

Objectives

- Finishing up string operations
- String Methods

Feb 15, 2016

Sprenkle - CSC111

1

Review: String Operations

- How do we represent text?
- How can we represent really long text?
- How can we combine strings?
- How can we combine strings multiple times?
- How can we find out how long a string is?
- How can we find out the character at a certain position?
- How can we iterate through a string?

Feb 15, 2016

Sprenkle - CSC111

2

Review: Strings

- Actually a *sequence* of characters

➤ Example:

band = "The Beatles"

'T'	'h'	'e'	' '	'B'	'e'	'a'	't'	'l'	'e'	's'
0	1	2	3	4	5	6	7	8	9	10

characters

Start at 0

index or position of characters

Length of the string: 11
Built-in function: `len(string)`
to find length of a string

End at `len()-1`

Feb 15, 2016

Sprenkle - CSC111

3

Review: Iterating Through a String

- For each character in the string

string of length 1

```
for char in mystring:
    print(char)
```

'T'	'h'	'e'	' '	'B'	'e'	'a'	't'	'l'	'e'	's'
0	1	2	3	4	5	6	7	8	9	10

An integer

```
for pos in range(len(mystring)):
    print(string[pos])
```

- For each position in the string

Index into the string

Feb 15, 2016

Sprenkle - CSC111

4

Review: Substrings Operator []

- Look at a particular character in the string
 - Syntax: `mystr[<integer expression>]`
 - [Positive value]: index of character
 - [Negative value]: count backwards from end
- Look at a sequence of characters in the string
 - Syntax: `mystr[<start>:<end>]`

Feb 15, 2016

Sprenkle - CSC111

5

Review: Testing for Substrings

- Using the `in` operator
 - Used `in` before with `for` loops
- Syntax:


```
substring in string:
```

 - Evaluates to `True` or `False`
- Example:

```
if "cat" in name:
    print(name, "contains 'cat'")
```

Feb 15, 2016

Sprenkle - CSC111

6

Strings are Immutable

You cannot change the value of strings

- For example, you **cannot** change a character in a string

~~`str[0] = 's'`~~

Feb 15, 2016

Sprenkle - CSCI111

7

Revised Pick4 Game

- To play: pick 4 numbers between 0 and 9
- To win: select the numbers that are selected by the magic ping-pong ball machine
- Done previously: Simulate the magic ping-pong ball machines
- Additional Functionality:
 - Determine if the user picks the winning number

Feb 15, 2016

Sprenkle - CSI

pick4winner.py

3

Revised Pick4 Numbers

- Tell the user how many numbers they got right
 - Get prizes for having some numbers right
- Examples:

Pick4 Num	User's Pick	Num Correct
"7737"	"1234"	1
"0204"	"1234"	2
"1234"	"1234"	4

Feb 15, 2016

Sprenkle - CSI

pick4num_places.py

3

USING THE STR API

Feb 15, 2016

Sprenkle - CSCI111

10

Review

- What is an API?
- How do we call methods on an object?

Feb 15, 2016

Sprenkle - CSCI111

11

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 15, 2016

Sprenkle - CSCI111

12

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:
 - `<str_obj>.methodname([arguments])`
 - Example: `filename.find(".py")`

Executed on this string

Feb 15, 2016

Sprenkle - CSC111

13

Common str Methods

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

Feb 15, 2016

Sprenkle - CSC111

string_methods.py

Common str Methods

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

Feb 15, 2016

Sprenkle - CSC111

15

String Methods vs. Functions

Functions

- All input comes from arguments/parameters
- Example: `len` is a built-in function
 - Called as `len(strobj)`

Methods

- Input comes from arguments *and* the string the method was called on
- Example:
 - `strobj.upper()`

Feb 15, 2016

Sprenkle - CSC111

16

Using the APIs

- Given a problem, break down the problem
 - Can any of the parts of the problem be solved using a method in the API?

Feb 16, 2016

Sprenkle - CSC111

17

Are You Smarter Than a 5th Grader?

- Problem in spelling from the show: How many a's are in abracadabra?
 - Solve using `str` methods
 - Silly problem but can generalize to other problems.

Feb 15, 2016

Sprenkle - CSC111

18

Verifying User Input

- How can we verify that the user entered the lottery number in the correct format?

`pick4winner_better_error_handling.py`

Feb 15, 2016

Sprengle - CSC111

19

Software Qualities


- Beyond functionality, what qualities do you like in software?
 - Web included

Feb 15, 2016

Sprengle - CSC111

20

Software Qualities

- Correct
- Efficient
 - Fast
 - Uses little memory
- Reliability
- Stability
- Robustness
- Secure
- Low cost
- Good support
- Usable 
 - Get what I need to do quickly
 - Pretty
 - ...
- Compatibility
- Portable (use anywhere)
- Maintainability

Feb 15, 2016

Sprengle - CSC111

21

Usability


- Want users to *like* to use your software
 - More revenue
 - Develop even better software
- How Apple makes money: best user interfaces → user buys products

Feb 15, 2016

Sprengle - CSC111

22

Usability Goals

- Pretty output, display, layout
 - Easy to read, understand, interpret 
- Clear navigation
- Easy to perform frequent tasks
- Undoability
- Difficult to make irrecoverable errors

Feb 15, 2016

Sprengle - CSC111

23

Escape Sequences

- Escape character: `\`
- Escape sequences
 - newline character (carriage return) → `\n`
 - tab → `\t`
 - quote → `\"` or `\'`
 - backslash → `\\`
- Example:
 - `print("To print a \\, you must use \"\\\\\\\\\\\\\\\\")`
 - What does this display?

Shell demonstration

Feb 15, 2016

Sprengle - CSC111

`demo_str.py`

24

Practice

- Display To print a tab, you must use `\t`.
- Display I said, "How are you?"

`escape_sequence.py`

Feb 15, 2016

Sprengle - CSCI111

25

Problem with print

- By default, `print` puts spaces in the output whenever there is a comma
- Example:

```
x = 13.54
print("You owe $", x, ". ")
```

Displays:

You owe \$ 13.54 .

Extra spaces

Possible solutions?

Feb 15, 2016

Sprengle - CSCI111

26

Solution: using print

- Use `sep` parameter

```
x = 13.54
print("You owe $", x, ".", sep="")
```

Any issues with this solution?

Only works if want that separator for *all* separators

Feb 15, 2016

Sprengle - CSCI111

27

Solution: using str()

- Recall: `str()` is constructor/convertor function to convert other data types to strings

➤ Example: `str(33) → '33'`

- Use constructor with the `+` (i.e., *concatenation*) operator when printing output

➤ Gets rid of extra spaces you don't want.

```
print("You owe $" + str(x) + ".")
```

Feb 15, 2016

Sprengle - CSCI111

28

Another problem with print

```
SALES_TAX=.053 # the sales tax in VA
value = eval(input("How much does your item cost? "))
with_tax = value * (1+SALES_TAX)
print("Your item that cost $", value, end=' ')
print("costs $", with_tax, "with tax.")
```

Feb 15, 2016

Sprengle - CSCI111

`sales_tax.py`

29

FORMATTING STRINGS

Feb 15, 2016

Sprengle - CSCI111

30

Solution: Format Operator & Specifiers

- Formatting operator: **%**
- Format specifiers allow us to control how output is displayed to user
 - Right, left justification
 - Number of decimals to display

Feb 15, 2016 Sprengle - CSC111 31

Formatting Strings

- Syntax is


```
><templatestring> % (<value1>
    <value2>, ..., <valuen>)
```
- Semantics: creates a **formatted string**
 - Means “format the templatestring, using the format(s) specified by **format specifiers** on the corresponding replacement values”
 - Evaluates to a the **str** data type
- Typically used with print statements

Feb 15, 2016 Sprengle - CSC111 32

Formatting Strings

- **templatestring** is a template for the resulting string with format specifiers instead of the values
 - For each format specifier in templatestring, should have a **replacement value**
 - Throws **TypeError** if not enough replacements for specifiers in templatestring
 - If only one replacement value, don't need ()

```
"%.2f" % 3.14159
```

↑ One format specifier in template string ↓ Corresponding replacement value

Evaluates to "3.14"

Feb 15, 2016 Sprengle - CSC111 33

Format Specifiers

[] mean optional

- General format: **%[flags][width][.precision]code**
- flags:
 - 0: zero fills
 - +: adds a + sign before positive values
 - -: left-justification (default is right-justification)
- width:
 - *Minimum* number of character spaces reserved to display the entire value
 - Includes decimal point, digits before and after the decimal point and the sign

Feb 15, 2016 Sprengle - CSC111 34

Format Specifiers

- General format: **%[flags][width][.precision]code**
- precision:
 - Number of digits after the decimal point for **floating point** values
- code:
 - Indicates the value's **type**/how to format

Code	Type
s	string
d or i	integer
f	float
e	floating point with exponent

Feb 15, 2016 Sprengle - CSC111 35

Example using Format Operator

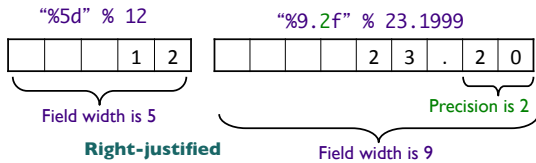
```
print("Your item that cost ($%.2f)" % value)
print("costs $%.2f with tax" % tax)
```

Alternative:

```
print("Your item that cost ($%.2f) costs $%.2f with tax" % (value, tax))
```

Feb 15, 2016 Sprengle - CSC111 36

Example Format Specifiers



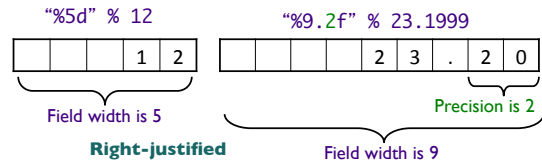
- What if precision is bigger than the decimal places?
 - What if field width is smaller than the length of the value?
- Any guesses? Try out in interpreter.

Feb 15, 2016

Sprengle - CSC111

37

Example Format Specifiers



- What if precision is bigger than the decimal places?
 > Fills decimal with 0s
- What if field width is smaller than the length of the value?
 > String contains entire value

Feb 15, 2016

Sprengle - CSC111

38

Formatting Practice

- `x = 10`
- `y = 3.5`
- `z = "apple"`
- `"%6d" % x`
- `"%6.2f" % x`
- `"%06.2f" % y`
- `"%+6.2f" % y`
- `"%-10s" % z`
- `"%5d %-7.3f" % (x,y)`

Feb 15, 2016

Sprengle - CSC111

39

Example: Printing Out Tables

- A table of temperature conversions

Temp F	Temp C	Temp K
-459.7	-273.1	0.0
0.0	-17.8	255.2
32.0	0.0	273.1

- If we want to print data in rows, what is the template for what a row looks like?
 > How do we make the column labels line up?

Feb 15, 2016

Sprengle - CSC111

`temp_table.py` 40

Looking Ahead

- Lab 5 tomorrow
- Broader Issue: Automated Cars

Feb 15, 2016

Sprengle - CSC111

41