One-on-One Sessions

- Starting **Monday**: 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**: `'Don't'` or `'0' tall`
  - What if need both `"` and `'?`
- **Solution**: escape characters
  - Format: `\ + letter`
  - Special or invisible chars:
    - \': single quote
    - \": double quote
    - \n: new line
    - \t: tab
    - \\: Backslash

String are Indexed

- `s = 'abc 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'`
  - **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` “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] == 'bracadabr'`

Defining a String Function

```python
>>> middle('abc')
'b'
>>> middle('aabbcc')
'bb'
>>> middle('aabbccoo')
'bbb'
```

```python
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
```

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(2.66,1)`

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)
>>>
```

```
x
```

Nothing here!

Method Calls

• Methods calls are unique (right now) to strings
  • Like a function call with a “string in front”

• Method calls have the form
  ```python
  string.name(x,y, ...)
  ```

• The string in front is an additional argument
  • Just one that is not inside of the parentheses
  • Why? Will answer this later in course.

Example: upper()

• upper(): Return an upper case copy
  ```python
  >>> 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

Examples of String Methods

• s.index(s2)
  • Returns position of the first instance of s2 in s1
  ```python
  >>> s = 'Prof (Walker) White'
  >>> firstparens(s)
  'Walker'
  >>> firstparens(' (A) B (C) D')
  'A'
  ```

String Extraction Example

```python
def firstparens(text):
    # 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]
```

```
def second(text):
    # SEARCH second elt in text
    start = text.index(',')
    tail = text[start+1:]
    end = tail.index(',')
    result = tail[:end]
    return result
```

```python
>>> s = 'Prof (Walker) White'
>>> firstparens(s)
'Walker'
>>> firstparens(' (A) B (C) D')
'A'
```