Data Structures
-- List and Tuple

江琳

Data Structures

- **Data structures are used to store a collection of related data.**
- **There are three built-in data structures in Python - list, tuple and dictionary.**

Sequence

- **The most basic data structure in Python is the sequence. Each element of a sequence is assigned a number - its position or index. The first index is zero, the second index is one, and so forth.**
- **the most common sequence are lists and tuples**

Python List

- A list is a data structure that holds an ordered collection of items.
- The list is enclosed in square brackets.

```
list1 = [1, 2, 3, 4, 5]
list2 = ['physics', 'chemistry', 'geography', 'history']
```

Python List

- **Items in a list need not be the same type.**

```
list3 = ['physics', 'chemistry', 1997, 2000]

list4 = ['physics', 'chemistry', 1997, True]

list5 = ['physics', ['chemistry', 2000], 1997]
```

Accessing Values in Lists

- To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index.

```
list1 = ['physics', 'chemistry', 1997, 2000]

print ('list1[0]: ', list1[0])

# 输出结果为:
list1[0]: physics
```

Accessing Values in Lists

- **Each element of a list is assigned a number - its index. The first index is zero, the second index is one, and so forth.**

List1:	***'physics'***	***'chemistry'***	***1997***	***2000***
Index:	***0***	***1***	***2***	***3***

Accessing Values in Lists

List1:	*'physics'*	*'chemistry'*	*1997*	*2000*
Index:	*0*	*1*	*2*	*3*

- **list[0] = 'physics'**
- **list[1] = 'chemistry'**
- **list[2] = 1997**
- **list[3] = 2000**

Accessing Values in Lists

```
list5 = ['physics', ['chemistry', 2000], 1997]
```

- How to get ‘chemistry’ in list5?

Accessing Values in Lists

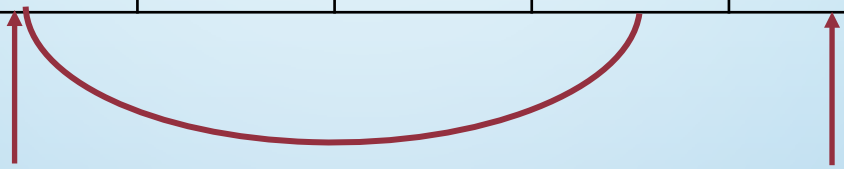
```
list2 = [1, 2, 3, 4, 5, 6, 7]

print ('list2[1:5]: ', list2[1:5])
```

- What's we get?

Accessing Values in Lists

List2:	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Index:	*0*	*1*	*2*	*3*	*4*	*5*	*6*



- **list2[1:5] = [2, 3, 4, 5]**

Updating List

- **You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator**

Updating List

```
list1 = ['physics', 'chemistry', 1997, 2000]

print ('Value available at index 2: ')

print (list1[2])

# update value in index 2
list1[2] = 2001;

print ('New value available at index 2: ')

print (list1[2])
```

Delete List Elements

```
list1 = ['physics', 'chemistry', 1997, 2000]

print (list1)

del list1[2]

print ('After deleting value at index 2 : ')

print (list1)
```

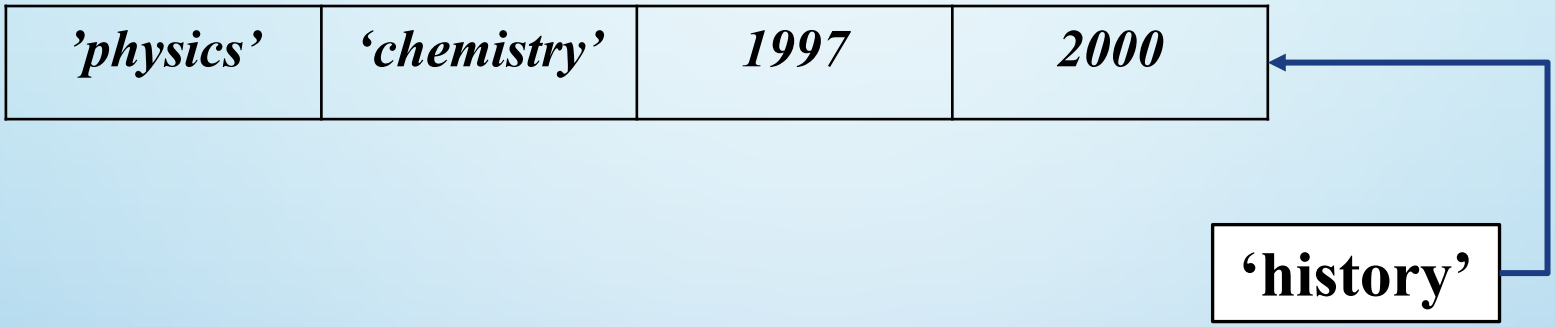
Basic List Operations

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition

Built-in List Functions & Methods

- list.append(obj)
  - Appends objects obj to list
- list.insert(index, obj)
  - Inserts object obj into list at offset index
- list.extend(seq)
  - Appends the contents of seq to list

list.append(obj)

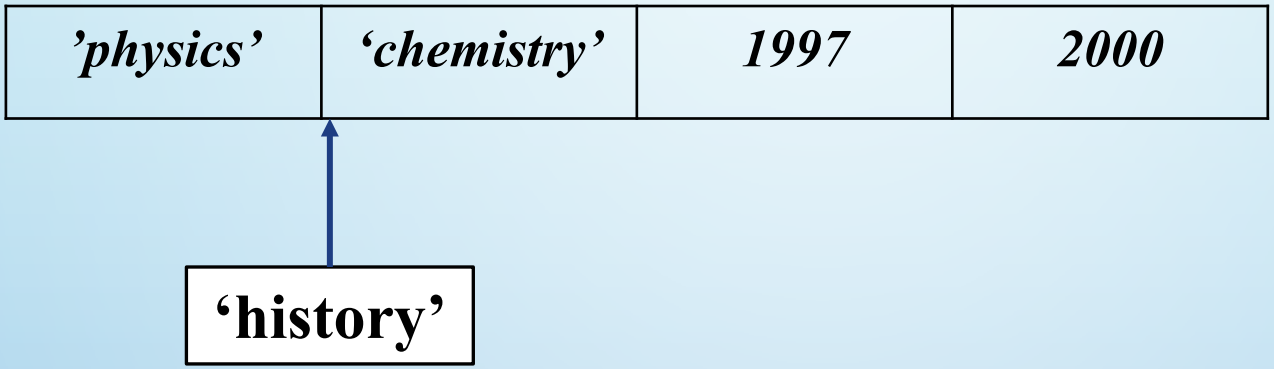



```
list1 = ['physics', 'chemistry', 1997, 2000]

list1.append('history')

print ('list1: ', list1)
```

list.insert(index, obj)

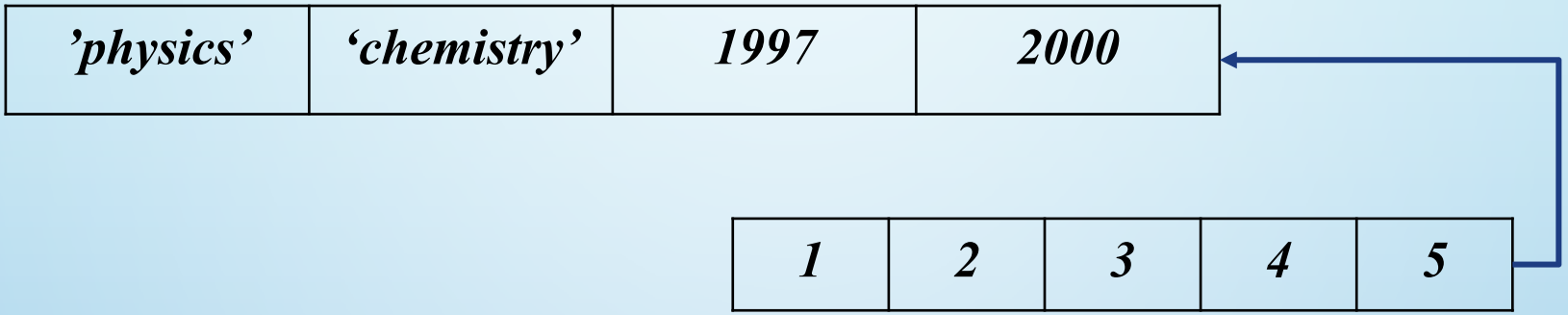



```
list1 = ['physics', 'chemistry', 1997, 2000]

list1.insert(1, 'history')

print ('list1: ', list1)
```

list.extend(seq)




```
list1 = ['physics', 'chemistry', 1997, 2000]

list2 = [1, 2, 3, 4, 5]

list1.extend(list2)

print ('list1: ', list1)
```

Built-in List Functions & Methods

- list.remove(obj)
  - Removes object obj from list
- list.pop()
  - Removes and returns last object

Built-in List Functions & Methods

```
list1 = ['physics', 'chemistry', 1997, 2000]

list1.remove(1997)

print ('list1: ', list1)

list1.pop()

print ('list1: ', list1)
```

Data Structures
-- List and Tuple

江琳

Tuple

- A tuple is a sequence of immutable Python objects.
- Tuples are sequences, just like lists.
- The differences between tuples and lists are:
  - the tuples cannot be changed unlike lists
  - tuples use parentheses, whereas lists use square brackets.

Creating Tuple

```
tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );


# without parentheses is also work
tup3 = 'a', 'b', 'c', 'd'


# To write a tuple containing a single value you
have to include a comma, even though there is
only one value
tup4 = (50, )
```

Accessing Values in Tuples

```
tup1 = ('physics', 'chemistry', 1997, 2000)

tup2 = (1, 2, 3, 4, 5, 6, 7 )

print ('tup1[0]: ', tup1[0])
print ('tup2[1:5]: ', tup2[1:5])



# results
# tup1[0]: physics
# tup2[1:5]: [2, 3, 4, 5]
```

Accessing Values in Tuples

```
tup1 = ('physics', 'chemistry', 1997, 2000)
a, b, c, d = tup1


# results
# a: 'physics'
# b: 'chemistry'
# c: 1997
# d: 2000
```

Update Tuple

- **Can not be updated**

Basic List Operations

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition

“mutable” Tuple

- Try the code below:

```
tup3 = ('physics', 'chemistry', [1997, 2000])

tup3[2][0] = 1998
print ('tup3: ', tup3)
# results
# ('physics', 'chemistry', [1998, 2000])

tup3[2].append(1991)
print ('tup3: ', tup3)
# results
# ('physics', 'chemistry', [1998, 2000, 1991])
```

“mutable” Tuple

- **Python tuples have a surprising trait: they are immutable, but their values may change. This may happen when a tuple holds a reference to any mutable object, such as a list. In fact, the tuple did not change, it still has the same references to other objects that it did before.**

Questions?