PYTHON

Chapter: Lists

Lists

- Data structure to hold a list of items
- These items can be of different data types too
- Can access items of the list using index
- The same slicing operations work well on lists too but at a list item level
  - [start:end:step]
  - [:end:step]
  - [::]
  - [::step]
  - [:end]
  - [:]
- List can contain lists too
- Can be as deeply nested as you want

Lists - creation

```
>>> a = []
>>> a = list()
>>> a
[]
>>> a = []
>>> a
[]
>>> a = list('cat')
>>> a
['c', 'a', 't']
>>> a= 'India,Japan,China,UK,USA'.split(',')
>>> a
['India', 'Japan', 'China', 'UK', 'USA']
```

Lists – accessing list element using index

```
>>> a = ['India', 'Japan', 'China', 'UK', 'USA']
>>> a[1]
'Japan'
>>> a[-1]
'USA'
>>> a[100]

Traceback (most recent call last):
  File "<pyshell#317>", line 1, in <module>
    a[100]
IndexError: list index out of range
```

Lists – slice

```
>>> a = ['India', 'Japan', 'China', 'UK', 'USA']
>>> a[1:3]
['Japan', 'China']
>>> a[-1:]
['USA']
>>> a[-1::-1]
['USA', 'UK', 'China', 'Japan', 'India']
```

Lists – Modifying Lists

```
>>> a
['India', 'Japan', 'China', 'UK',
'USA']
>>> a[3]='Burma'
>>> a
['India', 'Japan', 'China',
'Burma', 'USA']
>>> id(a)
59350280L
>>> a[3]='Russia'
>>> id(a)
59350280L
```

```
>>> b='hello'
>>> b.replace('h','d')
'dello'
>>> b
'hello'
>>> id(b)
59545200L
>>> b = b.replace('h','d')
>>> b
'dello'
>>> id(b)
56578616L
```

PYTHON

Lists – Modifying Lists

```
>>> a.append('UK')
>>> a
['India', 'Japan', 'China',
'Russia', 'USA', 'UK'] >>> b =
['Sri Lanka','Thailand','Nigeria']
>>> a + b
['India', 'Japan', 'China',
'Russia', 'USA', 'UK', 'Sri
Lanka', 'Thailand', 'Nigeria']
>>> a+=b
>>> a
['India', 'Japan', 'China',
'Russia', 'USA', 'UK', 'Sri
Lanka', 'Thailand', 'Nigeria']
```

```
>>> a = ['India', 'Japan',
'China', 'Russia', 'USA']
>>> a.extend(b)
>>> a
['India', 'Japan', 'China',
'Russia', 'USA', 'Sri Lanka',
'Thailand', 'Nigeria']
>>> a = ['India', 'Japan',
'China', 'Russia', 'USA']
>>> a.append(b)
>>> a
['India', 'Japan', 'China',
'Russia', 'USA', ['Sri Lanka',
'Thailand', 'Nigeria']]
```

Lists – Modifying Lists

```
>>> a = ['India', 'Japan',
'China', 'Russia', 'USA']
>>> a.insert(3,'Ukraine')
>>> a
['India', 'Japan', 'China',
'Ukraine', 'Russia', 'USA']

>>> a.insert(200, 'Bangkok')
>>> a
['India', 'Japan', 'China',
'Ukraine', 'Russia', 'USA',
'Bangkok']
```

```
>>> a.insert(-1, 'Indonesia')
>>> a
['India', 'Japan', 'China',
'Ukraine', 'Russia', 'USA',
'Indonesia', 'Bangkok']

>>> a.insert(-200, 'New York')
>>> a
['New York', 'India', 'Japan',
'China', 'Ukraine', 'Russia',
'USA', 'Indonesia', 'Bangkok']
```

Lists – Deleting

```
>>> a.remove('Bangkok')
>>> a
['New York', 'India', 'Japan', 'China', 'Ukraine', 'Russia', 'USA',
'Indonesia']

>>> del a[0]
>>> a
['India', 'Japan', 'China', 'Ukraine', 'Russia', 'USA', 'Indonesia']

>>> a.pop()
'Indonesia'
>>> a
['India', 'Japan', 'China', 'Ukraine', 'Russia', 'USA']
```

Lists – Test for a value

Pythonic way of coding

```
>>> a = ['India', 'Japan', 'China', 'UK', 'USA']
>>> 'India' in a
True
>>> country = 'India'
>>> country in a
True
```

A little digression

Pythonic way of coding

When a veteran Python developer (a “Pythonista”) calls portions of code not “Pythonic”, they usually mean that these lines of code do not follow the common guidelines and fail to express its intent in what is considered the best (hear: most readable) way.

```
#Pythonic way
>>> a = ['India', 'Japan',
'China', 'UK', 'USA']
>>> 'Japan' in a
True
```

```
# non Pythonic way
>>> a.index('Japan')
1
# then compare the index if its
positive or if it gave an error
and then confirm whether 'Japan'
exists or not
```

Lists – Copying

```
>>> a = ['India', 'Japan',
'China', 'UK', 'USA']
>>> b = a
>>> b
['India', 'Japan', 'China', 'UK',
'USA']
>>> a[1] = 'Nigeria'
>>> a
['India', 'Nigeria', 'China',
'UK', 'USA']
>>> b
['India', 'Nigeria', 'China',
'UK', 'USA']
```

```
>>> b = list(a)
>>> c = a[:]
```

Tuples

- Similar to lists
- Uses “(“ and “)” for being and end
- Tuples are Immutable. So they lack
  - append(), insert(), pop() etc

Tuples

```
>>> a_tuple= ()
>>> a_tuple = ('batman','spiderman','superman','ironman')
>>> a_tuple = 'batman','spiderman','superman','ironman'
>>> a_tuple
('batman', 'spiderman', 'superman', 'ironman')
>>> type(a_tuple)
<type 'tuple'>
```

Tuples - unpacking

```
>>> sh1, sh2, sh3, sh4 = a_tuple
>>> print sh1, sh2, sh3, sh4
batman spiderman superman ironman

>>> sh1, sh2 = a_tuple

Traceback (most recent call last):
  File "<pyshell#497>", line 1, in <module>
    sh1, sh2 = a_tuple
ValueError: too many values to unpack
```

Tuples – Slicing is same as in lists

```
>>> a_tuple[1:2]
('spiderman',)
>>> a_tuple[1:]
('spiderman', 'superman', 'ironman')
>>> a_tuple[1:100]
('spiderman', 'superman', 'ironman')
>>> a_tuple[1::2]
('spiderman', 'ironman')
```

Tuple Vs Lists

- Uses lesser space
- Immutable and hence cannot change by mistake
- Function arguments are passed as tuples