

## Objectives

- **str** methods

Feb 18, 2011

Sprenkle - CSCI111

1

## Review

- How does the computer represent information?
  - Numeric?
  - String?
- How do we figure out a character's ASCII value?
- How do we figure out the character represented by a number?

Feb 18, 2011

Sprenkle - CSCI111

2

## Review: Translating to/from ASCII

- Translate a character into its ASCII numeric code using **built-in function ord**
  - `ord('a') ==> 97`
- Translate an ASCII numeric code into its character using **built-in function chr**
  - `chr(97) ==> 'a'`

Feb 18, 2011

Sprenkle - CSCI111

`ascii_table.py`  
`ascii.py`

3

## USING THE STR API

Feb 18, 2011

Sprenkle - CSCI111

4

## str Methods

- **str** is a **class** or a **type**
- **Methods**: available operations to perform on **str** objects
  - Provide common functionality
- To see all methods available for **str** class
  - `help(str)`

Feb 18, 2011

Sprenkle - CSCI111

5

## str Methods

- Example method: **find(substring)**
  - Finds the index where substring is in string
  - Returns -1 if substring isn't found
- To call a method:
  - `<stringobj>.methodname([arguments])`
  - Example: `filename.find(".py")`

Executed on this string

Feb 18, 2011

Sprenkle - CSCI111

6

## Common str Methods

Method	Operation
center(width)	Returns a copy of string centered within the given number of columns
count(sub[, start [, end]])	Return # of non-overlapping occurrences of substring <b>sub</b> in the string.
endswith(sub), startswith(sub)	Return <b>True</b> iff string ends with/begins with <b>sub</b>
find(sub[, start [, end]])	Return first index where substring <b>sub</b> is found
isalpha(), isdigit(), isspace()	Returns <b>True</b> iff string contains letters/digits/whitespace only
lower(), upper()	Return a copy of string converted to lowercase/uppercase

Feb 18, 2011

Sprenkle - CSCI1111 string\_methods.py 7

## Common str Methods

Method	Operation
replace(old, new[, count])	Returns a copy of string with all occurrences of substring <b>old</b> replaced by substring <b>new</b> . If <b>count</b> given, only replaces first <b>count</b> instances.
split([sep])	Return a list of the words in the string, using <b>sep</b> as the delimiter string. If <b>sep</b> is not specified or is None, any whitespace string is a separator.
strip()	Return a copy of the string with the leading and trailing whitespace removed
join(<sequence>)	Return a string which is the concatenation of the strings in the sequence with the string this is called on as the separator
swapcase()	Return a copy of the string with uppercase characters converted to lowercase and vice versa.

Feb 18, 2011

Sprenkle - CSCI1111

8

## Functions vs Methods

### Functions

- All "input" as arguments/parameters
- Example: **len** is a built-in function
  - Called as **len(string)**

### Methods

- "Input" are argument/parameters **and** the string the method was called on
- Example:
  - **string.upper()**

Feb 18, 2011

Sprenkle - CSCI1111

9

## Are You Smarter Than a 5th Grader?

- Problem in spelling from the show: How many a's are in abracadabra?
  - Solve using **str** methods

Feb 18, 2011

Sprenkle - CSCI1111

10

## Get the Username

- Given the directory formatted as
  - **dir = "/home/www/users/username/"**
- Get the username out

Feb 18, 2011

Sprenkle - CSCI1111

11

## Using str Methods

- Modify **binaryToDecimal.py** to verify that the entered string contains only numbers
  - Keep asking them for a number until the string contains only numbers

Feb 18, 2011

Sprenkle - CSCI1111

12

## Using str Methods

- Modify `binaryToDecimal.py` to verify that the entered string contains only numbers
  - Keep asking them for a number until the string contains only numbers
- 2nd modification: How could we make sure that entered string contains only 0s and 1s?

Feb 18, 2011

Sprenkle - CSCI111

13

## Implementing Wheel of Fortune

- Simplifications: no money, no buying vowels, no keeping track of previous guesses, one player
- Functionality
  - Displaying puzzle appropriately
  - Gets guesses from user
    - Either letters or solve the puzzle
  - Keep track of the number of guesses
  - Displays puzzle with guesses filled in
- Think about ...
  - What do we need to model? How would we model it?
  - User input robustness?
  - Any special cases?

[wheeloffortune.py](#)

Feb 18, 2011

Sprenkle - CSCI111

14

## Implementing Wheel of Fortune

- Differences between real and simulated game
  - Players type in letter rather than say it
    - Case matters
    - What if enter more than one letter

Feb 18, 2011

Sprenkle - CSCI111

15

## Implementing Wheel of Fortune

- User input verification
  - How can we ensure that the user typed only one letter?
  - How can we ensure that the user typed a *letter*?
- Checking the guess
  - How can we tell if the guessed letter is in the puzzle?
  - How can report the number of times the guessed letter occurs in the puzzle?

Feb 18, 2011

Sprenkle - CSCI111

16

## Implementing Wheel of Fortune

- How many times should we prompt the user for a guess?
- How can we display the current puzzle?
  - What does the puzzle look like when we start the game?
  - What does it look like after we correctly guess a letter?

Feb 18, 2011

Sprenkle - CSCI111

17

## Wheel of Fortune

- Practice: Modify displayed puzzle to handle punctuation
  - Include punctuation in displayed puzzle
  - Original code:

```
displayedpuzzle = ""
for char in PHRASE:
    if char != " ":
        displayedpuzzle += "_"
    else:
        displayedpuzzle += " "
```

Feb 18, 2011

Sprenkle - CSCI111

18

## Broader Issues in Computer Science

- Testing isn't a broader issue
  - Glad you noticed lots of the issues with testing
  - We'll keep talking about testing because I love it!

Google Maps

Will  
Yates  
Callie  
Jean Paul  
Anh  
Minh

Excel

Ola  
Nick

Feb 18, 2011

Sprenkle - CSCI111

19

## Broader Issues in Computer Science

- Is the Excel 2007 or Google Maps bug a “reasonable” bug?
  - Why weren't they caught?
  - Should they have been caught?
- When should a company stop testing?
  - When do *you* stop testing?
- Why doesn't software have guarantees?

Feb 18, 2011

Sprenkle - CSCI111

20

## Broader Issues in Computer Science

- Have you ever encountered a bug in a program?
  - What happened?
  - How severe was the problem? Were you able to recover?
  - How did you respond? (Angry? Didn't think about? ...)
- If people can recover from a bug, when does it become important for software developers to fix the problem?
  - Tradeoffs between costs/revenues of implementing new features versus fixing existing code
  - What matters to you (as a consumer) more?

Feb 18, 2011

Sprenkle - CSCI111

21

## Notes from a Keynote Speech about Testing Microsoft Vista

- Users are “trained” to not use buggy features
  - After user encounters a certain bug when doing something enough times, the user stops trying to do that buggy activity

$$\text{User's Loss in Confidence} = \text{Disruption Frequency} \times \begin{matrix} \text{Recovery Time} \\ \text{Recover Effort} \\ \text{Lost data} \\ \text{Uncertainty} \end{matrix}$$

- Only ship fixes that affect many users

Feb 18, 2011

Sprenkle - CSCI111

22

## Relation to Our Class

- When do *you* stop testing?

Feb 18, 2011

Sprenkle - CSCI111

23

## Status from Official Excel Blog

- Post on 9/25/07 Happy Ending
  - We've come up with a fix for this issue and are in the final phases of a broad test pass in order to ensure that the fix works and *doesn't introduce any additional issues* - especially any other calculation issues. This fix then needs to make its way through our official build lab and onto a download site - which we expect to happen very soon.
- Post on 10/9/07
  - As of today, fixes for this issue in Excel 2007 and Excel Services 2007 are available for download ...
  - We are in the process of adding this fix to Microsoft Update so that it will get *automatically pushed* to users running Excel 2007 or Excel Services 2007. Additionally, the fix will also be contained in the first service pack of Office 2007 when it is released (the release date for SP1 of Office 2007 has not been finalized).

Feb 18, 2011

Sprenkle - CSCI111

24