ЛАБОРАТОРНАЯ РАБОТА №1

Курс «Кросс-платформенное программирование»

Тема: Основы языка Python. Системное программирование на Python.
Цель: Научиться писать простейшие Python-скрипты в процедурном стиле.

Темы для предварительной проработки [УСТНО]:

- Синтаксис Python. Типизация и управляющие конструкции Python.
- Работа со списками и строками. Регулярные выражения.
- Работа с файловой системой.

Разминка [КОД] :

1. Написать скрипт, позволяющий ввести с клавиатуры число N от 1 до 100 и вывести на экран грамматически верную фразу вида «N [лет | год | года]». Например: «21 год», «32 года», «57 лет» и т.д. В случае ввода отрицательного числа выдать сообщение об ошибке.
2. Ввести с экрана день, месяц и год (3 целых неотрицательных числа). Вывести на экран дату в формате dd/mm/yyyy (если одно из чисел однозначное, то слева дополнить одним нулем). В случае ввода отрицательного числа выдать сообщение об ошибке.
3. Написать скрипт, который преобразует введенное с клавиатуры дробное число в денежный формат. Например, число 12,5 должно быть преобразовано к виду 12 руб. 50 коп. В случае ввода отрицательного числа выдать сообщение *с помощью обработки исключения* в коде.
4. Написать скрипт для приближенного вычисления числа π на основе следующей формулы (количество слагаемых ввести с клавиатуры):
$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \dots$$
5. Вывести на экран все двузначные числа, сумма цифр которых делится на 7.
6. Написать скрипт, который предлагает ввести строку с консоли и определяет, начинается ли эта строка с «www». Если начинается, то вставить перед ней строку «http://», а в конце проверить, что строка заканчивается на «.com». Если не заканчивается, то вставить в конец эту подстроку.
7. Ввести с клавиатуры номер дебетовой карты (16 цифр). Вывести номер в скрытом виде – первые и последние 4 цифры отображены нормально, а между ними символы «*».
8. Написать скрипт, который разбивает введеное с клавиатуры предложение-строку на слова. Выбрать структуру данных для хранения слов. Вывести сначала слова, длина которых больше 7 символов, затем слова размером от 4 до 7 символов, затем – все остальные.
9. Сгенерировать случайным образом число N от 1 до 1000. Создать массив из N целых чисел, также сгенерированных случайным образом. Дополнить массив нулями до длины, равной ближайшей сверху степени двойки. Например, если в массиве было N = 100 элементов, то дополнить массив 28 нулями, чтобы в итоге был массив из 2^8=128 элементов (ближайшая степень двойки к 100 – это число 128, для 35 – это 64 и т.д.).
10. Написать скрипт, который выводит на экран TRUE, если элементы массива представляют собой возрастающую последовательность, иначе – FALSE.

Индивидуальные задания [КОД] :

11. Написать программу, которая позволяет ввести с клавиатуры натуральное число N и вывести на экран все комбинации натуральных чисел x, y, z, таких что $x^3+y^3+z^3=N$. Если число N невозможно разложить по кубам x, y, z, программа должна выводить сообщение «No such combinations!».
12. Модифицировать код задания 11: вывести все N от 1 до 100000 (и соответствующие им комбинации), которые раскладываются минимум по 3 разным суммам кубов.
13. Ввести с клавиатуры текст предложения, завершить точкой. Сформировать новую строку на основе исходной, в которой после каждого слова в скобках указать номер слова в предложении (слова разделяются запятыми, пробелами или тире). Например, если введено «*Донецк – прекрасный город*», результирующая строка должна выглядеть так: «*Донецк(1) – прекрасный(2) город(3)*».
14. Ввести с клавиатуры текст предложения, завершить точкой. Вывести на консоль все символы, которые входят в этот текст ровно по одному разу.
15. Ввести с клавиатуры текст. Программно найти в нем и вывести отдельно все слова, которые начинаются с большого латинского символа (от A до Z) и заканчиваются 2 цифрами, например «Petr93», «Johnny70», «Service02». Использовать *регулярные выражения*.
16. Написать скрипт, который читает текстовый файл и выводит символы в порядке убывания частоты встречаемости в тексте. Регистр символа не имеет значения. Программа должна учитывать только буквенные символы (символы пунктуации, цифры и служебные символы не подсчитывать). Проверить работу скрипта на нескольких файлах с текстом на английском и русском языках, сравнить результаты с таблицами, приведенными на wikipedia.org/wiki/Letter_frequencies. / Задание взято из книги [1] /.
17. Написать скрипт, позволяющий искать в заданной директории и в ее подпапках файлы-дупликаты на основе сравнения контрольных сумм (MD5). Файлы могут иметь одинаковое содержимое, но отличаться именами. Скрипт должен вывести группы имен обнаруженных файлов-дупликатов. Ниже приведен фрагмент кода для вычисления контрольной суммы:

```
import hashlib
...
f = open('file.txt', 'r' )
data = f.read()
f.close()
checksum = hashlib.md5(data).hexdigest()
```

    / Задание взято из книги [1] /.
18. Написать скрипт, который позволяет ввести с клавиатуры имя текстового файла, найти в нем с помощью регулярных выражений все подстроки определенного вида, в соответствии с вариантом (приложение А). Например, для варианта №1 скрипт должен вывести на экран примерно следующее:

```
Строка 3, позиция 10 : найдено «11-05-2014»
Строка 12, позиция 2 : найдено «23-11-2014»
Строка 12, позиция 17 : найдено «23-11-2014»
```

19. Написать скрипт reorganize.py, который в директории --source создает две директории: Archive и Small. В первую директорию помещаются файлы с датой изменения, отличающейся от текущей даты на количество дней более --days. Во вторую – все файлы размером меньше параметра --size. Каждая директория должна создаваться только в случае, если найден хотя бы один файл, который должен быть в нее помещен.
    Пример вызова:

    ```
    reorganize --source "C:\TestDir" --days 2 --size 4096
    ```

20. Написать скрипт trackmix.py, который формирует обзорный трек-микс альбома (попурри из коротких фрагментов mp3-файлов в пользовательской директории). Для манипуляций со звуковыми файлами можно использовать сторонние утилиты, например, FFmpeg.
    Пример вызова и работы скрипта:

    ```
    trackmix --source "C:\Muzak\Da Album" --count 5 --frame 15 -l -e

    --- processing file 1: 01 - Intro.mp3
    --- processing file 2: 02 - Outro.mp3
    --- done!
    ```

    Параметры:
    --source (-s) – имя рабочей директории с треками, обязателен
    --destination (-d) – имя выходного файла, необязателен (если не указан, то имя выходного файла – mix.mp3 в директории source)
    --count (-c) – количество файлов в "нарезке", необязателен (если он не указан, то перебираются все mp3-файлы в директории source)
    --frame (-f) – количество секунд на каждый файл, необязателен (если не указан, скрипт вырезает по 10 секунд из произвольного участка каждого файла)
    --log (-l) – необязательный ключ (если этот ключ указывается, то скрипт должен выводить на консоль лог процесса обработки файлов, как в примере)
    --extended (-e) – необязательный ключ (если этот ключ указывается, то каждый фрагмент микса начинается и завершается небольшим fade in / fade out)

Контрольные вопросы [ОТЧЕТ]:

1. Чем отличаются компилируемые и интерпретируемые языки программирования?
2. Типизация в языке Python. Преобразования типов. Переменные, объекты и ссылки.
3. Строки, списки, словари, кортежи, множества.
4. Пользовательские функции. Область видимости.
5. Анонимные функции. Функции map, filter, reduce.
6. Обработка исключений в языке Python.
7. Основные возможности Python по работе с файлами и файловой системой.
8. Регулярные выражения. Базовые синтаксические элементы.

Рекомендуемые источники:

[1] C. Severance "Python for informatics. Exploring information" [Электронный ресурс]. – URL: http://www.pythonlearn.com/book.php.

[2] Лутц М. Изучаем Python. – М.: Символ-плюс, 2008. – 848 с.

[3] Лутц М. Программирование на Python. – М.: Символ-плюс, 2002. – 1136 с.

[4] Саммерфильд М. Программирование на Python 3. Подробное руководство. – М.: Символ-плюс, 2009. – 608 с.

Приложение А. Варианты индивидуальных заданий.

Вариант 1.
Найти в тексте все даты – подстроки вида «11-05-2014».

Вариант 2.
Найти в тексте все значения времени – подстроки вида «23:15:59».

Вариант 3.
Найти в тексте все IPv4-адреса – подстроки вида «192.168.5.48».

Вариант 4.
Найти в тексте все строки вида «*type x = value*», где *type* – это тип (может принимать значение int, short или byte), *x* – любое слово, *value* – любое число.

Вариант 5.
Найти в тексте все номера телефонов – подстроки вида «(000)1112233» или «(000)111-22-33».

Вариант 6.
Найти в тексте все строки вида «*x: type [N]*», где *type* – это тип (может принимать значение int, short или byte), *x* – любое слово, *N* – любое положительное целое число.

Вариант 7.
Найти в тексте все «смайлы» – подстроки вида «:)», «:-)», «)))» (количество скобок может быть любым, начиная с 1).

Вариант 8.
Найти в тексте все логические выражения – подстроки вида «x && y», «x & y», где *x* и *y* – любые слова. Количество пробелов может быть также любым.

Вариант 9.
Найти в тексте все донецкие почтовые индексы – подстроки вида «83000, Донецк» (первые 2 символа строго закреплены).

Вариант 10.
Найти в тексте все полные имена директорий Windows – подстроки вида «C:\Dir\SubDir3».