Objectives

At the end of this section, you will be able to:

• Explain the difference between modules, classes and objects
• Explain the difference between variables and attributes
• Explain the difference between functions and methods
• Call methods using two different syntax
Classes

• We have already seen some Python constructs:
  – *Variables* store data
  – *Functions* take input, perform a process and produce output
  – *Modules* contain variables and functions

• A *class* is a construct which allows you to collect together both data and processes in a single entity
  – The class’s *attributes* store the data
  – The class’s *methods* perform processes on that data

• Programming which uses classes is called “object oriented”
  – Programming that only uses functions is called “imperative”

• Classes are typically defined inside modules just like variables or functions
  – Classes are often put in modules all by themselves
Objects

- An object is an instance of a class containing specific data
  - The definition of the class only specifies the general structure of the data (eg: the names of the attributes) and the processes (eg: the methods) that will be applied to that data
  - The object specifies the actual data (eg: the value of the attributes)
  - A running program can have many objects which are all instances of a single class

- A method is essentially a function that is part of a class and whose first argument must be an object of that class
  - Python provides a special syntax to make it easy to call methods

- All objects of a class have the same structure (same attributes but different values) and perform the same methods
Classes vs Objects Example: UBC Students

- Class: Students enrolled at UBC
- Objects: Individual students, such as student_a or student_b
- Class attributes: first_name, last_name, student_number, year, birthdate, program, bedtime_last_night, ...
- Class methods: Attend lecture, write exam, pay tuition, ask name, promote, ...
- student_a attribute values: first_name → 'Alice', last_name → 'Woods', bedtime_last_night → 11pm, ...
- student_b attribute values: first_name → 'Bob', last_name → 'Sethian', ..., bedtime_last_night → None, ...
- While student_a.attend_lecture(8) might return True 
  student_b.attend_lecture(8) might return False (or generate error HungoverError)
Classes vs Objects Example: Accounts

- Class: \textit{Account} (representing a bank account)
- Objects: individual accounts, such as \textit{john\_account}, etc.
- Class attributes: \textit{id, name, balance}
- Class methods:
  - \textit{deposit(amount)}: deposits that amount to the account
  - \textit{withdraw(amount)}: withdraws that amount to the account
- \textit{john\_account} attribute values:
  - id: 13345, name: John, balance: 500.50
- \textit{sarah\_account} attribute values:
  - id: 44455, name: Sarah, balance: 100.00
- Method results:
  - \textit{john\_account.withdraw(200.00)} will leave John's account with 300.50
  - \textit{sarah\_account.deposit(100.00)} will add 100 to Sarah's account
Example: Python's class str

- Typing `help(str)` will display:

```
class str(object):
    str(object='') -> str
    Create a new string object from the given object ...

    Methods defined here:
    capitalize(...)  
        S.capitalize() -> str
        Return a capitalized version of S ...
    find(...)  
        S.find(sub[, start[, end]]) -> int
        Return the lowest index in S where substring sub is found, such that sub
        is contained within S[start:end]. Return -1 on failure
    
    isnumeric(...)  
        S.isnumeric() -> bool
        Return True if there are only numeric characters in S, 
        False otherwise.
    
    replace(...)  
        S.replace(old, new[, count]) -> str
        Return a copy of S with all occurrences of substring old replaced by new. If
        the optional argument count is given, only the first count occurrences are replaced.
```
Example: Python's class str (cont')

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

The first way is the most common.

• Examples:

```python
>>> c = 'vancouver'
>>> c.capitalize()
'Vancouver'
>>> c
'vancouver'
>>> str.capitalize(c)
'Vancouver'
>>> dna = 'AGTTGGCAGAC'
>>> dna.replace('G', 'C')
'ACTTCCCCACAC'
>>> str.replace(dna, 'G', 'C')
'ACTTCCCCACAC'
>>> 
```