Code for Teachers

A practical approach to programming

Chapter 5:
The Palindrome Detector

Lesson Objectives

- Understanding palindromes
- Creating command-line tools
- Handling user input
- Adding functionality

Lesson Design

- Discuss palindromes
- Make up some!
- Write out the procedure (in English) how to detect palindromes by hand
- Identify pieces of the procedure that can easily be translated into Python
- Get ready to code!

Palindromes

- Phrases that are the same when reversed (ignoring spaces)
  - Race car
  - So many dynamos
  - 1001
  - Taco cat

Detecting Palindromes, Human-Style

Race Car

racecar

r aceca r

r a cec a r

r a c e c a r

r a c e c a r

If at any point the ends don’t match, the test fails

Function Design

- `isPal(test) -> bool`
  - Takes a string `test` and returns `True` or `False`
- Two approaches to this function: iterative and recursive
  - Iterative uses a loop
  - Recursive calls itself
  - For now, iterative

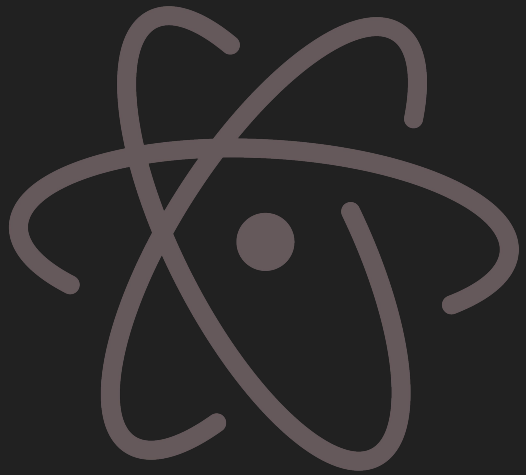
Iterative Approach

- For the length of the string, starting at `i = 0`
  - Check `i` against `i` in the reverse string
    - If they match, keep going
    - If `False`, return `False`
      - Otherwise, do nothing
- If we get to the end of the loop, return `True`

An application vs. a program

- We shouldn’t have to change variables every time we want to test a different string
- In fact, we should have to open the code at all.
- Programmers write *user interfaces* for their code so others can use it
- Most UIs are graphical these days (GUIs)
- But before GUIs, there were command-line interfaces (CLIs)

New Tools: A proper text editor



https://atom.io/

https://sublimetext.com/

https://code.visualstudio.com/

The Command Line

- Text-based interface to run programs and navigate files/folders
- Different operating systems use different command-line interfaces
  - Windows: cmd.exe
  - macOS / Linux: Terminal
- We’re going to use a Unix-like terminal for our purposes
  - If you’re on Mac, that’s Terminal!
  - If you’re on Windows, try using Cygwin

Bonus! Recursive Version

- Instead of a loop, a recursive function calls itself with varying arguments
  - When the function finds a *base case*, it returns something
  - A *base case* is the smallest possible reduction of the information the function is working on
  - Recursion, besides being efficient, is cool because it demonstrates a powerful programming concept:
    - Take a big problem and break it down into smaller, easy-to-solve problems

Recursive function design

- Identify base cases
  - What are the simplest possible palindromes?
  - A single-character string is always a palindrome
  - A two-character string `s` is a palindrome when `s[0] == s[1]`
- Check the arguments against the base cases
  - If they match, return the appropriate value
- Otherwise, check the first character against the last
  - If they match, send the interior characters back around into the same function
  - Otherwise, return `False`

Recursion

racecar

r aceca r

r a cec a r

r a c e c a r

r a c e c a r

is_pal("racecar")

first/last match, but not base case: is_pal("aceca")

first/last match, but not base case: is_pal("cec")

first/last match, but not base case: is_pal("e")

"e" is a base case: True

theforeverstudent.com

@mttaggart

mttaggart