http://www.cs.cornell.edu/courses/cs1110/2020sp

Lecture 5: Strings

(Sections 8.1, 8.2, 8.4, 8.5, 1st paragraph of 8.9)

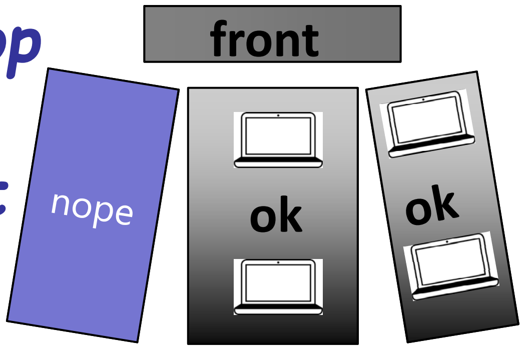
CS 1110

Introduction to Computing Using Python

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

Announcements

No-laptop zone on your left



- No laptop use stage right (your left)
- We will use clickers, but not for credit. Therefore no need to register your clicker.
- "Partner Finding Social" Tues Feb 4th 5-6pm Gates Hall 3rd floor Lounge (1xxx-2xxx courses)
- Before next lecture, read Sections 4.9, 9.5
- To access video of lecture, log in using NetID and password "through Canvas", but we don't use Canvas otherwise. Course website is https://www.cs.cornell.edu/courses/cs1110/2020sp/

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
a	b	c		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"

- t = 'Hello all'

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- What is t[3:6]?

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

Strings are Indexed (Solution 1)

- s = 'abc d'

0	1	2	3	4
a	b	c		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”

- t = 'Hello all'

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- What is t[3:6]?

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

Strings are Indexed (Question 2)

- s = 'abc d'

0	1	2	3	4
a	b	c		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"

- t = 'Hello all'

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- What is t[:3]?

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

Strings are Indexed (Solution 2)

- s = 'abc d'

0	1	2	3	4
a	b	c		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"

- t = 'Hello all'

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- What is t[:3]?

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

Other Things We Can Do With Strings

Operator in: s_1 in s_2

- Tests if s_1 "a part of" (or a *substring* of) s_2
- 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'
>>>
```

Defining a String Function

Want to write function **middle**, which returns the middle 3rd of a string (length divisible by 3).

How we want it to behave:

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

Important Questions:

1. What are the parameters?
2. What is the return value?
3. What goes in the body?

```
def middle(text):

        ???

    return middle_third
```

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)

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 3rd of text
    Param text: a string with
    length divisible by 3"""
```

IMPORTANT:
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.

Definition of middle

```
def middle(text):
    """Returns: middle 3rd of text
    Param text: a string with
    length divisible by 3"""

    # Get length of text
    size = len(text)
    # Start of middle third
    start2 = size//3
    # End of middle third
    start3 = (2*size)//3
    # Get the substring
    middle_third = text[start2:start3]
    return middle_third
```

IMPORTANT:
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.upper()
'HELLO WORLD’
>>> s
'Hello World’
```

```
>>> s[1:5].upper()
'ELLO'
>>> ‘scream'.upper()
‘SCREAM'
>>> 'cs1110'.upper()
'CS1110'
```

Examples of String Methods

- s_1.index(s_2)
  - Returns position of the first instance of s_2 in s_1
  - error if s_2 is not in s_1
- s_1.count(s_2)
  - Returns number of times s_2 appears inside of s_1
- s.strip()
  - Returns a copy of s with white-space removed at ends

- s = 'abracadabra’

0	1	2	3	4	5	6	7	8	9	10
a	b	r	a	c	a	d	a	b	r	a

• s.index('a')	0
• s.index('rac')	2
• s.count('a')	5
• s.count('b')	2
• s.count('x')	0
• ' a b '.strip()	'a b'

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 ()"""
```

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

String Extraction, Round 1

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

    # Find the open parenthesis
    start = text.index('(')

    # Find the close parenthesis
    end = text.index(‘)’)

    inside = text[start+1:end]

    return inside
```

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

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** *Think of all the corner cases*
7. Debug (if necessary) *What could possibly go wrong?*

String Extraction, Round 2

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

    # Find the open parenthesis
    start = text.index('(')

    # Store part AFTER paren
    substr = text[start+1:]

   # Find the close parenthesis
    end = substr.index(')')

    inside = substr[:end]
    return inside
```

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

String Extraction Puzzle

```
def second(thelist):
    """Returns: second word in a list
    of words separated by commas, with
    any leading or trailing spaces from the
    second word removed
    Ex: second('A, B, C') => 'B'
    Param thelist: a list of words with
    at least two commas """

1   start = thelist.index(',')
2   tail = thelist[start+1:]
3   end = tail.index(',')
4   result = tail[:end]
5   return result
```

Is there an error?

A: Yes, Line 1
B: Yes, Line 2
C: Yes, Line 3
D: Yes, Line 4
E: There is no error

String Extraction Puzzle

```
def second(thelist):
    """Returns: second word in a list
    of words separated by commas, with
    any leading or trailing spaces from the
    second word removed
    Ex: second('A, B, C') => 'B'
    Param thelist: a list of words with
    at least two commas """

1   start = thelist.index(',')
2   tail = thelist[start+1:]
3   end = tail.index(',')
4   result = tail[:end]
5   return result
```

```
>>> second('cat, dog, mouse, lion')
expecting: 'dog'            get: ' dog'

>>> second('apple, pear, banana')
expecting: 'pear'           get: ' pear'
```

Is there an error?

A: Yes, Line 1
B: Yes, Line 2
C: Yes, Line 3
D: Yes, Line 4
E: There is no error

String Extraction Puzzle, v2

```
def second(thelist):
    """Returns: second word in a list
    of words separated by commas, with
    any leading or trailing spaces from the
    second word removed
    Ex: second('A, B, C') => 'B'
    Param thelist: a list of words with
    at least two commas """

    start = thelist.index(',')
    tail = thelist[start+1:]
    end = tail.index(',')
    result = tail[:end]
    return result
```

```
>>> second('cat, dog, mouse, lion')
expecting: 'dog'              get: ' dog'

>>> second('apple,pear    , banana')
expecting: 'pear'             get: 'pear    '
```

String Extraction Fix

```
def second(thelist):
    """Returns: second word in a list
    of words separated by commas, with
    any leading or trailing spaces from the
    second word removed
    Ex: second('A, B, C') => 'B'
    Param thelist: a list of words with
    at least two commas """

    start = thelist.index(',')
    tail = thelist[start+1:]
    end = tail.index(',')
    result = tail[:end]
    return result
```

```
>>> second('cat, dog, mouse, lion')
expecting: 'dog'            get: ' dog'

>>> second('apple,pear   , banana')
expecting: 'pear'           get: 'pear   '
```

`tail = thelist[start+1:]` → `tail = thelist[start+2:]   #possible fix ??`

What if there are *multiple* (*or no!*) spaces?

`result = tail[:end]` → `result = tail[:end].strip()   #better fix!`

String: Text as a Value

- String are quoted characters
  - 'abc d' (Python prefers)
  - "abc d" (most languages)
- How to write quotes in quotes?
  - Delineate with “other quote”
  - **Example**: " ' " or ' " '
  - What if need both " and ' ?
- **Solution**: escape characters
  - Format: \ followed by letter (character)
  - Special or invisible chars

Type: str

Char	Meaning
\'	single quote
\"	double quote
\n	new line
\t	tab
\\	backslash