

One-on-One Sessions

- Starting tomorrow: 1/2-hour one-on-one sessions
 - Bring computer to work with instructor, TA or consultant
 - Hands on, dedicated help with Lab 3 (or next lecture)
 - To prepare for assignment, **not for help on assignment**
- Limited availability: we cannot get to everyone**
 - Students with experience or confidence should hold back
- Sign up online in CMS: first come, first served
 - Choose assignment One-on-One
 - Pick a time that works for you; will add slots as possible
 - Can sign up starting at 1pm **TODAY**

9/6/16

Strings

1

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: \ + letter
 - Special or invisible chars

Type: str

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

9/6/16

Strings

2

String are Indexed

- `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 e)
 - `s[2:]` is 'c d'
 - Called "string slicing"
- `s = 'Hello all'`
 - 0 1 2 3 4 5 6 7 8
 - H e l l o a l l
 - What is `s[3:6]`?
 - A: 'lo a'
 - B: 'lo'
 - C: 'lo'
 - D: 'o'
 - E: I do not know

9/6/16

Strings

3

Other Things We Can Do With Strings

- Operation** in: `s1 in s2`
 - Tests if `s1` "a part of" `s2`
 - Say `s1` a *substring* of `s2`
 - Evaluates to a bool
- Examples:**
 - `s = 'abracadabra'`
 - 'a' in `s` == True
 - 'ead' in `s` == True
 - 'foo' in `s` == False
- 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'

9/6/16

Strings

4

Defining a String Function

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

```
def middle(text):
    """Returns: middle 3rd of text
    Param text: a string"""
    # Get length of text
    size = len(text)
    # Start of middle third
    start = size/3
    # End of middle third
    end = 2*size/3
    # Get the text
    result = text[start:end]
    # Return the result
    return result
```

9/6/16

Strings

5

Not All Functions Need a Return

```
def greet(n):
    """Prints a greeting to the name n

    Parameter n: name to greet
    Precondition: n is a string"""
    print 'Hello '+n+'!'
    print 'How are you?'
```

Displays these strings on the screen

No assignments or return
The call frame is **EMPTY**

9/6/16

Strings

6

Procedures vs. Fruitful Functions

Procedures

- Functions that **do** something
- Call them as a **statement**
- Example: `greet('Walker')`

Fruitful Functions

- Functions that give a **value**
- Call them in an **expression**
- Example: `x = round(2.56,1)`

Historical Aside

- Historically “function” = “fruitful function”
- But now we use “function” to refer to both

9/6/16

Strings

7

Print vs. Return

Print

- Displays a value on screen
 - Used primarily for **testing**
 - Not useful for calculations

```
def print_plus(n):
    print(n+1)
>>> x = print_plus(2)
3
>>>
```

x
Nothing here!

9/6/16

Strings

8

Return

- Defines a function’s value
 - Important for **calculations**
 - But does not display anything

```
def return_plus(n):
    return(n+1)
>>> x = return_plus(2)
>>>
```

x 3

Advanced String Features: Method Calls

- Method calls are unique (right now) to strings
- Like a function call with a “string in front”
 - Usage: *string.method(x,y,...)*
 - The string is an *implicit argument*
- Example: `upper()`
 - `s = 'Hello World'`
 - `s.upper() == 'HELLO WORLD'`
 - `s[1:5].upper() == 'ELLO'`
 - `'abc'.upper() == 'ABC'`

Will see why we do it this way later in course

9/6/16

Strings

9

Examples of String Methods

- `s1.index(s2)`
 - Position of the first instance of `s2` in `s1`
- `s1.count(s2)`
 - Number of times `s2` appears inside of `s1`
- `s.strip()`
 - A copy of `s` with white-space removed at ends

- `s = 'abracadabra'`
- `s.index('a') == 0`
- `s.index('rac') == 2`
- `s.count('a') == 5`
- `s.count('b') == 2`
- `s.count('x') == 2`
- `' a b '.strip() == 'a b'`

See Python Docs for more

9/6/16

Strings

10

String Extraction Example

```
def firstparens(text):
    """Returns: substring in ()
    Uses the first set of parens
    Param text: a string with ()"""
    # Find the open parenthesis
    start = s.index('(')
    # Store part AFTER paren
    tail = s[start+1:]
    # Find the close parenthesis
    end = tail.index(')')
    # Return the result
    return tail[end:]

>>> s = 'Prof (Walker) White'
>>> firstparens(s)
'Walker'
>>> t = '(A) B (C) D'
>>> firstparens(t)
'A'
```

9/6/16

Strings

11

String Extraction Puzzle

```
def second(thelist):
    """Returns: second elt in thelist
    The list is a sequence of words
    separated by commas, spaces.
    Ex: second('A, B, C') => 'B'
    Param thelist: a list of words"""
    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')
'dog'
>>> second('apple, pear, banana')
'pear'
```

9/6/16

Strings

12