Python Summary Documentation

Выпуск 1

Dee

сент. 27, 2017

Оглавление

Оглавление:

Глава 1

Python 3: Строки. Функции и методы строк

Базовые операции

```
# Конкатенация (сложение)
>>> s1 = 'spam'
>>> s2 = 'eggs'
>>> print(s1 + s2)
'spameggs'

# Дублирование строки
>>> print('spam' * 3)
spamspamspam

# Длина строки
>>> len('spam')
4

# Доступ по индексу
>>> S = 'spam'
>>> S[0]
's'
>>> S[2]
'a'
>>> S[-2]
'a'

# Срез
>>> s = 'spameggs'
>>> s[3:5]
'me'
>>> s[2:-2]
'ameg'
>>> s[:6]
'spameg'
```

```
>>> s[1:]
'pameggs'
>>> s[:]
'spameggs'

# Шаг, извлечения среза
>>> s[::-1]
'sggemaps'
>>> s[3:5:-1]
''
>>> s[2::2]
'aeg'
```

Другие функции и методы строк

```
# Литералы строк
S = 'str'; S = "str"; S = '''str'''; S = """str"""
# Экранированные последовательности
S = "s\np\ta\nbbb"
# Неформатированные строки (подавляют экранирование)
S = r"C:\temp\new"
# Строка байтов
S = b"byte"
# Конкатенация (сложение строк)
S1 + S2
# Повторение строки
S1 * 3
# Обращение по индексу
S[i]
# Извлечение среза
S[i:j:step]
# Длина строки
len(S)
# Поиск подстроки в строке. Возвращает номер первого вхождения или -1
S.find(str, [start],[end])
# Поиск подстроки в строке. Возвращает номер последнего вхождения или -1
S.rfind(str, [start],[end])
# Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
S.index(str, [start],[end])
# Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError
S.rindex(str, [start],[end])
# Замена шаблона
S.replace(шаблон, замена)
# Разбиение строки по разделителю
S.split(символ)
# Состоит ли строка из цифр
S.isdigit()
# Состоит ли строка из букв
S.isalpha()
# Состоит ли строка из цифр или букв
S.isalnum()
# Состоит ли строка из символов в нижнем регистре
S.islower()
# Состоит ли строка из символов в верхнем регистре
S.isupper()
```

```
# Состоит ли строка из неотображаемых символов (пробел, символ перевода страницы ('\f'), "новая␣
↪строка" ('\n'), "перевод каретки" ('\r'), "горизонтальная табуляция" ('\t') и "вертикальная␣
↪табуляция" ('\v'))
S.isspace()
# Начинаются ли слова в строке с заглавной буквы
S.istitle()
# Преобразование строки к верхнему регистру
S.upper()
# Преобразование строки к нижнему регистру
S.lower()
# Начинается ли строка S с шаблона str
S.startswith(str)
# Заканчивается ли строка S шаблоном str
S.endswith(str)
# Сборка строки из списка с разделителем S
S.join(список)
# Символ в его код ASCII
ord(символ)
# Код ASCII в символ
chr(число)
# Переводит первый символ строки в верхний регистр, а все остальные в нижний
S.capitalize()
# Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию)
S.center(width, [fill])
# Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и␣
↪длина строки по умолчанию)
S.count(str, [start],[end])
# 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.expandtabs([tabsize])
# Удаление пробельных символов в начале строки
S.lstrip([chars])
# Удаление пробельных символов в конце строки
S.rstrip([chars])
# Удаление пробельных символов в начале и в конце строки
S.strip([chars])
# 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.partition(шаблон)
# 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.rpartition(sep)
# Переводит символы нижнего регистра в верхний, а верхнего - в нижний
S.swapcase()
# Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний
S.title()
# Делает длину строки не меньшей width, по необходимости заполняя первые символы нулями
S.zfill(width)
# Делает длину строки не меньшей width, по необходимости заполняя последние символы символом␣
↪fillchar
S.ljust(width, fillchar=" ")
# Делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar
S.rjust(width, fillchar=" ")
```

Форматирование строк

```
S.format(*args, **kwargs)
```

Примеры

Python: Определение позиции подстроки (функции str.find и str.rfind)

Определение позиции подстроки в строке с помощью функций `str.find` и `str.rfind`.

```
In [1]: str = 'ftp://dl.dropbox.com/u/7334460/Magick_py/py_magick.pdf'
```

Функция `str.find` показывает первое вхождение подстроки. Все позиции возвращаются относительно начало строки.

```
In [2]: str.find('/')
Out[2]: 4

In [3]: str[4]
Out[3]: '/'
```

Можно определить вхождение в срезе. первое число показывает начало среза, в котором производится поиск. Второе число — конец среза. В случае отсутствия вхождения подстроки выводится -1.

```
In [4]: str.find('/', 8, 18)
Out[4]: -1

In [5]: str[8:18]
Out[5]: '.dropbox.c'

In [6]: str.find('/', 8, 22)
Out[6]: 20

In [7]: str[8:22]
Out[7]: '.dropbox.com/u'

In [8]: str[20]
Out[8]: '/'
```

Функция `str.rfind` осуществляет поиск с конца строки, но возвращает позицию подстроки относительно начала строки.

```
In [9]: str.rfind('/')
Out[9]: 40

In [10]: str[40]
Out[10]: '/'
```

Python: Извлекаем имя файла из URL

Понадобилось мне отрезать от URL всё, что находится после последнего слэша, т.е.названия файла. URL можеть быть какой угодно. Знаю, что задачу запросто можно решить с помощью специально-

го модуля, но я хотел избежать этого. Есть, как минимум, два способа справиться с поставленным вопросом.

Способ №1

Достаточно простой способ. Разбиваем строку по слэшам с помощью функции `split()`, которая возвращает список. А затем из этого списка извлекаем последний элемент. Он и будет названием файла.

```
In [1]: str = 'http://dl.dropbox.com/u/7334460/Magick_py/py_magick.pdf'

In [2]: str.split('/')
Out[2]: ['http:', '', 'dl.dropbox.com', 'u', '7334460', 'Magick_py', 'py_magick.pdf']
```

Повторим шаг с присвоением переменной:

```
In [3]: file_name = str.split('/')[-1]

In [4]: file_name
Out[4]: 'py_magick.pdf'
```

Способ №2

Второй способ интереснее. Сначала с помощью функции `rfind()` находим первое вхождение с конца искомой подстроки. Функция возвращает позицию подстроки относительно начала строки. А далее просто делаем срез.

```
In [5]: str = 'http://dl.dropbox.com/u/7334460/Magick_py/py_magick.pdf'

In [6]: str.rfind('/')
Out[6]: 41
```

Делаем срез:

```
In [7]: file_name = str[42:]

In [8]: file_name
Out[8]: 'py_magick.pdf'
```

Дополнительные материалы

- Учим старую собаку новым трюкам или как я научился любить str.format и отказался от %
- Строки. Функции и методы строк
- Работа со строками в Python: литералы
- Погружение в Python 3 (Пилгрим)/Строки

Python: удаление не пустых папок Модуль os содержит ряд функций для работы с файлами, в том числе функции os.remove(path) os.removedirs(path) os.rmdir(path) Однако они могут удалять только пустые папки.

Для удаления не пустых папок нужно использовать модуль shutil и функцию из него shutil.rmtree(path, ignore_errors=False, onerror=None)

Глава 2

Python 3: Генерация случайных чисел (модуль random)

«Генерация случайных чисел слишком важна, чтобы оставлять её на волю случая»

— Роберт Кавью

Python порождает случайные числа на основе формулы, так что они не на самом деле случайные, а, как говорят, псевдослучайные[1]. Этот способ удобен для большинства приложений (кроме онлайновых казино)[2].

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

```
import random
```

random.random

`random.random()` — возвращает псевдослучайное число от 0.0 до 1.0

```
random.random()
0.07500815468466127
```

random.seed

`random.seed(<Параметр>)` — настраивает генератор случайных чисел на новую последовательность. По умолчанию используется системное время. Если значение параметра будет одиноким, то генерируется одинокое число:

[1] Википедия: Генератор псевдослучайных чисел
[2] Доусон М. Программируем на Python. — СПб.: Питер, 2014. — 416 с.: ил. — 3-е изд

```
random.seed(20)
random.random()
0.9056396761745207

random.random()
0.6862541570267026

random.seed(20)
random.random()
0.9056396761745207

random.random()
0.7665092563626442
```

random.uniform

`random.uniform(<Начало>, <Конец>)` — возвращает псевдослучайное вещественное число в диапазоне от `<Начало>` до `<Конец>`:

```
random.uniform(0, 20)
15.330185127252884

random.uniform(0, 20)
18.092324756265473
```

random.randint

`random.randint(<Начало>, <Конец>)` — возвращает псевдослучайное целое число в диапазоне от `<Начало>` до `<Конец>`:

```
random.randint(1,27)
9
random.randint(1,27)
22
```

random.choince

`random.choince(<Последовательность>)` — возвращает случайный элемент из любой последовательности (строки, списка, кортежа):

```
random.choice('Chewbacca')
'h'
random.choice([1,2,'a','b'])
2
random.choice([1,2,'a','b'])
'a'
```

random.randrange

`random.randrange(<Начало>, <Конец>, <Шаг>)` — возвращает случайно выбранное число из последовательности.

random.shuffle

`random.shuffle(<Список>)` — перемешивает последовательность (изменяется сама последовательность). Поэтому функция не работает для неизменяемых объектов.

```
List = [1,2,3,4,5,6,7,8,9]
List
[1, 2, 3, 4, 5, 6, 7, 8, 9]
random.shuffle(List)
List
[6, 7, 1, 9, 5, 8, 3, 2, 4]
```

Вероятностные распределения

`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
# Щепотка цифр
str1 = '123456789'
# Щепотка строчных букв
str2 = 'qwertyuiopasdfghjklzxcvbnm'
# Щепотка прописных букв. Готовится преобразованием str2
в верхний     регистр.
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('123456789qwertyuiopasdfghjklzxc
vbnmQWERTYUIOPASDFGHJKLZXCVBNM')) 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.

Ссылки

- Официальная документация по модулю random (англ.)
- Python 3 для начинающих: Модуль random
- Модуль random — генерация случайных чисел
- Безопасность случайных чисел в Python

Глава 3

Список используемых материалов

Строки. Функции и методы строк

- Учим старую собаку новым трюкам или как я научился любить str.format и отказался от %
- Строки. Функции и методы строк
- Работа со строками в Python: литералы
- Погружение в Python 3 (Пилгрим)/Строки
- Хабрахабр: Python: вещи, которых вы могли не знать

Работа с файлами

- Python. Замена строки в файле без дополнительного файла
- ‘ <>‘_
- ‘ <>‘_
- ‘ <>‘_

Регулярные выражения

- Хабрахабр: Regexp и Python: извлечение токенов из текста
- Блог Ростислава Дзинько: Регулярные выражения в Python: изучение и оптимизация
- IBM devWorks: Секреты регулярных выражений (regular expressions): Часть 1. Диалекты и возможности. Составление регулярных выражений

Python: вещи, которых вы могли не знать * ‘ <>‘_ * ‘ <>‘_

Компиляция программ на python 3

- Компиляция программы на python 3 в exe с помощью программы cx_Freeze
- Хабрахабр: Облегчаем использование pyinstaller для создания exe
- Хабрахабр: Немного про py2exe
- IT записки: cx-freeze: exe из скрипта Python 3

Глава 4

Indices and tables

- genindex
- modindex
- search