Разработка научных приложений

Введение в Python. Домашнее задание №2

Для выполнения заданий создайте новый репозиторий Git, в котором будет вестись разработка всех скриптов. Файлы, относящиеся к отдельным заданиям, размещаются в подпапках с разумным образом выбранными названиями. Не забывайте коммитить все важные этапы написания скрипта. После завершения задания делайте итоговый коммит с соответствующим комментарием и присваивайте тег вида «задание 1» и т.д.

После завершения работы запакуйте папку с репозиторием в архив **.zip** (обязательно zip !) с названием вида Фамилия_Имя_ДЗ2.zip (кириллицей). Отправьте его на адрес **ogulenko.a.p@onu.edu.ua** до 20.03 включительно.

В качестве литературы по Python рекомендуется официальная документация на сайте python.org и книга М.Саммерфельда (прилагается). Очень большой объем литературы и информации доступен на английском языке.

Задание 1. *Поработайте с документацией*

Напишите скрипт, который печатает значение случайной величины, равномерно распределенной в промежутке от -1 до 1. Значение должно быть напечатано с четырьмя значащими цифрами после запятой (форматная строка `%.4f`).

Стандартный модуль (библиотека) языка Python для генерации равномерно распределенных случайных величин называется **random**. Чтобы разобраться, как им пользоваться, изучите документацию в Python Library Reference на сайте python.org или встроенную справку (для этого в интерпретаторе Python подключите модуль командой `import random`, а затем выполните команду `help(random)`; выйти из справочной системы можно, нажав **q**).

Не стоит называть ваш скрипт **random.py**. Так как имя совпадает с именем стандартного модуля и находится «ближе», при выполнении команды `import random` будет импортирован ваш собственный скрипт.

Задание 2. *Добавьте цикл*

Измените скрипт из задания 1 так, чтобы он печатал n случайных чисел и затем их среднее значение. При этом число n задается в командной строке как аргумент скрипта, т.е. вызов его примерно такой (знак доллара означает приглашение командной строки операционной системы) :

```
$ python task2.py 10
```

Задание 3. *Найдите ошибки*

Пусть некоторый скрипт содержит следующий код:

```
import sys, random

def compute(n):
    i = 0; s = 0
    while i <= n:
        s += random.random()
```

```
            i += 1
    return s/n

n = sys.argv[1]
print 'average of %d random numbers is %g" % (n, compute(n))
```

В нем есть несколько ошибок. Найдите их и исправьте по коммиту за раз. Суть ошибки можно указать в комментарии к коммиту.

Задание 4. *Используйте базовые конструкции*

Чтобы потренироваться в использовании базовых управляющих конструкций и функций, напишите скрипт, который читает произвольное количество чисел–аргументов командной строки и печатает натуральный алгоритм каждого из них на экране.

Например, если передать скрипту в качестве аргумента набор

```
1.0 -0.9 2.1
```

скрипт должен напечатать что-то вроде

```
ln(1) = 0
ln(-0.9) is illegal
ln(2.1) = 0.741937
```

Реализуйте четыре вида циклов перебора аргументов командной строк:

- цикл `for r in sys.argv[1:]` (т.е. цикл по элементам массива `sys.argv`, начиная с индекса 1 и до последнего возможного значения индекса;
- цикл с целым счетчиком `i`, перебирающим индексы массива `sys.argv` (используйте функцию **range**, чтобы сгенерировать соответствующий набор индексов);
- цикл `while` с целым счетчиком, перебирающим валидные индексы в массиве `sys.argv`;
- «бесконечный» цикл `while 1:` с целым счетчиком и **try–except** блоком, в котором осуществляется выход из цикла, если `sys.argv[i]` является незаконной операцией.

Почитайте документацию по стандартному модулю **math**, чтобы узнать, как можно вычислить натуральный логарифм числа. Поскольку все четыре цикла, в общем, похожи, имеет смысл выделить в отдельную функцию общие для всех них действия: преобразование аргумента командной строки в переменную типа **float**, проверку положительности (все таки вычисляем логарифм!) и красивую печать результата.

Задание 5. *Читайте поток данных*

Имеется следующий скрипт, который мы будем называть *datatrans.py*:

```
import sys, math

try:
    infilename = sys.argv[1]
    outfilename = sys.argv[2]
```

```
except:
    print "Usage:", sys.argv[0], "infile outfile"
    sys.exit(1)

ifile = open(infilename, 'r')
ofile = open(outfilename, 'w')

def myfunc(y):
    if y >= 0.0:
        return y**5*math.exp(-y)
    else:
        return 0.0

for line in ifile:
    pair = line.split()
    x = float(pair[0])
    y = float(pair[1])
    fy = myfunc(y)
    ofile.write('%g %12.5e\n' % (x,fy))

ifile.close()
ofile.close()
```

Разберитесь в том, что делает каждая строка этого скрипта. Модифицируйте скрипт так, чтобы он читал поток пар чисел `(x, y)` из командной строки и записывал пару чисел `(x, myfunc(y))` в файл. Использовать такой скрипт можно будет так:

```
python task5.py log.txt 1.1 3 2.6 8.3 7 -0.1675
```

Задание 6. *Сумма строк чисел из файла*

Модифицируйте скрипт *datatrans.py* так, чтобы он мог читать файл с произвольным числом столбцов вещественных чисел. Нужно найти среднее значение по каждой строке и записать в новый скрипт исходные столбцы чисел, добавив к ним в конец столбец со средними значениями. Все числа в выходном файле должны иметь формат `12.6f`.

Задание 7. *Приближенное вычисления вероятности*

Напишите скрипт, который приближенно вычисляет вероятность выпадения минимум одной шестерки при бросании двух игральных костей.

Если забыть все, что вы знаете о теории вероятностей, самый простой способ сделать это — «разыграть» очень большое количество раз бросок двух костей и подсчитать количество нужных исходов. Вы должны помнить, что такой подход к вычислению величин называется методом Монте–Карло.

Таким образом, нужно организовать цикл от `1` до `n`, в котором генерируются две целые случайные величины, равномерно распределенные между 1 и 6. При этом надо подсчитать, сколько раз встретиться хотя бы одна 6. Разделив это число на общее число испытаний, получим приближеннное значение требуемой вероятности.

Определите, какого числа испытаний будет достаточно, чтобы получить верными первые три цифры после запятой. Для этого, конечно, надо будет найти аналитическое решение, вспомнив теорию вероятностей.

Задание 8. *Игра в кости (hazard game)*

Некто предлагает сыграть в следующую игру. Вы платите 1 единицу денег и получаете возможность бросить четыре кости. Если сумма выпавших очков меньше 9, вы выигрываете 10 единиц денег. В противном случае вы теряете свой взнос. Используя решение предыдущего задания, определите, следует ли вам соглашаться играть в такую игру (т.е. велика ли вероятность выигрыша)?

Примечание: это упрощенный вариант одной игры в кости, которая была известна и очень популярна в Англии начиная с 13 века. В 14 веке она упоминается в «Кентерберийских рассказах» Г. Чосера