Python Tutorial – Part 2: Objects and Classes

Mark A. Austin

University of Maryland

austin@umd.edu
ENCE 688P, Spring Semester 2022

February 21, 2023

Overview

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:

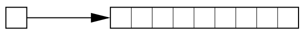
- Choose an appropriate mathematical formalism.
- Develop software to support each formalism.

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

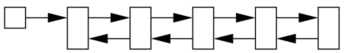
- Arrays
- Linked lists and queues (lists in Python).
- HashMaps (dictionaries in Python).
- Trees and Graphs.

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

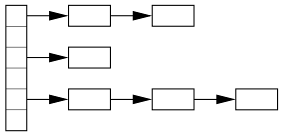
Arrays



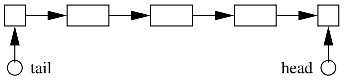
Linked List



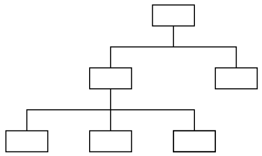
Hash Map



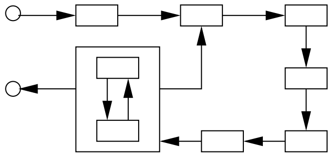
Queues



Trees



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:

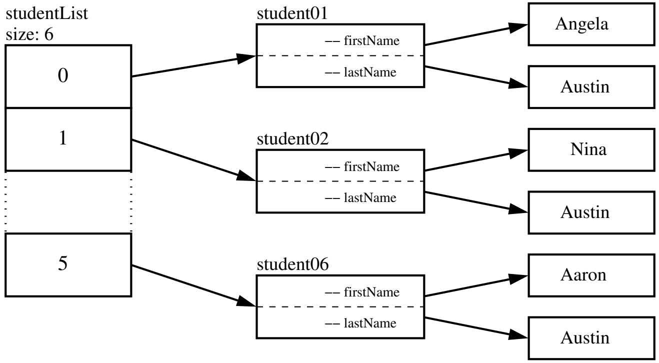
- Dictionaries store data values as key:value pairs.
- As of Python 3.7, a dictionary is a collection which is ordered, changeable and do not allow duplicates.

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 8: Create List of Student Objects

Part I: Program Architecture



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

Example 8: Create List of Student Objects

Part II: Assemble Student Objects ...

```
# ====================================================================
# TestStudents02.py: Assemble list of students ...
#
# Written by: Mark Austin                                 February 2023
# ====================================================================

from Student import Student
from datetime import date

# main method ...

def main():
    print("--- Enter TestStudents02.main()                           ... ");
    print("--- ================================================= ... ");

    print("--- ")
    print("--- Part 1: Create student objects ...")

    student01 = Student( "Angela", "Austin", date(2002, 3, 2), 2023)
    student01.setGpa(3.5), student01.setSSN(1234)

    student02 = Student( "Nina", "Austin", date(2001, 4, 12), 2025)
    student02.setGpa(3.2), student02.setSSN(2134)

    student03 = Student( "David", "Austin", date(2000, 6, 8), 2025)
    student03.setGpa(2.9), student03.setSSN(2143)
```

Example 8: Create List of Student Objects

Part II: Assemble Student Objects ...

```

    student04 = Student( "Marie", "Austin", date(2005, 8, 5), 2026)
    student04.setGpa(3.9), student04.setSSN(1243)

    student05 = Student( "Albert", "Austin", date(1999, 10, 20), 2026)
    student05.setGpa(3.8), student05.setSSN(3124)

    student06 = Student( "Aaron", "Austin", date(2002, 12, 2), 2026)
    student06.setGpa(4.0), student06.setSSN(1131)

    print("--- ")
    print("--- Part 2: String description of student parameters ...")

    print( student01.__str__() )
    print( student02.__str__() )
    print( student03.__str__() )
    print( student04.__str__() )
    print( student05.__str__() )
    print( student06.__str__() )

    print("--- ")
    print("--- Part 3: Add students to list ... ")

    studentList = [];
    studentList.append(student01)
    studentList.append(student02)
    studentList.append(student03)
```

Example 8: Create List of Student Objects

Part II: Assemble Student Objects ...

```
    studentList.append(student04)
    studentList.append(student05)
    studentList.append(student06)

    print("--- ")
    print("--- Part 4: Print contents of list ... ")

    i = 0
    for student in studentList:
       print ("---   list01[{:d}]: {:6s} --> {:.2f} ...".format( i, student.getFirstName
       i = i + 1

    print("--- ")
    print("--- Part 5: Sort list items by first name ... ")

    sort_values = sorted( studentList, key = lambda x: x._firstname )

    i = 0
    for student in sort_values:
       print ("---   list01[{:d}]: {:6s} --> {:.2f} ...".format( i, student.getFirstName
       i = i + 1

    print("--- ")
    print("--- Part 6: Sort list items by gpa ... ")

    sort_values = sorted( studentList, key = lambda x: x._gpa )

    i = 0
```

Example 8: Create List of Student Objects

Part II: Assemble Student Objects ...

```
    for student in sort_values:
       print ("---   list01[{:d}]: {:6s} --> {:.2f} ...".format( i, student.getFirstName
       i = i + 1

    print("--- ==================================================== ... ");
    print("--- Finished TestStudents02.main()                        ... ");

# call the main method ...

main()
```

Example 8: 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 8: 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 ...
```

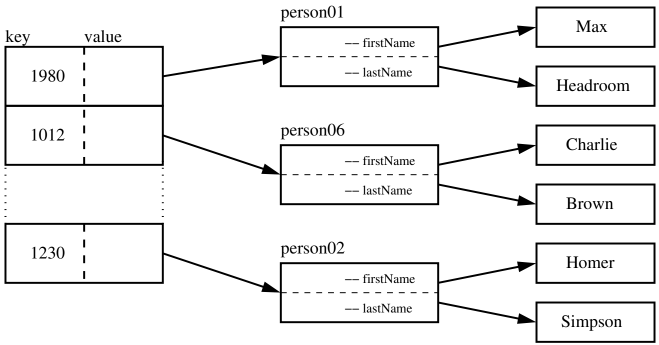
Example 8: Create List of Student Objects

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 9: Create Dictionary of Objects

Part I: Program Architecture



Assemble dictionary of six students (key = SSN, value = reference to object). Convert dictionary to list. Sort by age.

Example 9: Create Dictionary of Objects

Part II: Dictionary of Student Objects:

```
# ===============================================================
# TestDictionary03.py: Create dictionary of objects ...
#
# Last Modified:                                   February 2023
# ===============================================================

from Person import Person

# main method ...

def main():
    print("--- Enter TestDictionary03.main()    ... ");
    print("--- ======================================= ... ");

    # Create cartoon characters ...

    print ("--- Part 01: Create cartoon character objects ...")

    person01 = Person( "Max", "Headroom" )
    person01.setAge(42)
    person01.setSSN(1980)

    person02 = Person( "Homer", "Simpson" )
    person02.setAge(55)
    person02.setSSN(1230)
```

Example 9: Create Dictionary of Objects

Part II: Dictionary of Student Objects:

```
    person03 = Person( "Bart", "Simpson" )
    person03.setAge(35)
    person03.setSSN(1231)

    person04 = Person( "Yogi", "Bear" )
    person04.setAge(65)
    person04.setSSN(1111)

    person05 = Person( "Charlie", "Brown" )
    person05.setAge(72)
    person05.setSSN(1012)

    print ("--- ")
    print ("--- Part 02: Print sample objects ...")
    print ("--- ")

    print("--- person01 --> {:s} ...".format(person01.__str__() ))
    print("--- person05 --> {:s} ...".format(person05.__str__() ))

    print ("--- ")
    print ("--- Part 03: Assemble dictionary of cartoon characters ...")

    cartoon = {}
    cartoon[ person01.getSSN() ] = person01
    cartoon[ person02.getSSN() ] = person02
    cartoon[ person03.getSSN() ] = person03
    cartoon[ person03.getSSN() ] = person03
```

Example 9: Create Dictionary of Objects

Part II: Dictionary of Student Objects:

```
    cartoon[ person04.getSSN() ] = person04
    cartoon[ person05.getSSN() ] = person05

    print ("--- ")
    print ("--- Part 04: Retrieve items from dictionary ...")
    print ("--- ")

    key = 1980
    personItem = cartoon.get(key)
    print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )

    key = 1230
    personItem = cartoon.get(key)
    print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )

    key = 1231
    personItem = cartoon.get(key)
    print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )

    key = 1111
    personItem = cartoon.get(key)
    print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )

    key = 1012
    personItem = cartoon.get(key)
    print("--- key = {:d} --> {:s} ...".format( key, personItem.__str__() ) )
```

Example 9: Create Dictionary of Objects

Part II: Dictionary of Student Objects:

```
    print ("--- ")
    print ("--- Part 04: Convert dictionary to list ...")

    keysList = list( cartoon.keys() )
    cartoonlist = [];
    for person in keysList:
       cartoonlist.append( cartoon.get(person) )

    print ("--- ")
    print ("--- Part 05: Sort list of cartoon items by age ...")
    print ("--- ")

    sorted_items = sorted( cartoonlist )

    i = 1
    for person in sorted_items:
       print ("---    person[%d]: %s --> %s ..." %( i, person.getFirstName(), person.getA
       i = i + 1

    print("--- ======================================= ... ");
    print("--- Leave TestDictionnary03.main()            ... ");

# call the main method ...

main()
```

Example 9: Create Dictionary of Person Objects

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 9: Create Dictionary of Person Objects

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 TestDictionnary03.main()          ...
```