Санкт-Петербургский государственный инженерно-экономический университет.

Кафедра вычислительных систем и программирования

Лабораторная работа «Антивирусные технологии»

Составитель: Федоров Д.Ю.

http://pycode.ru/edu/

Оглавление

1. Предварительные сведения

Вы сотрудник антивирусной лаборатории. В распоряжение лаборатории поступило несколько экземпляров новых вирусов. Руководитель отдела поручил вам разработать программу, детектирующую вирусы и их возможные модификации. Задержка в разработке программы может повлечь за собой эпидемию заражения компьютеров по всему миру.

Исследования вирусов показали, что все они содержат текстовые строки (представлены разные вариации строк в зависимости от модификации вируса):

«~Fun Loving Criminal-1~»

«~Fun Loving Criminal-2~»

«~Fun Loving Criminal-3~»

«~Fun Loving Criminal-4~»

«~Fun Loving Criminal-5~»

«~Fun Loving Criminal-6~»

Условия работы программы, детектирующей вирусы в системе:

- если файл заражен, то выводится текст «Attention! Virus!»
- если файл не заражен, то ничего не выводится

Подумайте, подвержена ли ваша программа ложным срабатываниям?

2. Теоретические сведения

Для реализации поставленной задачи необходимо познакомиться с принципами работы с файлами и концепцией регулярных выражений в языке Python.

2.1. Файлы

Файлами называются области постоянной памяти на компьютере, которыми управляет операционная система.

Встроенная функция *open* создает объект файла, который обеспечивает связь с файлом, размещенным в компьютере. После вызова функции *open* можно выполнять операции чтения и записи во внешний файл, используя методы полученного объекта.

Рассмотрим небольшие примеры, демонстрирующие основы работы с файлами.

```
>>> myfile = open('myfile.txt', 'w')
```

В первом примере создается объект файла, ссылка на который помещается в переменную *myfile*. Выполняется открытие нового файла в режиме для записи.

```
>>> myfile.write('hello text file\n')
```

Воспользовавшись методом *write* объекта файла, в открытый файл записывается строка (завершающаяся символом новой строки \n),

```
>>> myfile.close()
```

после чего файл закрывается. Вызов метода *close* разрывает связь с внешним файлом.

Далее этот же файл открывается в режиме для чтения строки из него (по умолчанию файл открывается для чтения, если не указан иной спецификатор открытия).

```
>>> myfile = open('myfile.txt')
```

```
>>> myfile.readline()
```

```
'hello text file\n'
>>> myfile.readline()
''
```

Обратите внимание, что второй вызов метода *readline* возвращает пустую строку – таким способом методы файла в языке Python сообщают о том, что был достигнут конец файла.

Этот пример записывает единственную строку текста в файл, добавляя в нее символ конца строки, поэтому нам необходимо добавлять его в выводимые строки (в противном случае следующая операция записи дополнит текущую строку в файле).

Выполните следующие выражения в командной строке интерпретатора, попытайтесь объяснить полученный результат:

```
>>> myfile = open('myfile.txt', 'w')
>>> myfile.write('hello text file\n')
>>> myfile.write('hi text file\n')
>>> myfile.write('text file\n')
>>> myfile.close()
>>> myfile = open('myfile.txt')
>>> aString = myfile.read()
>>> aString
 'hello text file\nhi text file\n text file\n'
>>> myfile.close()
>>>
```

Функция *read* читает файл в виде строки.

2.2. Импортирование модулей

Каждый файл с исходным текстом на языке Python, имя которого оканчивается расширением *.py*, является модулем. Другие файлы могут обращаться к функциональным возможностям, объявляемым модулем, *импортируя* этот модуль.

```
>>> import script
```

Инструкция *import* выполняет загрузку другого файла и обеспечивает доступ к его содержимому. Эта модульная модель является центральной идеей, лежащей в основе архитектуры программ на языке Python. Крупные программы организованы в виде множества файлов модулей, которые импортируют и используют функциональные возможности из других модулей.

Интерпретатор Python поставляется с обширной коллекцией дополнительных модулей, которая известна как *стандартная библиотека*. Эта коллекция насчитывает порядка 200 крупных модулей и содержит платформонезависимую поддержку распространенных задач программирования: интерфейсы операционных систем, поиск по шаблону, сетевые взаимодействия и многих других. Ни один из этих инструментов не является непосредственной частью языка Python, но вы можете использовать их, импортируя соответствующие модули (так же называемые *библиотеками* инструментов).

Рассмотрим примеры импортирования модулей. С помощью интерактивного интерпретатора Python создайте новый файл модуля в корневом каталоге Python с именем *myfile.py* со следующим содержимым:

title = “Life”

Это один из самых простых модулей Python. При импортировании этого модуля выполняется его программный код, который создает *атрибут модуля* с именем *title*.

Доступ к атрибуту *title* можно получить из других программных компонентов.

Выполните следующие команды в интерактивном режиме интерпретатора Python.

```
>>> import myfile
>>> print(myfile.title)
Life
>>>
```

Обратите внимание, при импортировании модуля расширение «.py» указывать не надо. Интерпретатор самостоятельно анализирует каталоги программы в поисках указанного файла. Здесь мы использовали нотацию *object.attribute* для обращения к строковой переменной *title*, определенной внутри модуля *myfile*, т.е. *myfile.title*.

2.3. Регулярные выражения

Для того чтобы воспользоваться функциональными возможностями одного из стандартных модулей, импортируем его и обратимся к одному из его атрибутов (функции).

```
>>> import re
```

Этой командой мы импортировали модуль для работы с регулярными выражениями в Python.

Регулярные выражения – это один из способов поиска подстрок (соответствий) в строках. Осуществляется это с помощью просмотра строки в поисках некоторого шаблона. Общеизвестным примером могут быть символы «*» и «?», используемые в командной строке MS-DOS. Первый из них заменяет ноль или более произвольных символов, второй же – один произвольный символ. Так, использование шаблона поиска типа "*text?.**" найдет файлы *textf.txt, text1.asp* и другие аналогичные, но не найдет *text.txt* или *text.htm*.

Примеры шаблонов в MS-DOS:

. - все файлы с любым расширением;

*.*txt* - все файлы с расширением .txt;

*command.** - все файлы с именем “command” и любым расширением;

*.*do?* - все файлы, расширение которых начинается с “do”;

.???* - все файлы, имеющие трехбуквенное расширение;

Простейшее регулярное выражение – это обычные литералы символов, такие как *a* или *5*. В отсутствие явного *квантификатора* (квантификаторы определяют число совпадений с выражением) такое выражение подразумевает «совпадение с одним вхождением». Например, регулярное выражение *tune* будет совпадать с tune и attuned.

Во многих случаях вместо совпадения с единственным символом бывает необходимо отыскать совпадение с одним из множества символов. Реализовать это можно с помощью *символьного класса* (это термин регулярных выражений и он не имеют никакого отношения к классам в языке Python) – один или более символов, заключенные в квадратные скобки. Например, *[ea]* означает любой символ из набор в скобках, т.е. регулярное выражение *r[ea]d* совпадает с red и radar, но не со словом read. Точно так же, чтобы отыскать совпадение с единственной цифрой, можно использовать регулярное выражение *[0123456789]*. Для удобства можно указывать диапазон символов с помощью символа дефиса: *[0-9]*.

Выполните в интерактивном режиме интерпретатора Python:

```
>>> import re
>>> re.search(“r[ea]d”, “rad”)
<_sre.SRE_Match object at 0x00F1E020>
>>> re.search("r[ea]d", "read")
>>>
```

В первой строке мы импортировали модуль *re* для работы с регулярными выражениями. Затем, воспользовавшись нотацией *object.attribute* (см. раздел «импортирование модулей»), использовали функцию *search*, которая

возвращает объект совпадения *SRE_Match*, если обнаружено совпадение с регулярным выражением *“r[ea]d”* (первый аргумент функции) в любом месте строки *“rad”* (второй аргумент функции), в противном случае возвращает *None*.

Теперь, используя полученные знания, перейдем к пошаговой постановке задачи.

1) Создайте файл (*virus.txt*) в корневой директории с одной из перечисленных в предварительных сведениях строк. К примеру, содержимое файла может иметь следующий вид:

   AAAAAAAAAAAAAAAAA ~Fun Loving Criminal-4~ AAAAAAAAAAAAAAAAAAAAAAAAA

   Созданный файл будет служить тестовым экземпляром.

2) Первым делом импортируйте модуль *re*. Затем созданный в 1) файл необходимо открыть для чтения и прочитать в виде строки.

3) Составьте регулярное выражение, соответствующее семейству строк, указанных в предварительных сведениях, и сопоставьте его со строкой, полученной на шаге 2).

4) Если строка соответствует регулярному выражению, то выведете “Attantion! Virus”, иначе - ничего.

3. Литература для дополнительного чтения

1. Документация Python версии 2.7: http://docs.python.org/library/re.html
2. Марк Саммерфилд. Программирование на Python 3. Подробное руководство
3. Регулярные выражения:

   http://www.intuit.ru/department/pl/python/6/4.html
4. Джеффри Фридл. Регулярные выражения, 3-е издание.