Think Python 2e, Chapter 9 Notes

Case study: word play

September 26, 2022

Words

- The exercises for this chapter need a list of 113,783 English words, `words.txt`
- Available from:
  - The lecture folder for this chapter on github.
  - `http://thinkpython2.com/code/words.txt`

```
aa
aah
aahed
aahing
aahs
aal
aalii
aaliis
aals
aardvark
...
```

```
...
zymogene
zymogenes
zymogens
zymologies
zymology
zymoses
zymosis
zymotic
zymurgies
zymurgy
```

Process a file line by line

```
aa
aah
aahed
aahing
aahs
aal
aalii
aaliis
aals
aardvark
...
```

```
>>> fin = open('words.txt')
>>> fin.readline()
'aa\n'
>>> fin.readline()
'aah\n'
>>> fin.readline().strip()
'aahed'
```

Process a file line by line

```
aa
aah
aahed
aahing
aahs
aal
aalii
aaliis
aals
aardvark
...
```

```
fin = open('words.txt')
for line in fin:
    word = line.strip()
    print(word)
```

Process a file line by line

```
aa
aah
aahed
aahing
aahs
aal
aalii
aaliis
aals
aardvark
...
```

```
fin = open('words.txt')
for line in fin:
    word = line.strip()
    print(word)
```

```
fin = open('words.txt')
for line in fin:
    word = line.strip()
    if len(word) > 20:
        print(word)
```

Process a file line by line

```
aa
aah
aahed
aahing
aahs
aal
aalii
aaliis
aals
aardvark
...
```

```
fin = open('words.txt')
for line in fin:
    word = line.strip()
    print(word)
```

```
fin = open('words.txt')
for line in fin:
    word = line.strip()
    if len(word) > 20:
        print(word)
```

See text exercises

Reduction to a previously solved problem

```
1 def uses_only(word, available):
2     for letter in word:
3         if letter not in available:
4             return False
5     return True
```

```
1 def uses_all(word, required):
2     for letter in required:
3         if letter not in word:
4             return False
5     return True
```

Reduction to a previously solved problem

```
def uses_only(word, available):
    for letter in word:
        if letter not in available:
            return False
    return True
```

```
def uses_all(word, required):
    for letter in required:
        if letter not in word:
            return False
    return True
```

```
def uses_all(word, required):
    return uses_only(required, word)
```

is_abecedarian

Use a `for` loop

is_abecedarian

Use a for loop

```
1 def is_abecedarian(word):
2     previous = word[0]
3     for c in word:
4         if c < previous:
5             return False
6         previous = c
7     return True
```

is_abecedarian

Use a for loop

```
def is_abecedarian(word):
    previous = word[0]
    for c in word:
        if c < previous:
            return False
        previous = c
    return True
```

for with range

```
def is_abecedarian(word):
    for i in range(len(word)-1):
        if word[i+1] < word[i]:
            return False
    return True
```

is_abecedarian

Use recursion

is_abecedarian

Use recursion

```
1 def is_abecedarian(word):
2     if len(word) <= 1:
3         return True
4     if word[0] > word[1]:
5         return False
6     return is_abecedarian(word[1:])
```

is_abecedarian

Use `while` loop

is_abecedarian

Use while loop

```
1 def is_abecedarian(word):
2     i = 0
3     while i < len(word)-1:
4         if word[i+1] < word[i]:
5             return False
6         i = i+1
7     return True
```

is_palindrome

```
1  def is_palindrome(word):
2      i = 0
3      j = len(word)-1
4
5      while i<j:
6          if word[i] != word[j]:
7              return False
8          i = i+1
9          j = j-1
10
11     return True
```

is_palindrome

```
1  def is_palindrome(word):
2      i = 0
3      j = len(word)-1
4
5      while i<j:
6          if word[i] != word[j]:
7              return False
8          i = i+1
9          j = j-1
10
11     return True
```

Reduced to a previous problem

```
1  def is_palindrome(word):
2      return is_reverse(word, word)
```

Special cases

Testing `has_no_e`

- Test words with e
- Test words with no e
- Test words with e at the beginning
- Test words with e at the end
- Test words with e in the middle
- Test very long words
- Test very short words, like e
- Test very short words, like the empty string

Testing

Program testing can be used to show the presence of bugs, but never to show their absence!

— *Edsger W. Dijkstra*

Vocabulary

file object: A value that represents an open file.

reduction to a previously solved problem: A way of solving a problem by expressing it as an instance of a previously solved problem.

special case: A test case that is atypical or non-obvious (and less likely to be handled correctly).