Today

- More about the **str** type
  - New ways to use strings
- More examples of functions
  - Functions with strings!
- Learn the difference between `print` and `return`

Strings are Indexed (Question 1)

- `s = 'abc d'`
  
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

  - 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”

- `t = 'Hello all'
  
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>e</td>
<td>l</td>
<td>l</td>
<td>o</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  - What is `t[3:6]`?

    - A: 'lo a'
    - B: 'lo'
    - C: 'lo '
    - D: 'o '  
    - E: I do not know

Strings are Indexed (Question 2)

- `s = 'abc d'`
  
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

  - 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”

- `t = 'Hello all'
  
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>e</td>
<td>l</td>
<td>l</td>
<td>o</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  - What is `t[3:6]`?

    - A: 'lo a'
    - B: 'lo'
    - C: 'lo '
    - D: 'o '  
    - E: I do not know

Other Things We Can Do With Strings

**Operator `in`:** `s_1 in s_2`
- Tests if `s_1` “a part of”
  - (or a **substring** of) `s_2`
- Evaluates to a bool

**Examples:**
- `>>> s = 'bracadabra'
  >>> 'a' in s`
  - `True`
- `>>> 't' in s`
  - `False`
- `>>> 'o' in s`
  - `True`
- `>>> 'foo' in s`
  - `False`

**Built-in Function `len`:** `len(s)`
- Value is # of chars in `s`
- Evaluates to an int

**Examples:**
- `>>> s = 'bracadabra'
  >>> len(s)`
  - `11`
- `>>> s[:len(s)-1]`
  - `'bracadab'`
Defining a String Function

Want to write function `middle`, which returns the middle 3rd of a string (length divisible by 3).

How we want it to behave:

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

**Important Questions:**
1. What are the parameters?
2. What is the return value?
3. What goes in the body?

```python
def middle(text):
    return middle_third
```

Steps to writing a program

1. Work an instance yourself
2. Write down exactly what you just did
3. Generalize your steps from 2
4. Test your steps
5. Translate to Code
6. Test program
7. Debug (if necessary)

Advanced String Features: Method Calls

- Strings have some useful **methods**
  - Like functions, but “with a string in front”
- **Format**: `<string name>,<method name>(x,y,…)
- **Example**: `upper()` returns an upper case version

```python
>>> s = 'Hello World'
>>> s[1:5].upper()
'ELLO'
>>> 'HELLO WORLD'.upper()
'SCREAM'
>>> 'cs1110'.upper()
'CS1110'
```

Examples of String Methods

- `s1.index(s2)`
  - Returns position of the first instance of `s2` in `s1`
  - **error** if `s2` is not in `s1`
- `s1.count(s2)`
  - Returns number of times `s2` appears inside of `s1`
- `s.strip()`
  - Returns a copy of `s` with white-space removed at ends

```
s = 'abracadabra'
s1 = 'a b c d e'
s1.upper()
s1.index('a')
s1.count('b')
s1.index('x')
s1.strip()
s1.strip(' a ')
```

Definition of `middle`

```python
def middle(text):
    # IMPORTANT:
    # Precondition requires that arguments to `middle` have length divisible by 3.
    # If not? Bad things could happen, and we blame the user (not the author) of the function.
    return middle_third
```

Examples of String Methods

- `s1.index(s2)`
  - Returns position of the first instance of `s2` in `s1`
  - **error** if `s2` is not in `s1`
- `s1.count(s2)`
  - Returns number of times `s2` appears inside of `s1`
- `s.strip()`
  - Returns a copy of `s` with white-space removed at ends

```
s = 'abracadabra'
s1 = 'a b c d e'
s1.upper()
s1.index('a')
s1.count('b')
s1.index('x')
s1.strip()
s1.strip(' a ')
```
**String Extraction Example**

```python
def firstparens(text):
    r"""Returns: substring in ()
    Uses the first set of parens
    Param text: a string with ()"""
    >>> s = 'One (Two) Three'
    >>> firstparens(s)
    'Two'
    >>> t = '(A) B (C) D'
    >>> firstparens(t)
    'A'
```

**Steps to writing a program**

1. Work an instance yourself
2. Write down exactly what you just did
3. Generalize your steps from 2
4. Test your steps
5. Translate to Code
6. **Test program** Think of all the corner cases
7. Debug (if necessary) What could possibly go wrong?

**String Extraction Puzzle**

```python
def second(thelist):
    r"""Returns: second word in a list
    of words separated by commas, with
    any leading or trailing spaces from the
    second word removed
    Param thelist: a list of words with
    at least two commas"""
    start = thelist.index(',')
    tail = thelist[start+1:]
    end = tail.index(',')
    result = tail[:end]
    return result
```

**Not All Functions Need a Return**

```python
def greet(n):
    r"""Prints a greeting to the name n
    Parameter n: name to greet
    Precondition: n is a string"""
    print('Hello ' + n + '!
    print('How are you?')
```

### print vs. return

- **print**: Displays a value on screen
  - Used primarily for testing
  - Sends a value from a function call frame back to the caller
  - Important for calculations
  - Does not display anything

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

```python
def return_plus(n):
    return n+1
```

```python
>>> print_plus(2)
3
>>> return_plus(2)
3
```

---

### Python Interactive Mode

- Executes both **statements** and **expressions**
- If **expression**:
  1. Evaluates
  2. Prints value (if one exists)

```python
>>> 2+2  # evaluates (performs addition)
4
>>> return_plus(2)  # evaluates (makes function call, gets return value)
3
>>> print value (3)
```
**return_plus** in action

```python
def return_plus(n):
    return n+1
```

Python Interactive Mode

```python
>>> return_plus(2)
3
```

1. Evaluates: makes function call, evaluates to return value
2. Python interactive mode prints that value

**print_plus** in action

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

Python Interactive Mode

```python
>>> print_plus(2)
3
```

1. Evaluates:
   1. makes function call
   2. prints (executes line 1)
   3. return value is **NONE**
2. does not print that value because it's **NONE**

**hybrid_plus** in action

```python
def hybrid_plus(n):
    print(n)
    return n+1
```

Python Interactive Mode

```python
>>> hybrid_plus(2)
3
```

1. Evaluates: makes function call, evaluates to return value
2. Python interactive mode prints that returned value

See the difference in the Python Tutor

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

```python
x1 = print_plus(2)
x2 = return_plus(2)
print(x1)
print(x2)
```

http://cs1110.cs.cornell.edu/tutor/#mode=edit

**Exercise 1**

<table>
<thead>
<tr>
<th>Module Text</th>
<th>Python Interactive Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td># module.py</td>
<td>&gt;&gt;&gt; import module</td>
</tr>
<tr>
<td></td>
<td>&gt;&gt;&gt; print(module.x)</td>
</tr>
<tr>
<td>def foo(x):</td>
<td></td>
</tr>
<tr>
<td>x = 1+2</td>
<td></td>
</tr>
<tr>
<td>x = 3*x</td>
<td></td>
</tr>
</tbody>
</table>

What does Python give me?

- A: 9
- B: 10
- C: 1
- D: None
- E: Error

**Exercise 2**

<table>
<thead>
<tr>
<th>Module Text</th>
<th>Python Interactive Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td># module.py</td>
<td>&gt;&gt;&gt; import module</td>
</tr>
<tr>
<td></td>
<td>&gt;&gt;&gt; print(module.y)</td>
</tr>
<tr>
<td>def foo(x):</td>
<td></td>
</tr>
<tr>
<td>x = 1+2</td>
<td></td>
</tr>
<tr>
<td>x = 3*x</td>
<td></td>
</tr>
</tbody>
</table>

What does Python give me?

- A: 9
- B: 10
- C: 1
- D: None
- E: Error
### Exercise 3

**Module Text**

```python
# module.py

def foo(x):
    x = 1+2
    x = 3*x
    return x+1

y = foo(0)
```

**Python Interactive Mode**

```python
>>> import module
>>> module.y
9
```

**A:** 9  
**B:** 10  
**C:** 1  
**D:** None  
**E:** Error

### Exercise 4

**Function Definition**

```python
def foo(a,b):
    x = a
    y = b
    return x*y+y
```

**Function Call**

```python
>>> x = 2
>>> foo(3,4)
>>> x
16
```

**A:** 2  
**B:** 3  
**C:** 16  
**D:** None  
**E:** I do not know