

# Python Workshop

## File Input/Output

Mike Fienen

U.S. Geological Survey  
Wisconsin Water Science Center, Middleton, Wisconsin USA

USGS National Groundwater Workshop, August 2012



# Outline

# Overview

- Much of what is useful to do in Python is reading files, manipulating the data, and writing out results in another format
- Python and Numpy provide ways to read and write ASCII and binary files
- We will focus on ASCII files

# Reading and Writing with Strings

- The simplest way to write information to a string is using  
`str`

```
>>>a = 5.4
>>>str(a)
'5.4'
```

- We typically want more control. Two main ways to do it.  
Old school (%) and new school (`format`)
- Formatted input and output are a key difference between  
Python 2.X and 3.X

# Writing Strings the Old School Way (%)

- The general syntax is to make a string with conversion types for variables. For example:

```
>>>outstr = 'I have %21.1f kg of %s and %d bins of %s' %(3.99983,'eggs',53,'spam')
>>> outstr
'I have          4.0 kg of eggs and 53 bins of spam'
>>>outstr = 'I have %-21.1f kg of %s and %0.3d bins of %s' %(3.99983,'eggs',53,'spam')
>>> outstr
'I have 40.0      kg of eggs and 053 bins of spam'
```

- The general idea is to make a string including ' % ', a conversion flag (optional), a width and resolution (optional), and a conversion type (required).

For example:

%<flag><width>.<resolution><type>  
%-12.3f Is a left-justified, floating point value with width of 12  
and 3 decimal places.

- Following the format string must be a list of values as a tuple identified by %

'%4d is the %s\n' %(42,'answer')

# Writing Strings the Old School Way (%)

Details about formatted output available at:

<http://docs.python.org/library/stdtypes.html>

- Conversion flag characters

- '#' Invokes alternate behavior (see website for details)
- '0' Pads numeric values with zeros
- '-' Left-adjusts the output
- ' ' Leave a space before signed positive values so they line up with negative ones

- Most common conversion types.

%d or %i Signed integer

%f or %F Floating point

%e or %E Floating point exponential (lower or upper case)

%g or %G Combination of %f and %e depending on resolution

%s or %r String. Width is used, but not resolution

# Writing Strings the Old School Way (%)

Try it out! '%4d is the %s\n' %(42,'answer')  
%<flag><width>.<resolution><type>

- Conversion flag characters

- '#' Invokes alternate behavior (see website for details)
- '0' Pads numeric values with zeros
- '-' Left-adjusts the output
- ' ' Leave a space before signed positive values so they line up with negative ones

- Most common conversion types.

- %d or %i Signed integer
- %f or %F Floating point
- %e or %E Floating point exponential (lower or upper case)
- %g or %G Combination of %f and %e depending on resolution
- %s or %r String. Width is used, but not resolution

# Writing Strings the New School Way (format)

Details about new school string formatting at:

<http://docs.python.org/library/string.html#formatstrings>

- The general syntax is similar, but conversion information is supplied differently. For example:

```
>>>outstr = 'I have {:.2f} kg of {:s} and {:0=3d} bins of {:s}'.  
format(3.99983, 'eggs', 53, 'spam')  
>>> outstr  
'I have' 4.0 kg of eggs and 053 bins of spam'
```

- In this case, make a string including `{...}`, statements with conversion information.  
The general pattern is `{[index] : [format]}`

- ▶ The `[index]` argument refers to the item index being mapped in
- ▶ The `[format]` argument is similar to those in the old school way, but with some additional flexibility

# A Slight Aside on User Input

- There are two main ways to get user input into a script from the command line
  - ➊ Ask the user for it
  - ➋ Get it from the script call

# Ask the user

- Asking the user for input

```
>>> st = raw_input("What is your favorite color? ")  
What is your favorite color? blue, no green!  
>>> st  
'blue, no green!'
```

- The input will always be a string which you can parse and convert as you wish.

# Parse the call string

- Arguments in the call string can also be passed

```
>>> import sys  
$ python testscript.py arg1 arg2  
>>>scriptname = sys.argv[0]  
>>>arg1 = sys.argv[1]  
>>>arg2 = sys.argv[2]  
...
```

- The list returned by `sys.argv` is all strings
  - 0th argument is the script name
  - space-delimited arguments follow from there

Now that we can write strings, we can write them to files

But first, We need to read stuff back in *from* files

# Interacting with Text Files

- The first thing is to open a file and make a file object

```
ifp = open('somefile.txt', 'r')
ofp = open('someotherfile.txt', 'w')
```

- ▶ This object can be used to read or write from.  
I use `ifp` for “input file pointer” and  
`ofp` for “output file pointer”  
The arguments ‘`r`’ and ‘`w`’ indicate “read” and “write”  
respectively.

- To read the file can use `readline()` or `readlines()`

- ▶ The difference is that `readlines()` reads the entire file  
into memory rather than `readline()` which reads one line  
at a time. Most of the time, `readlines()` is better
- ▶ With `readlines()` once the data are read in, the result is  
a list with each element representing a line in the text file

# Readlines Example

Here's a file called `example_file.txt`

```
This is a file.  
It is a very nice file, no?  
It has several lines.  
Some of them are numbers.  
This is NOT a poem.  
3.14 pi 5  
But code --> now THAT's poetry!
```

Now let's read it in:

```
>>>import os  
>>>filepath = os.path.join('data', 'example_file.txt')  
>>>whole_file = open(filepath, 'r').readlines()  
>>> whole_file  
['This is a file.\n', 'It is a very nice file, no?\n', 'It has several lines.\n',  
 'Some of them are numbers.\n', 'This is NOT a poem.\n', '3.14 pi 5\n',  
 "But code --> now THAT's poetry!"]
```

What if it's a windows file?

```
>>>filepath = os.path.join('data', 'example_file_win.txt')  
>>>whole_file = open(filepath, 'r').readlines()  
>>> whole_file  
['This is a file.\r\n', 'It is a very nice file, no?\r\n', 'It has several lines.\r\n',  
 'Some of them are numbers.\r\n', 'This is NOT a poem.\r\n', '3.14 pi 5\r\n',  
 "But code --> now THAT's poetry!"]
```

# Parsing input strings

- Using `strip` and `split`

- ▶ `strip()` removes newline and tab characters from the end

```
>>>line = 'USGS      430406089232901 2010-12-03      15.04  P\t\r\n'  
>>>line.strip()  
'USGS\t430406089232901\t2010-12-03\t15.04\tP'
```

- ▶ `split()` breaks up a string on whitespace
  - ▶ Can take any character as an argument (usually ',' or ' ')

```
>>>line.strip().split()  
['USGS', '430406089232901', '2010-12-03', '15.04', 'P']  
>>>line.strip().split('0')  
['USGS\t43', '4', '6', '892329', '1\t2', '1', '-12-', '3\t15.', '4\tP']
```

- ▶ Stacking `strip` and `split` is a common violation of the general rule not to stack up function calls

# Parsing input strings (continued)

- Using pop

```
>>>a=line.strip().split()  
>>>a  
['USGS', '430406089232901', '2010-12-03', '15.04', 'P']  
>>>a.pop()  
'P'  
>>>a  
['USGS', '430406089232901', '2010-12-03', '15.04']  
>>>a.pop(1)  
'430406089232901'  
>>>a  
['USGS', '2010-12-03', '15.04']
```

- ▶ `pop()` *both* returns an element from a list *and* removes it from the remaining list

- Regular expressions are very flexible but another topic

```
>>>import re  
>>>allints = re.findall("[0-9]",line)  
>>>allints  
['4', '3', '0', '4', '0', '6', '0', '8', '9', '2', '3', '2', '9', '0', '1', '2',  
'0', '1', '0', '1', '2', '0', '3', '1', '5', '0', '4']
```

# An Example Text File from NWIS

```
# ----- WARNING -----
# Provisional data are subject to revision. Go to
# http://waterdata.usgs.gov/nwis/help/?provisional for more information.
#
# File-format description: http://waterdata.usgs.gov/nwis/?tab_delimited_format_info
# Automated-retrieval info: http://waterdata.usgs.gov/nwis/?automated_retrieval_info
#
# Contact: gs-w_support_nwisweb@usgs.gov
# retrieved: 2012-07-16 17:24:35 EDT (vaas01)
#
# Data for the following 2 site(s) are contained in this file
#    USGS 430406089232901 DN-07/09E/23-1297
#    USGS 430427089284901 DN-07/09E/19-0064
#
#
# Data provided for site 430406089232901
#   DD parameter statistic   Description
#   01    72019      00001   Depth to water level, feet below land surface (Maximum)
#
# Data-value qualification codes included in this output:
#   P  Provisional data subject to revision.
#
agency_cd site_no datetime 01_72019_00001 01_72019_00001_cd
5s 15s 20d 14n 10s
USGS 430406089232901 2010-12-03 15.04 P
USGS 430406089232901 2010-12-04 14.92 P
...
...
```

# Reading NWIS Output File

```
def NWIS_reader(infile):
    """
    NWIS_reader(infile)
    A function to read in an NWIS file generated using USGS webservices.
    Mike Fienen - 7/16/2012
    <mnfienen *at* usgs *dot* gov>

    INPUT:
    infile --> the name of an input file in USGS RDB (tab-delimited) format

    OUTPUT:
    indat --> a dictionary with keys corresponding to site numbers and each
              element being a dictionary with keys date and depth to water.
    ...
    # tell the user what's happening
    print 'Reading Data from file: %s' %(infile)
    #
    # set up a couple initial variables
    #
    # format for reading the date --> formats noted at
    #           http://docs.python.org/library/datetime.html (bottom of the page)
    indatefmt = "%Y-%m-%d"
```

# Reading NWIS Output File (Continued)

```
# open the text file and read all lines into a variable "tmpdat"
tmpdat = open(infile,'r').readlines()

# make empty lists to temporarily hold the data
Site_ID = []      # site ID
dates = []         # date of measurement (daily values)
DTW = []           # depth to water below land surface (feet)
prov_code = []     # provisional code: [P] is provisional, [A] is accepted
# loop over the input data, keep only proper data rows. Parse and assign to lists
for lnum, line in enumerate(tmpdat):
    # first read the lookup information from the header of the file
    if ("data for the following" in line.lower()):
        nWells = int(re.findall("[0-9]+",line)[0])
        statnums = []
        countynums = []
        for cwell in np.arange(nWells):
            nextline = lnum+1+cwell
            tmp = tmpdat[nextline].strip().split()
            statnums.append(tmp[2])
            countynums.append(tmp[3])
        station_lookup = dict(zip(statnums,countynums))

    if (('usgs' in line.lower()) and ('#' not in line)):
        tmp = line.strip().split() # strip newline off the end and split on whitespace
        Site_ID.append(tmp[1])
        dates.append(datetime.strptime(tmp[2],indatefmt)) #convert date to a time tuple
        DTW.append(tmp[3])
        prov_code.append(tmp[4].lower()) # --> note conversion to lower case!
```

# Try Running the Code

Quick aside on `os` module...

Windows uses "\ " for separating path components, but everyone else uses "/"

So, you can make a Windows path as:

```
>>> infile =  
'..\\data\\NWIS_data.dat'
```

That works, but only on Windows.  
Better style is:

```
>>> infile =  
os.path.join('..', 'data', 'NWIS_data.dat')  
>>> infile # on a Mac  
'..../data/NWIS_data.dat'  
>>> infile # on Windows  
'..\\data\\NWIS_data.dat'
```

*That is platform independent*

Now, let's try an example in the EXERCISES/FILE\_I\_O directory

```
>>> import NWIS_read_parse as NWR, os  
>>> infile = os.path.join('..', 'data', 'dane_county_GW_wells.dat')  
>>> indat, county_lookup = NWR.NWIS_reader(infile)  
Reading Data from file: ..../data/dane_county_GW_wells.dat  
Reading ..../data/dane_county_GW_wells.dat complete!  
>>> indat.keys()  
>>> county_lookup.keys()
```

# np.genfromtxt: flexible way to read columns

## Example file: STATE\_FIPS.csv

```
State Abbreviation,FIPS Code,State Name
AK,02,ALASKA
AL,01,ALABAMA
AR,05,ARKANSAS
AS,60,AMERICAN SAMOA
AZ,04,ARIZONA
CA,06,CALIFORNIA
...
```

```
import numpy as np
filename = 'STATE_FIPS.csv'
indat = np.genfromtxt(filename,delimiter=',',dtype=None,names=True)
```

delimiter=',' delimiter can be *anything*

dtype=None Numpy interprets column data types. If unknown, makes it a string

names=True Each column gets a data type and a name

```
In [10]: indat
```

```
Out[7]:
```

```
array([('AK', 2, 'ALASKA'), ('AL', 1, 'ALABAMA'), ('AR', 5, 'ARKANSAS'), ...
dtype=[('State_Abbreviation', '|S2'), ('FIPS_Code', '<i4'), ('State_Name', '|S20')])
```

*N.B. → underscore replaces space in names*

# Writing Back out to a File

- First need a file object in the same way as reading

```
ofp = open('some_outfile.txt', 'w')
```

- Next, create a string of output
- Write the string using

```
ofp.write(<string>)
```

- Remember to put a newline character '\n' at the end of each line

# Pulling a data file from the Web

- An example using REST (Representational State Transfer a.k.a. a RESTful query) of USGS water data.

```
import urllib
fullURL = 'www.place.gov/some_path_to_a_data_file.txt'
datastream = urllib.urlopen(fullURL).read()
outfilename = 'local_filename.txt'
open(outfilename, 'wb').write(datastream)
```

- urllib enables simple interaction with a URL
- BeautifulSoup allows for much more sophisticated complete web-scraping applications (not built in)
- Writing the local version of the file as binary is most robust:
  - ▶ Retains a copy of exactly what was downloaded
  - ▶ Writes the copy without respect to formatting issues
- One could work only in memory and not make a local copy

# Using Stackoverflow

Questions containing "python" ↗ stackoverflow.com/search?q=python+download+file

mishaf 270 1 10 review chat meta about faq python download file

## stackoverflow Questions Tags Users Badges Unanswered Ask Question

### Search Results

python download file relevance newest votes active search

Want better search results? See our search tips!

Python is an interpreted, general-purpose high-level programming language whose design philosophy emphasizes code readability.

item more | improve tag wiki | top users | synonyms (1)

**2 answers 3 questions** python 140 views

**13 votes 6 answers** python django 9k views

**6 votes 4 answers** python 2k views

**10 votes 3 answers** python http 3k views

**0 votes 1 answer** python 139 views

**3,901** search results for posts containing download file tagged with python

**Favorite Tags** python mysql sql ignored tags

**CAREERS 2.0** Develop Higher Pay CAREERS 2.0 CAREERS 2.0 Technical Project Manager MDx Medical Inc dba Vitalis.com Lyndhurst, NJ, United States 11/14 Konstantin 11/14 944 8 23

Programmer/Hayneedle Digital Schoolhouse Columbus, OH Programmer/Hayneedle Association of College 8... Columbus, OH More jobs near Columbus...

# Using Stackoverflow

How to download a file in python

StackExchange v misha 270 10 review chat meta about faq search

Questions Tags Users Badges Unanswered Ask Question

## How to download a file in python

I tried to download something from the internet with python. I use urllib.retriever from the urllib module but I just can't get it work. I would like to be able to save the downloaded file to a location of my choice. If someone could explain me how to do it with clear examples, that would be VERY appreciated. =)

python link | edit | flag

add comment start a bounty

1 Answer

I suggest using urllib2 like so:

```
source = urllib2.urlopen("http://someUrl.com/somePage.html").read()
open("/path/to/someFile", "wb").write(source)
```

You could even shorten it to (although, you wouldn't want to shorten it if you plan to enclose each individual call in a `try - except`):

```
open("/path/to/someFile", "wb").write(urllib2.urlopen("http://someUrl.com/somePage
```

link | edit | flag

active oldest votes

answered Nov 14 '11 at 2:23 by user1044824 18 2

1 You might consider using `"wb"` rather than `"w"` in case the downloaded file is binary. – icktoofay Nov 14 '11 at 2:25

@icktoofay Good call, fixed. – chown Nov 14 '11 at 2:25

Thanks it working very well, now I just have to understand why exactly. =) – user1044824 Nov 14 '11 at 3:15

@user which part doesn't make sense? I'll try to explain more if possible. – chown Nov 14 '11 at 18:50

add comment

Your Answer

B I

tagged python × 119744

asked 8 months ago viewed 140 times active 8 months ago

CAREERS 2.0

Technical Project Manager MDx Medical Inc d/b/a Vitalis.com Lyndhurst, NJ; Tulsa, OK

Senior NetLead Engineer Digital School Network Columbus, OH

Programmer/Analyst-Lead Association of College &... Columbus, OH

More jobs near Columbus...

Linked

Need clear explanation about how to download a file in python

Related

How do I download a file over HTTP using Python?

How to download a file using python in a 'smarter' way?

Serving file download with python

How to Improve performance of

# Some Useful Resources

- Building queries for RESTful<sup>1</sup> queries of USGS water data  
<http://waterservices.usgs.gov/>
- USGS Water data type pm code lookup  
<http://nwis.waterdata.usgs.gov/nwis/pmcodes/>
- General I/O information in Python documentation  
<http://docs.python.org/tutorial/inputoutput.html>

# Some More Useful Resources

- Getting data through CIDA/ACWI from the National Groundwater monitoring Network (NGWMN) Portal

- Main page

[http://cida.usgs.gov/gw\\_data\\_portal/](http://cida.usgs.gov/gw_data_portal/)

- List of wells where data are available

<http://cida.usgs.gov/ngwmn/wells>

Hover over a link to see code information

- Data are returned in WaterML2 Format

<http://www.waterml2.org/>

- Similar RESTful Query

<http://cida.usgs.gov/ngwmn/data/<agencyCd>/<siteNo>/<data-type>>