Homework Solutions 2 Lab 11

hw11_1.py

```
#!/usr/bin/env python
"""Makes an ascii chart."""

START = 32
END = 126

def GiveAscii(start=START, end=END, width=4):
    """Returns an ascii chart as a string.  Readable."""
    entries = end - start + 1
    entries_per_column = entries/width
    if entries % width:
        entries_per_column += 1
    ret = []
    for row in range(entries_per_column):
        for column in range(width):
            entry = entries_per_column * column + row + start
            if entry > end:
                break
            ret += ["%3d = %-6s" % (entry, chr(entry))]
        ret += ['\n']
    return ''.join(ret)

def GiveAscii(start=START, end=END, width=4):
    "Returns an ascii chart as a string.  NOT readable."
    entries = end - start + 1
    entries_per_column = entries/width + (1 if entries % width else 0)
    entries = [[entries_per_column * column + row + start \
                for column in range(width)] \
               for row in range(entries_per_column)]
    return '\n'.join(['''.join(['%3d = %-6s' % (e, chr(e))\
                                for e in entries[r] if e <= end])\
                      for r in range(entries_per_column)])

def main():
    print GiveAscii()

if __name__ == '__main__':
    main()
"""
$ ./hw11_1.py
 32 =        56 = 8      80 = P     104 = h
 33 = !      57 = 9      81 = Q     105 = i
```

```
43  34 = "      58 = :      82 = R     106 = j
44  35 = #      59 = ;      83 = S     107 = k
45  36 = $      60 = <      84 = T     108 = l
46  37 = %      61 = =      85 = U     109 = m
47  38 = &      62 = >      86 = V     110 = n
48  39 = ’      63 = ?      87 = W     111 = o
49  40 = (      64 = @      88 = X     112 = p
50  41 = )      65 = A      89 = Y     113 = q
51  42 = *      66 = B      90 = Z     114 = r
52  43 = +      67 = C      91 = [     115 = s
53  44 = ,      68 = D      92 = \     116 = t
54  45 = -      69 = E      93 = ]     117 = u
55  46 = .      70 = F      94 = ^     118 = v
56  47 = /      71 = G      95 = _     119 = w
57  48 = 0      72 = H      96 = ‘     120 = x
58  49 = 1      73 = I      97 = a     121 = y
59  50 = 2      74 = J      98 = b     122 = z
60  51 = 3      75 = K      99 = c     123 = {
61  52 = 4      76 = L     100 = d     124 = |
62  53 = 5      77 = M     101 = e     125 = }
63  54 = 6      78 = N     102 = f     126 = ~
64  55 = 7      79 = O     103 = g
65 $"""
```

hw11_2.py

```
#!/usr/bin/env python
"""Provides Palindromize(phrase)"""

import string

table = ''.join([chr(i) for i in range(256)])
delete_chars = string.punctuation + string.whitespace

def Palindromize(phrase):
    """Returns lowercase version of the phrase with whitespace and
    punctuation removed if the phrase is a palindrome.  If not, it
    returns None."""

    phrase = str(phrase).lower().translate(table, delete_chars)
    half_len = len(phrase)/2
    if half_len <= 1:
        return None
    for i, ch in enumerate(phrase[:half_len]):
        if ch != phrase[-(i+1)]:
            return None
    return phrase

def main():
    DATA = ('Murder for a jar of red rum', 12321,
            'nope', 'abcbA', 3443, 'what',
            'Never odd or even', 'Rats live on no evil star')
    for phrase in DATA:
        answer = Palindromize(phrase)
        if answer:
            print "%s -> %s" % (phrase, answer)

if __name__ == '__main__':
    main()

"""
$ hw11_2.py
./hw11_2.py
Murder for a jar of red rum -> murderforajarofredrum
12321 -> 12321
abcbA -> abcba
3443 -> 3443
Never odd or even -> neveroddoreven
Rats live on no evil star -> ratsliveonnoevilstar
$"""
```

hw11_3.py

```
#!/usr/bin/env python
"""Palindrom searcher.  Call with:

hw11_3.py starting_dir
"""

import os
import sys

if __name__ == '__main__':
    sys.path.insert(0, "..")
else:
    sys.path.insert(0, os.path.join(os.path.split(__file__)[0], '..'))

import utils.hw11_2 as palindromizer

STARTING_DIR = '/home/marilyn/python/mm/labs/lab_11_os_Module'

def _FindPalindromes(found_d, dir_, files):
    for file_name in files:
        whole_path = os.path.join(dir_, file_name)
        if os.path.isdir(whole_path):
            continue
        for line in open(whole_path):
            for word in line.split():
                answer = palindromizer.Palindromize(word)
                if not answer:
                    continue
                if answer in found_d:
                    found_d[answer] += [whole_path]
                else:
                    found_d[answer] = [whole_path]

def ReportPalindromes(starting_dir):
    """
    Reports the one-word palindromes in all the files in the
    directory structure starting at starting_dir.
    """
    palindromes_d = {}
    os.path.walk(starting_dir, _FindPalindromes, palindromes_d)
    for each in palindromes_d:
        print "%s in %d files" % (each, len(palindromes_d[each]))

def main():
```

```
45     try:
46         starting_dir = sys.argv[1]
47     except IndexError:
48         starting_dir = STARTING_DIR
49     ReportPalindromes(starting_dir)
50 
51 if __name__ == '__main__':
52     main()
53 
54 """
55 $ hw11_3.py
56 abcba in 13 files
57 stats in 21 files
58 murderforajarofredrum in 1 files
59 3443 in 13 files
60 12321 in 13 files
61 neveroddoreven in 1 files
62 ratsliveonnoevilstar in 1 files
63 $ """
```