Lists

BY DR. SABEEHUDDIN HASAN

Lists

- A *list* is a collection of items in a particular order
- It allows you to store sets of information in one place, whether they are just a few or millions of items
- It can include the letters of the alphabet, the digits from 0 to 9, or the names of all the people in your family
- In Python, square brackets ([]) indicate a list,
  - and individual elements in the list are separated by commas
- **cars = ['Suzuki', 'Toyota', 'Honda', 'Changan']**
- **print(cars)**
- ['Suzuki', 'Toyota', 'Honda', 'Changan']

Lists

- ***Accessing Elements in a List***
- Lists are ordered collections, so any element in a list by telling Python the position, or *index*, of the item desired
- For example, let's pull out the first bicycle in the list bicycles:
  - **bicycles = ['trek', 'cannondale', 'redline', 'specialized']**
  - **print(bicycles[0])**
  - trek
- You can also use print(bicycles[0].title()) to print 0^{th} element in titlecase

Lists

- ***Index Positions Start at 0, Not 1***
- Python considers the first item in a list to be at position 0, not position 1
- This is true for most programming languages
- The second item in a list has an index of 1
- The following code accesses bicycles at index 1 and 3
  - **bicycles = ['trek', 'cannondale', 'redline', 'specialized']**
  - **print(bicycles[1])**
  - **print(bicycles[3])**
- This code returns the second and fourth bicycles in the list:
  - cannondale
  - specialized

Lists

- Python has a special syntax for accessing the last element in a list
- If you ask for the item at index -1, Python always returns the last item in the list:
  - **bicycles = ['trek', 'cannondale', 'redline', 'specialized']**
  - **print(bicycles[-1])**
  - This code returns the value '**specialized**'
- The index -2 returns the second item from the end of the list
- the index -3 returns the third item from the end, and so forth

Lists

- ***Using Individual Values from a List***
- You can use individual values from a list just as you would any other variable
- For example, you can use f-strings to create a message based on a value from a list
  - **bicycles = ['trek', 'cannondale', 'redline', 'specialized']**
  - **message = f"My first bicycle was a {bicycles[0].title()}."**
  - **print(message)**
  - My first bicycle was a Trek.

Modifying, Adding, and Removing Elements

- Most lists you create will be *dynamic*,
  - meaning you'll build a list and then add and remove elements from it as your program runs its course
- ***Modifying Elements in a List***
- syntax for modifying an element is similar to the syntax for accessing an element in a list
  - **motorcycles = ['honda', 'yamaha', 'suzuki']**
  - **print(motorcycles)**
  - **motorcycles[0] = 'ducati'**
  - **print(motorcycles)**
  - ['honda', 'yamaha', 'suzuki']
  - ['ducati', 'yamaha', 'suzuki']

Modifying, Adding, and Removing Elements

- ***Adding Elements to a List***
  - **Appending Elements to the End of a List**
  - The simplest way to add a new element to a list is to *append* the item to the list
    - **motorcycles = ['honda', 'yamaha', 'suzuki']**
    - **print(motorcycles)**
    - **motorcycles.append('ducati')**
    - **print(motorcycles)**
  - Here the append() method adds 'ducati' to the end of the list
    - ['honda', 'yamaha', 'suzuki']
    - ['honda', 'yamaha', 'suzuki', 'ducati']

Modifying, Adding, and Removing Elements

- The append() method makes it easy to build lists dynamically
- start with an empty list and then add items to the list using a series of append() calls
  - **motorcycles = []**
  - **motorcycles.append('honda')**
  - **motorcycles.append('yamaha')**
  - **motorcycles.append('suzuki')**
  - **print(motorcycles)**
- Resulting list looks exactly the same as the lists in the previous examples
  - ['honda', 'yamaha', 'suzuki']

Modifying, Adding, and Removing Elements

- ***Adding Elements to a List***
  - **Inserting Elements into a List**
  - You can add a new element at any position in your list by using the insert() method
    - **motorcycles = ['honda', 'yamaha', 'suzuki']**
    - **motorcycles.insert(0, 'ducati')**
    - **print(motorcycles)**
  - The insert() method opens a space at position 0 and stores the value 'ducati' at that location
    - ['ducati', 'honda', 'yamaha', 'suzuki']

Modifying, Adding, and Removing Elements

- ***Removing Elements from a List***
  - If a user decides to cancel his/her account on a web application you created, you'll want to remove that user from the list of active users
  - You can remove an item according to its position in the list or according to its value
  - **Removing an Item Using the del Statement**
    - If you know the position of the item you want to remove from a list, you can use the del statement
    - **motorcycles = ['honda', 'yamaha', 'suzuki']**
    - **print(motorcycles)**
    - **del motorcycles[0]**
    - **print(motorcycles)**
    - ['honda', 'yamaha', 'suzuki']
    - ['yamaha', 'suzuki']

Modifying, Adding, and Removing Elements

- **Removing an Item Using the pop() Method**
  - The pop() method removes the last item in a list, but it lets you work with that item after removing it
  - Think of list as a stack of items and popping one item off the top of the stack
    - **motorcycles = ['honda', 'yamaha', 'suzuki']**
    - **print(motorcycles)**
    - **popped_motorcycle = motorcycles.pop()**
    - **print(motorcycles)**
    - **print(popped_motorcycle)**
  - 'suzuki' was removed from the end of the list and is now assigned to the variable popped_motorcycle
    - ['honda', 'yamaha', 'suzuki']
    - ['honda', 'yamaha']
    - suzuki

Modifying, Adding, and Removing Elements

- **Removing an Item Using the pop() Method**
  - Imagine that the motorcycles in the list are stored in chronological order according to the date of purchase
  - If this is the case, we can use the pop() method to print a statement about the last motorcycle we bought
    - **motorcycles = ['honda', 'yamaha', 'suzuki']**
    - **last_owned = motorcycles.pop()**
    - **print(f"The last motorcycle I bought was a {last_owned.title()}.")**
  - The output is a simple sentence about the most recent motorcycle we bought
    - The last motorcycle I bought was a Suzuki.

Modifying, Adding, and Removing Elements

- **Removing an Item by Value**
  - If you only know the value of the item you want to remove, you can use the remove() method
  - For example, say we want to remove the value 'ducati' from the list of motorcycles
    - **motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']**
    - **print(motorcycles)**
    - **motorcycles.remove('ducati')**
    - **print(motorcycles)**
  - Here the remove() method tells Python to figure out where 'ducati‘ appears in the list and remove that element
    - ['honda', 'yamaha', 'suzuki', 'ducati']
    - ['honda', 'yamaha', 'suzuki']

Modifying, Adding, and Removing Elements

- **Removing an Item by Value**
  - You can also use the remove() method to work with a value that's being removed from a list
    - **motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']**
    - **print(motorcycles)**
    - **too_expensive = 'ducati'**
    - **motorcycles.remove(too_expensive)**
    - **print(motorcycles)**
    - **print(f"\nA {too_expensive.title()} is too expensive for me.")**
  - *The remove() method deletes only the first occurrence of the value you specify*
    - ['honda', 'yamaha', 'suzuki', 'ducati']
    - ['honda', 'yamaha', 'suzuki']
    - A Ducati is too expensive for me.

Organizing a List

- Sometimes you'll want to preserve the original order of your list, and other times you'll want to change the original order
- Python provides a number of different ways to organize your lists, depending on the situation
- ***Sorting a List Permanently with the sort() Method***
  - Python's sort() method makes it relatively easy to sort a list
  - Imagine we have a list of cars and want to change the order of the list to store them alphabetically
    - **cars = ['bmw', 'audi', 'toyota', 'subaru']**
    - **cars.sort()**
    - **Print (cars)**
    - ['audi', 'bmw', 'subaru', 'toyota']

Organizing a List

- You can also sort this list in reverse-alphabetical order by passing the
  - **argument reverse=True to the sort() method**
  - **cars = ['bmw', 'audi', 'toyota', 'subaru']**
  - **cars.sort(reverse=True)**
  - **print(cars)**
- Again, the order of the list is permanently changed:
  - ['toyota', 'subaru', 'bmw', 'audi']

Organizing a List

- ***Sorting a List Temporarily with the sorted() Function***
  - The sorted() function lets you display your list in a particular order, but doesn't affect the actual order of the list
    - **cars = ['bmw', 'audi', 'toyota', 'subaru']**
    - **print("Here is the original list:")**
    - **print(cars)**
    - **print("\nHere is the sorted list:")**
    - **print(sorted(cars))**
    - **print("\nHere is the original list again:")**
    - **print(cars)**

Organizing a List

- The output is very straight forward
  - Here is the original list:
  - ['bmw', 'audi', 'toyota', 'subaru']
  - Here is the sorted list:
  - ['audi', 'bmw', 'subaru', 'toyota']
  - Here is the original list again:
  - ['bmw', 'audi', 'toyota', 'subaru']
- Notice that the list still exists in its original order 1 after the sorted() function has been used
- The sorted() function can also accept a **reverse=True** argument if you want to display a list in reverse-alphabetical order

Organizing a List

- ***Printing a List in Reverse Order***
  - To reverse the original order of a list, you can use the reverse() method
    - cars = ['bmw', 'audi', 'toyota', 'subaru']
    - print(cars)
    - cars.reverse()
    - print(cars)
  - Notice that reverse() doesn't sort backward alphabetically; it simply reverses the order of the list:
    - ['bmw', 'audi', 'toyota', 'subaru']
    - ['subaru', 'toyota', 'audi', 'bmw']

Organizing a List

- ***Finding the Length of a List***
  - You can quickly find the length of a list by using the len() function
  - **>>> cars = ['bmw', 'audi', 'toyota', 'subaru']**
  - **>>> len(cars)**
  - 4

Avoiding Index Errors When Working with Lists

- There's one type of error that's common to see when you're working with lists for the first time
  - **motorcycles = ['honda', 'yamaha', 'suzuki']**
  - **print(motorcycles[3])**
- This example results in an *index error*:

```
Traceback (most recent call last):
      File "motorcycles.py", line 2, in <module>
            print(motorcycles[3])
                  ~~~~~~~~~~^^^
IndexError: list index out of range
```

- Because of the off-by-one nature of indexing in lists, this error is typical
- **Python starts indexing from 0 and not 1**

Avoiding Index Errors When Working with Lists

- Keep in mind that whenever you want to access the last item in a list, you should use the index -1
  - **motorcycles = ['honda', 'yamaha', 'suzuki']**
  - **print(motorcycles[-1])**
- The index -1 always returns the last item in a list, in this case the value 'suzuki'
- The only time this approach will cause an error is when you request the last item from an empty list:
  - motorcycles = []
  - print(motorcycles[-1])
- No items are in motorcycles, so Python returns another index out of range error