Домашнее задание №4

Дедлайн 1 (12 баллов): 4 октября, 13:00
Дедлайн 2 (6 баллов): 11 октября, 13:00

Домашнее задание нужно написать на Python 3 и сдать в виде одного файла. Правило именования файла: `name_surname_04.py`. Например, если вас зовут Иван Петров, то имя файла должно быть: `ivan_petrov_04.py`.

Обратите внимание, что при выполнении задания можно использовать только то, что мы уже успели обсудить в рамках курса.

1 В следующих заданиях необходимо использовать методы класса `str`, описанные в лекции.

а Напишите функцию `capwords`, которая принимает строку и опциональный разделитель и возвращает новую строку, в которой каждое слово начинается с заглавной буквы. Логика работы функции `capwords` должна быть аналогична методу `str.split`.

```
>>> capwords("foo,,bar,", sep=",")
'Foo,,Bar,'
>>> capwords(" foo  \nbar\n")
'Foo Bar'
```

б Напишите функцию `cut_suffix`, которая принимает строку и суффикс. Функция должна вернуть эту строку, но без заданного суффикса.

```
>>> cut_suffix("foobar", "bar")
'foo'
>>> cut_suffix("foobar", "boo")
'foobar'
```

в Напишите функцию `boxed`, которая принимает строку и два аргумента: символ `fill` и число `pad`. Результатом работы функции `boxed` должна стать исходная строка, обрамлённая символами `fill`, как это показано в примере.

```
>>> print(boxed("Hello world", fill="*", pad=2))
******************
** Hello world **
******************
>>> print(boxed("Fishy", fill="#", pad=1))
#########
# Fishy #
#########
```

г Напишите функцию `find_all`, которая возвращает список индексов всех вхождений подстроки в строке.

```
>>> find_all("abracadabra", "a")
[0, 3, 5, 7, 10]
```

д Напишите функцию `common_prefix`, которая возвращает наибольший общий префикс двух или более строк.

```
>>> common_prefix("abra", "abracadabra", "abrasive")
"abra"
>>> common_prefix("abra", "foobar")
""
```

2 В следующих заданиях стоит использовать методы файловых объектов, описанные в лекции.

а Напишите функцию `reader`, которая принимает путь к файлу и возвращает файловый объект. Функция должна работать с архивированными файлами, сжатыми с помощью `gzip` и `bzip2`. Несколько примеров:

```
>>> reader("./example.txt")
<_io.TextIOWrapper name='./example.txt' mode='rt' encoding='UTF-8'>
>>> reader("./example.txt.gz", mode="rt", encoding="ascii")
<_io.TextIOWrapper name='./example.txt.gz' encoding='ascii'>
>>> reader("./example.txt.bz2", mode="wb")
<bz2.BZ2File object at 0x1066e6978>
```

Можно считать, что файл, оканчивающийся на `.gz`, всегда является `gzip` архивом, аналогично для `bz2` и `bzip2`. Для реализации вам потребуются функции `open` из соответствующих модулей стандартной библиотеки:

```
>>> import gzip
>>> gzip.open
<function open at 0x1066dc730>
>>> import bz2
>>> bz2.open
<function open at 0x1066fde18>
```

Найти к ним документацию можно по ссылкам:

- `http://docs.python.org/3/library/bz2.html`
- `http://docs.python.org/3/library/gzip.html`

б She-bang[1] — последовательность `#!`, которая используется Unix-подобными системами для запуска исполняемых скриптов. She-bang **всегда** пишется на первой строчке в скрипте. После she-bang следует путь к программе-интерпретатору, например:

```
#! /bin/sh                    #!/usr/bin/env python -v
```

Напишите функцию `parse_shebang`, которая принимает путь к исполняемому скрипту и возвращает путь к программе-интерпретатору, если скрипт содержит she-bang, и `None` в обратном случае.

Для скриптов из примера выше:

```
>>> parse_shebang("./example1.txt")
"/bin/sh"
>>> parse_shebang("./example2.txt")
"/usr/bin/env python -v"
```

[1] `http://en.wikipedia.org/wiki/Shebang_(Unix)`

3 Вероятностная модель языка описывает фрагменты текста на некотором языке в терминах случайных процессов. Одна из самых простых моделей языка формулируется следующим образом. Предположим, что нам известно множество всех слов в языке. Будем порождать слова в предложении слева направо одно за другим:

- Случайно выберем первые два слова из множества слов.
- Каждое t-е слово будем порождать при условии известных нам $(t-1)$-го и $(t-2)$-го слов.

Попробуем построить модель языка по лирике известного хип-хоп исполнителя Снуп Догга.

а Напишите функцию `words`, которая принимает текстовый файл и возвращает список слов из файла:

```
>>> import io
>>> handle = io.StringIO("""Ignorance is the curse of God;
... knowledge is the wing wherewith we fly to heaven.""")
>>> words(handle)
['Ignorance', 'is', 'the', 'curse', 'of', 'God;\n', 'knowledge', 'is', 'the',
 'wing', 'wherewith', 'we', 'fly', 'to', 'heaven.']
```

Обратите внимание на то, что знаки препинания и символы переноса строки на конце слов остаются без изменений.

б Напишите функцию `transition_matrix`, которая принимает список слов и возвращает словарь. В этом словаре для каждой пары слов `(u, v)` содержится список из слов `w`, которые встречаются во входном списке после последовательности слов `u v`. Для примера выше:

```
>>> language = words(handle)
>>> m = transition_matrix(language)
>>> m["is", "the"]
['curse', 'wing']
```

в Напишите функцию `markov_chain`, генерирующую предложения определённой длины. Функция принимает три аргумента:

- список слов, результат работы функции `words`,
- словарь, построенный с помощью функции `transition_matrix`,
- целое число — количество слов в предложении, которое нужно сгенерировать.

Напомню, как генерировать случайные предложения. Будем порождать слова в предложении слева направо одно за другим:

- Случайно выберем первые два слова из всех слов `words`.
- Каждое t-е слово будем порождать при условии известных нам $(t-1)$-го и $(t-2)$-го слов (воспользуемся `transition_matrix`).
- Если такая пара раньше не встречалась (в словаре `transition_matrix` её нет), то t-е слово берётся случайно из всех слов.

Вам потребуются функции `random.randint` и `random.choice`, их названия не требуют комментариев:

```
>>> import random
>>> random.randint(0, 42)
26
>>> random.choice(["foo", "bar", "baz"])
'bar'
```

r Напишите функцию `snoop_says`, которая принимает путь к файлу `snoop279.txt`[2] и целое число — длину предложения и возвращает случайное предложение указанной длины на языке Снуп Догга.

```
>>> print(snoop_says("./snoop279.txt", 23))
 And check the file
 Under G you will see, that ever since Snoopy was a juvenile
 They cut me loose in nine-deuce swore
```

Файл `snoop279.txt` доступен по ссылке

`https://gist.github.com/superbobry/26c108e17e9a02cae197`

[2]Файл состоит из сконкатенированных текстов Снуп Догга, скачанных с сайта AZLyrics (`http://azlyrics.com/s/snoopdogg.html`).