

Python Tutorial II – Objects and Classes

Mark A. Austin

University of Maryland

austin@umd.edu

ENCE 201, Spring Semester 2025

August 29, 2025

Overview

- 1 Working with Objects and Classes
- 2 Data Hiding and Encapsulation
- 3 Relationships Among Classes
- 4 Inheritance Mechanisms
- 5 Composition of Object Models
- 6 Working with Groups of Objects
- 7 Case Study: GeoModeling Spatial Entities

Working with Groups of Objects

Pathway From Objects to Groups of Objects

Data Structures

Now that we know how to create objects, the next subject is how to **organize collections** of **objects** so that they are **easy to store**, **easy to find**, and **easy to modify**?

Approach: Two-step procedure:

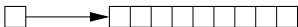
- Choose an appropriate **mathematical formalism**.
- Develop **software** to **support each formalism**.

As a starting point, of objects can be organized into:

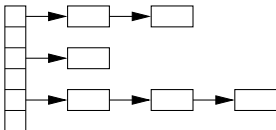
- Arrays
- Linked lists and queues (lists in Python).
- HashMaps (dictionaries in Python).
- Trees and Graphs.

Memory Layout: Arrays, Lists, Queues, Trees, and Graphs

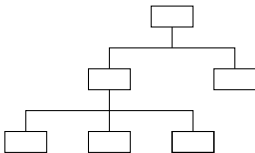
Arrays



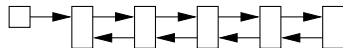
Hash Map



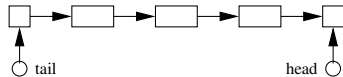
Trees



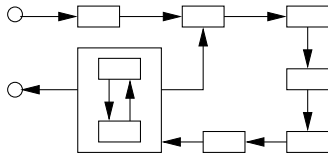
Linked List



Queues



Graphs



Linear and Nonlinear Data Structures

Linear Data Structure:

- Items are arranged in a linear fashion.
 - Simple to implement.
-

Examples:

- **Array:** Sequential arrangement of data elements paired with the index of the data element.
- **Linked List:** Each data element contains a link to another element along with the data present in it.
- **Stack:** LIFO (last in First Out) or FILO (First in Last Out).
- **Queue:** Similar to Stack, but the order of operation is only FIFO (First In First Out).

Linear and Nonlinear Data Structures

Nonlinear Data Structure:

- Items are not ordered in any particular way.
 - Often, items are often organized into hierarchies.
-

Examples:

- **Binary Tree:** Each data element can be connected to maximum two other data elements and it starts with a root node.
- **Hash Table:** Retrieves values using keys rather than index from a data element.
- **Graph:** Arrangement of vertices and nodes where some of the nodes are connected to each other through links.

Python Builtin Data Structures

Lists:

- Lists are used to **store multiple items** in a **single variable**.
- A list may store **multiple types** (heterogeneous) of **elements**.

Dictionary:

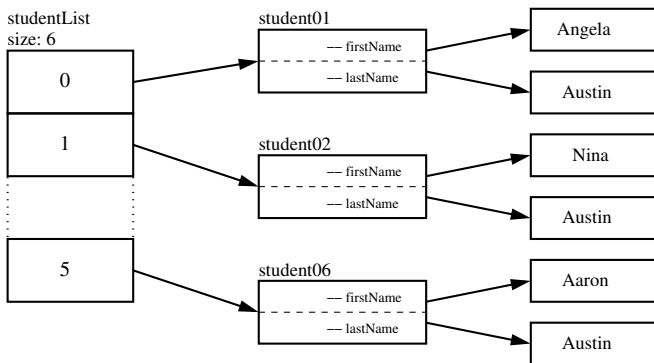
- Dictionaries store **data values** as **key:value pairs**.
- As of Python 3.7, a dictionary is a collection which is ordered, changeable and do not allow duplicates.

Set:

- Sets store **multiple items** in a **single variable**.
- A set is a collection which is unordered, unchangeable (but you can remove items and add new items) and unindexed.

Example 9: Create List of Student Objects

Part I: Program Architecture



Assemble list of six students. Sort and print by name and gpa.

Example 9: Create List of Student Objects

Part II: Assemble Student Objects ...

```
1 # =====
2 # TestStudents02.py: Assemble list of students ...
3 #
4 # Written by: Mark Austin                                February 2023
5 # =====
6
7 from Student import Student
8 from datetime import date
9
10 # main method ...
11
12 def main():
13     print("--- Enter TestStudents02.main()                ... ");
14     print("--- =====                                ... ");
15
16     print("--- ")
17     print("--- Part 1: Create student objects ...")
18
19     student01 = Student( "Angela", "Austin", date(2002, 3, 2), 2023)
20     student01.setGpa(3.5), student01.setSSN(1234)
21
22     student02 = Student( "Nina", "Austin", date(2001, 4, 12), 2025)
23     student02.setGpa(3.2), student02.setSSN(2134)
24
25     student03 = Student( "David", "Austin", date(2000, 6, 8), 2025)
26     student03.setGpa(2.9), student03.setSSN(2143)
```

Example 9: Create List of Student Objects

Part II: Assemble Student Objects ...

```
27
28     student04 = Student( "Marie", "Austin", date(2005, 8, 5), 2026)
29     student04.setGpa(3.9), student04.setSSN(1243)
30
31     student05 = Student( "Albert", "Austin", date(1999, 10, 20), 2026)
32     student05.setGpa(3.8), student05.setSSN(3124)
33
34     student06 = Student( "Aaron", "Austin", date(2002, 12, 2), 2026)
35     student06.setGpa(4.0), student06.setSSN(1131)
36
37     print("--- ")
38     print("--- Part 2: String description of student parameters ...")
39
40     print( student01.__str__() )
41     print( student02.__str__() )
42     print( student03.__str__() )
43     print( student04.__str__() )
44     print( student05.__str__() )
45     print( student06.__str__() )
46
47     print("--- ")
48     print("--- Part 3: Add students to list ... ")
49
50     studentList = [];
51     studentList.append(student01)
52     studentList.append(student02)
53     studentList.append(student03)
```

Example 9: Create List of Student Objects

Part II: Assemble Student Objects ...

```
54     studentList.append(student04)
55     studentList.append(student05)
56     studentList.append(student06)
57
58     print("--- ")
59     print("--- Part 4: Print contents of list ... ")
60
61     i = 0
62     for student in studentList:
63         print ("---   list01[{:d}]: {:6s} --> {:.2f} ...".format( i, student.getFirstName
64             i = i + 1
65
66     print("--- ")
67     print("--- Part 5: Sort list items by first name ... ")
68
69     sort_values = sorted( studentList, key = lambda x: x._firstname )
70
71     i = 0
72     for student in sort_values:
73         print ("---   list01[{:d}]: {:6s} --> {:.2f} ...".format( i, student.getFirstName
74             i = i + 1
75
76     print("--- ")
77     print("--- Part 6: Sort list items by gpa ... ")
78
79     sort_values = sorted( studentList, key = lambda x: x._gpa )
80
81     i = 0
```

Example 9: Create List of Student Objects

Part II: Assemble Student Objects ...

```
82     for student in sort_values:
83         print ("--- list01[{:d}]: {:6s} --> {:.2f} ...".format( i, student.getFirstName
84             i = i + 1
85
86     print("--- ===== ... ");
87     print("--- Finished TestStudents02.main()           ... ");
88
89     # call the main method ...
90
91     main()
```

Example 9: Create List of Student Objects

Part III: Abbreviated Output:

```
--- Enter TestStudents02.main() ...
--- ===== ...
--- Part 1: Create student objects ...
--- Part 2: String description of student parameters ...

--- Student: Angela Austin ...
--- -----
---   Gpa = 3.50, Age = 20, Graduation year = 2023 ..
--- -----

--- Student: Nina Austin ...
--- -----
---   Gpa = 3.20, Age = 21, Graduation year = 2025 ..
--- -----

--- Student: David Austin ...
--- -----
---   Gpa = 2.90, Age = 22, Graduation year = 2025 ..
--- -----
```

Example 9: Create List of Student Objects

Part III: Abbreviated Output: (Continued) ...

```
--- Student: Marie Austin ...
-----
---   Gpa = 3.90, Age = 17, Graduation year = 2026 ..
-----

--- Student: Albert Austin ...
-----
---   Gpa = 3.80, Age = 23, Graduation year = 2026 ..
-----

--- Student: Aaron Austin ...
-----
---   Gpa = 4.00, Age = 20, Graduation year = 2026 ..
-----

--- Part 4: Print contents of list ...
---
---   list01[0]: Angela --> 3.50 ...
---   list01[1]: Nina   --> 3.20 ...
---   list01[2]: David  --> 2.90 ...
```

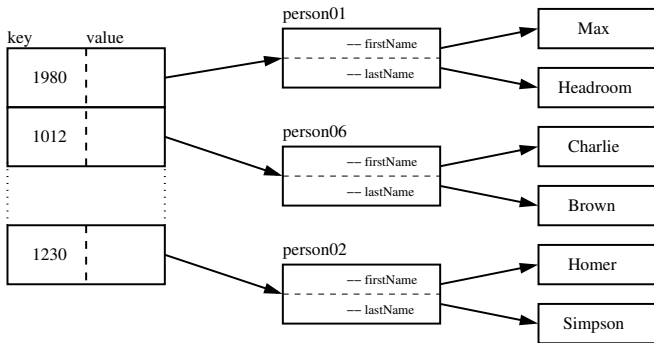
Example 9: Create List of Student Objects

Part III: Abbreviated Output: (Continued) ...

```
--- list01[3]: Marie --> 3.90 ...
--- list01[4]: Albert --> 3.80 ...
--- list01[5]: Aaron --> 4.00 ...
---
--- Part 5: Sort list items by first name ...
--- list01[0]: Aaron --> 4.00 ...
--- list01[1]: Albert --> 3.80 ...
--- list01[2]: Angela --> 3.50 ...
--- list01[3]: David --> 2.90 ...
--- list01[4]: Marie --> 3.90 ...
--- list01[5]: Nina --> 3.20 ...
---
--- Part 6: Sort list items by gpa ...
--- list01[0]: David --> 2.90 ...
--- list01[1]: Nina --> 3.20 ...
--- list01[2]: Angela --> 3.50 ...
--- list01[3]: Albert --> 3.80 ...
--- list01[4]: Marie --> 3.90 ...
--- list01[5]: Aaron --> 4.00 ...
```

Example 10: Dictionary of Fictional Characters

Part I: Program Architecture



Assemble dictionary of six cartoon characters (key = SSN, value = reference to object). Convert dictionary to list, then sort by age.

Example 10: Dictionary of Fictional Characters

Part II: Dictionary of Fictional Characters

```
1 # =====
2 # TestDictionary03.py: Create dictionary of objects ...
3 #
4 # Last Modified: February 2023
5 # =====
6
7 from Person import Person
8
9 # main method ...
10
11 def main():
12     print("--- Enter TestDictionary03.main()    ... ");
13     print("--- ===== ... ");
14
15     # Create cartoon characters ...
16
17     print ("--- Part 01: Create cartoon character objects ...")
18
19     person01 = Person( "Max", "Headroom" )
20     person01.setAge(42)
21     person01.setSSN(1980)
22
23     person02 = Person( "Homer", "Simpson" )
24     person02.setAge(55)
25     person02.setSSN(1230)
```

Example 10: Dictionary of Fictional Characters

Part II: Dictionary of Fictional Characters:

```
27     person03 = Person( "Bart", "Simpson" )
28     person03.setAge(35)
29     person03.setSSN(1231)
30
31     person04 = Person( "Yogi", "Bear" )
32     person04.setAge(65)
33     person04.setSSN(1111)
34
35     person05 = Person( "Charlie", "Brown" )
36     person05.setAge(72)
37     person05.setSSN(1012)
38
39     print ( "--- " )
40     print ( "--- Part 02: Print sample objects ..." )
41     print ( "--- " )
42
43     print( "--- person01 --> {:s} ...".format(person01.__str__() ) )
44     print( "--- person05 --> {:s} ...".format(person05.__str__() ) )
45
46     print ( "--- " )
47     print ( "--- Part 03: Assemble dictionary of cartoon characters ..." )
48
49     cartoon = {}
50     cartoon[ person01.getSSN() ] = person01
51     cartoon[ person02.getSSN() ] = person02
52     cartoon[ person03.getSSN() ] = person03
53     cartoon[ person03.getSSN() ] = person03
```

Example 10: Dictionary of Fictional Characters

Part II: Dictionary of Fictional Characters:

```
54     cartoon[ person04.getSSN() ] = person04
55     cartoon[ person05.getSSN() ] = person05
56
57     print ("--- ")
58     print ("--- Part 04: Retrieve items from dictionary ...")
59     print ("--- ")
60
61     key = 1980
62     personItem = cartoon.get(key)
63     print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )
64
65     key = 1230
66     personItem = cartoon.get(key)
67     print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )
68
69     key = 1231
70     personItem = cartoon.get(key)
71     print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )
72
73     key = 1111
74     personItem = cartoon.get(key)
75     print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )
76
77     key = 1012
78     personItem = cartoon.get(key)
79     print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )
```

Example 10: Dictionary of Fictional Characters

Part II: Dictionary of Fictional Characters:

```

81     print ("--- ")
82     print ("--- Part 04: Convert dictionary to list ...")
83
84     keysList = list( cartoon.keys() )
85     cartoonlist = [];
86     for person in keysList:
87         cartoonlist.append( cartoon.get(person) )
88
89     print ("--- ")
90     print ("--- Part 05: Sort list of cartoon items by age ...")
91     print ("--- ")
92
93     sorted_items = sorted( cartoonlist )
94
95     i = 1
96     for person in sorted_items:
97         print ("---   person[%d]: %s --> %s ..." % ( i, person.getFirstName(), person.getA
98             i = i + 1
99
100    print("--- ===== ... ");
101    print("--- Leave TestDictionary03.main()           ... ");
102
103    # call the main method ...
104
105    main()

```

Example 10: Dictionary of Fictional Characters

Part III: Abbreviated Output:

```
--- Enter TestDictionary03.main()    ...
--- ===== ...
--- Part 01: Create cartoon character objects ...
---
--- Part 02: Print sample objects ...
---
--- person01 --> Person: Max Headroom: age = 42.00 ...
--- person05 --> Person: Charlie Brown: age = 72.00 ...
---
--- Part 03: Assemble dictionary of cartoon characters ...
---
--- Part 04: Retrieve items from dictionary ...
---
--- key = 1980 --> Person: Max Headroom: age = 42.00 ...
--- key = 1230 --> Person: Homer Simpson: age = 55.00 ...
--- key = 1231 --> Person: Bart Simpson: age = 35.00 ...
--- key = 1111 --> Person: Yogi Bear: age = 65.00 ...
--- key = 1012 --> Person: Charlie Brown: age = 72.00 ...
```

Example 10: Dictionary of Fictional Characters

Part III: Abbreviated Output: (Continued) ...

```

--- Part 05: Convert dictionary to list ...
---
--- Part 06: Sort list of cartoon items by age ...
---
---   person[1]: Bart --> 35 ...
---   person[2]: Max --> 42 ...
---   person[3]: Homer --> 55 ...
---   person[4]: Yogi --> 65 ...
---   person[5]: Charlie --> 72 ...
--- ===== ...
--- Leave TestDictionary03.main()          ...

```