

## Lecture 5: Strings

(Sections 8.1, 8.2, 8.4, 8.5, 1<sup>st</sup> paragraph of 8.9)

## CS 1110

## Introduction to Computing Using Python

[E. Andersen, A. Bracy, D. Fan, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]

#### Announcements

- Did you try the 6 questions at the end of the slides from the previous lecture? Check answers on course website and ask at office/consulting hrs if you have questions!
- Want to find an assignment partner?
  - Make/answer a post on Ed Discussion megathread "[Where can I] find a partner for this class?"
  - · Learning Strategy Center (LSC) has a study partner finding service
  - Talk to classmates in lab!
- · Check out "Assignment Advice" on course website
- Note "Policies" on the course website

#### Today

- More about the str type New ways to use strings
- More examples of functions
  - Functions with strings!
- · Learn the difference between print and return

## **Strings are Indexed (Question 1)**

- s = 'abc d'
  - 0 1 2 3 4 abc d
- Access characters with []
  - s[0] is 'a'
  - s[4] is 'd'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'
- · Called "string slicing"



- What is t[3:6]?

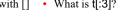
A: 'lo a'
B: 'lo'
C: 'lo '
D: 'o '
E: I do not know

## **Strings are Indexed (Question 2)**

 s = 'abc d' 0 1 2 3 4

- abc d
- Access characters with []
   What is t[:3]?
  - s[0] is 'a'
  - s[4] is 'd'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'
- Called "string slicing"





A: 'all'
B: 'l'
C: 'Hel'
D: Error!
E: I do not know

# **Other Things We Can Do With Strings**

#### Operator in: S1 in S2 Tests if S<sub>1</sub> "a part of" (or a substring of) S2

· Evaluates to a bool

#### Examples:

- >>> s = 'abracadabra' >>> 'a' in s True >>> 'cad' in s True >>> 'foo' in s False
- Built-in Function len: len(s) Value is # of chars in s
  - Evaluates to an int

#### Examples:

#### >>> s = 'abracadabra' >>> len(s) 11

- >>> len(s[1:5])
- 4
- >>> s[1:len(s)-1] 'bracadabr >>>



8

#### **Defining a String Function**

Want to write function middle, which returns the middle 3<sup>rd</sup> of a string (length 2. What is the return value? divisible by 3).

Important Questions: 1. What are the parameters?

#### How we want it to behave:

>>> middle('abc') 'b' >>> middle('aabbcc') 'bb' >>> middle('aaabbbccc') 'bbb'

3. What goes in the body? def middle(text): ??? return middle\_third 11

#### Steps to writing a program

- 1. Work an instance yourself
- 2. Write down exactly what you just did
- Generalize your steps from 2 3.
- 4. Test your steps
- 5. Translate to Code
- 6. Test program
- 7. Debug (if necessary)

#### Steps to writing a program

- 1. Work an instance yourself
- 2. Write down exactly what you just did
- 3. Generalize your steps from 2
- 4. Test your steps
- 5. Translate to Code
- >>> middle('abc') middle\_third = text[1] Too easy!!
- >>> middle('aabbcc') middle\_third = text[2:4] Still too easy!!

>>> middle('It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way ... ')

## **Definition of middle**

## def middle(text): ""Returns: middle 3<sup>rd</sup> of text Param text: a string with length divisible by 3""

#### **IMPORTANT:**

12

**Precondition requires** that arguments to middle have length divisible by 3.

If not? Bad things could happen, and we blame the user (not the author) of the function.

### **Advanced String Features: Method Calls**

- · Strings have some useful methods Like functions, but "with a string in front"
- Format: <*string name*>.<*method name*>(*x*,*y*,...)
- Example: upper() returns an upper case version

_	
>>> s = 'Hello World'	>>> s[1:5].upper()
>>> s.upper()	'ELLO'
'HELLO WORLD'	>>> 'scream'.upper()
>>> \$	'SCREAM'
'Hello World'	>>> 'cs1110'.upper()
	'CS1110'

16

## **Examples of String Methods**

s <sub>1</sub> .index(s <sub>2</sub> )	<ul> <li>s = 'abracadabra'</li> </ul>
<ul> <li>Returns position of the first</li> </ul>	0 1 2 3 4 5 6 7 8 9 10
instance of $S_2$ in $S_1$	a b r a c a d a b r a
error if s <sub>2</sub> is not in s <sub>1</sub>	
	<ul> <li>s.index('a') 0</li> </ul>
s <sub>1</sub> .count(s <sub>2</sub> )	<ul> <li>s.index('rac')</li> </ul>
<ul> <li>Returns number of times S<sub>2</sub> appears inside of S<sub>1</sub></li> </ul>	<ul> <li>s.count('a') 5</li> </ul>
appears inside of S <sub>1</sub>	• s.count('b') 2
<ul> <li>s.strip()</li> </ul>	<ul> <li>s.count('x')</li> <li>0</li> </ul>
Returns a copy of S with	<ul> <li>' a b '.strip() 'a b'</li> </ul>
white-space removed at ends	
See Python	Docs for more

#### **String Extraction Example**

#### def firstparens(text):

"""Returns: substring in () Uses the first set of parens Param text: a string with ()"""

>>> firstparens(s) 'Two' >>> t = '(A) B (C) D' >>> firstparens(t) 'A'

>>> s = 'One (Two) Three'

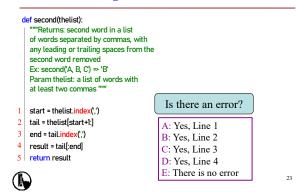
18

#### Steps to writing a program

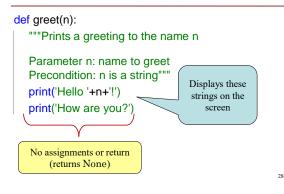
- 1. Work an instance yourself
- 2. Write down exactly what you just did
- 3. Generalize your steps from 2
- 4. Test your steps
- 5. Translate to Code
- 6. Test program
- 7. Debug (if necessary)
- Think of all the corner cases
- What could possibly go wrong?

20

## **String Extraction Puzzle**

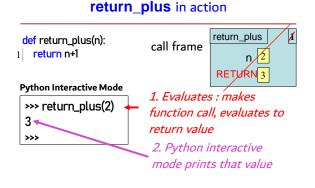


### Not All Functions Need a Return



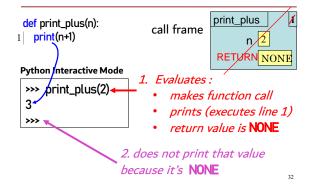
print	vs.	return
<ul> <li>Displays a value on screen</li> <li>Used primarily for testing</li> </ul>	cal • Imj	nds a value from a function I frame back to the caller portant for <b>calculations</b> es not display anything
def print_plus(n): print(n+1)		return_plus(n): eturn n+1
>>> print_plus(2) 3 >>>	>>>   3 >>>	return_plus(2)

unexpected printing courtesy of:	
<b>Python Interactive Mode</b>	
executes both <i>statements</i> and <i>expressions</i>	
• if <i>expression:</i>	
1. evaluates	
2. prints value (if one exists)	
>>> 2+2 🔶 evaluates (performs addition)	
4 <i>— prints value (4)</i>	
>>> return_plus(2)	
3	
>>>	30
	50

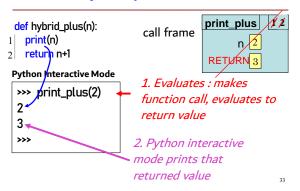


31

#### print\_plus in action



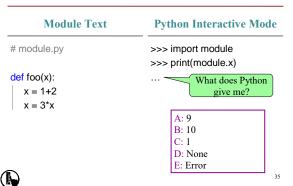
## hybrid\_plus in action



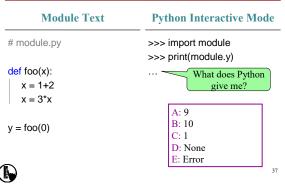
## See the difference in the Python Tutor

def return_plus(n): return n+1 x1 = print_plus(2) x2 = return_plus(2)	Program output: 3 None 3
print(x1)	
print(x2)	
http://cs1110.cs.cornell.e	du/tutor/#mode=edit 34

## **Exercise 1**



# Exercise 2



## Exercise 3

>>> import module
>>> module.y
What does Python
give me?
A: 9
B: 10
C: 1
D: None
E: Error

Exercise 4			
<b>Function Definition</b>	<b>Function Call</b>		
def foo(a,b):	>>> x = 2		
1 x = a	>>> foo(3,4)		
2 y = b	>>> X What does Python		
3 return x*y+y	give me?		
	A: 2		
	B: 3		
	C: 16		
	D: None		
<u>_</u>	E: I do not know		
	41		

# 5