Lecture 25: Practice Programming (focus on while-loop)

CS 1110
Introduction to Computing Using Python

[E. Andersen, A. Bracy, D. Fan, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]
Revisit word guessing game

- There is a secret word.
- The user has 10 chances to guess letters until the word has been spelled out.

- We implemented a class SecretWord to keep track of both the word being guessed and what the user sees / has guessed so far.

Play the game.
How does the game go?

word_list = [ ... candidate words for user to guess ... ]

N_CHANCES = 10

*Set the secret word*

User guesses

until no more guesses or *secret is solved*

*Reveal the word*

Let’s solve the problem with a while-loop this time (instead of recursion)!
Setting up a **while-loop**

0. Situation is to do something until an event happens

1. Write the **continuation condition**
   - Create var names as necessary to express condition
   - May be easier to negate stop condition to get continuation condition

2. **Initialize loop vars** (vars in loop condition) as necessary

3. In loop body: **update loop vars**
   - to possibly change loop condition from True to False

4. Write the rest of the loop body
Start next video:

Use while-loop get and check user input
Get and check user input with **while-loop**

- User may not enter appropriate input
- Can use **assert** and error out if user provides inappropriate input—not friendly
- Can **re-prompt** user for appropriate input
- Re-prompt how many times? Can **re-prompt until user does the right thing**

Indefinite iteration! Use a **while-loop**.
Other changes to word guessing game?

- Allow 6 strikes instead of 10 guesses
  - Change in game module
- Accommodate space and hyphen
  E.g., “ice cream” displayed as ___ _____
  “high-rise” displayed as _____-_____
  - Change in class SecretWord
- Change instance attribute display_word from a string to a list of letters. How about secret_word?

Great opportunity for extra practice! And fun 😊
Start next video:
Search algorithms
(linear search, binary search)
Search Algorithms

- Search for a target $x$ in a list $v$
- Start at index 0, keep checking *until* you find it
  or *until no more element to check*

$$v = [12, 35, 33, 15, 42]$$

$x = 14$

**Linear search**
Search Algorithms

- Search for a target $x$ in a list $v$
- Start at index 0, keep checking until you find it or until no more elements to check

Linear search

- Search for a target $x$ in a *sorted* list $v$

Searching in a sorted list should require less work!

Binary search
How do you search for a word in a dictionary? (NOT linear search)

To find the word “tanto” in my Spanish dictionary...

while dictionary is longer than 1 page:
   Open to the middle page
   if first entry comes before “tanto”:
      Rip* and throw away the 1st half
   else:
      Rip* and throw away the 2nd half

* For dramatic effect only--don’t actually rip your dictionary! Just pretend that the part is gone.
Repeated halving of “search window”

Original: 3000 pages
After 1 halving: 1500 pages
After 2 halvings: 750 pages
After 3 halvings: 375 pages
After 4 halvings: 188 pages
After 5 halvings: 94 pages

After 12 halvings: 1 page
Binary Search

• Repeatedly halve the “search window”
• An item in a sorted list of length $n$ can be located with just $\log_2 n$ comparisons.
• “Savings” is significant!

<table>
<thead>
<tr>
<th>n</th>
<th>$\log_2 (n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td>10000</td>
<td>13</td>
</tr>
</tbody>
</table>
**Binary Search: target** \(x = 70\)

\[
v[\text{mid}] \text{ is not } x \\
v[\text{mid}] < x \\
\text{So throw away the left half...}
\]
Binary Search: target $x = 70$

- $v$ = [12, 15, 33, 35, 42, 45, 51, 62, 73, 75, 86, 98]
- $i$: 6
- mid: 8
- $v[\text{mid}]$ is not $x$
  - $x < v[\text{mid}]$
- j: 11
- So throw away the right half...
Binary Search: target $x = 70$

$v_{[6]}$ is not $x$
$v_{[mid]} < x$

So throw away the left half...
Binary Search: target $x = 70$

$0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11$

$v$

12 15 33 35 42 45 51 62 73 75 86 98

$v[\text{mid}]$ is not $x$
$v[\text{mid}] < x$

So throw away the left half...
Binary Search: \textit{target }x = 70

\begin{tabular}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\hline
12 & 15 & 33 & 35 & 42 & 45 & 51 & 62 & 73 & 75 & 86 & 98 \\
\end{tabular}

\begin{itemize}
\item \textbf{i:} 8
\item \textbf{mid:} 7
\item \textbf{j:} 7
\end{itemize}

\textbf{DONE} because \\
i no longer less than j \\
\rightarrow \text{no valid search window}