บทที่ 6 ลิสต์ (Lists)

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

What is Not a “Collection”?

Most of our variables have one value in them - when we put a new value in the variable, the old value is overwritten

```
>>> x=2
>>> x
2
>>> x=4
>>> x
4
```

A List is a Kind of Collection

- A collection allows us to put many values in a single "variable"
- A collection is nice because we can carry all many values around in one convenient package.

```
friends = [ 'Joseph', 'Glenn', 'Sally' ]

carryon = [ 'socks', 'shirt', 'perfume' ]
```

List Constants

- List constants are surrounded by square brackets and the elements in the list are separated by commas
- A list element can be any Python object - even another list
- A list can be empty

```
>>> print([1, 24, 76])
[1, 24, 76]
>>> print(['red', 'yellow', 'blue'])
['red', 'yellow', 'blue']
>>> print(['red', 24, 98.6])
['red', 24, 98.6]
>>> print([ 1, [5, 6], 7])
[1, [5, 6], 7]
>>> print([])
[]
```

We Already Use Lists!


```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Blastoff!')
```

```
5
4
3
2
1
Blastoff!
```

Lists and Definite Loops - Best Pals


```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
    print('Happy New Year:',  friend)
print('Done!')
```

```
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
Done!
```

```
z = ['Joseph', 'Glenn', 'Sally']
for x in z:
    print('Happy New Year:',  x)
print('Done!')
```

Looking Inside Lists

Just like strings, we can get at any single element in a list using an index specified in square brackets

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> print(friends[1])
Glenn
>>>
```

Lists are Mutable

- Strings are "immutable" - we cannot change the contents of a string - we must make a new string to make any change
- Lists are "mutable" - we can change an element of a list using the index operator

```
>>> fruit = 'Banana'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not
support item assignment
>>> lotto = [2, 14, 26, 41, 63]
>>> print(lotto)
[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print(lotto)
[2, 14, 28, 41, 63]
```

How Long is a List?

- The len() function takes a list as a parameter and returns the number of elements in the list
- Actually len() tells us the number of elements of any set or sequence (such as a string...)

```
>>> greet = 'Hello Bob'
>>> print(len(greet))
9
>>> x = [ 1, 2, 'joe', 99]
>>> print(len(x))
4
>>>
```

A Tale of Two Loops...


```
friends = ['Joseph', 'Glenn', 'Sally']

for friend in friends :
    print('Happy New Year:',  friend)

for i in range(len(friends)) :
    friend = friends[i]
    print('Happy New Year:',  friend)
```

```
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print(len(friends))
3
>>> print(range(len(friends)))
[0, 1, 2]
```

```
Happy New Year:  Joseph
Happy New Year: Glenn
Happy New Year: Sally
```

Concatenating Lists Using +

We can create a new **list** by adding two existing lists together

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

Lists Can Be Sliced Using :


```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41,12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
>>> t[-1]
15
>>> t[-2]
74
>>> t[1:-1]
[41, 12, 3, 74]
>>> t[::-1]
[15, 74, 3, 12, 41, 9]
```

Remember: the second number is “up to but not including”

List Methods


```
>>> x = list()
>>> type(x)
<class 'list'>
>>> dir(x)
['__add__', '__class__', '__class_getitem__', '__contains__',
'__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__getitem__',
'__gt__', '__hash__', '__iadd__', '__imul__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__reversed__', '__rmul__', '__setattr__',
'__setitem__', '__sizeof__', '__str__', '__subclasshook__',
'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']
```

การสร้างตัวแปรลิสต์ (list)


```
>>> stuff = []
>>> stuff
[]
>>> things = list()
>>> things
[]
>>> stuff = "pot fork spoon mango"
>>> stuff
'pot fork spoon mango'
>>> fruits = "apple mango papaya".split()
>>> fruits
['apple', 'mango', 'papaya']
```

Is Something in a List?


```
>>> some = [1, 9, 21, 10, 16]
>>> 9 in some
True
>>> 15 in some
False
>>> 20 not in some
True
```

Lists are in Order


```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> friends.sort()
>>> print(friends)
['Glenn', 'Joseph', 'Sally']
>>> print(friends[1])
Joseph

>>> x = [5,9,4,1,3]
>>> x.sort()
>>> x
[1, 3, 4, 5, 9]

>>> x = [3,7,"Hi"]
>>> x.sort()
Traceback (most recent call last):
  File "<pyshell#33>", line 1, in <module>
    x.sort()
TypeError: '<' not supported between instances of 'str' and 'int'
```

Built-in Functions and Lists

- There are a number of functions built into Python that take lists as parameters
- Remember the loops we built? These are much simpler.

```
>>> nums = [3, 41, 12, 9, 74, 15]
>>> print(len(nums))
6
>>> print(max(nums))
74
>>> print(min(nums))
3
>>> print(sum(nums))
154
>>> print(sum(nums)/len(nums))
25.6
```

Split delimiter


```
>>> line = 'A lot                of spaces'
>>> etc = line.split()
>>> print(etc)
['A', 'lot', 'of', 'spaces']
>>>
>>> line = 'first;second;third'
>>> thing = line.split()
>>> print(thing)
['first;second;third']
>>> print(len(thing))
1
>>> thing = line.split(';')
>>> print(thing)
['first', 'second', 'third']
>>> print(len(thing))
3
```

When you do not specify a delimiter, multiple spaces are treated like one delimiter

You can specify what delimiter character to use in the splitting

List copying


```
>>> x = [1,2,3,4,5]
>>> a = x
>>> x
[1, 2, 3, 4, 5]
>>> a
[1, 2, 3, 4, 5]
>>> a[2] = 9
>>> a
[1, 2, 9, 4, 5]
>>> x
[1, 2, 9, 4, 5]
```

```
x = [1,2,3,4,5]
a = list(x)
x
[1, 2, 3, 4, 5]
a
[1, 2, 3, 4, 5]
a[2] = 9
a
[1, 2, 9, 4, 5]
x
[1, 2, 3, 4, 5]
```

```
x = [1,2,3,4,5]
a = x.copy()
x
[1, 2, 3, 4, 5]
a
[1, 2, 3, 4, 5]
a[2] = 9
a
[1, 2, 9, 4, 5]
x
[1, 2, 3, 4, 5]
```

List uppack


```
x = [1,2,3,4,5]
a,b,c,d,e = x
print(a,b,c,d,e)
1 2 3 4 5
a,b = x
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    a,b = x
ValueError: too many values to unpack (expected 2)
```

```
a,*b=x
print(a,b)
1 [2, 3, 4, 5]
*a,b=x
print(a,b)
[1, 2, 3, 4] 5
a,*b,c = x
print(a,b,c)
1 [2, 3, 4] 5
```

List comprehension


```
s = "1 2 3 4 5".split()
print("s=",s)
length = len(s)
x = []
for ele in s:
    num = int(ele)
    x.append(num)
print("x=",x)
a = [int(m) for m in s]
print("a=",a)
```

```
s= ['1', '2', '3', '4', '5']
x= [1, 2, 3, 4, 5]
a= [1, 2, 3, 4, 5]
```

List iteration with index


```
s = [10,20,30,40,50]
for idx in range(len(s)):
    print(f"s[{idx}] = {s[idx]}")
print()
i=0
for ele in s:
    print(f"s[{i}] = {ele}")
    i += 1
```

```
s[0] = 10
s[1] = 20
s[2] = 30
s[3] = 40
s[4] = 50

s[0] = 10
s[1] = 20
s[2] = 30
s[3] = 40
s[4] = 50
```

Enumerate

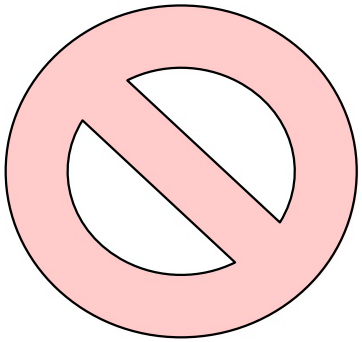

```python
s = [10,20,30,40,50]
for idx, ele in enumerate(s):
    print(f"s[{idx}] = {ele}")
```

```
s[0] = 10
s[1] = 20
s[2] = 30
s[3] = 40
s[4] = 50
```

Program bug?


```
>>> x = "Hello Kmitl"
>>> len = len(x)
>>> print(len)
11
>>> m = "Hi"
>>> print(len(m))
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    print(len(m))
TypeError: 'int' object is not callable
```

DO NOT name variable
the same as function name



List method


```
>>> x = [4,3,7]
>>> x
[4, 3, 7]
>>> x.sort()
>>> x
[3, 4, 7]
>>> x = [4,3,7]
>>> x
[4, 3, 7]
>>> check = x.sort()
>>> check
>>> print(check)
None
>>> x
[3, 4, 7]
>>> print(type(check))
<class 'NoneType'>
```

```
>>> x = [1,2,3]
>>> x
[1, 2, 3]
>>> x.append(4)
>>> x
[1, 2, 3, 4]
>>> x.clear()
>>> x
[]
```

List method


```
>>> x = [1,2,3]
>>> x
[1, 2, 3]
>>> a = x.copy()
>>> a
[1, 2, 3]
>>> x
[1, 2, 3]
>>> a == x
True
>>> a is x
False
>>> a is not x
True
>>> x == a
True
```

```
>>> x
[1, 2, 3]
>>> x.append(1)
>>> x
[1, 2, 3, 1]
>>> x.count()
Traceback (most recent call last):
  File "<pyshell#34>", line 1, in
<module>
    x.count()
TypeError:    list.count()    takes
exactly one argument (0 given)
>>> x.count(2)
1
>>> x.count(1)
2
```

List method


```
>>> x
[1, 2, 3, 1, 1, 2, 3]
>>> x.index(1)
0
>>> x.index(3)
2
>>> x.remove(2)
>>> x
[1, 3, 1, 1, 2, 3]
>>> x.pop()      # last element
3
>>> x
[1, 3, 1, 1, 2]
>>> x.pop()      # last element
2
>>> x
[1, 3, 1, 1]
```

```
>>> x.pop(2)     # index 2
1
>>> x
[1, 3, 1]
>>> x.insert(0,7)  # modify x
>>> x
[7, 1, 3, 1]
>>> x.reverse()
>>> x
[1, 3, 1, 7]
>>> x[::-1]      # x intact
[7, 1, 3, 1]
>>> x[::-1]       # x intact
[7, 1, 3, 1]
>>> x
[1, 3, 1, 7]
```

List method


```
>>> a = [1,2,3]
>>> b = [4,5,6]
>>> c = a + b
>>> a
[1, 2, 3]
>>> b
[4, 5, 6]
>>> c
[1, 2, 3, 4, 5, 6]
```

```
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6]
>>> b
[4, 5, 6]
>>> c
[1, 2, 3, 4, 5, 6]
```

List summary

- List creation (3)
- List concatenation (+)
- List iteration
- List comprehension
- Sum, max, min function
- Len function
- Enumerate
- List assignment
- Immutable
- List slicing
- Dir

Method

- Append
- Sort
- Copy
- Remove
- Pop
- Index
- Extend
- Count
- Insert
- reverse