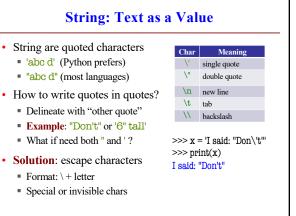
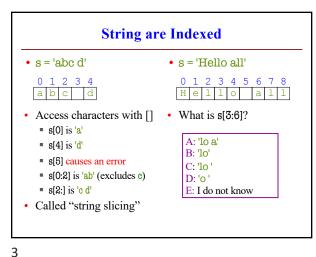
### **Announcements For This Lecture** Assignment 1 **Getting Help** · Will post it on Thursday • Can work in pairs ■ Need Thurs. lecture Will set up And associated lab Submit one for both • Due Fri Sep. 20th · Lots of consultant hours ■ Revise until correct Come early! Beat the rush Final version Sep 27th Also use TA office hours • Do not put off until end! · One-on-Ones next week 9/9/21



1



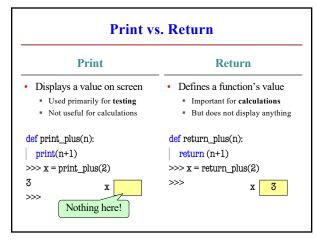
Other Things We Can Do With Strings Operation in: s1 in s2 • Function len: len(s) ■ Tests if s<sub>1</sub> "is inside" s<sub>2</sub> Value is # of chars in s • We say  $s_1$  a *substring* of  $s_2$  Evaluates to an int ■ Evaluates to a bool • Examples: • Examples: s = 'abracadabra' s = 'abracadabra' ■ 'a' in s == True len(s) == 11 ■ 'cad' in s == True len(s[1:5]) == 4 'foo' in s == False ■ s[1:len(s)-1] == 'bracadabr'

#### **Defining a String Function** def middle(text): >>> middle('abc') """Returns: middle 3rd of text 'b' Param text: a string""" >>> middle('aabbcc') # Get length of text 'bb' size = len(text)>>> middle('aaabbbccc') # Start of middle third start = size//3'bbb' # End of middle third end = 2\*size//3# Get the text result = text[start:end] # Return the result return result

**Procedures vs. Fruitful Functions Fruitful Functions Procedures** • Functions that **do** something • Functions that give a **value** Call them as a statement · Call them in an expression Example: greet('Walker') • Example: x = round(2.56,1)Historical Aside Historically "function" = "fruitful function" But now we use "function" to refer to both

5 6

1



**Method Calls** 

- Methods calls are unique (right now) to strings
  - Like a function call with a "string in front"
- Method calls have the form



- The string in front is an additional argument
  - Just one that is not inside of the parentheses
  - Why? Will answer this later in course.

8

7

## Example: upper()

- upper(): Return an upper case copy
  - >>> s = 'Hello World'
  - >>> s.upper()
  - 'HELLO WORLD'
  - >>> s[1:5].upper() # Str before need not be a variable
  - 'ELLO'
- >>> 'abc'.upper() # Str before could be a literal 'ABC'
- Notice that only argument is string in front

**Examples of String Methods** 

- s<sub>1</sub>.index(s<sub>2</sub>)
  - Returns position of the first instance of s<sub>2</sub> in s<sub>1</sub>
- $s_1.count(s_2)$
- Returns number of times
   s<sub>2</sub> appears inside of s<sub>1</sub>
- s.strip()
  - Returns copy of s with no white-space at *ends*

- >>> s = 'abracadabra'
- >>> s.index('a')
- U
- >>> s.index('rac') 2
- >>> s.count('a')
- Ð
  - >>> s.count('x')
- 0
  - >>> ' a b '.strip()
    'a b'

9

10

## **String Extraction Example**

## def firstparens(text):

- """Returns: substring in ()
  Uses the first set of parens
  Param text: a string with ()"""
- # SEARCH for open parens
  start = text.index('(')
  # CUT before paren
  tail = text[start+1:]
- # SEARCH for close parens end = tail.index(')')
- end = tail.index(')')
  # CUT and return the result
  return tail[:end]
- >>> s = 'Prof (Walker) White'
- >>> firstparens(s)
- 'Walker'
- >>> t = '(A) B (C) D'
- >>> firstparens(t)
- 'A'

# String Extraction Puzzle

#### def second(text):

"""Returns: second elt in text
The text is a sequence of words
separated by commas, spaces.
Ex: second('A, B, C') rets 'B'
Param text: a list of words"""

>>> second('cat, dog, mouse, lion')

>>> second('apple, pear, banana')

- start = text.index(',') # SEARCH
  tail = text[start+1:] # CUT
- end = tail.index(',') # SEARCH
- result = tail[:end] # CUT
- 5 return result

11 12

2