

Инсталляция питона

- Установить python
- Установить 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 expression](#)

Разложить число на простые делители

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

строковые функции

Написать строковые функции `xfind`, `xreplace`, `xsplrit`, `xjoin` используя срезы строк (без применения встроенных методов строк). Для получения требований на функцию посмотрите документацию к соответствующему методу строки в `ipython`.

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

Декодирование АОН

- Нужно преобразовать строку по следующим правилам:
- Если символ идет 2 и больше раз подряд - 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}

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

Двоичный поиск

Найти элемент в упорядоченном массиве методом дихотомии.

Обработка файла

Написать конвейерные 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 элементы к `int/float` соответственно, остальные оставляет строками. Возвращает итератор по этим элементам
- `get_ints` - возвращает из входного потока только целые
- `my_sum` - считает сумму элементов целых во входном потоке

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



```
4 list(strip_spaces(iter_lines(fd))) == \
5     ["1 2 3 3.45 abra_cadabra", "", "12"]
6 list(drop_empty(["1 2 3 3.45 abra_cadabra", "", "12"])) == \
7     ["1 2 3 3.45 abra_cadabra", "12"]
8 list(split_items(["1 2 3 3.45 abra_cadabra", "12"])) == \
9     [1, 2, 3, 3.45, "abra_cadabra", 12]
10 list(get_ints([1, 2, 3, 3.45, "abra_cadabra", 12])) == \
11     [1, 2, 3, 12]
12 my_sum([1, 2, 3, 12]) == 18
13
14 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 страницы. По умолчанию он пишет в footer текущее время (модуль datetime). А в header - "name текущее имя пользователя".

CherryPy превращает 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    cherry.py . quickstart ( HelloWorld ( ) )
```

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

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, соответственно, 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.

Обход дерева

Найти 10 самых больших файлов 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 следующего вида - для каждого элемента в словаре, у которого значение не None - создается файл с путем `os.path.join(root, "ключ элемента")` 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. Функция получает два пути к папкам в файловой системе и находит все файлы, которые присутствуют только в одном из деревьев, но отсутствуют в другом. Файл считается отсутствующим, 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 вверх 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`.

Разбор HTTP запроса

Сделать функцию, которая получает на вход строку HTTP запроса и возвращает удобный для обработки объект

```
1
2 request = ""
3 POST /x/y/z HTTP/1.0
4 Host: some.host.org
5 content-type: application/x-www-form-urlencoded; charset=utf-8
6 content-length:207
7
8 Action=GetStatus
9 &JobId=JOBID
10 &AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
11 &SignatureMethod=HmacSHA256
12 &SignatureVersion=2
13 &Version=2010-06-03
14 &Signature=lBP67vCvGIDMBQ1dofZxg8E8SUEXAMPLE
15 &Timestamp=2011-06-20T22%3A30%3A59.556Z
16 ""
```

subprocess

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