http://www.cs.cornell.edu/courses/cs1110/2020sp

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.

2

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)!

3

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

4

Start next video:

Use while-loop get and check user input

5

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.

6

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 ☺

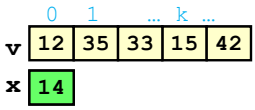
7

Start next video:
Search algorithms
(linear search, binary search)

8

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*

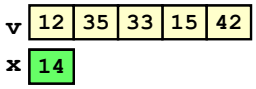


Linear search

10

Search Algorithms

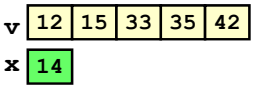
- Search for a target x in a list v
- Start at index 0, keep checking *until* you find it or *until no more elements to check*



Linear search

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

Searching in a sorted list should require less work!



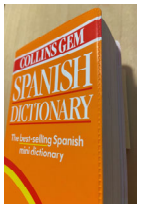
Binary search

11

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.

12

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
```

13

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!

n	log2(n)
100	7
1000	10
10000	13

14

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

i: 0
mid: 5
j: 11

v[mid] is not x
v[mid] < x

So throw away the left half…

15

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

i: 6
mid: 8
j: 11

v[mid] is not x
x < v[mid]

So throw away the right half…

16

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

i: 6
mid: 6
j: 7

v[mid] is not x
v[mid] < x

So throw away the left half…

17

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

i: 7
mid: 7
j: 7

v[mid] is not x
v[mid] < x

So throw away the left half…

18

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

i: 8
mid: 7
j: 7

DONE because
i no longer less than j
→ no valid search window

19