**One-on-One Sessions**

- Starting **Sunday**: 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 5pm **TOMORROW**

**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

<table>
<thead>
<tr>
<th>Char</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>'</td>
<td>single-quote</td>
</tr>
<tr>
<td>&quot;</td>
<td>double-quote</td>
</tr>
<tr>
<td>\n</td>
<td>new line</td>
</tr>
<tr>
<td>\t</td>
<td>tab</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
</tr>
</tbody>
</table>

**String are Indexed**

- s = 'abc d'
  - [0] is 'a'
  - [4] is 'd'
  - [5] causes an error
  - [0:2] is 'ab' (excludes c)
  - [2:] is 'c d'
- Called “string slicing”

- s = 'Hello all'
  - What is s[3:8]?
    - 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 “a part of” s2
  - 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] == 'bracadab'

**Defining a String Function**

```python
def middle(text):
    # 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
```

**Not All Functions Need a Return**

```python
def greet(n):
    # Prints a greeting to the name n
    # Parameter n: name to greet
    # Precondition: n is a string
    print('Hello "{}"!'.format(n))
    print('How are you?')
```

Displays these strings on the screen

No assignments or return
The call frame is **EMPTY**
Procedures vs. Fruitful Functions

<table>
<thead>
<tr>
<th>Procedures</th>
<th>Fruitful Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Functions that do something</td>
<td>• Functions that give a value</td>
</tr>
<tr>
<td>• Call them as a statement</td>
<td>• Call them in an expression</td>
</tr>
<tr>
<td>• Example: <code>greet('Walker')</code></td>
<td>Example: <code>x = round(2.56,1)</code></td>
</tr>
</tbody>
</table>

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

Print vs. Return

<table>
<thead>
<tr>
<th>Print</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Displays a value on screen</td>
<td>• Defines a function’s value</td>
</tr>
<tr>
<td>• Used primarily for testing</td>
<td>• Important for calculations</td>
</tr>
<tr>
<td>• Not useful for calculations</td>
<td>• But does not display anything</td>
</tr>
</tbody>
</table>

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

```python
def return_plus(n):
    return(n+1)
>>> x = return_plus(2)
3
```

Advanced String Features: Method Calls

- Methods 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`

```
>>> s = 'Prof(Walker) White'
>>> firstparens(s)
'Walker'
```

```
>>> t = '(A) B (C) D'
>>> firstparens(t)
'A'
```

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
```

```python
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'
```

```
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'
```

**String Extraction Example**

```python
def firstparens(text):
    start = text.index('(')
    tail = text[start+1:]
    end = tail.index(')')
    return tail[:end]
```

```python
>>> s = 'Prof (Walker) White'
>>> firstparens(s)
'Prof (Walker)'
```

```python
>>> t = '(A) B (C) D'
>>> firstparens(t)
'(A) B (C) D'
```

**String Extraction Puzzle**

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

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

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