
File Edit Tools Syntax Buffers Window Help
import re
from datetime import date
# The parse_adif function came from http://web.bxhome.org/blog/ok4bx/2012/05/adif-parser-python
def parse_adif(fn):
raw = re.split('<eor>|<eoh>(?i)',open(fn).read())
raw.pop(0) #remove header
raw.pop() #remove last empty item
logbook =[]
for record in raw:
qso = {}
tags = re.findall('<(.*?):(\d+).*?>([^<\t\n\r\f\v\Z]+)',record)
for tag in tags:
qso[tag[0].lower()] = tag[2][:int(tag[1])]
logbook.append(qso)
return logbook
dat = open(file_dat)
stn = {}
for a in dat :
b = a.rstrip().spl
if b[0] != '' :
stn[b[0]]
nam = open(file_nam)
sname = {}
for a in nam :
b = a.rstrip().split(',')
if b[0] != '' :
sname[b[0]] = b[1]
g = {}
for i in range(0,len(l)):
if l[i]['app_writelog_p'] != '0':
bandmode = l[i]['band'] + '-' + l[i]['mode']
My Programming
Language Evolution
... and why I now use
Python

The Programming Languages

- FORTRAN IV - IBM Mainframe
- BASIC - IBM Mainframe and Z80 PC
- C (not C++) - Z80 PC
- Perl - Linux
- Python - Linux/Windows
- Others including Pascal, LISP, 360 Assmebler, Z80 Assembler, PL/1, SNOBOL and several others I have conveniently forgotten.

"Hello World!"

- Adopted as the introduction to the simplest programs in most modern languages.
- Started with "C" in *The C Programming Language*.
- Can of course be programmed in ANY language.

"Hello World!" in FORTRAN IV

On punched cards:

```
//C0122689  JOB  (0,0,,,,,1,,0),
// MA105092.B.M.FERRY,MSGLEVEL=1
//A EXEC FORTGCLG
//FORT.SYSIN DD *
      WRITE(6,10)
10    FORMAT('HELLO WORLD!')
      STOP
      END
/*
//GO.FT06F001 DD SYSOUT=A
```

"Hello World!" in BASIC

```
10 PRINT "Hello World!"
```

"Hello World!" in C

The official version from: *The C Programming Language*

```
main()
  {
        printf("hello, world");
  }
```

"Hello World!" in Perl

Perl is a staple of Linux. I learned it from experimentation and a couple of books. It was the first language I used to interact with MySQL.

```
print "Hello World!";
```

"Hello World!" in Python

Python is my latest new language. I wish I had pursued it sooner.

```
print "Hello World!"
```

A More Complex Example

- In C:

```
#include <stdio.h>
main () {
static int i;
for (i=1; i<11; ++i) {
    printf("%d\n",i);
    }
}
```

- In Perl:

```
for ($i=0; $i<11; ++$i) {
    printf("%d\n",$i);
    }
```

- In Python:

```
for i in range(1,11) :
    print "%d" % i    # "print i" would also work
```

Python Features

- Interpreted/compiled
- Variables need not be pre-defined and don't have an extra character to denote them like Perl's '$' sign.
- All of the usual programming constructs are there but are implemented differently sometimes.
- No silly braces {} or semicolons needed. Blocks are denoted by indenting.
- Comments are easy with a '#' symbol.

Python Features - 2

- Programs tend to be shorter than Perl.
- Variable types are automatic based on content.
- Linear arrays work in natural ways. For example: 'for a in b :' where 'b' is an array executes the block using each value in array 'b' assigned to variable 'a'. Array 'b' could also be an array of arrays.
- Dictionaries, similar to Perl hashes, work pretty much like arrays but need to be defined so Python knows what you're up to.

Python Features - 3

- To nest deeper blocks just use more indenting.
- There are no particular limits on any program structure. A number, string or array can be as big as you need it to be.
- Python programs can be pre-compiled to make execution faster.
- File reading and writing as well as a wealth of built in functions are handy and well documented.

Python Features 4

- The substring function is such a popular thing to do that it is extra easy: a[2:3] means substr(a,2,3). a[4:] means substr(a,4). a[-4:] means the last 4 characters.
- Concatenating strings is easy – just add them: c = a + b. Or you can use something like: a += b.
- To add an element to an array just add it: a[ ] += [b]
- Splitting as string ('c') is easy too: c.split(',') returns an array of strings.
- Lots of other functions, almost anything you can think of.

Python Unique Things

- When you create a variable the LOCATION of the value is stored in the variable descriptor. EVERY Python element POINTS to that element value somewhere.
- Like values can be shared. READ THE BOOK to make sure you understand this. REALLY odd things can happen if you don't.
- Arrays like [1,2,3] are different from tuples like (1,2,3). Tuples cannot be changed but arrays can. Tuples elements are protected and cannot be changed. You can concatenate two tuples together though.
- Odd structures are easy. Things like a dictionary of arrays:

```
a = {}      # 'define' variable 'a' as a dictionary
a['The'] = ['quick','brown']
a['The'] += ['fox','jumped','over']
print a
{'The': ['quick', 'brown', 'fox', 'jumped', 'over']}
```

- There are other really unique things that I haven't dug into yet.

Python Code Examples

- I have found out that a lot of code examples exist out there. Google can find them.
- One of the early programs I wrote needed a way to parse ADIF format files. This is not the easiest file format to break down in any other language but in Python it was simple. I found the code I needed at the top of my first Google search: 'python ADIF'

Python Versions and Extras

- Know what version of Python you are using. Some things are different between versions! Most significant is the change of 'print' from a statement to a function in version 3.
- If you install it on Windows pick the same version you are using on Linux.
- Expect to add extra packages to support things like MySQL. Once installed they're easy to use. They usually install as RPMs (or the usual Windows install things).

A Problem for Python

I have a file, shown below, that I want to categorize. It's a file of "Field Notes" from a recent Geocaching day and I want to know how many I found and how many I didn't find. Here's what the file looks like:

```
GC44PGQ,2013-09-18T12:21Z,Found it,""
GC3G4NV,2013-09-18T13:06Z,Didn't find it,""
GC4AEB0,2013-09-18T13:07Z,Found it,""
GC44GYF,2013-09-18T13:20Z,Found it,""
GC49G8K,2013-09-18T14:08Z,Didn't find it,""
GC3XQBM,2013-09-18T14:08Z,Found it,""
```

There are more lines but you get the idea. It's four fields separated by commas. I only care about summarizing what's in the third field.

The Python Program

```
gv = open('geocaching_visits.txt')   # Open the file
tally = {}                  # Create 'Dictionary' (like Perl hash)
for a in gv :               # Read each line in the file into 'a'
    b = a.split(',')          # Split 'a' into an array 'b' at commas
    if b[2] not in tally :    # If the key isn't in 'tally' add it
        tally[b[2]] = 0       # Count to zero
    tally[b[2]] += 1          # add one to the count
tkeys = tally.keys()        # extract the keys
for k in tkeys :            # loop through the keys
    print "%-20s %d" % (k, tally[k])   # print it
```

The Python Program Condensed Version

```
tally = {}
for a in open('geocaching_visits.txt') :
   b = a.split(',')
   if b[2] not in tally : tally[b[2]] = 0
   tally[b[2]] += 1
for k in tally.keys() : print "%-20s %d" % (k, tally[k])
```

OR:

```
tally = {}
for a in open('geocaching_visits.txt') :
   b = a.split(',')[2]
   if b not in tally : tally[b] = 0
   tally[b] += 1
for k in tally.keys() : print "%-20s %d" % (k, tally[k])
```

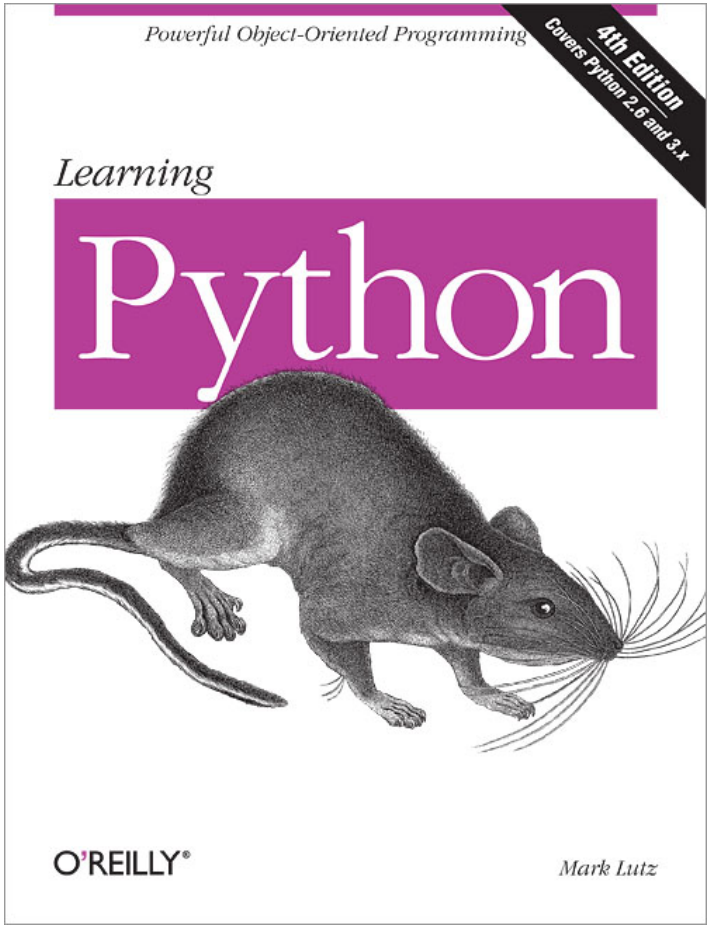
The Output

```
Didn't find it          7
Found it                38
```

Why I Like Python over Perl

- Python is so different from everything else I don't mix up how to do things as I do in Perl. Perl is ALMOST like C but just enough different to be annoying.
- Python does complex things with much less fuss than Perl. It's easier to handle arrays both linear and dictionaries.
- The online documentation for Perl is less than useful. It's great if you know what you need and are just looking for details. If you're stumbling around trying to find out HOW to do something it falls short. Python is much better.
- Python, like Perl, is cross platform. I use it under both Windows and Linux. There are some differences but nothing major.

How to Learn Python



- Get the O'Reilly book. It takes you through the various aspects of the language in a very logical way. It does assume you know something about programming.
- There are several online resources also available.
- DO NOT try to just learn Python by trial and error. Python is just too different. That's where I went wrong when I first tried to learn Python. It took me years to give it another try.