Storing Collections of Data Using Lists

- Up to this point, we have seen numbers, Boolean values, strings, functions, and a few other types.

- Once one of these objects has been created, it can't be modified.

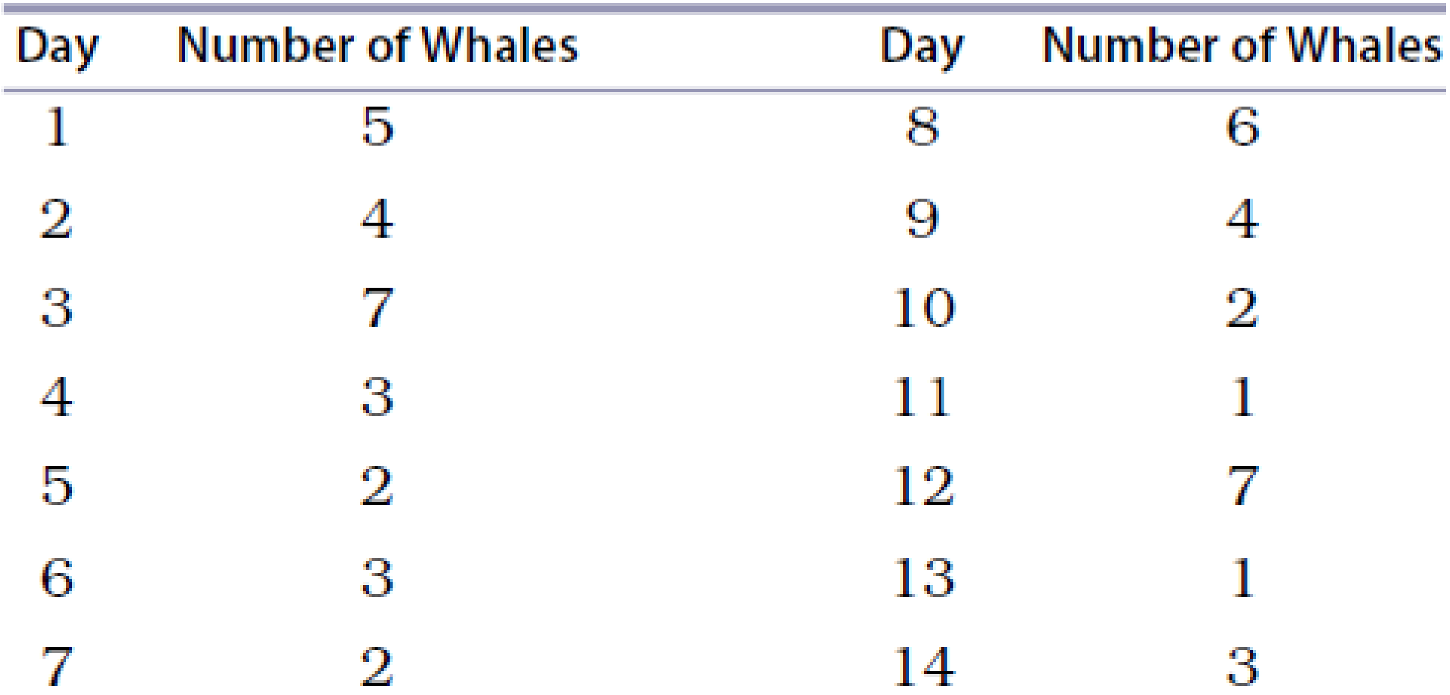
- In this lecture, you will learn how to use a Python type named list.

- Lists contain zero or more objects and are used to keep track of collections of data.

- Unlike the other types you've learned about, lists can be modified.

Storing and Accessing Data in Lists

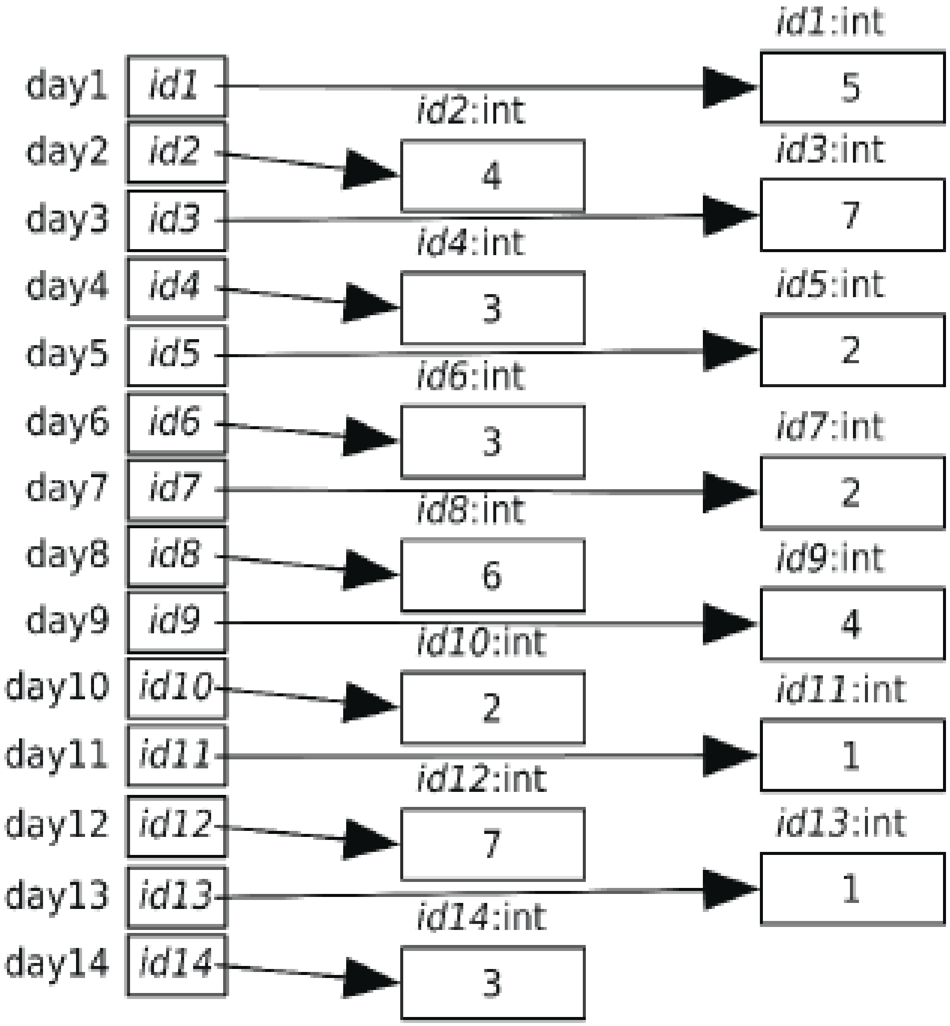
shows the number of gray whales counted in research study in two weeks:



Day	Number of Whales	Day	Number of Whales
1	5	8	6
2	4	9	4
3	7	10	2
4	3	11	1
5	2	12	7
6	3	13	1
7	2	14	3

Question: How many variables are needed to keep track of the number of whales counted each day?

Storing and Accessing Data in Lists



Storing and Accessing Data in Lists

Rather than dealing with this programming nightmare, we can use a *list* to keep track of the 14 days of whale counts.

That is, we can use a list to keep track of the 14 int objects that contain the counts:

```
>>> whales = [5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
>>> whales
[5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
```

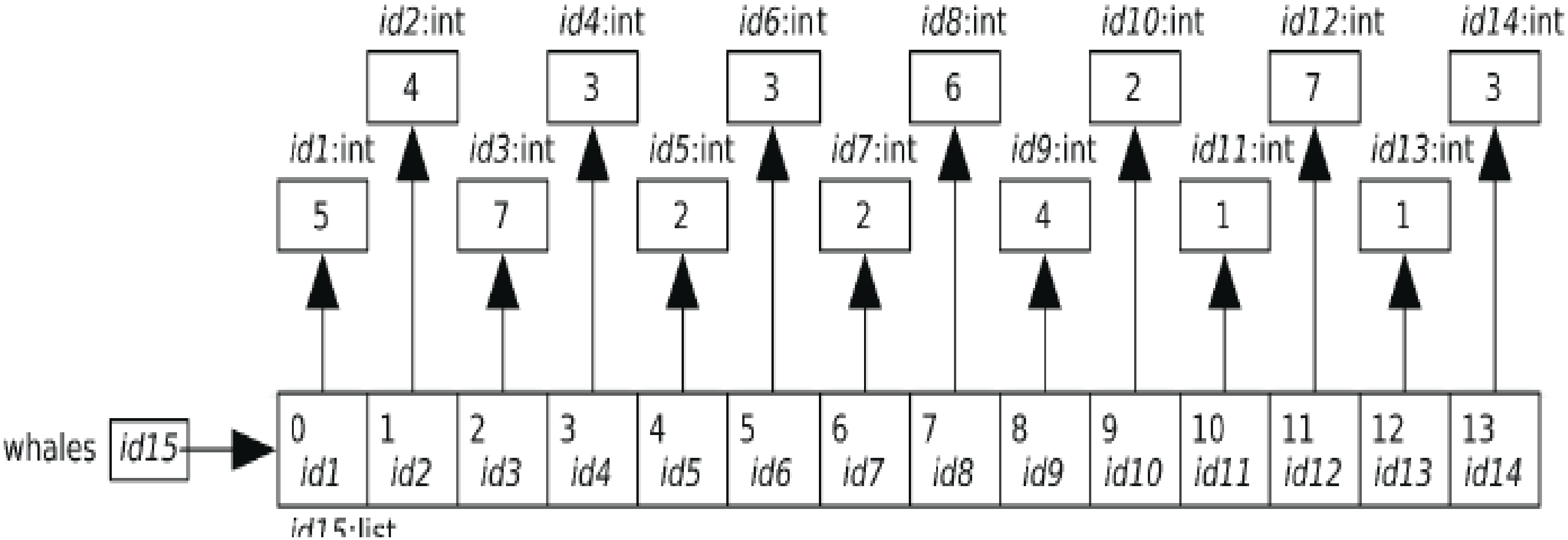
The general form of a list expression is as follows:

[*«expression1»*, *«expression2»*, ... , *«expressionN»*]

The empty list is expressed as [].

Storing and Accessing Data in Lists

A list is an object; like any other object, it can be assigned to a variable. Here is what happens in the memory model:



Question: What is the difference between the structure in slide 3 and this structure?

Storing and Accessing Data in Lists

- List is an object with memory address

- The items in a list are ordered, and each item has an *index* indicating its position in the list.

- The first item in a list is at index 0, the second at index 1, and so on.

- To refer to a particular list item, we put the index in brackets after a reference to the list (such as the name of a variable):

Ex:

```
>>> whales = [5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
>>> whales[0]
>>> whales[1]
>>> whales[7]
>>> whales[100]
```

Storing and Accessing Data in Lists

Unlike most programming languages, Python lets us index backward from the end of a list.

The last item is at index -1, the one before it at index -2, and so on.

Negative indices provide a way to access the list reversibly:

```
>>> whales = [5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
>>> whales[-1]
>>> whales[-2]
>>> whales[-14]
>>> whales[-15]
```

Also we can assign any item to variable

Ex:

```
>>> third = whales[2]
>>> print('Third day:', third)
```

Lists Are Heterogeneous

Lists can contain any type of data, including integers, strings, and even other lists.

Here is a list of information about the element krypton, including its name, symbol, melting point (in degrees Celsius), and boiling point (also in degrees Celsius):

```
>>> krypton = ['Krypton', 'Kr', -157.2, -153.4]
>>> krypton[1]
>>> krypton[2]
```

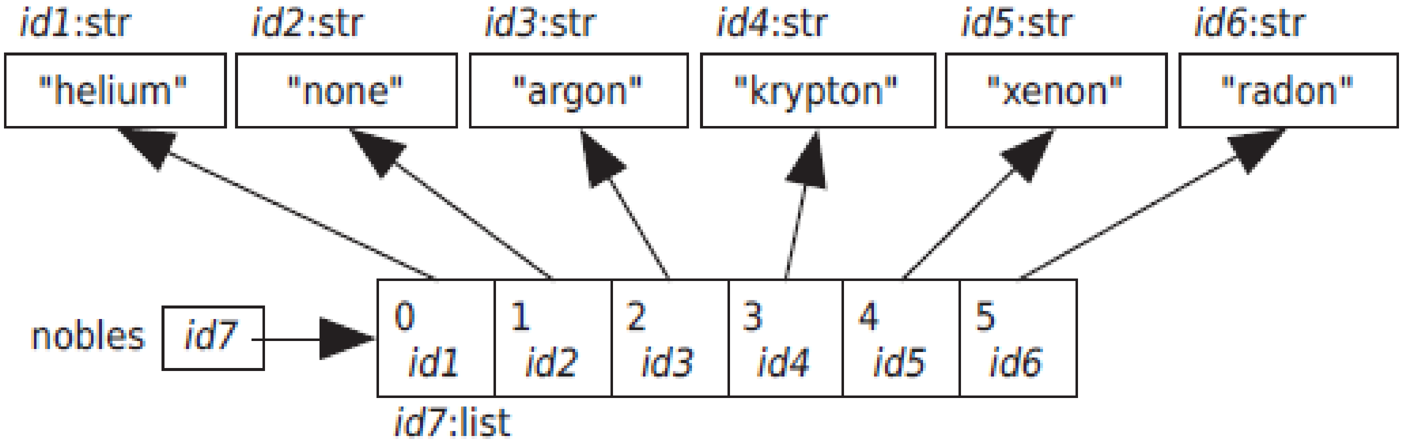
Modifying Lists

Suppose you’re typing in a list of the noble gases and your fingers slip:

>>> nobles = ['helium', 'none', 'argon', 'krypton', 'xenon', 'radon']

The error here is that you typed 'none' instead of 'neon'.

Here’s the memory model

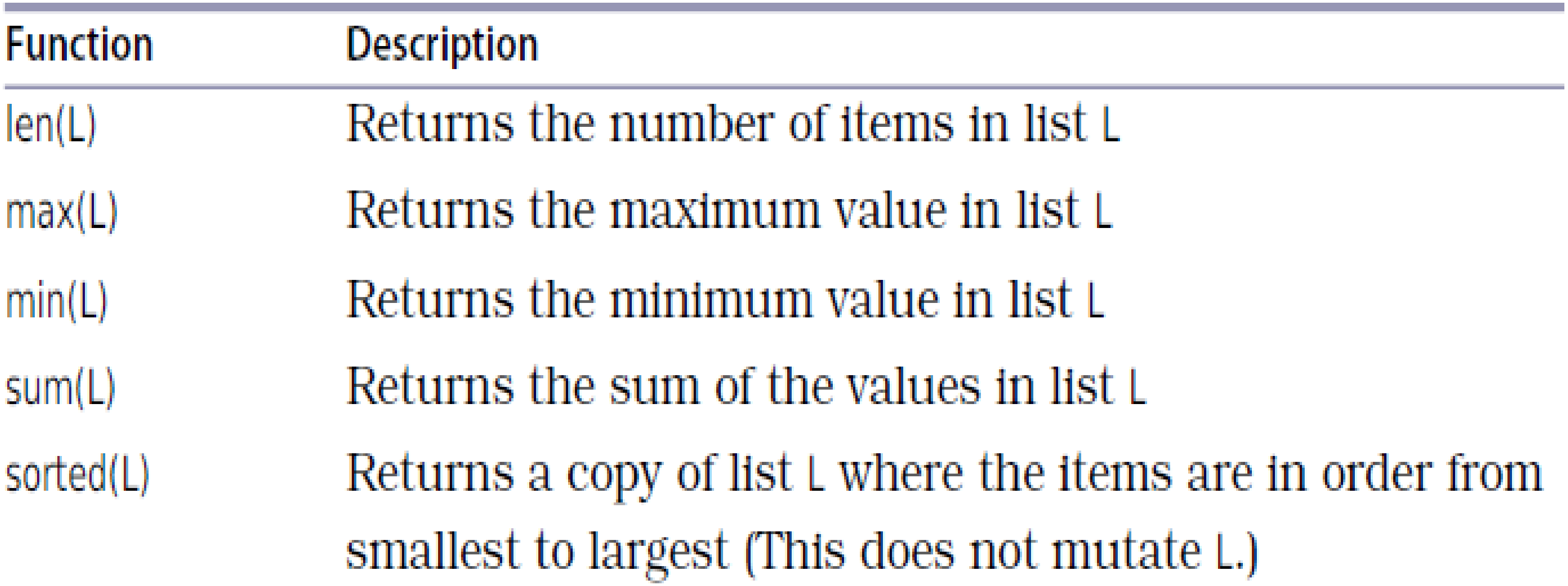


Rather than retyping the whole list, you can assign a new value to a specific element of the list:

>>> nobles[1] = 'neon'

>>> nobles

Operations on Lists



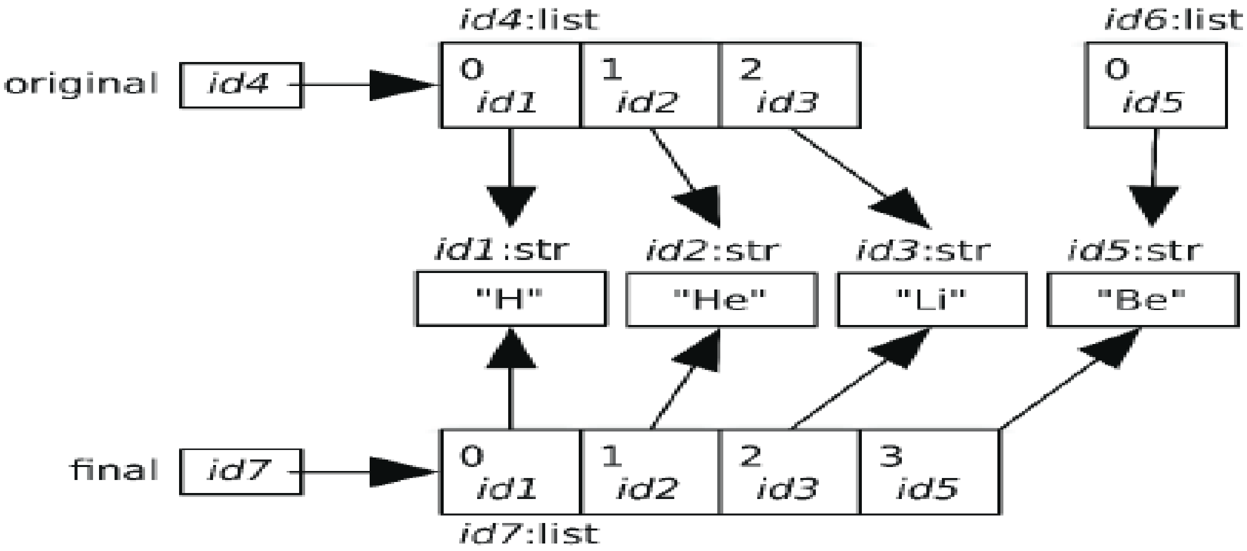
Function	Description
len(L)	Returns the number of items in list L
max(L)	Returns the maximum value in list L
min(L)	Returns the minimum value in list L
sum(L)	Returns the sum of the values in list L
sorted(L)	Returns a copy of list L where the items are in order from smallest to largest (This does not mutate L.)

Operations on Lists

some of the operators can be applied to lists. Like strings, lists can be combined using the concatenation (+) operator:

```
>>> original = ['H', 'He', 'Li']
>>> final = original + ['Be']
>>> final
```

This code doesn't mutate either of the original list objects. Instead, it creates a new list whose entries refer to the items in the original lists.



Question: Can This operation works (why)?

```
>>> ['H', 'He', 'Li'] + 'Be'
```

Operations on Lists

multiply a list by an integer

```
>>> metals = ['Fe', 'Ni']
>>> metals * 3
```

Delete function:

```
>>> metals = ['Fe', 'Ni']
>>> del metals[0]
>>> metals
```

The In Operator on Lists:

```
>>> nobles = ['helium', 'neon', 'argon', 'krypton', 'xenon', 'radon']
>>> gas = input('Enter a gas: ')
Enter a gas: argon
>>> if gas in nobles:
... print('{} is noble.'.format(gas))
...
argon is noble.
>>> gas = input('Enter a gas: ')
Enter a gas: nitrogen
>>> if gas in nobles:
... print('{} is noble.'.format(gas))
```

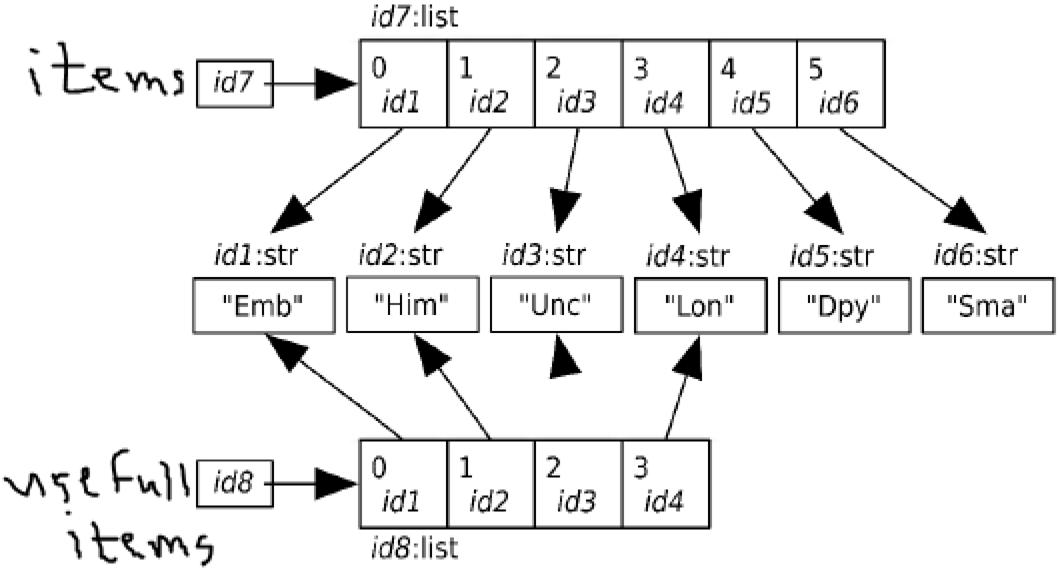
Slicing Lists

>>> items = ['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']

We can create useful item according to a specific criteria in the problem case

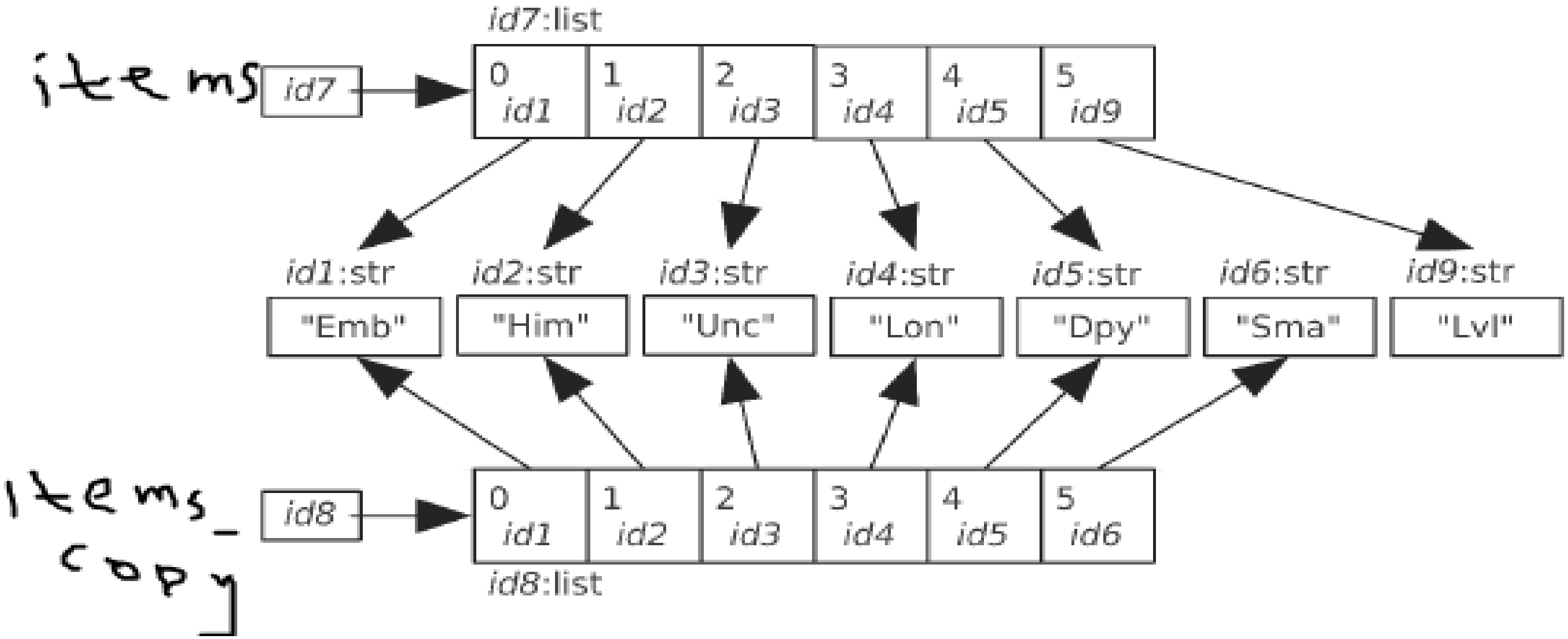
useful_items = items[0:4]

This creates a new list consisting of only the four useful items, which are the first four items from the list items:



Slicing Lists

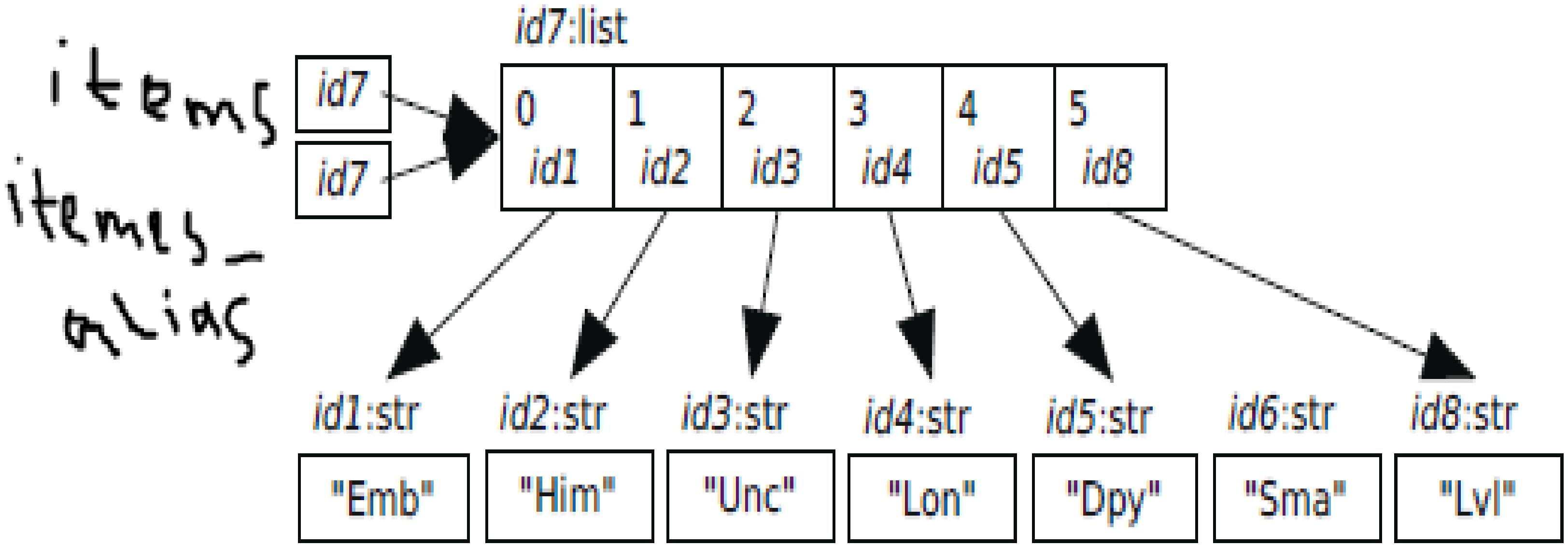
```
>>> items= ['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
>>> items[:4]       4 exclusive,
>>> items[4:]        4 incluisve
>>> items = ['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
>>> items_copy=items[:]
>>> items[5] = 'Lvl'
>>> item_copy
```



Aliasing: What's in a Name?

An *alias* is an alternative name for something. In Python, two variables are said to be aliases when they contain the same memory address.

```
>>> items = ['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
>>> items_alias = items
>>> items[5] = 'Lvl'
```



Mutable Parameters

Here is a simple function that takes a list, removes its last item, and returns the list:

```
>>> L=[1,2,3,4]
>>> L
[1, 2, 3, 4]
>>> del L[-1]
```

del used to release the memory immediately in python list.

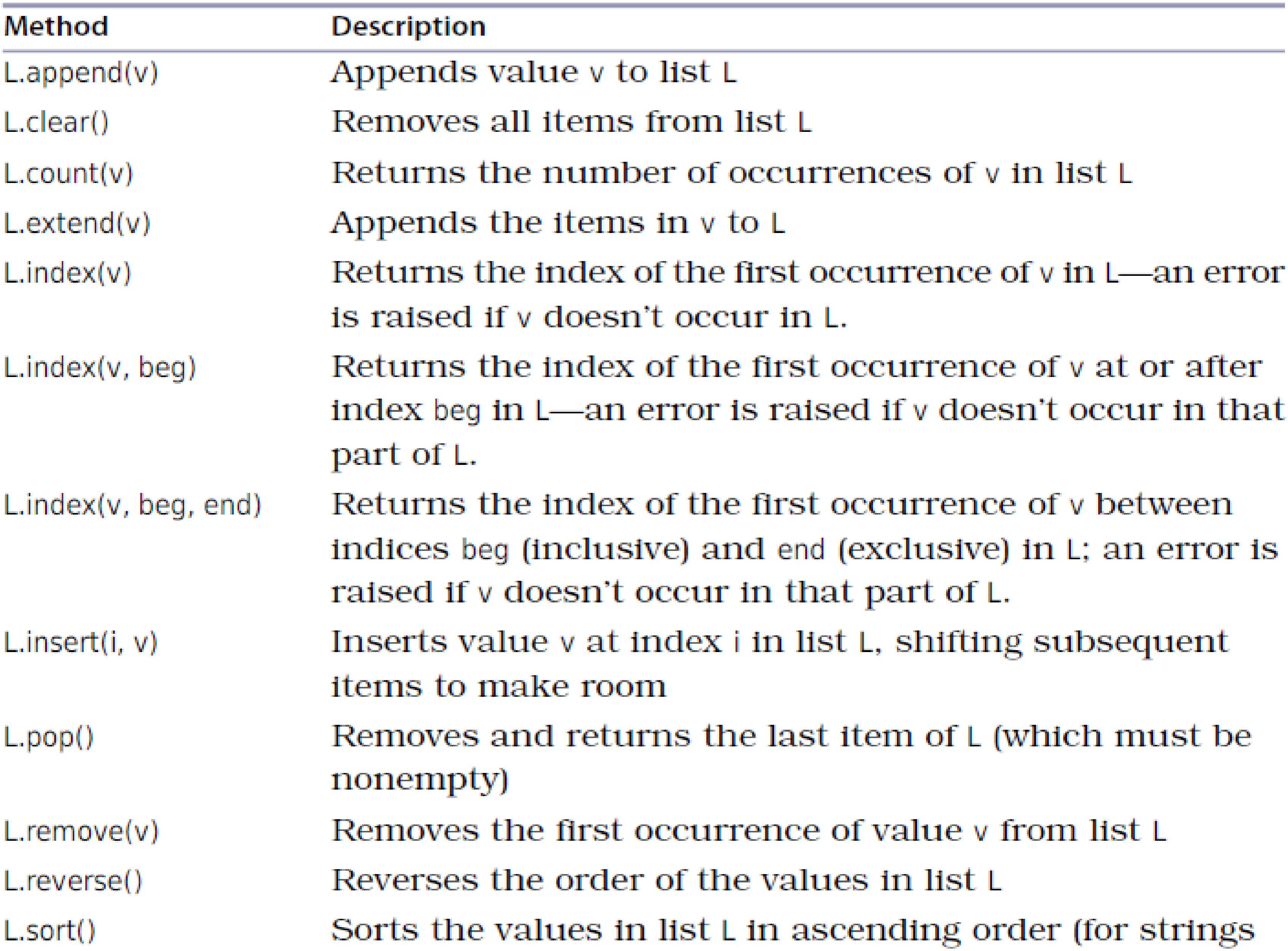
List Methods

List like any other objects, has many methods – accessible via list period(.) and the method name.

Here are some methods:

```
>>> colors = ['red', 'orange', 'green']          ?
>>> colors.extend(['black', 'blue'])             ?
>>> colors.append('purple')                      ?
>>> colors.remove('black')                       ?
```

List Methods



Method	Description
L.append(v)	Appends value v to list L
L.clear()	Removes all items from list L
L.count(v)	Returns the number of occurrences of v in list L
L.extend(v)	Appends the items in v to L
L.index(v)	Returns the index of the first occurrence of v in L—an error is raised if v doesn't occur in L.
L.index(v, beg)	Returns the index of the first occurrence of v at or after index beg in L—an error is raised if v doesn't occur in that part of L.
L.index(v, beg, end)	Returns the index of the first occurrence of v between indices beg (inclusive) and end (exclusive) in L; an error is raised if v doesn't occur in that part of L.
L.insert(i, v)	Inserts value v at index i in list L, shifting subsequent items to make room
L.pop()	Removes and returns the last item of L (which must be nonempty)
L.remove(v)	Removes the first occurrence of value v from list L
L.reverse()	Reverses the order of the values in list L
L.sort()	Sorts the values in list L in ascending order (for strings

Important note to know

Note that method does not create a list rather it performs operation on list

Programmers occasionally forget that many list methods return None rather than creating and returning a new list. As a result, lists sometimes seem to disappear:

```
>>> colors = 'red orange yellow green blue purple'.split()   ?
>>> colors                    ?
>>> sorted_colors = colors.sort()                    ?
>>> print(sorted_colors)                             ?
```

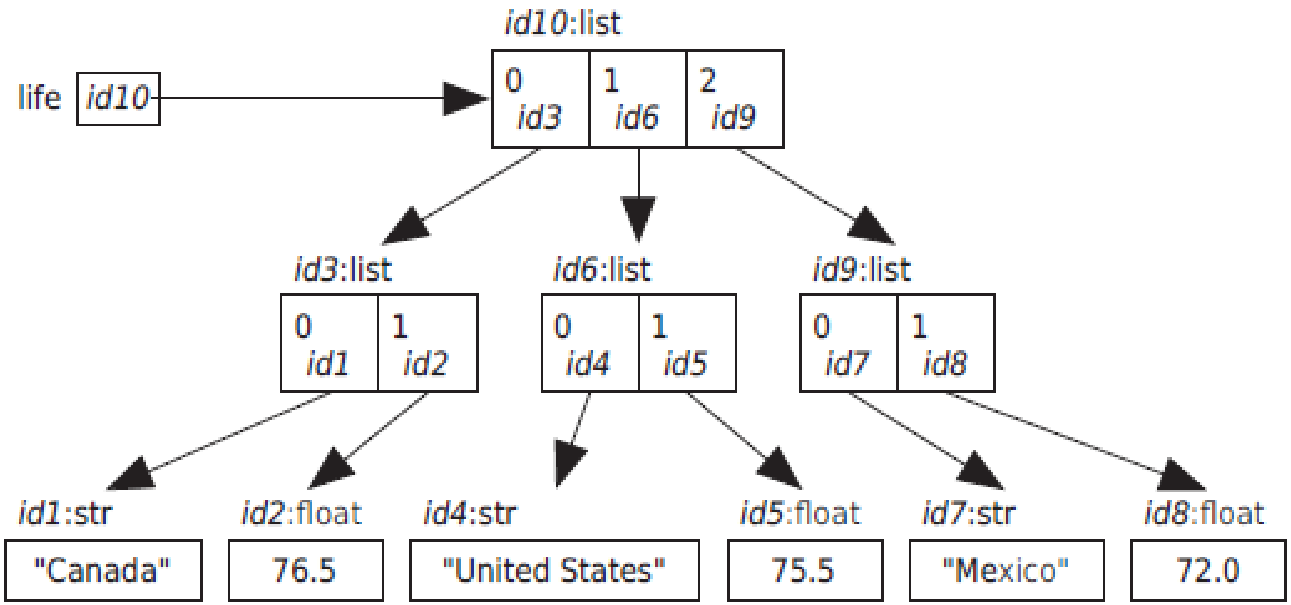
In this example, colors.sort() did two things: it sorted the items in the list, and it returned the value None.

Working with a List of Lists

Lists Are Heterogeneous, lists can contain any type of data. That means that they can contain other lists. A list whose items are lists is called a *nested list*.

Ex:

```
>>> life = [['Canada', 76.5], ['United States', 75.5], ['Mexico', 72.0]]
```



Working with a List of Lists

Notice that each item in the outer list is itself a list of two items. We use the standard indexing notation to access the items in the outer list:

```
>>> life = [['Canada', 76.5], ['United States', 75.5], ['Mexico', 72.0]]
>>> life[0]
?
>>> life[1]
?
[>>> life[2]
?
```

How to access an item in the nested list

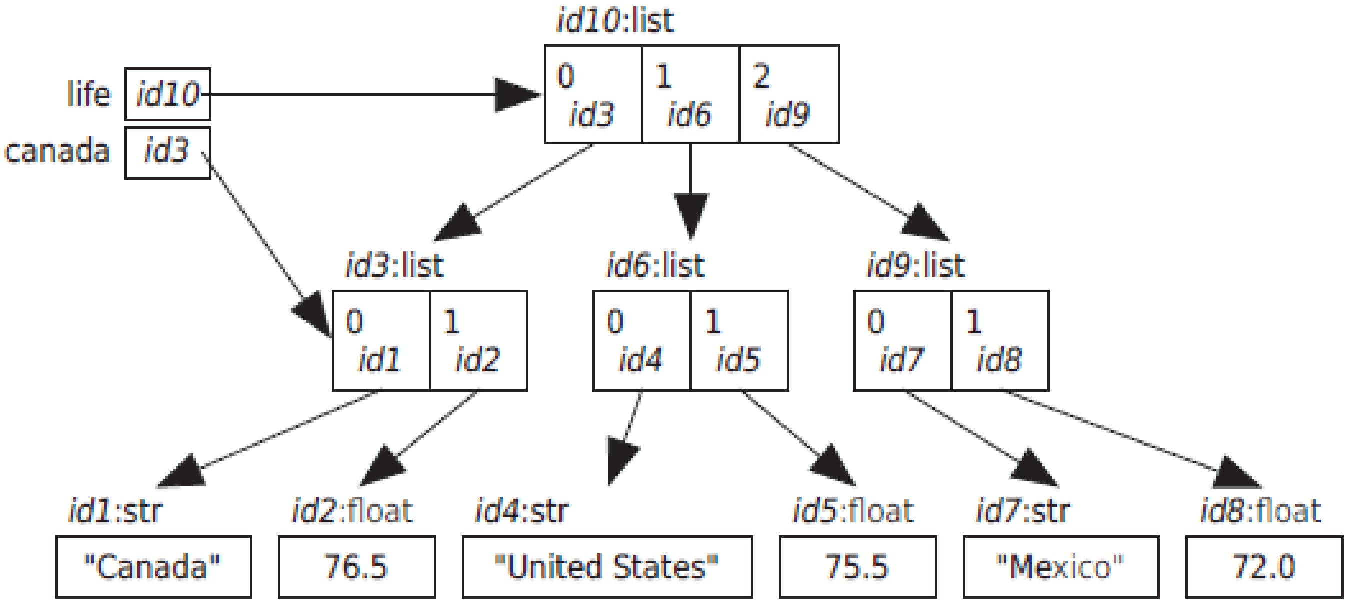
Since each of these items is also a list, we can index it again, just as we can chain together method calls or nest function calls:

```
>>> life = [['Canada', 76.5], ['United States', 75.5], ['Mexico', 72.0]]
>>> life[1]
['United States', 75.5]
>>> life[1][0]
?
>>> life[1][1]
?
```

Sublists assignment

```
>>> life = [['Canada', 76.5], ['United States', 75.5], ['Mexico', 72.0]]
>>> canada = life[0]
>>> canada                    ?
>>> canada[0]                  ?
>>> canada[1]                 ?
```

Assigning a sublist to a variable creates an alias for that sublist:



alias for that sublist

As before, any change we make through the sublist reference will be seen when we access the main list, and vice versa:

EX:

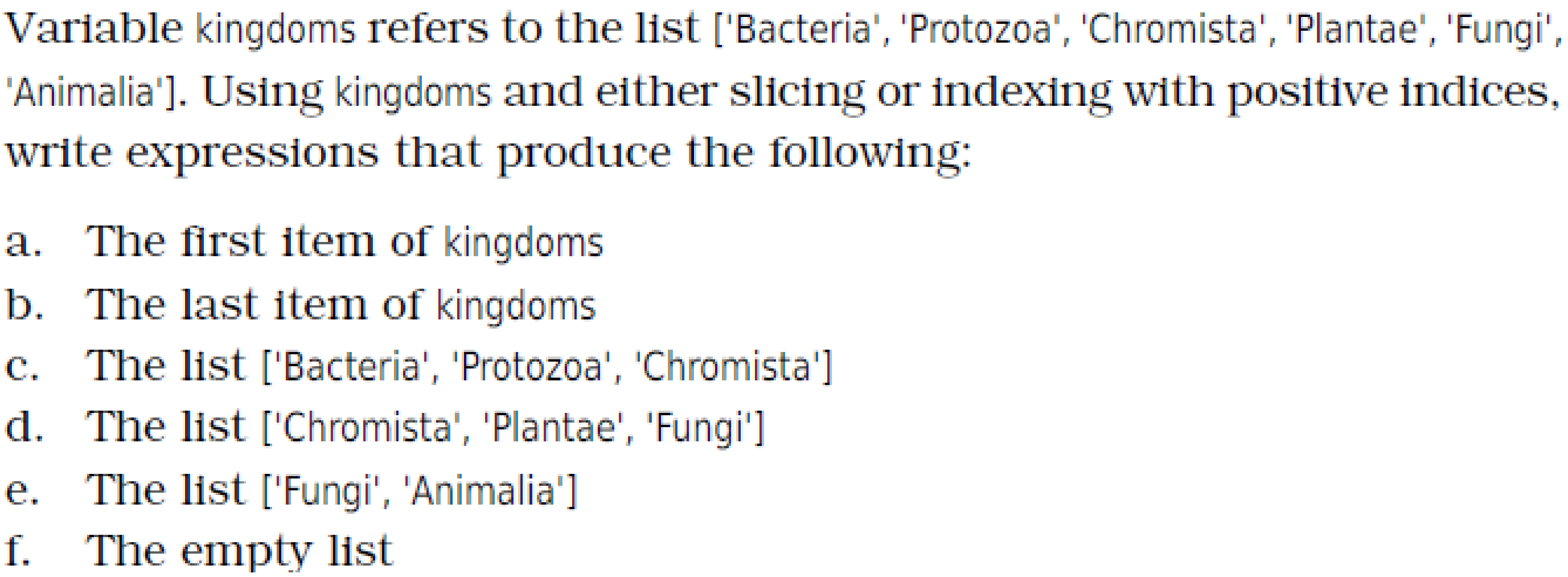
```
>>> life = [['Canada', 76.5], ['United States', 75.5], ['Mexico',
72.0]]
>>> canada = life[0]
>>> canada[1] = 80.0
>>> canada
 ?
>>> life
?
```

A Summary List

In this lecture(s) we learned :

- Lists are used to keep track of zero or more objects.
- The objects in a list are called items or elements.
- Each item has a position in the list called an index and that position ranges from zero to one less than the length of the list.
- Lists can contain any type of data, including other lists.
- Lists are mutable, which means that their contents can be modified.
- Slicing is used to create new lists that have the same values or a subset of the values of the originals.
- When two variables refer to the same object, they are called aliases.

In Class Act-1



Variable `kingdoms` refers to the list `['Bacteria', 'Protozoa', 'Chromista', 'Plantae', 'Fungi', 'Animalia']`. Using `kingdoms` and either slicing or indexing with positive indices, write expressions that produce the following:

a. The first item of `kingdoms`

b. The last item of `kingdoms`

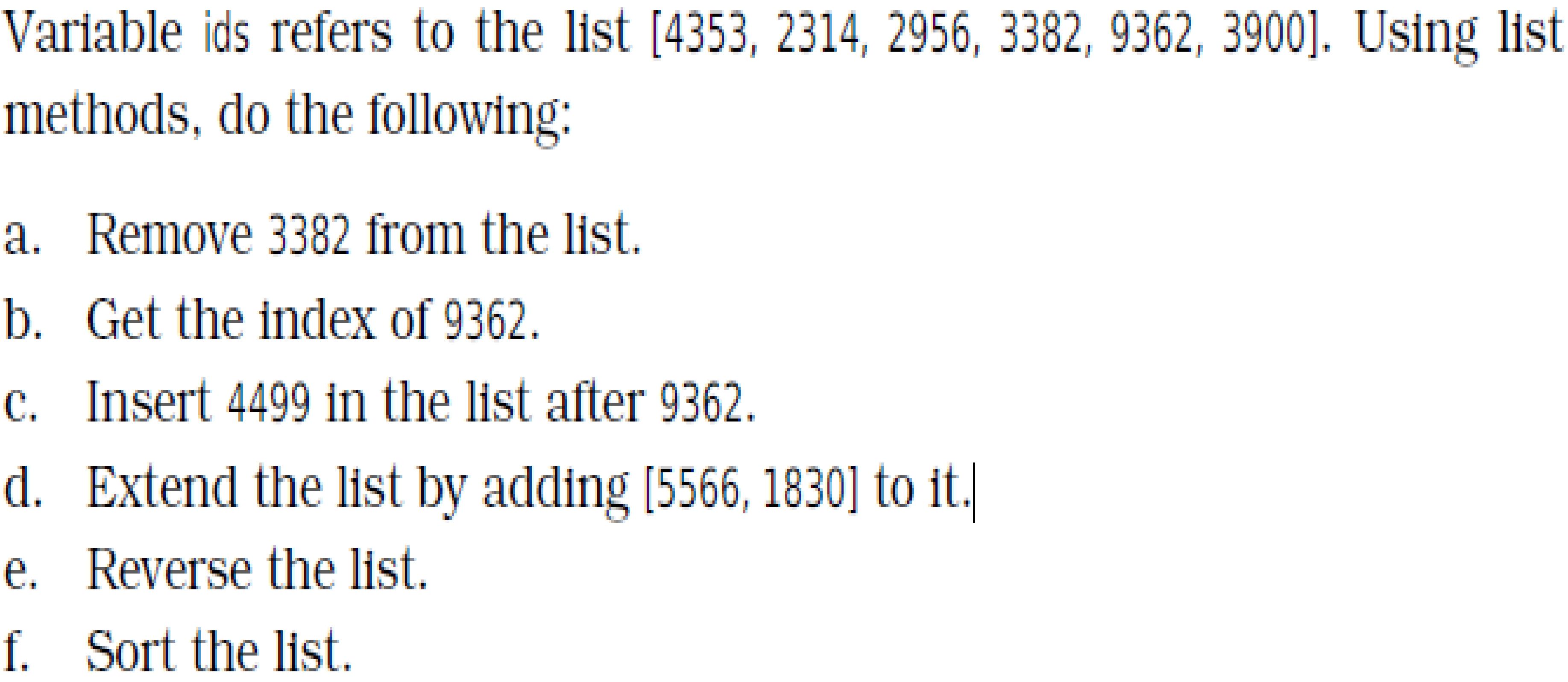
c. The list `['Bacteria', 'Protozoa', 'Chromista']`

d. The list `['Chromista', 'Plantae', 'Fungi']`

e. The list `['Fungi', 'Animalia']`

f. The empty list

In Class Act-2



Variable ids refers to the list [4353, 2314, 2956, 3382, 9362, 3900]. Using list methods, do the following:

a. Remove 3382 from the list.

b. Get the index of 9362.

c. Insert 4499 in the list after 9362.

d. Extend the list by adding [5566, 1830] to it.

e. Reverse the list.

f. Sort the list.