Carlson School of Management
University of Minnesota

Chapter 7
Introduction to Lists

Carlson School of Management

Outline

- Sequence Data Structures
- Creating and Displaying Lists
- Basic List Operations
  - Indexing, iterating, modifying, concatenating
  - Slicing, searching, copying
- List Methods and Functions
  - Appending, inserting, sorting, removing
  - Finding min/max values

Carlson School of Management

Sequence Data Structures

sequence

- Sequence – an object containing multiple data items
  - Lists, tuples, strings, dictionaries and sets
  - Lists are the most general and versatile
    - Can hold a variety of data types
    - Are mutable (unlike tuples) , i.e., their contents can be modified
    - Are dynamic data structures that can expand and contract
    - Have many built-in methods and functions
  - Tuples are more restrictive
    - Immutable, i.e., their contents cannot be changed
    - Better suited for situations when data will not change
    - More secure and faster to process

Carlson School of Management

Creating and Displaying Lists

list()

- What is a **list**?
  - An object that contains multiple items
  - Items are listed in brackets [] separated by commas
  - Each item is called an **element**

element_1	element_2	element_3	....	element_n

- **Lect7_Loan_Lists.py**
  - List of strings

```
>>> customers = ['Ryan', 'Ellen', 'Bob']
```

  - List of numbers

```
>>> payments = [661.44, 1705.96, 1239.68, 3925.64]
>>> loan_ids = list(range(1024, 1031, 3))
```

  - List of items of different data types

```
>>> loan = [1027, 'Ellen Harper', 1705.96]
```

Carlson School of Management

Basic List Operations

0	1	2	3	4	5
element_1	element_2	element_3	element_4	element_5	element_6
-6	-5	-4	-3	-2	-1

- Indexing
  - Access individual list elements through an **index**
  - Index of first element in a list is 0, all the way to length-1

```
>>> nums = [1, 2] * 3  # Repetition operator
>>> nums[0] = 1; nums[3] = 2
>>> size = len(nums) = 6; nums[-2] = 1
>>> nums[size] # IndexError, nums[size-1] = 2
```

- Iterating
  - Without an index

```
>>> for num in nums:
        print(num)
```

0	1	2	3	4	5
1	2	1	2	1	2
-6	-5	-4	-3	-2	-1

  - With an index

```
>>> for idx in range(size):
        print(idx, num[idx])
```

Carlson School of Management

Basic Operations (cont.)

0	1	2	3	4	5
element_1	element_2	element_3	element_4	element_5	element_6
-6	-5	-4	-3	-2	-1

- Modifying
  - Lists are mutable, their elements can be reassigned values

```
>>> for idx in range(2, size):
        nums[idx] = idx + 1
```

0	1	2	3	4	5
1	2	3	4	5	6
-6	-5	-4	-3	-2	-1

  - Initializing and populating

```
>>> rev_nums = [0] * 6
>>> for idx in range(len(rev_nums)):
        rev_nums[idx] = len(rev_nums) - idx
```

0	1	2	3	4	5
6	5	4	3	2	1
-6	-5	-4	-3	-2	-1

- Concatenating
  - Join two lists together into a third list

```
>>> all_nums = nums + rev_nums
```

  - Append one list to another

```
>>> nums += rev_nums
```

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	6	5	4	3	2	1
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Carlson School of Management

Basic Operations (cont.)

0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	6	5	4	3	2	1
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- Slicing
  - Creates a **sublist** from **start** index to **end** index – 1
    ```
    >>> nums[2:4] # Returns 3rd & 4th elements at indx 2 & 3
    ```
  - Variations
    ```
    >>> first4 = nums[:4], remain = nums[7:]
    >>> skip2 = nums[1:10:2], last3 = nums[-3:]
    ```
- Searching
  - Use the `in` operator to determine an item is in the list
  - Use the `not in` operator to determine an item is not in the list
    - Search for customers in the `customers` list
- Copying
  - Assigning one list variable to another will NOT make a copy
  - Omit both the start and the end slicing indices
  - Concatenate blank list to the existing list
  - Initialize a list and populate it with elements in a loop

Carlson School of Management

List Methods and Functions

- `append()` method adds item to the end of the list
  - Append new loan ID and customer to respective lists
    ```
    >>> loan_ids.append(new_id)
    >>> customers.append(new_cust)
    ```
- `index()` method finds the location of an item on the list
  - Get the location of the loan ID and change it
    ```
    >>> id_indx = loan_ids.index(new_id)
    >>> loan_ids[id_indx] = 1034
    ```
  - If the item does not exist `ValueError` is thrown
    ```
    >>> no_indx = loan_ids.index(1031)
    ```
- `insert()` method allows us to insert an item to a specific position on the list
  - Insert another loan ID and customer right before recent additions
    ```
    >>> loan_ids.insert(id_indx, 1031)
    >>> customers.insert(id_indx, 'Max Entermann')
    ```

Carlson School of Management

List Methods and Functions (cont.)

- `remove()` method removes an item from the list
  - Remove the loan id from the list
    ```
    >>> loan_ids.remove(1034)
    ```
  - If the item does not exist `ValueError` is thrown
    ```
    >>> loan_ids.remove(1111)
    ```
- `del` statement removes an element at a specific index
  - Delete customer at the specific index
    ```
    >>> del_indx = customers.index('Craig Holden')
    >>> del customers[del_indx]
    ```

Carlson School of Management

List Methods and Functions (cont.)

- **`sort()`** method rearranges items in ascending order
  - Sort the payments

```
>>> payments.sort()
```

- **`reverse()`** method reverses items in the list
  - Reversing the sorted payments sorts them descending

```
>>> payments.reverse()
```

- **`min/max`** return the lowest/highest item in the list
  - Find lowest/highest payment

```
>>> pmt_min = min(payments)
>>> pmt_max = max(payments)
```

  - Find first/last customer

```
>>> cust_first = min(customers)
>>> cust_last = max(customers)
```

Carlson School of Management

Summary

- Defined sequences and lists
- Showed how to create and manage lists
- Demonstrated many basic list operations
  - Indexing, iterating, modifying and concatenating
  - Slicing, searching and copying
- Used list methods for dynamic updates
  - Append, index, insert, remove
  - Sort, reverse and find min/max values