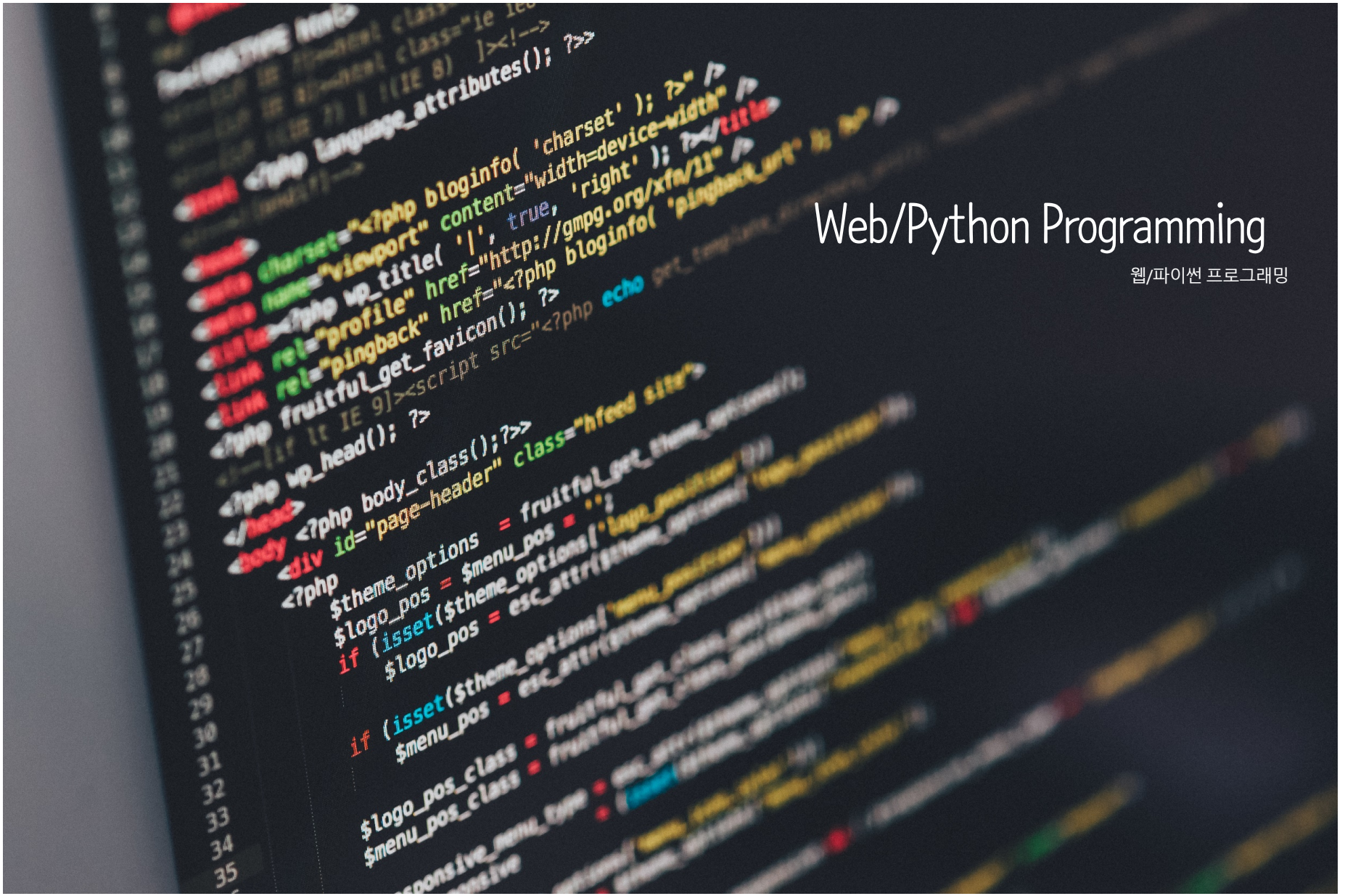


Web/Python Programming

웹/파이썬 프로그래밍

Today

- Lists

List is a type

- Integer
- Float
- Boolean
- String

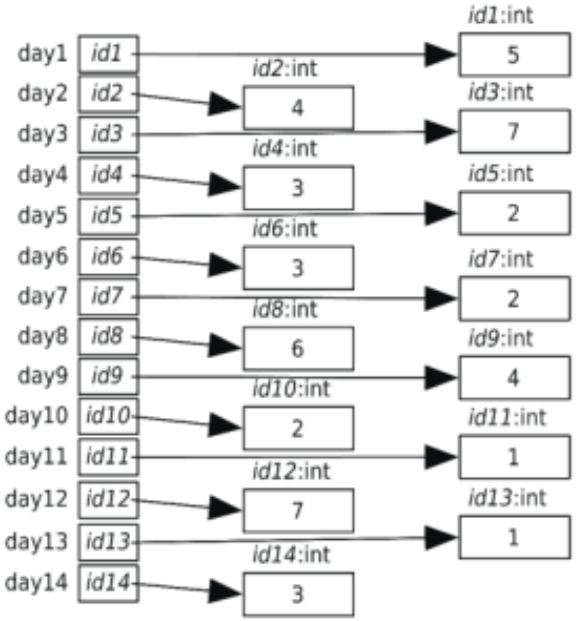
- Lists contain 0 or more objects
- Collection of data!

Collection of data

- Number of gray whales counted near the Coal Oil Point Natural Reserve

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

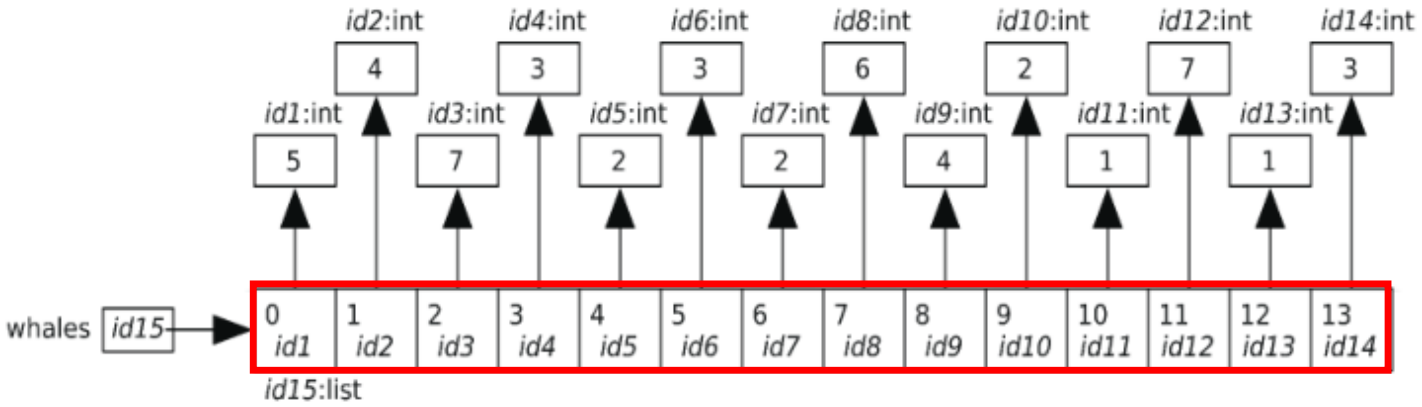
Table 8—Gray Whale Census




```
>>> 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]
```

List

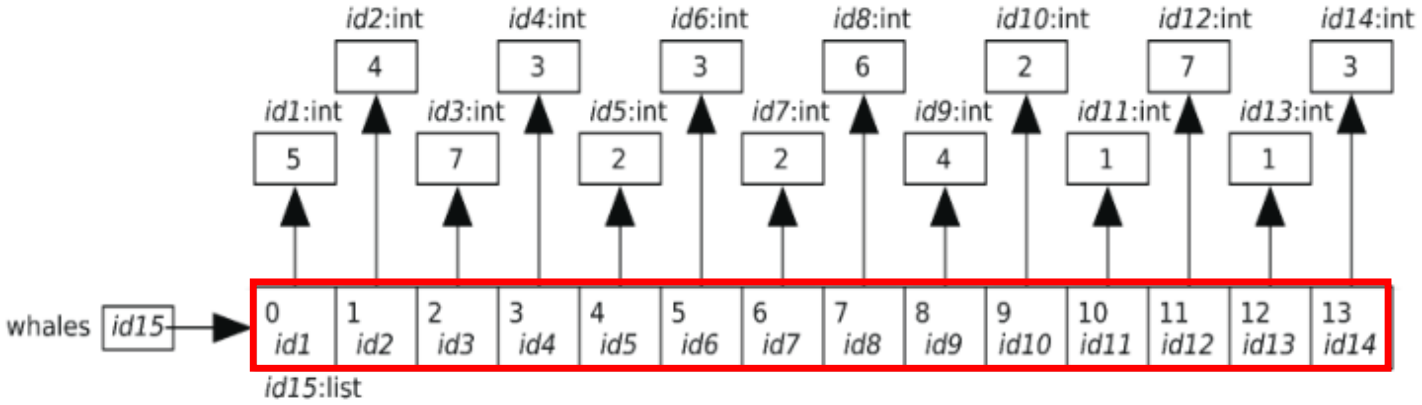
```
>>> whales = [5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
>>> whales
[5, 4, 7, 3, 2, 3, 2, 6, 4, 2, 1, 7, 1, 3]
```



- List is a type
- A list is an object
- An object can be assigned to a variable

Elements

```
>>> 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]
```



- `whales` refers to a list with 14 elements
- [] is an empty list: a list with no elements
- A list is an object, but it also contains the memory addresses of other objects

Index

```
>>> 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]
>>> whales[0]
5
>>> whales[1]
4
>>> whales[13]
3
>>> whales[14]
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    whales[14]
IndexError: list index out of range
```

```
>>> whales[-1]
3
>>> whales[-14]
5
>>> whales[-15]
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    whales[-15]
IndexError: list index out of range
```

- Each item has an index indicating its position in the list
- The first item in a list is at index 0
- The length of whales is 14. Its indices are in the range from 0 to 13.

Element

```
>>> 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]
>>> third = whales[2]
>>> third
7

>>> whales = []
>>> whales[0]
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    whales[0]
IndexError: list index out of range
>>> whales[-1]
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    whales[-1]
IndexError: list index out of range
```

Lists can contain any type of data

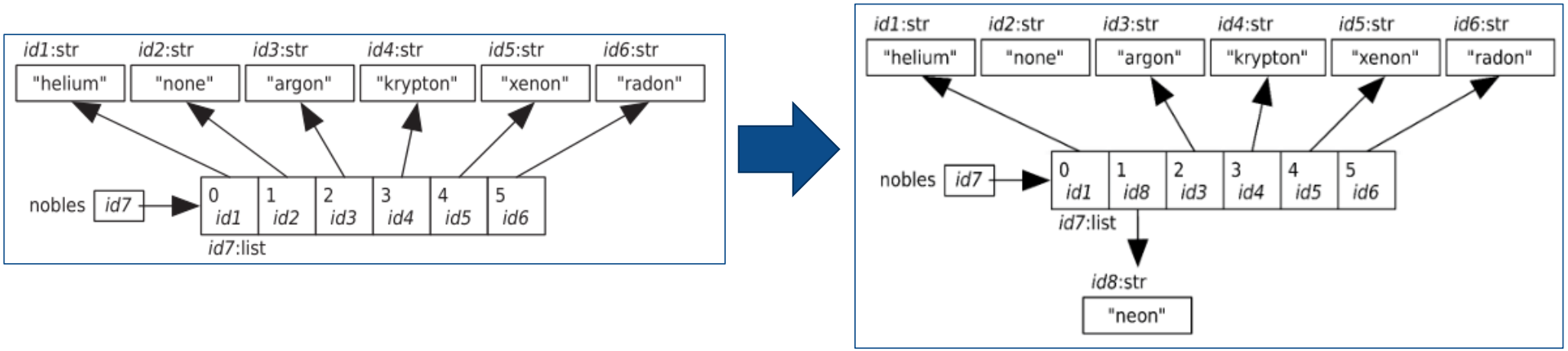
- Lists can contain integers, strings, and even other lists.

```
>>> krypton = ['Krypton','Kr',-157.2,-153.4]
>>> krypton[1]
'Kr'
>>> krypton[2]
-157.2
>>> type(krypton[1])
<class 'str'>
>>> type(krypton[2])
<class 'float'>
```

- But usually, a list is used to contain items of the same kind.

Modifying lists

```
>>> nobles = ['helium', 'none', 'argon', 'krypton', 'xenon', 'radon']
>>> nobles
['helium', 'none', 'argon', 'krypton', 'xenon', 'radon']
>>> nobles[1] = 'neon'
>>> nobles
['helium', 'neon', 'argon', 'krypton', 'xenon', 'radon']
```



- `nobles[1]` behaves just like a simple variable

Lists are mutable

```
>>> x = L[i]
```

- `L[i]`: Get the value referred to by the memory address at index `i` of list `L`

```
>>> L[i] = x
```

- `L[i]`: Look up the memory address at index `i` of list `L` so it can be overwritten

- Lists are mutable (can be mutated)
- Strings are immutable

```
>>> number_list = [2,4,5,8,10]
>>> x = number_list[2]
>>> x
5
>>> number_list[2] = 6
>>> number_list
[2, 4, 6, 8, 10]
>>> a = 'Pithon'
>>> a
'Pithon'
>>> a[1]
'i'
>>> a[1] = 'y'
Traceback (most recent call last):
  File "<pyshell#59>", line 1, in <module>
    a[1] = 'y'
TypeError: 'str' object does not support i
tem assignment
>>> name = 'Darwin'
>>> name.upper()
'DARWIN'
>>> name
'Darwin'
>>> capitalized = name.upper()
>>> capitalized
'DARWIN'
>>> name
'Darwin'
```

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.)

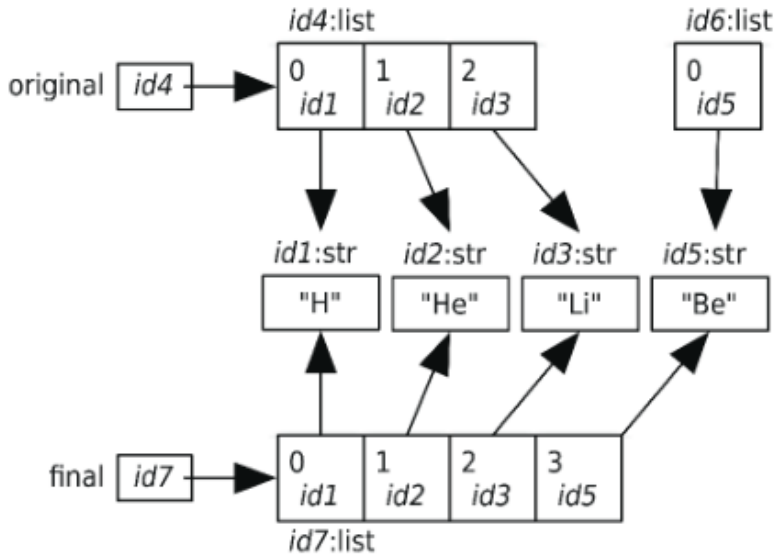
Table 9—List Functions

```
>>> a = [2,4,1,7,3]
>>> len(a)
5
>>> max(a)
7
>>> min(a)
1
>>> sum(a)
17
>>> sorted(a)
[1, 2, 3, 4, 7]
>>> sorted(a,reverse=True)
[7, 4, 3, 2, 1]
>>> a
[2, 4, 1, 7, 3]
```

+, *, del operators on lists

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

>>> a = [2,4,1,7,3]
>>> b = 3
>>> a+b
Traceback (most recent call last):
  File "<pyshell#95>", line 1, in <module>
    a+b
TypeError: can only concatenate list (not
"int") to list
>>> c = [3]
>>> a+c
[2, 4, 1, 7, 3, 3]
>>> metals = ['Fe','Ni']
>>> metals *3
['Fe', 'Ni', 'Fe', 'Ni', 'Fe', 'Ni']
>>> del metals[0]
>>> metals
['Ni']
```



In operator on lists

```
nobles = ['helium', 'neon', 'argon', 'krypton', 'xenon', 'radon']

gas = input('Enter a gas: ')
if gas in nobles:
    print('{} is nobel.'.format(gas))
else:
    print('{} is not nobel gas'.format(gas))
```

```
Enter a gas: neon
neon is nobel.
>>>
```

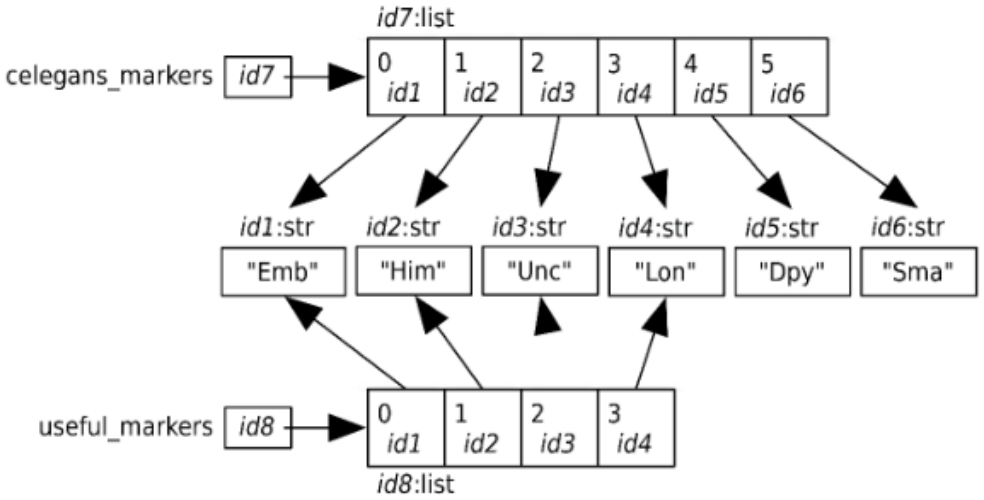
```
Enter a gas: hydrogen
hydrogen is not nobel gas
>>>
```

```
Enter a gas: Radon
Radon is not nobel gas
```

```
>>> 1 in [0,1,2,3]
True
>>> [1] in [0,1,2,3]
False
>>> [1,2] in [0,1,2,3]
False
>>> 'ar' in 'Bargain'
True
>>> 'ar' in ['Bar','Bargain']
False
>>> 'ar' in ['B','a','r','g','a','i','n']
False
>>> 'ar' in ['ar','br','cr']
True
```

Slicing lists

- Geneticists describe C. elegans phenotypes (nematodes, a type of microscopic worms) using three-letter short-form markers.
  - Emb: embryonic lethality
  - Him: high incidence of males
  - Unc: uncoordinated
  - Dpy: dumpy – short and fat
  - Sma: small
  - Lon: long

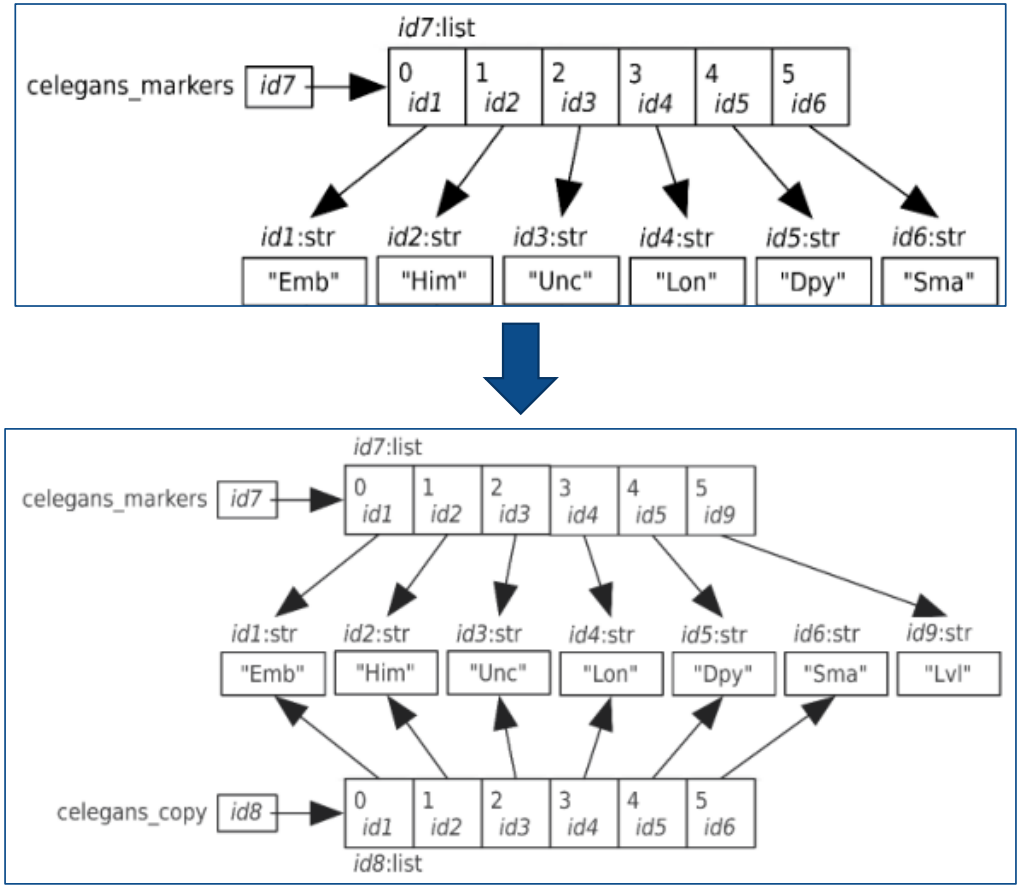



```
>>> celegans_phenotypes = ['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
>>> celegans_phenotypes
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
>>> useful_markers = celegans_phenotypes[0:4]
>>> useful_markers
['Emb', 'Him', 'Unc', 'Lon']
```

Slicing/Copying lists

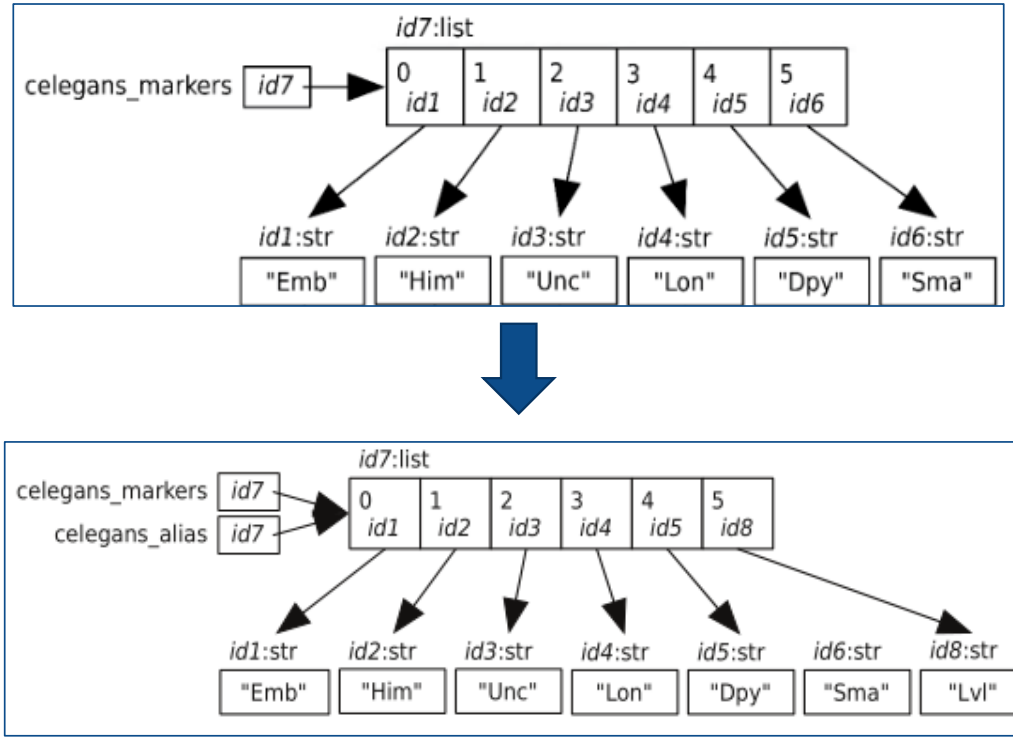
```
>>> celegans_phenotypes = ['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
>>> celegans_phenotypes[:4]
['Emb', 'Him', 'Unc', 'Lon']
>>> celegans_phenotypes[4:]
['Dpy', 'Sma']
>>> celegans_phenotypes[::2]
['Emb', 'Unc', 'Dpy']
>>> celegans_phenotypes[::-1]
['Sma', 'Dpy', 'Lon', 'Unc', 'Him', 'Emb']
```

```
>>> celegans_copy = celegans_phenotypes[:]
>>> celegans_phenotypes[5] = 'Lvl'
>>> celegans_phenotypes
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Lvl']
>>> celegans_copy
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
```



Aliasing

```
>>> celegans_phenotypes = ['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
>>> celegans_phenotypes
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Sma']
>>> celegans_alias = celegans_phenotypes
>>> celegans_phenotypes[5] = 'Lvl'
>>> celegans_phenotypes
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Lvl']
>>> celegans_alias
['Emb', 'Him', 'Unc', 'Lon', 'Dpy', 'Lvl']
```



Aliasing vs. Copying

```
>>> x = [1,2,3,4,5]
>>> y = x
>>> z = x[:]
>>> x
[1, 2, 3, 4, 5]
>>> y
[1, 2, 3, 4, 5]
>>> z
[1, 2, 3, 4, 5]
>>> x[5] = 7
>>> x
[1, 2, 3, 4, 7]
>>> y
[1, 2, 3, 4, 7]
>>> z
[1, 2, 3, 4, 5]
```

Mutable parameters

```
def remove_last_item(L):
    """(list) -> list

    Return list L with the last item removed.

    Precondition: len(L) >=0

    >>> remove_last_item([1,3,2,4])
    [1,3,2]
    """
    del L[-1]
    return L
```

```
def remove_last_item(L):
    """(list) -> list

    Return list L with the last item removed.

    Precondition: len(L) >=0

    >>> remove_last_item([1,3,2,4])
    [1,3,2]
    """
    del L[-1]
```

```
>>> celegans_markers = ['Emb','Him','Unc','Lon','Dpy','Lvl']
>>> result_list = remove_last_item(celegans_markers)
>>> result_list
['Emb', 'Him', 'Unc', 'Lon', 'Dpy']
>>> celegans_markers
['Emb', 'Him', 'Unc', 'Lon', 'Dpy']
```

```
>>> celegans_markers = ['Emb','Him','Unc','Lon','Dpy','Lvl']
>>> remove_last_item(celegans_markers)
>>> celegans_markers
['Emb', 'Him', 'Unc', 'Lon', 'Dpy']
```

No need to return the modified list because the input list (parameter) is modified!!

List methods

- List is a type
- List is a class
- List has methods.

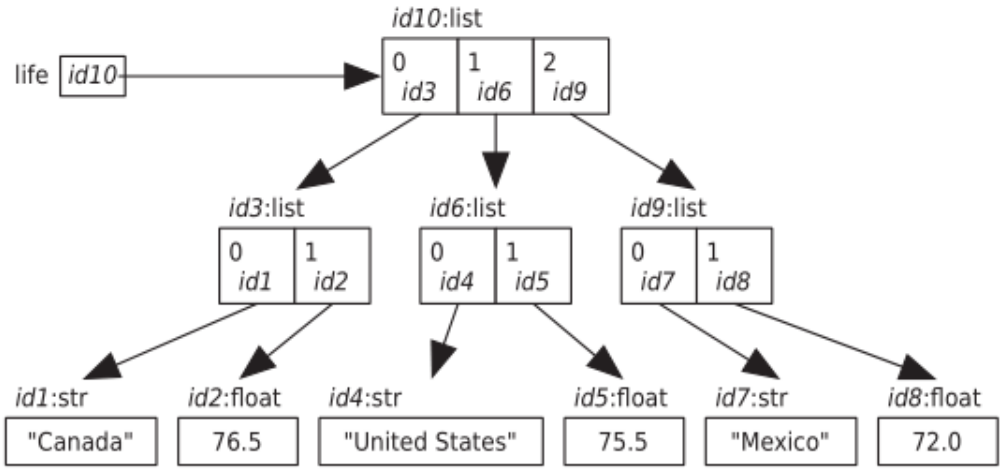
Method	Description
L.append(v)	Appends value v to list L
L.clear()	Removes all items from list L
L.count(v)	Returns the number of occurrences of v in list L
L.extend(v)	Appends the items in v to L
L.index(v)	Returns the index of the first occurrence of v in L—an error is raised if v doesn't occur in 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 with the same letter case, it sorts in alphabetical order)
L.sort(reverse=True)	Sorts the values in list L in descending order (for strings with the same letter case, it sorts in reverse alphabetical order)

Table 10—List Methods

A list of lists

- A nested list
- The following nested list describe life expectancies in different countries

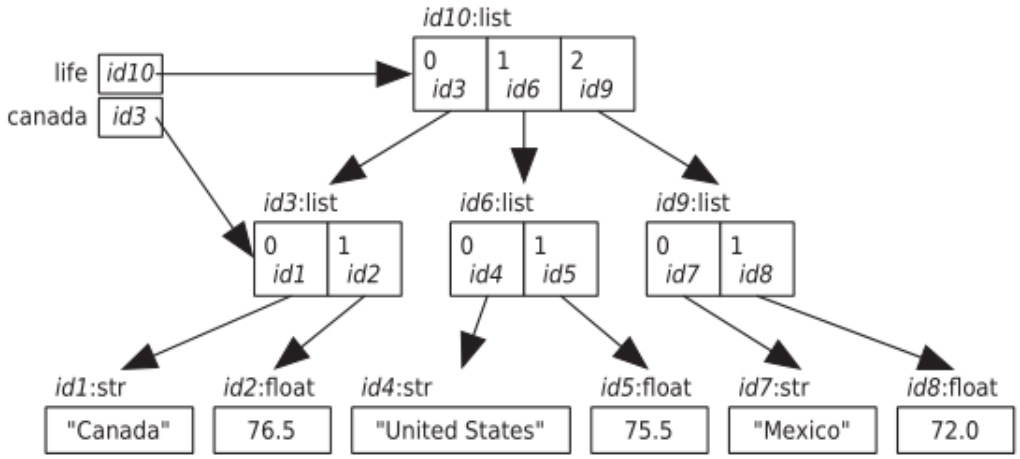
```
>>> life = [['Canada',76.5], ['United States',75.5], ['Mexico', 72.0]]
>>> life[0]
['Canada', 76.5]
>>> life[1]
['United States', 75.5]
>>> life[2]
['Mexico', 72.0]
>>> life[0][0]
'Canada'
>>> life[0][1]
76.5
>>> canada = life[0]
>>> canada[0]
'Canada'
>>> canada[1]
76.5
```



A list of lists

- Aliasing

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



Summary

- 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.

Ref. List methods

```
>>> colors = ['red','orange','green']
>>> colors.extend(['black','blue'])
>>> colors
['red', 'orange', 'green', 'black', 'blue']
>>> colors.insert(2,'yellow')
>>> colors
['red', 'orange', 'yellow', 'green', 'black', 'blue']
>>> colors.pop()
'blue'
>>> colors
['red', 'orange', 'yellow', 'green', 'black']
>>> colors.remove('green')
>>> colors
['red', 'orange', 'yellow', 'black']
>>> colors.reverse()
>>> colors
['black', 'yellow', 'orange', 'red']
>>> colors.count('yellow')
1
>>> colors.count('e')
0
```

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 with the same letter case, it sorts in alphabetical order)
L.sort(reverse=True)	Sorts the values in list L in descending order (for strings with the same letter case, it sorts in reverse alphabetical order)

Table 10—List Methods

Ref. List methods

```
>>> colors.append('blue')
>>> colors
['black', 'yellow', 'orange', 'red', 'blue']
>>> colors.append(['green','white'])
>>> colors
['black', 'yellow', 'orange', 'red', 'blue', ['green', 'wh
ite']]
>>> colors.pop()
['green', 'white']
>>> colors
['black', 'yellow', 'orange', 'red', 'blue']
>>> colors.extend(['green','white'])
>>> colors
['black', 'yellow', 'orange', 'red', 'blue', 'green', 'whi
te']
>>> colors.pop()
'white'
>>> colors.sort()
>>> colors
['black', 'blue', 'green', 'orange', 'red', 'yellow']
>>> colors.sort(reverse=True)
>>> colors
['yellow', 'red', 'orange', 'green', 'blue', 'black']
>>> colors.index(1)
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    colors.index(1)
ValueError: 1 is not in list
>>> colors.index('red')
1
```

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 with the same letter case, it sorts in alphabetical order)
L.sort(reverse=True)	Sorts the values in list L in descending order (for strings with the same letter case, it sorts in reverse alphabetical order)

Table 10—List Methods