Инсталляция питона

- Установить python
- Установаить pip/easy_install
- Установить ipython (со всеми зависимостями)
- Проверить работу ipython qtconsole, ipython notebook
- sublime-text-2 / notepad++ / vim / emacs /eclipse + pydev
- pylint
- winpdb
- В ipython справка по объекту - obj?<enter>
-
- Более подробное описание

Строки

- Прочитать описание строковых операций
- Прочитать описание format
- Pragmatic Unicode, or, How do I stop the pain?
- Google Python Class Day 2 Part 1: regular exression

Разложить число на простые делители

Написать функцию factorize, которая возвращает все простые делители у переданного числа.

строковые функции

Написать строковые функции xfind, xreplace, xsplit, xjoin используя срезы строк (без применения других методов строк).

- `xfind(s1, s2) == s1.find(s2)`
- `xreplace(s1, s2, s3) == s1.replace(s2, s3)`
- `xsplit(s1, s2) == s1.split(s2)`
- `xjoin(s, array) == s.join(array)`

Функтор

Функтор

1) Написать объект, который запоминает арифметические операции, проделанные над ним и может их повторить. 2) Написать объект, который трансформирует операции сравнения над собой в WHERE для SQL.

```
x = Var()
f = 3 * x + x ** 2
f(1) == 4

y = SQL("table.field")
str(y > 12 && y != 17)  == "table.field > 12 AND table.field
```

Декодирование АОН

- Нужно преобразовать строку по следующим правилам:
- Если символ идет 2 и больше раз подрят - записать его в результат 1 раз
- Если символ повторяется 1 раз - отбросить
- Если # повторяется два и более раз - последний символ, записанный в результат записать еще раз

```
1       decode("") == ""
2       decode("1") == ""
3       decode("11") == "1"
4       decode("11111") == "1"
5       decode("11#") == "1"
6       decode("11##") == "11"
7       decode("11122234###55") == "1225"
```

Разбор логов

Разобрать файл логов на записи и записи на отдельные компоненты. На выходе должен быть массив словарей {field_name → field_value}

```
Nov  6 03:23:32 some-laptop anacron[9343]: Job 'cron.daily' terminated
Nov  6 03:23:32 some-laptop anacron[9343]: Normal exit (1 job run)
Nov  6 04:17:01 some-laptop CRON[13180]: (root) CMD (
    cd / && run-parts --report /etc/cron.hourly)
Nov  6 05:17:01 some-laptop CRON[13573]: (root) CMD (
    cd / && run-parts --report /etc/cron.hourly)
Nov  6 06:00:37 some-laptop NetworkManager[1049]: <info>
    sleep requested (sleeping: no  enabled: yes)
Nov  6 06:00:37 some-laptop NetworkManager[1049]:
    <info> sleeping or disabling...
```

Гномья сортировка

В массиве сравниваются соседние элементы. Если они неупорядоченны - они меняются местами и делается шаг назад. Если они упорядоченны, то шаг вперед. Если дошли до конца, то сортировка оконченна.

Двоичный поиск

Найти елемент в упорядоченном массиве методом дихотомии.

Кодирование Шеннона — Фано

- Сообщение бъется на элементы
- Изначально коды для всех элементов пустые
- Элементы множества выписывают в порядке убывания вероятностей.
- Множество делится на две части, суммарные вероятности символов которых максимально близки друг другу.
- К коду первой половины элементов дописывается "0", второй "1"
- Алгоритм повторяется для обоих частей

Кодирование и декодирование файла по Хаффману.

На диске есть файл с именем "input.txt". Его нужно прочитать, закодировать символы используя не адаптивный алгоритм Хаффмана и записать результат в output.bin. В решении должно быть две функции hf_encode(string) str->str, и hf_decode(string) str->str. Первая кодирует, вторая декодирует. Входными элементами для алгоритма являются отдельные байты файла.

Интерпретатор minilisp

Программа на mini-lisp имеет вид (oper param1 param2 para3 .... paramn), здесь oper это имя функции - любой набор символов, кроме пробелов. param2 - целое, строка в кавычках (без кавычек внутри) или другая программа на mini-lisp. Допустимые oper - '+' (складывает все операнды), '-' (вычитает из первого все операнды), print (печатает все операнды через пробел). Сложение имеет такой же смысл, как и в питоне. Вычитание для строк не определено. Нужно написать функция eval_minilisp, которая исполнить программу переданную параметром. По умолчанию из main вызывать eval_minilisp("example.lst")

- eval_minilisp ('(+ 1 2 3)') должна вернуть 6
- eval_minilisp ('(print (+ "a" "bc")) ') => должны напечатать 'abc'

Интерпретатор minilisp с промежуточным деревом

- Сделать задание "Интерпретатор minilisp" но промежуточно необходимо преобразовать дерево в форму, удобную для промежуточной обработки.
- Сделать систему разбора расширяемой снаружи новыми командами.

интерпретатор подмножества языка forth

Программа на Forth состоит из набора команд(слов), некоторые из которых имеют параметры. Для хранения данных используется стек - команды получают свои операнды с вершины стека и туда же сохраняют результаты. В подмножестве 5 команд:

- put значение - помещает значение в стек. Значение может быть числом или строкой. Строка заключается в кавычки, внутри строки кавычек быть не может.
- pop - убирает значение из стека
- add - убирает из стека 2 значения, складывает их и помещает результат в стек
- sub - убирает из стека 2 значения, вычитает их и помещает результат в стек
- print - вынимает из стека 1 значение и печатает его.

```
1       put 3
2       put "asdaadasdas"
```

Каждая команда начинается с новой строки. Строки, начинающиеся с '#' - комментарии. Ваша программа должна содержать функцию eval_forth(), принимающую строку на языке forth и исполняющую ее. По умолчанию из main вызывать eval_forth("example.frt") Пример, если в example.rft будет:

```
1       put 1
2       put 3
3       add
4       print
```

То программа должна напечатать '4'.

Сложение имеет такой же смысл, как и в питоне. Вычитание для строк не определено Программа должна содержать функцию eval_forth(), принимающую строку на языке forth и исполняющую ее. По умолчанию из main вызывать eval_forth("example.frt")

Умножение больших чисел

Реализовать алгоритм Карацубы для умножения больших чисел.

$$AB * CD == (A + B * 2^m) * (C + D * 2^m)$$
$$== A * C + 2^{2*m} * B * D + 2^m * (B * C + A * D)$$
$$== A * C + 2^{2*m} * B * D + 2^m * ((A + B)(C + D) - A * C - B * D)$$

Острова

Задан двумерный массив из 0,1. Островом называется связная группа единиц, т.е. такие что от любой из них можно дойти до любой другой, перемещаюсь за шаг на одну клетку вверх, вниз, вправо, влево или по диагонали и не попадая на клетки с '0'. Посчитать количество островов.

Операции над множествами через сортировку

Написать следующие функции над массивами. Все они должны исполняться за $O(n * log(n))$, где n - количество элементов в большем массиве. В результирующем массиве не должно быть дубликатов.

- union - пересечение двух множеств. Все элементы, которые есть хотя-бы в одном из множеств.
- difference - все элементы, которые есть в одном из множеств, но отсутствуют во втором.

Поиск удаленных и созданных файлов

Написать функцию, которая получает два пути, проходит по указанным директориям и всем вложенным и находит какие файлы есть только в одной из них. Возвращает пару списков с именами файлов, имеющимися только в одной из дирректорий.

FP

Написать функции:

- my_map, принимает функцию и список, возвращяет список полученный в итоге применения переданной функции к каждому элементу из списка-параметра. map(func, lst) == [func(lst[0]), func(lst[1]), ..., func(lst[N])]
- my_filter(func, lst1) -> lst2. lst2 содержит только те элементы из lst1, для которых func возвращает True
- my_reduce(my_fold) описание reduce.
- рекурсивные варианты всех этих функций
- Функцию bind, которая принимает функцию func и список параметров params1 и возвращает функцию, при вызове которой со списком параметров params2 вызывается func с объединенным списком параметров params1 + params2. То-же, но с поддержекой именованных аргументов. bind(func, 1, 2, "3")(2, 4) == func(1, 2, "3", 2, 4)
- my_map_gen, my_filter_gen, my_reduce_gen, которые принимают генераторы и возвращают генераторы

super

Написать my_super, аналогичный по поведению встроенному

Обработка файла

Написать конвеерные генераторы для обработки тектовых потоков.

- iter_lines(fd) получает имя файла итерирует по строкам. Для чтения можно использовать только fd.read(1)
- strip_spaces(iter) -> принимает итератор, получает из него строки и возвращает строки без стартовых и финальных пробельных символов
- drop_empty - получает итератор и возвращает только не пустые строки
- split_items - получает итератор, считывает из него строки, разбивает их по пробелам и для каждого элемента определяет является ли он строковым представлением целого или числа с плавающей запятой. Приводит опознанные елементы к int/float соответсвенно, остальные оставляет строками. Возращает итератор по этим элементам
- get_ints - возращает из входного потока только целые
- my_sum - считает сумму элементов целых во входном потоке

```
# fd == "1 2 3 3.45 abra_cadabra    \n\n12"
list(iter_lines(fd)) == \
    ["1 2 3 3.45 abra_cadabra    ", "", "12"]
```

```
    list(strip_spaces(iter_lines(fd))) == \
        ["1 2 3 3.45 abra_cadabra", "", "12"]
    list(drop_empty(["1 2 3 3.45 abra_cadabra", "", "12"])) == \
        ["1 2 3 3.45 abra_cadabra", "12"]
    list(split_items(["1 2 3 3.45 abra_cadabra", "12"])) == \
        [1, 2, 3, 3.45, "abra_cadabra", 12]
    list(get_ints([1, 2, 3, 3.45, "abra_cadabra", 12])) == \
        [1, 2, 3, 12]
    my_sum([1, 2, 3, 12]) == 18

    my_sum(get_ints(drop_empty(strip_spaces(iter_lines(fd))))) ==
```

Сайт на CherryPy

CherryPy это библиотека для написания простых веб приложений. Необходимо написать иерархию классов для сайта.

- Пустой класс Router
- Класс BaseSite с методом index, который возвращает "Hello"
- Класс PolliteSite, наследующий BaseSite, имеющий метод set_name, принимающий имя в качестве параметра. В методе index он должен возвращать "Hello" + name. name по умолчанию пустое.
- Класс StructuredSite от PolliteSite, который добавляет два метода - header и footer. Заголовок и подложка, которые должны выводиться вверху и внизу каждой страници. По умолчанию он пишет в footer текущее время (модуль datetime). А в header - "name текущщее имя пользователя".

CherryPy привращает HTTP запрос (та строка, которую вы вводите в адресной строке браузера) вида http://domain/x/y/z?var1=val1&var2=val2 в вызов RootObject.x.y.z(var1=val1, var2=val2) или в вызов RootObject.x.y.z.index(var1=val1, var2=val2) в зависимости от того, что есть.

```
1 import cherrypy
```

```
2
3       class Router(object):
4           pass
5
6       Router.base = BasicSite()
7       Router.pollite1 = PolliteSite()
8       Router.pollite2 = PolliteSite()
9
10      cherrypy.quickstart(HelloWorld())
```

Запускаем эту программу, запускаем браузер и вводим в адресной строке '127.0.0.1:8080'.

Задание - func_info

Написать функцию func_info, которая принимает функцию и печатает ее

- Имя
- Количество параметров
- Документацию
- Значения параметров по умолчанию
- Поля искать через ipython/google/python doc

Задание - композиция функций

Написать функцию haskell_dot, которая принимает неограниченное количество функций и возвращает новую функцию, поторая при вызове последовательно применяет все сохраненные функции к параметру.

haskell_dot(f1, f2, f3, ...) -> fC
fC(x) = f1(f2(f3(...(x))))

ООП 1

- Сделать класс рациональное число (BasicRationalNumber), имеющий два поля - числитель и знаменатель. Значения этих полей должны передаваться в конструктор класса BasicRationalNumber.
- Сделать класс CalcRationalNumber, который имеет методы add, sub, mul and div. Которые, соответсвенно, вычисляю сумму, разность, произведение и отношение текущего RationalNumber и переданного в качестве параметра.
- Сделать класс SimplRationalNumber, который автоматически делит числитель и знаменатель на наибольший общий делитель после каждой операции
- Добавить в класс BasicRationalNumber метод __str__ из которого возвращать строковое представление объекта

```
1       def mk_str(rn):
2           return "{}/{}".format(rn.numer, rn.denom)
3
4       rn = BasicRationalNumber(2, 3)
5       print mk_str(rn) # 2/3
```

```
6
7       x1 = CalcRationalNumber(1, 2)
8       x2 = CalcRationalNumber(1, 3)
9       x3 = x1.sub(x2)
10      print mk_str(x3) # 1/6
11
12      x3 = x3.add(x3)
13      print mk_str(x3) # 2/6
14
15      x4 = SimplRationalNumber(1, 6)
16      x4.add(x4)
17      print mk_str(x4) # 1/3
```

web crawler

Написать программу для загрузки веб сайтов. На вход передается url и глубина. Программа должна загрузить указанную страницу и все страницы с того же домена, на которые она ссылается прямо или не более чем через X-1 страниц, где X - указанная глубина.

Сериализация

Написать функции, умеющуе сериализовать и десериализовать следующие типы данных: list, int, str, dict и все их комбинации. Например такое значение {'a':1, 'b':[1,2,3,['3']], 4:7}. Длинные целые поддерживать не надо. Сериализаций называется превращение значения в строку, представление которой не зависит от аппаратных особенностей компьютера. serialize должна возвращать строку, вызов deserialize от которой вернет значение, равное значению, переданному в serialize. Нельзя использовать eval/pickle/marshal и прочие готовые решения. Строки внутри передаваемого значения могут содержать любые символы (например '
x00', '{', '}', '[', ']', '(', ')', etc). Не стоит для преобразования полагаться на встроенное преобразование объектов в строку с помошью функций str/repr. Необходимые модули: struct.

RPC

Используя функцию из задачи "Сериализация" или pickle.dumps и сокеты реализовать сервер и клиент для удаленного вызова функций. Удаленный вызов означает, что на одном компьютере(клиенте) вызывается специальная процедура-прокси которая передает параметры по сети на сервер. На сервере необходимая процедура вызывается с этими параметрами. Результат передаетмя назад на клиент и процедура-прокси возвращает его. Таким образом код, испольщующий прокси не замечает факта общения по сети.

Обход дерева

Найти 10 самых больших файлой в дереве, начинающемся с указанной папки.
Необходимые функции: os.stat, os.walk.

Создание дерева папок

Написать функцию, которая получает словарь вида {str: str or None} и строку(root) и создает на диске дерево файлов следующиего вида - для каждого элемента в словаре, у которого значение не None - создается файл с путем os.path.join(root, "ключ елемента") и в него записывается значение элемента. Если значение None - то по аналогичному пути создается директория. пример:

вызов create_tree ("/tmp/ test_dir ", \{'a':None, 'b/c':'xxxx', 'm/t/y/u':None\}) создаст пустые папки '/tmp/ test_dir /a' и '/tmp/ test_dir /m/t/y/u' и файл "/tmp/test_dir/b/c", содержащий 'xxxx'.

Сравнение папок

Сравнить два дерева папок в файловой системе. Фукция получает два пути к папкам в файловой системе и находит все файлы, которые присутвуют только в одном из деревьев, но отсутвуют в другом. Файл считается отсутвующим, если в аналогичной папке в другом дереве нет файла с таким-же именем. Наличие файла с таким же именем в другой папке второго дерева не учитывать. Файлы сравниваются только по имени, содержимое и атрибуты не учитываются. Аналогичной называется папка имеющая такой же путь от своего корня. Например - папки /x/y/z и /t/r/z считаются аналогичными, если корни /x/y и /t/r соответственно. Функция итерирует по всем таким файлам, для каждого из них выбрасывая наверх пару (bool {True если файл только в первой папке, False - если только во второй}, путь от того корня, под которым файл найден)

Для юнит-тестирования используйте функцию из задания "Создание дерева папок".

Необходимые функции: os.path.join, os.walk, os.listdir, shutil.rmtree.

Алгоритм Кнута-Морриса-Пратта

-

Тетрис

ООП, сделать GUI - заготовку

subprocess

Используя subprocess запустить в фоне tcpdump и выводить на количество траффика по отдельным протоколам.