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 '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"
```

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: s_1 in s_2
  - Tests if s_1 “a part of” s_2
  - Say s_1 a *substring* of s_2
  - 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

```
>>> 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
```

Procedures vs. Fruitful Functions

Procedures	Fruitful Functions
• Functions that **do** something	• Functions that give a **value**
• Call them as a **statement**	• Call them in an **expression**
• Example: greet('Walker')	• 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

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

Nothing here!

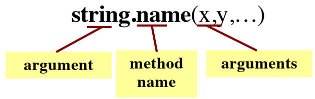
Return

- 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
```

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.

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

Examples of String Methods

- s_1.index(s_2)
  - Returns position of the *first* instance of s_2 in s_1
- s_1.count(s_2)
  - Returns number of times s_2 appears inside of s_1
- s.strip()
  - Returns copy of s with no white-space at *ends*

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

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'
```

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"""

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

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