Recursion on non-numerics

- How could we check whether a string of characters is a palindrome, i.e., reads the same forwards and backwards
  - “Able was I ere I saw Elba” – attributed to Napolean
  - “Are we not drawn onward, we few, drawn onward to new era?”

How to we solve this recursive?

- First, convert the string to just characters, by stripping out punctuation, and converting upper case to lower case
- Then
  - Base case: a string of length 0 or 1 is a palindrome
  - Recursive case:
    - If first character matches last character, then is a palindrome if middle section is a palindrome

Example

- ‘Able was I ere I saw Elba’ → ‘ablewasiereisawleba’
- isPalindrome(‘ablewasiereisawleba’) is same as
  - ‘a’ == ‘a’ and isPalindrome(‘blewasiereisawleb’)

```python
def isPalindrome(s):

    def toChars(s):
        s = s.lower()
        ans = ''
        for c in s:
            if c in 'abcdefghijklmnopqrstuvwxyz':
                ans = ans + c
        return ans

    def isPal(s):
        if len(s) <= 1:
            return True
        else:
            return s[0] == s[-1] and isPal(s[1:-1])

    return isPal(toChars(s))
```

Divide and conquer

- This is an example of a “divide and conquer” algorithm
  - Solve a hard problem by breaking it into a set of sub-problems such that:
    - Sub-problems are easier to solve than the original
    - Solutions of the sub-problems can be combined to solve the original