

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 Labs 5 & 6 (and related)
 - 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**

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 ' ?

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

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

- Solution**: escape characters
 - Format: \ + letter
 - Special or invisible chars

2

String are Indexed

- `s = 'abc d'`

0	1	2	3	4
a	b	c		d
- `s = 'Hello all'`

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l
- 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'
- What is `s[3:6]`?

A: 'lo a'

B: 'lo'

C: 'lo '

D: 'o '

E: I do not know
- Called "string slicing"

3

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'`

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 | Fruitful Functions |
|---------------------------------------|---|
| Functions that do something | Functions that give a value |
| Call them as a statement | Call them in an expression |
| Example: <code>greet('Walker')</code> | Example: <code>x = round(2.56,1)</code> |

Historical Aside

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

6

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
>>>
```

Nothing here!

Return

- Defines a function's value
 - Important for **calculations**
 - But does not display anything

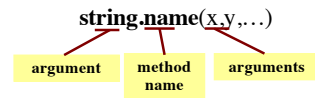
```
def return_plus(n):
    return (n+1)
>>> x = return_plus(2)
>>> x
3
```

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

- s₁.index(s₂)
 - Returns position of the *first* instance of s₂ in s₁

```
>>> s = 'abracadabra'
>>> s.index('a')
0
>>> s.index('rac')
2
```
- s₁.count(s₂)
 - Returns number of times s₂ appears inside of s₁

```
>>> s.count('a')
5
>>> s.count('x')
0
>>> ' a b '.strip()
'a b'
```
- s.strip()
 - Returns copy of s with no white-space at *ends*

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