Example: Python's class str (cont')

• Two ways to apply a method:
  – Using the **object**. Notation:
    \[ \text{object.method}(\text{rest-of-arguments}) \]
  – Using the regular function notation (defined in a module):
    \[ \text{class.method}(\text{object, rest-of-arguments}) \]

The first way is the most common.

• Examples:

```python
>>> c = 'vancouver'
>>> c.capitalize()
'Vancouver'
```
```
>>> c
'veancouver'
```
```
>>> str.capitalize(c)
'Vancouver'
```
```
>>> dna = 'AGTTGGCAGAC'
>>> dna.replace('G', 'C')
'ACTTCCGCACAC'
```
```
>>> str.replace(dna, 'G', 'C')
'ACTTCCGCACAC'
```
```
>>> dna = 'AGTTGGCAGAC'
>>> dna.replace('G', 'C')
'ACTTCCGCACAC'
```
```
>>> str.replace(dna, 'G', 'C')
'ACTTCCGCACAC'
```
```
```
Any identifier (variable, function, method, ...) starting and ending with a double underscore (__) performs some special purpose in Python

- In particular, classes often have many double underscore methods

These methods are called by Python when certain Python syntax is used; for example consider object `obj`

- If you type `str(obj)`, method `obj.__str__()` is called
- If you type `obj % 2`, method `obj.__mod__(2)` is called
- If you type `obj < 2`, method `obj.__lt__(2)` is called
- If you type `help(obj)`, string `obj.__doc__` is shown
Summary

• Classes store data and processes together
  – A class attribute is a variable within a class
  – A class method is a function within a class
• An object is an instance of a class with specific data values
  – Each class can have many objects
  – Each object has only one class
• Class methods are accessed using the same “.” notation as is used to access variables or functions in an imported module
  – Called directly: class.method(object, ...)
  – More common approach: object.method(...)  
• All the data types we have seen in Python are actually classes
END OF MIDTERM MATERIAL