

## Announcements For This Lecture

### Assignment 1

- Will post it on Thursday
  - Need Thurs. lecture
  - And associated lab
- Due Sun Sep. 17th
  - Revise until correct
  - Final version Sep 24th
- Do not put off until end!

### Getting Help

- Can work in pairs
  - Will set up
  - Submit one for both
- Lots of consultant hours
  - Come early! Beat the rush
  - Also use TA office hours
- One-on-Ones next week

9/9/21

Strings

1

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:** "Don't" or '6" tall'
  - What if need both " and ' ?
- **Solution:** escape characters
  - Format: \ + letter
  - Special or invisible chars

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

```
>>> x = 'I said: "Don\'t"'
>>> print(x)
I said: "Don't"
```

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 c)
  - `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

## Other Things We Can Do With Strings

- **Operation** in: `s1 in s2`
  - Tests if `s1` "is inside" `s2`
  - We say `s1` a *substring* of `s2`
  - Evaluates to a bool
- **Examples:**
  - `s = 'abracadabra'`
  - `'a' in s == True`
  - `'cad' 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'`

9/9/21

Strings

4

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
```

5

## 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(2.86,1)`

### Historical Aside

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

6

## Print vs. Return

| Print                                                                                                                                                                                                                      | Return                                                                                                                                                                                                                       |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>Displays a value on screen                     <ul style="list-style-type: none"> <li>Used primarily for <b>testing</b></li> <li>Not useful for calculations</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>Defines a function's value                     <ul style="list-style-type: none"> <li>Important for <b>calculations</b></li> <li>But does not display anything</li> </ul> </li> </ul> |
| <pre>def print_plus(n):     print(n+1) &gt;&gt;&gt; x = print_plus(2) 3 &gt;&gt;&gt;</pre>                                                                                                                                 | <pre>def return_plus(n):     return (n+1) &gt;&gt;&gt; x = return_plus(2) &gt;&gt;&gt; x</pre>                                                                                                                               |

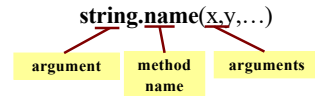
Nothing here!

7

## 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

## 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

9

## Examples of String Methods

- |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>s<sub>1</sub>.index(s<sub>2</sub>)                             <ul style="list-style-type: none"> <li>Returns position of the <i>first</i> instance of s<sub>2</sub> in s<sub>1</sub></li> </ul> </li> <li>s<sub>1</sub>.count(s<sub>2</sub>)                             <ul style="list-style-type: none"> <li>Returns number of times s<sub>2</sub> appears inside of s<sub>1</sub></li> </ul> </li> <li>s.strip()                             <ul style="list-style-type: none"> <li>Returns copy of s with no white-space at <i>ends</i></li> </ul> </li> </ul> | <pre>&gt;&gt;&gt; s = 'abracadabra' &gt;&gt;&gt; s.index('a') 0 &gt;&gt;&gt; s.index('rac') 2 &gt;&gt;&gt; s.count('a') 5 &gt;&gt;&gt; s.count('x') 0 &gt;&gt;&gt; ' a b '.strip() 'a b'</pre> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

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(')')
    # CUT and return the result
    return tail[:end]

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

11

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

>>> second('cat, dog, mouse, lion')
'dog'
>>> second('apple, pear, banana')
'pear'
```

12