

## Objectives

- Strings

Feb 14, 2011

Sprenkle - CSCI111

1

## Motivation: Text Processing

- Mostly focused on numbers so far
  - A little on graphics
- We can manipulate strings to do useful work
  - Web search: finding most relevant documents to a query
  - Analyzing web logs (who is looking at my web page?)
  - Many, many others
- **Today's Focus:** the **str** data type and what you can do with them

Feb 14, 2011

Sprenkle - CSCI111

2

## String Comparisons

- Same operations as with numbers:
  - `==`, `!=`
  - `<`, `<=`
  - `>`, `>=` } Alphabetical comparison
- Use in conditions for **if** statements and **while** loops

```
if userpick == pick4num:
    print "We have a winner!"
else:
    print "You lose."
```

Feb 14, 2011

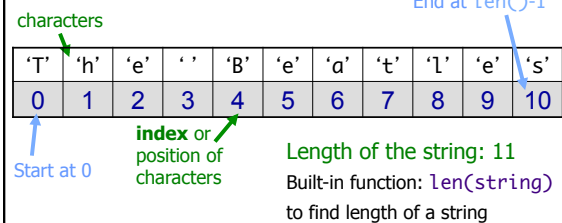
Sprenkle - CSCI111 string\_compare.py 3

## Strings

- A *sequence* of characters

➢ Example:

band = "The Beatles"



Feb 14, 2011

Sprenkle - CSCI111

4

## Iterating Through a String

- Use a **for** loop to iterate through *characters* in a string

string of length 1

```
for char in string:
    print char
```

➢ Read as "for each character in the string"

Feb 14, 2011

Sprenkle - CSCI111

Python shell

5

## Substrings Operator: `[]`

Literally, **not** optional

- Look at a particular character in the string

➢ Syntax: `str[<integer_expression>]`

➢ [Positive value]: index of character

➢ [Negative value]: count backwards from end

- Examples:

➢ `<sequence>[0]` returns the first element/char

➢ `<sequence>[-1]` returns the last element/char

We will deal with sequences beyond strings later.

Feb 14, 2011

Sprenkle - CSCI111

Examples in shell

6

## Substrings Operator: []

- Look at a particular character in the string
  - Syntax: `string[<integer_expression>]`
- Examples with `string = "The Beatles"`

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Expression	Result
<code>string[0]</code>	"T"
<code>string[3]</code>	" "
<code>string[len(string)]</code>	<b>IndexError</b>
<code>string[len(string)-1]</code>	"s"
<code>string[-1]</code>	"s"

Feb 14, 2011

7

## Substrings Operator: []

- Look at a particular character in the string
  - Syntax: `string[<integer_expression>]`
- Examples with `string = "The Beatles"`

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Expression	Result
<code>string[0]</code>	"T"
<code>string[3]</code>	" "
<code>string[len(str)]</code>	<b>IndexError</b>
<code>string[len(str)-1]</code>	"s"
<code>string[-1]</code>	"s"

Feb 14, 2011

8

## Iterating Through a String

- Alternatively, can iterate through the *positions* in a string
  - Could write as a **while** loop as well

An integer

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

Index into the string

Feb 14, 2011

Sprenkle - CSCI1111 whilestr.py

9

## Summary: Iterating Through a String

- For each character in the string

string of length 1

```
for char in string:
    print char
```

Determines loop's behavior

- For each position in the string

An integer

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

Index into the string

Feb 14, 2011

Sprenkle - CSCI1111

10

## Substrings Operator: [:]

- Select a substring (zero or more characters) using the `[]` and `:`
- `<sequence>[<start>:<end>]`
  - returns the subsequence from **start** up to and **not** including **end**
- `<sequence>[<start>:]`
  - returns the subsequence from **start** to the end of the sequence
- `<sequence>[:<end>]`
  - returns the subsequence from the first element up to and **not** including **end**
- `<sequence>[:]`
  - returns a copy of the entire sequence

Feb 14, 2011

Sprenkle - CSCI1111

11

## Substrings Operator: [:]

- Select a substring (one or more characters) using the `[]` and `:`

- Examples: `file = "program.py"`

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
<code>file[:]</code>	"program.py"
<code>file[0:2]</code>	"pr"
<code>file[:3]</code>	"pro"
<code>file[8:]</code>	".py"
<code>file[-2:]</code>	"py"

Feb 14, 2011

12

## Substrings Operator: [:]

- Select a substring (one or more characters) using the [] and :
- Examples: file = "program.py"

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
file[0:]	"program.py"
file[0:2]	"pr"
file[:3]	"pro"
file[8:]	"py"
file[-2:]	"py"

Feb 14,

13

## Testing for Substrings

- Using the **in** operator
  - Used **in** before **in** for loops
- Syntax:

```
substring in string:
```

➤ Evaluates to True or False

- Example:

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

Feb 14, 2011

Sprengle - CSCI1111

14

## String Search Comparison

- What do the two **if** statements test for?

```
PYTHON_EXT = ".py"  
  
filename = raw_input("Enter a filename: ")  
  
if filename[-(len(PYTHON_EXT)):] == PYTHON_EXT:  
    # Appropriate output  
if PYTHON_EXT in filename:  
    # Appropriate output
```

- How would the program execution change if it were an **if-elif**?

Feb 14, 2011

Sprengle - CSCI1111

search.py

15

## Strings are Immutable

You cannot change the value of strings

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

➤ ~~str[0] = 'S'~~

Feb 14, 2011

Sprengle - CSCI1111

16

## Practice: 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 14, 2011

Sprengle - CS

pick4num\_places.py

17

## USING THE STR API

Feb 14, 2011

Sprengle - CSCI1111

18

## 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 14, 2011

Sprenkle - CSCI1111

19

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

Executed on this string

Feb 14, 2011

Sprenkle - CSCI1111

20

## 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 <b>sub</b> in the string.
<code>endswith(sub), startswith(sub)</code>	Return <b>True</b> iff string ends with/begins with <b>sub</b>
<code>find(sub[, start [, end]])</code>	Return first index where substring <b>sub</b> is found
<code>isalpha(), isdigit(), isspace()</code>	Returns <b>True</b> iff string contains letters/digits/whitespace only
<code>lower(), upper()</code>	Return a copy of string converted to lowercase/uppercase

Feb 14, 2011

Sprenkle - CSCI1111 [string\\_methods.py](#)

## Common str Methods

Method	Operation
<code>replace(old, new[, count])</code>	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.
<code>split([sep])</code>	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.
<code>strip()</code>	Return a copy of the string with the leading and trailing whitespace removed
<code>join(&lt;sequence&gt;)</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 14, 2011

Sprenkle - CSCI1111

22

## String Methods vs. Functions

### 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 14, 2011

Sprenkle - CSCI1111

23

## Looking Ahead

- Tuesday: Lab
- Friday: Broader Issue: BUGS!
- Today – Wednesday
  - Jeopardy! Challenge
  - Answer questions on Sakai

Feb 14, 2011

Sprenkle - CSCI1111

24