Week 6: Boggle!

Practice? Cheat?
Super-human boggle player

Boggle solver for ...
Recursion ... depth-first search
Modules
(and a first glance at classes)

Recursion - Wikipedia, the free encyclopedia
Recursion is the process of repeating items in a self-similar way. For instance, when the surfaces of two mirrors are exactly parallel with each other the nested ...
Seems like we’re missing a base case here ... and the progress case doesn’t seem to be making the problem smaller.
$ python3 boggler.py "oydliexennoktati" dict.txt

ament  nine
annex  not
ant    oat
anti   one
atone  oxen
den    tan
dent   tanned
dye    tat
eon    tike
eon    toe
eon    toed
eon    toke
eon    ton
eon    tone
ktion
ion
iota
kit
led
leonine
nation
neon

Press enter to end
Let’s see it
Let's look at a simpler example ...

```
APPAPA
SVAR
XUTMN
NKMP
```

We can explore all 8 directions ...

```
APPAPA
SVR
XUTMN
NKMP
```
Depth-first search logic

Given a position and a prefix ...
  If the position is off the board, do nothing
  If the position is already in use, do nothing
  New prefix = prefix + tile
  If it’s a complete word, note it
  If it’s a valid prefix
    Mark current tile as “in use”
    Recursively search in all 8 directions
    Unmark current tile before returning
Remove duplicates ... how?

Two ways
1) Python “set” data structure
2) Sort, scan

```python
$ python3 boggler.py "oydliexennoktati" dict.txt
anent
annex
ant
anti
atone
den
dent
dye
eon
ikon
inane
inn
into
ion
iota
kit
led
leonnine
nation
neon

nine
not
oat
one
oxen
tan
tanned
tat
tike
toe
toed
toke
ton
tone
yen
yin
```

Press enter to end

---

**boggler.py**

```python
# Boggle solver.
# Usage: python3 boggler.py "board" dict.txt
# where "board" is 16 characters of board, in left-to-right reading order
# and dict.txt can be any file containing a list of words in alphabetical order

Author: Michal Young, michal@cs.uoregon.edu 2012.10.26

```python
from boggle_board import BoggleBoard
```

In addition to boggler.py, you write game_dict.py

```python
import game_dict  # Dictionary of legal game words
import sys  # for command line arguments: board, dictionary
```

---

**MODULES**
game_dict.py

dict = [ ]

# Codes for result of search
WORD = 1
PREFIX = 2
NO_MATCH = 0

def read( filename, min_length=3 ):
    ...
def search( str ):
    ...

---

dict.search( str ):

"""
Search for a prefix string in the dictionary.
Args:
    str: A string to look for in the dictionary
Returns:
    code WORD if str exactly matches a word in the dictionary,
    PREFIX if str does not match a word exactly but is a prefix
    of a word in the dictionary, or
    NO_MATCH if str is not a prefix of any word
    in the dictionary
"""

(You can use a binary search or a linear search)

---

in boggler.py

...
match = game_dict.search(prefix)
if match == game_dict.NO_MATCH:
    return
...

---

Summary: Boggle solver

Depth first search, again
Break into modules

Dictionary (game_dict.py)

class BoggleBoard (boggle_board.py)
    (which uses grid.py for display,
     which uses graphics.py for display)

Solver (boggler.py)
    (main program uses dictionary
     and board)