Reading/Writing from configuration files

```
# databaseconfig.py
#!/usr/bin/env python
import preprocessing

mysql = {
    "host": "localhost",
    "user": "root",
    "passwd": "my secret password",
    "db": "write-math",
}
preprocessing_queue = [
    preprocessing.scale_and_center,
    preprocessing.dot_reduction,
    preprocessing.connect_lines,
]
use_anonymous = True

#!/usr/bin/env python
import databaseconfig as cfg

connect(cfg.mysql["host"], cfg.mysql["user"],
cfg.mysql["password"])
```

```
# Reading gzipp'ed log file
import gzip

if __name__ == '__main__':
    with gzip.open('us-log1.log.gz') as fh:
        for line in fh:
            print(line)
```

```
# JSON data file, config.json

{
    "mysql":{
        "host":"localhost",
        "user":"root",
        "passwd":"my secret password",
        "db":"write-math"
    },
    "other":{
        "preprocessing_queue":[
            "preprocessing.scale_and_center",
            "preprocessing.dot_reduction",
            "preprocessing.connect_lines"
            ],
        "use_anonymous":true
    }
}

# Reading the file
import json

with open("config.json") as json_data_file:
    data = json.load(json_data_file)
print(data)

# Writing the file
import json

with open("config.json", "w") as outfile:
    json.dump(data, outfile)
```

```
# YAML data file, config.yml
mysql:
    host: localhost
    user: root
    passwd: my secret password
    db: write-math
other:
    preprocessing_queue:
        - preprocessing.scale_and_center
        - preprocessing.dot_reduction
        - preprocessing.connect_lines
    use_anonymous: yes


# config1.yml
document: 1
name: 'erik'
---
document: 2
name: 'config'

..Hos to read file with above contents..
..(previous lines stripped)
docs = yaml.safe_load_all(file)
for doc in docs:
        print(doc)
O/P:
{'document': 1, 'name': 'erik'}
{'document': 2, 'name': 'config'}
```

```
# YAML data file, config.yml

# Reading the file
import yaml

with open("config.yml", "r") as ymlfile:
    cfg = yaml.safe_load(ymlfile)

for section in cfg:
   print(section)
   print(cfg["mysql"])
   print(cfg["other"])


O/P:
other
mysql
{
    "passwd": "my secret password",
    "host": "localhost",
    "db": "write-math",
    "user": "root",
}
{
    "preprocessing_queue": [
        "preprocessing.scale_and_center",
        "preprocessing.dot_reduction",
        "preprocessing.connect_lines",
    ],
    "use_anonymous": True,
}

## Writing to a file in yml format
with open("config.yml", "r") as ymlfile:
    data = yaml.safe_load(ymlfile)

with open('config.out.yml','w',encoding='utf-8') as of:
    yaml.dump(data, of, default_flow_style=False, allow_unicode=True)
```

```
# CSV file toolhire.csv
ItemID,Name,Description,Owner,Borrower,DateLent,DateReturned
1,LawnMower,Small Hover mower,Fred,Joe,4/1/2012,4/26/2012
2,LawnMower,Ride-on mower,Mike,Anne,9/5/2012,1/5/2013
3,Bike,BMX bike,Joe,Rob,7/3/2013,7/22/2013
4,Drill,Heavy duty hammer,Rob,Fred,11/19/2013,11/29/2013
5,Scarifier,"Quality, stainless steel",Anne,Mike,12/5/2013,
6,Sprinkler,Cheap but effective,Fred,,,

# Reading the file using list
import csv
with open ('toolhire.csv' ) as th:
  toolreader = csv. reader (th)
  print (list (toolreader))

O/P: (note, we lost the double quotes):
[['ItemID', 'Name', 'Description', 'Owner', 'Borrower',
DateLent', 'DateReturned'],
```

```
# Writing the file using list
# writer.wroiterow() returns the no of characters
written , ignore that.
# note that we got the double quote back

import csv
items = [   # this is a list of lists
['2','Lawnmower','Ride-on
mower','Mike','$370','Fair','2012-04-01'],
['3','Bike','BMX bike','Joe','$200','Good','2013-03-
22'],
['4','Drill','Heavy duty
hammer','Rob','$100','Good','2013-10-28'],

with open('tooldesc.csv','w', newline='') as tooldata:
  toolwriter = csv.writer(tooldata)
  for item in items:
    toolwriter.writerow(item)
```

```
# Reading the csv file using Dict
with open('tooldesc.csv') as th:
  rdr = csv.DictReader(th)

for item in rdr: print(item)

O/P:
{'DateReturned': '4/26/2012', 'Description':
'Small Hover mower', 'Owner': 'Fred', 'ItemID':
'1', 'DateLent': '4/1/2012',
'Name': 'LawnMower', 'Borrower': 'Joe'}
{'DateReturned': '1/5/2013', 'Description':
'Ride-on mower', 'Owner': 'Mike', 'ItemID': '2',
'DateLent': '9/5/2012',
'Name': 'LawnMower', 'Borrower': 'Anne'}
```

```
# Adding Label to csv file
import csv
fields = ['ItemID', 'Name', 'Description', 'Owner', 'Price',
'Condition', 'DateRegistered']

with open('tooldesc2.csv') as td_in:
  rdr = csv.DictReader(td_in, fieldnames = fields)
  items = [item for item in rdr]


with open('tooldesc3.csv', 'w', newline='') as td_out:
  wrt = csv.DictWriter(td_out, fieldnames=fields)
  wrt.writeheader()
  wrt.writerows(items)
```

```
# Find all items rented by Fred

with open('toolhire.csv') as th:
  rdr = csv.DictReader(th)
  items = [item for item in rdr]

[item['Name'] for item in items if item['Owner'] ==
'Fred']

O/P:
['LawnMower', 'Sprinkler']

# example: reading from tab delimited csv
..
try:
  with open(fname) as fh:
    reader = csv.reader(fh, dialect=csv.excel_tab)
    header = reader.next()
    data = [row for row in reader]
except csv.Error as e:
  print("blah ..")
  sys.exit(-1)

if header:
  print(header)
for datarow in data:
  print(datarow)
```

```
# Reformat Date and write to csv file

import csv
from datetime import datetime

def convertDate(item):
    theDate = item[-1]
    dateObj = datetime.strptime(theDate,'%Y-%m-%d')
    dateStr = datetime.strftime(dateObj,'%m/%d/%Y')
    item[-1] = dateStr
return item


with open('tooldesc.csv') as td:
    rdr = csv.reader(td)
    items = list(rdr)

items = [convertDate(item) for item in items]

with open('tooldesc2.csv', 'w', newline='') as td:
    wrt = csv.writer(td)
    for item in items:
      wrt.writerow(item)
```

```
# ConfigParser

[DEFAULT]
Option1=value1

[SECTION1]
Option2=value2
Option3=value3

[SECTION2]
Option4=value4

import configparser as cp
conf = cp.ConfigParser()


conf['DEFAULT'] = {'lending_period' : 0, 'max_value' : 0}

conf['Fred'] = {'max_value' : 200} # Fred's a bit rough with
things!

conf['Anne'] = {'lending_period' : 30} # She is a bit forgetful
sometimes


with open('toolhire.ini', 'w') as toolhire:
  conf.write(toolhire)

del(conf) # get rid of the old one

conf = cp.ConfigParser()
conf.read('toolhire.ini')
['toolhire.ini']



conf.sections()
['Fred', 'Anne']


conf['DEFAULT']['max_value'] '0'
conf['Anne']['max_value'] '0'
conf['Anne']['lending_period']


conf['Fred']['max_value'] '200'
conf.options('DEFAULT') Traceback (most recent call last):


File "<interactive input>", line 1, in <module>
File "C:\Python33\lib\configparser.py", line 667, in options
raise NoSectionError(section) configparser.NoSectionError: No
section: 'DEFAULT'

conf.defaults()
OrderedDict([('lending_period', '0'), ('max_value', '0')])
```

```
# My ref code on config parser

import configparser as cp

def read_config(apptype, file=CFGFILE):

conf = cp.ConfigParser()
if conf.read(file):
  defconfig = {x:y for x,y in conf.items('default')}
  appconfig = {k:v for k,v in conf.items(apptype)}
  defconfig.update(appconfig)
  netconfig = defconfig

return netconfig


# Reading from stdin
```

```
# fixed-width-file.log
207152670 3984356804116 9532
207152671 3984356804117 9533

import struct

datafile = 'fixed-width-file.log'
mask = '9s14s5s'

with open(datafile) as f: # not working
  for line in f:
    fields = struct.Struct(mask).unpack_from(line)
    print("fields:", [field.strip() for field in fields])

# fields = struct.unpack_from(mask, line)
```

```
# Reading from stdin
```

EMPTY BLOCK	EMPTY BLOCK