**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**

---

**String: Text as a Value**

- String are quoted characters
  - "abo d" (Python prefers)
  - "abo 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
- **String are Indexed**
  - \[0\] is 'a'
  - \[4\] is 'd'
  - \[5\] causes an error
  - \[0:2\] is 'ab' (excludes c)
  - \[2:\] is 'c d'
- Called “string slicing”

---

**Other Things We Can Do With Strings**

- **Operation in \(s_1 \text{ in } s_2\)**
  - Tests if \(s_1\) “a part of” \(s_2\)
  - Say \(s_1\) a subset of \(s_2\)
  - Evaluates to a bool
- **Examples:**
  - \(a = \text{‘abra}\text{cabra’}\)
  - ‘a’ in \(a\) => True
  - ‘ad’ in \(a\) => True
  - ‘foo’ in \(a\) => False
- **Function \(\text{len}(s)\)**
  - Value is # of chars in \(s\)
  - Evaluates to an int
- **Examples:**
  - \(a = \text{‘abra}\text{cabra’}\)
  - \(\text{len}(a) == 11\)
  - \(\text{len}(a[1:8]) == 4\)
  - \(a[1: \text{len}(a)-1] == \text{‘bracadabra’}\)

---

**Defining a String Function**

```python
>>> middle(\'abc\')
'b'
```

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

---

**Not All Functions Need a Return**

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

- Displays these strings on the screen
- No assignments or return
- The call frame is **EMPTY**
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

Print vs. Return

Print
- Displays a value on screen
- Used primarily for testing
- Not useful for calculations
- Example:

```python
def print_plus(n):
    print(n + 1)
```

>>> x = print_plus(2)
3

Return
- Defines a function’s value
- Important for calculations
- But does not display anything
- Example:

```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[1:8].upper()` == 'ELLO'
  - `'abc'.upper()` == 'ABC'

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
  - 'abc'.upper() == 'ABC'

String Extraction Example

```python
def firstparens(text):
    # Find the open parenthesis
    start = text.index('(')
    # Store part AFTER parenthesis
    tail = text[start + 1:
    # Find the close parenthesis
    end = tail.index(')')
    # Return the result
    return tail[end]
```

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

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

String Extraction Puzzle

```python
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
```

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

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