки
S.rpartition(sep)	Возвращает кортеж, содержащий часть перед последним шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий две пустых строки, а затем саму строку
S.swapcase()	Возвращает строку, в которой переведены символы нижнего регистра в верхний, а верхнего – в нижний
S.title()	Возвращает строку, в которой первая буква каждого слова переведена в верхний регистр, а все остальные в нижний

Метод или команда	Что делает
S.zfill(width)	Возвращает строку, в которой делает длину строки не меньшей width, по необходимости заполняя первые символы нулями
S.ljust(width, fillchar=" ")	Возвращает строку, в которой делает длину строки не меньшей width, по необходимости заполняя последние символы символом fillchar
S.rjust(width, fillchar=" ")	Возвращает строку, в которой делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar
S.format(*args, **kwargs)	Форматирование строки. Подробнее здесь: Форматирование строки
int(s)	Пример использования преобразования типов данных. Возвращает число, записанное в строке s.

Материалы этого справочника подготовлены с использованием материалов сайтов:
https://pythonworld.ru/tipy-dannyx-v-python/stroki-funkcii-i-metody-strok.html
https://pythontutor.ru/lessons/lists/

Генерация псевдослучайных чисел в языке программирования Python (модуль random)

Модуль random позволяет генерировать случайные числа. Прежде чем использовать модуль, необходимо подключить его с помощью инструкции:

```
import random
```

Метод или команда	Что делает
random.random()	возвращает псевдослучайное число от 0.0 до 1.0
random.seed()	настраивает генератор случайных чисел на новую последовательность. По умолчанию используется системное время. Если указать в скобочках необязательное значение параметра, то генерируется одна и та же последовательность
random.uniform(Начало, Конец)	возвращает псевдослучайное вещественное число в диапазоне от Начало до Конец
random.randint(Начало, Конец)	возвращает псевдослучайное целое число в диапазоне от Начало до Конец
random.choince(Последовательность)	возвращает случайный элемент из любой последовательности (строки, списка, кортежа)
random.randrange(Начало, Конец, Шаг)	возвращает случайно выбранное число из последовательности
random.shuffle(Список)	перемешивает последовательность (изменяется сама последовательность). Поэтому функция не работает для неизменяемых объектов

Вероятностные распределения. (Для умных взрослых, которые занимаются математикой и решили почитать наш справочник)

`random.triangular(low, high, mode)` — случайное число с плавающей точкой, `low ≤ N ≤ high`. `Mode` - распределение.

`random.betavariate(alpha, beta)` — бета-распределение. `alpha>0`, `beta>0`. Возвращает от 0 до 1.

`random.expovariate(lambd)` — экспоненциальное распределение. `lambd` равен 1/среднее желаемое. `Lambd` должен быть отличным от нуля. Возвращаемые значения от 0 до плюс бесконечности, если `lambd` положительно, и от минус бесконечности до 0, если lambd отрицательный.

`random.gammavariate(alpha, beta)` — гамма-распределение. Условия на параметры `alpha>0` и `beta>0`.

random.gauss(значение, стандартное отклонение) — распределение Гаусса.

`random.lognormvariate(mu, sigma)` — логарифм нормального распределения. Если взять натуральный логарифм этого распределения, то вы получите нормальное распределение со

средним `mu` и стандартным отклонением `sigma`. `mu` может иметь любое значение, и `sigma` должна быть больше нуля.

`random.normalvariate(mu, sigma)` — нормальное распределение. `mu` — среднее значение, `sigma` — стандартное отклонение.

`random.vonmisesvariate(mu, kappa)` — `mu` — средний угол, выраженный в радианах от 0 до 2π, и `kappa` — параметр концентрации, который должен быть больше или равен нулю. Если каппа равна нулю, это распределение сводится к случайному углу в диапазоне от 0 до 2π.

`random.paretovariate(alpha)` — распределение Парето.

`random.weibullvariate(alpha, beta)` — распределение Вейбулла.

Примеры

Генерация произвольного пароля

Хороший пароль должен быть произвольным и состоять минимум из 6 символов, в нём должны быть цифры, строчные и прописные буквы. Приготовить такой пароль можно по следующему рецепту:

```
import random
# Набор цифр
str1 = '123456789'
# Набор строчных букв
str2 = 'qwertyuiopasdfghjklzxcvbnm'
# То же, но в верхнем регистре
str3 = str2.upper()
print(str3)
# Выведет: 'QWERTYUIOPASDFGHJKLZXCVBNM'
# Соединяем все строки в одну
str4 = str1+str2+str3
print(str4)
# Выведет:
'123456789qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM'
# Преобразуем получившуюся строку в список
ls = list(str4)
# Тщательно перемешиваем список
random.shuffle(ls)
# Извлекаем из списка 12 произвольных значений
psw = ''.join([random.choice(ls) for x in range(12)])
# Пароль готов
print(psw)
# Выведет: '1t9G4YPsQ5L7'
```

Этот же скрипт можно записать всего в две строки:

```
import random
print(''.join([random.choice(list('123456789qwertyuiopasdfghjklz
xcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) for x in range(12)]))
```

Данная команда является краткой записью цикла for, вместо неё можно было написать так:

```
import random
psw = '' # предварительно создаем переменную psw
for x in range(12):
```

```
    psw = psw + random.choice(list('123456789qwertyuiopasdfgh
    jklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM'))
print(psw)
# Выведет: Ci7nU6343YGZ
```

Данный цикл повторяется 12 раз и на каждом круге добавляет к строке psw произвольно выбранный элемент из списка.

Подробнее здесь: http://ps.readthedocs.io/ru/latest/random.html

а это «на потом»: проверка наличия элемента в словаре

```
my_dict = {'key': 'value'}
key_exists = my_dict.has_key('key') # Устаревший способ.
key_exists = 'key' in my_dict # Актуальный способ.
```

Черепашья графика в языке программирования Python

Для работы с черепашьей графикой необходимо подключить специальный модуль и создать черепаху на черепашьем поле:

```
import turtle
t=turtle.Pen()
```

Таблица функций и методов работы с черепашьей графикой. В таблице используем черепаху t.

Метод или команда	Что делает	Пример
up()	Поднятие "пера", чтобы не оставалось следа его при перемещении	t.up()
down()	Опускание "пера", чтобы при перемещении оставался след (рисовались линии)	t.down()
goto(x,y)	Перемещение "пера" в точку с координатами x,y в системе координат окна рисования	t.goto(50,20)
color ('цвет')	Установка цвета "пера" в значение, определяемое строкой цвета	t.color('blue') t.color('#0000ff')
bgcolor ('цвет')	Установка цвета фона в значение, определяемое строкой цвета. Вызываем не для Pen, а для turtle	turtle.bgcolor('#909090')
width(n)	Установка толщины "пера" в точках экрана	t.width(3)
forward(n)	Передвижение "вперёд" (в направлении острия стрелки) на n точек	t.forward(100)
backward(n)	Передвижение "назад" на n точек	t.backward(100)
right(k)	Поворот направо (по часовой стрелке) на k единиц	t.right(75)
left(k)	Поворот налево (против часовой стрелки) на k единиц	t.left(45)
radians()	Установка единиц измерения углов в радианы	t.radians()
degrees()	Установка единиц измерения углов в градусы (включён по умолчанию)	t.degrees()
circle(r)	Рисование окружности радиусом r точек из текущей позиции "пера". Если r положительно, окружность рисуется против часовой стрелки, если отрицательно — по часовой стрелке.	t.circle(40) t.circle(-50)
circle(r,k)	Рисование дуги радиусом r точек и	t.circle(40,45)

Метод или команда	Что делает	Пример
	углом k единиц.	t.circle(-50,275)
write ('строка')	Вывод текста в текущей позиции пера	t.write('Начало координат!')
clear()	Очистка области рисования	t.clear(0)

Прочитать подробнее можно здесь: https://www.intuit.ru/studies/courses/3489/731/lecture/25771

Файлы в языке программирования Python

Для работы с файлами будем использовать файловые переменные, которые будут представлять файлы в нашей программе.

```
ou = open('c://spam.txt', 'w')  - Открывает файл spam.txt для записи ('w' означает write – запись)

inp = open('data', 'r') -  Открывает файл data из той-же папки, в которой находится наша программа для чтения  ('r' означает read – чтение)
```

Таблица функций и методов работы с файлами. В таблице используем описанные выше переменные ou и inp.

Пример команды	Что делает
a = inp.read()	Чтение файла целиком в строку a
a = inp.read(N)	Чтение следующих N символов (или байтов) в строку a
a = inp.readline()	Чтение следующей текстовой строки (включая символ конца строки) в строку a. Для удаления символа '\n' из конца файла удобно использовать метод строки rstrip(). Например: a = a.rstrip()
Li = inp.readlines()	Чтение файла целиком в список строк Li (включая символ конца строки)
ou.write(S)	Запись строки символов S (или байтов) в файл
ou.writelines(Li)	Запись всех строк из списка Li в файл
ou.close()	Закрытие файла (выполняется по окончании работы с файлом)
ou.flush()	Выталкивает выходные буферы на диск, файл остается открытым
anyFile.seek(N)	Изменяет текущую позицию в файле для следующей операции, смещая ее на N байтов от начала файла.
for line in open('data'): операции над line	Переменная line в цикле по очереди принимает значение каждой строки файла data
f1=open('f.txt', encoding='latin-1')	Открытие файла с кодировкой Юникод
f1=open('f.bin', 'rb')	Открытие двоичного файла

Сохранение и интерпретация объектов Python в файлах

Следующий пример записывает различные объекты в текстовый файл.

```
>>> X, Y, Z = 43, 44, 45          # Объекты языка Python должны
>>> S = 'Spam'                    # записываться в файл только
                                    в виде строк
>>> D = {'a': 1, 'b': 2}
>>> L = [1, 2, 3]
>>>
>>> F = open('datafile.txt', 'w') # Создает файл для записи
>>> F.write(S + '\n')             # Строки завершаются символом
\n
>>> F.write('%s,%s,%s\n' % (X, Y, Z))# Преобразует числа в
строки
>>> F.write(str(L) + '$' + str(D) + '\n') # Преобразует
                                          и разделяет символом $
>>> F.close()
```

Использование инструкции `print`:

```
>>> chars = open('datafile.txt').read() # Отображение строки
>>> chars                          # в неформатированном виде
"Spam\n43,44,45\n[1, 2, 3]${'a': 1, 'b': 2}\n"
>>> print(chars)                    # Удобочитаемое
представление
Spam
43,44,45
[1, 2, 3]${'a': 1, 'b': 2}
```

Теперь нам необходимо выполнить обратные преобразования, чтобы получить из строк в текстовом файле действительные объекты языка Python.

```
>>> F = open('datafile.txt') # Открыть файл снова
>>> line = F.readline()      # Прочитать одну строку
>>> line
'Spam\n'
>>> line.rstrip()            # Удалить символ конца строки
'Spam'
```

Пока что мы прочитали ту часть файла, которая содержит строку. Теперь прочитаем следующий блок, в котором содержатся числа, и выполним разбор этого блока (то есть извлечем объекты):

```
>>> line = F.readline()        # Следующая строка из файла
>>> line                       # Это - строка
'43,44,45\n'
>>> parts = line.split(',')    # Разбить на подстроки по запятым
>>> parts
['43', '44', '45\n']
```

Здесь был использован метод split, чтобы разбить строку на части по запятым, которые играют роль символов-разделителей, – в результате мы получили список строк, каждая из которых содержит отдельное число. Теперь нам необходимо преобразовать эти строки в целые числа, чтобы можно было выполнять математические операции над ними:

```
>>> int(parts[1])       # Преобразовать строку в целое число
44
>>> numbers = [int(P) for P in parts] # Преобразовать весь
список
>>> numbers
[43, 44, 45]
```

Как мы уже знаем, функция int преобразует строку цифр в объект целого числа.

Наконец, чтобы преобразовать список и словарь в третьей строке файла, можно воспользоваться встроенной функцией eval, которая интерпретирует строку как программный код на языке Python:

```
>>> line = F.readline()
>>> line
"[1, 2, 3]${'a': 1, 'b': 2}\n"
>>> parts = line.split('$')  # Разбить на строки по символу $
>>> parts
['[1, 2, 3]', "{'a': 1, 'b': 2}\n"]
>>> eval(parts[0])           # Преобразовать строку в объект
[1, 2, 3]
>>> objects = [eval(P) for P in parts] # То же самое для всех
строк в списке
>>> objects
[[1, 2, 3], {'a': 1, 'b': 2}]
```

Поскольку теперь все данные представляют собой список обычных объектов, а не строк, мы сможем применять к ним операции списков и словарей.