Python List, Set, Dict

- prepared by Frank Zhu for GTSC-ITClub Python class

Table of Contents

In [1]:
```
%%javascript
$.getScript('https://kmahelona.github.io/ipython_notebook_goodies/ipython_note
book_toc.js')
```

List

Review

In [2]:
```
# construct list
fruits = ['apple', 'pear', 'strawberry']
```

In [3]:
```
# enumeration
for fruit in fruits:
    print(fruit)
```

```
apple
pear
strawberry
```

In [4]:
```
# sometimes it is handy to add an index
for i, fruit in enumerate(fruits):
    print(i, fruit)
```

```
0 apple
1 pear
2 strawberry
```

list comprehension

[transformation for item in item_list]

In [5]:
```
# no transformation
[ i for i in range(10)]
```

Out[5]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [6]:
```
[ i*2 for i in range(10)]
```

Out[6]:
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

In [7]:
```
# append condition
[ i for i in range(10) if i%2 == 0 ]
```

Out[7]:
```
[0, 2, 4, 6, 8]
```

Use list to implement queue and stack

- queue: when you wait in line to checkout in a grocery store, the clerk will process in First-In-First-Out manner, we call this line a queue
- stack: in contrast to queue, the stack is First-In-Last-Out. Think of go into an elevator, the first person will get out last.

In [8]:
```
# customers form a line
customers = [ i for i in range(5)]
print(customers)

# treat them in FIFO
while customers:
    # pop(0) remove the first element
    print(customers.pop(0))

print(customers)
```

```
[0, 1, 2, 3, 4]
0
1
2
3
4
[]
```

In [9]:
```
# simulate 5 persons go into elevator
# this time we use different method: for loop
persons= []
for i in range(5):
    persons.append(i)

print(persons)

while persons:
    # pop() remove the last element
    print(persons.pop())

print(persons)
```

```
[0, 1, 2, 3, 4]
4
3
2
1
0
[]
```

difference between append and extend

In [10]:
```
a = [1, 2, 3]
b = [4, 5]
a.append(b) # this return None, but changes a
print(a)

a = [1, 2, 3]
b = [4, 5]
a.extend(b)
print(a)
```

```
[1, 2, 3, [4, 5]]
[1, 2, 3, 4, 5]
```

matrix and multi-dimensional arrays

- this is rarely used, therefore optional,

In [11]:
```
rows, cols = 3, 2
m = [ [ 0 for j in range(cols) ] for i in range(rows) ]
print(m)
print(m[2][1])
```

```
[[0, 0], [0, 0], [0, 0]]
0
```

Set

set is a special list with no duplicates

Set construction

```
In [12]: # construct a set from list, duplicates removed automatically
         set([1,1,2,2,3])

Out[12]: {1, 2, 3}
```

Why use set?

- set operations (union and except): |, -
- test membership : in

```
In [13]: boys_like_hockey = ['Joshua', 'Allen']
         boys_like_violin = ['Joshua', 'David']

         # find out who like either hockey or violin
         # Attention! Use | operator, not '+'
         print(set(boys_like_hockey) | set(boys_like_violin))

         # wrong! this contains duplicates
         print(boys_like_hockey + boys_like_violin)

         # boys like hockey but don't like violin
         print(set(boys_like_hockey) - set(boys_like_violin))

         {'Joshua', 'Allen', 'David'}
         ['Joshua', 'Allen', 'Joshua', 'David']
         {'Allen'}
```

```
In [14]: # does Allen like hockey? does he like violin?
         'Allen' in boys_like_hockey

Out[14]: True
```

```
In [15]: 'Allen' in boys_like_violin

Out[15]: False
```

In [16]:
```
# modify a set
s = set([1,1,2,2,3])
print(s)
s.add(4)
print(s)
s.remove(3)
print(s)
```
```
{1, 2, 3}
{1, 2, 3, 4}
{1, 2, 4}
```

In [17]:
```
# Catchoa !!! set is orderless, s[0] is an Error!
```

Dict

a set of key-value pairs

construct dict

In [18]:
```
states = {'NC':'North Carolina', 'SC': 'South Carolina', 'CA': 'California'}
```

In [19]:
```
states
```
Out[19]:
```
{'CA': 'California', 'NC': 'North Carolina', 'SC': 'South Carolina'}
```

In [20]:
```
states['CA']
```
Out[20]:
```
'California'
```

In [21]:
```
states.keys()
```
Out[21]:
```
dict_keys(['NC', 'SC', 'CA'])
```

In [22]:
```
states.items()
```
Out[22]:
```
dict_items([('NC', 'North Carolina'), ('SC', 'South Carolina'), ('CA', 'Calif
ornia')])
```

In [23]:
```
states.values()
```
Out[23]:
```
dict_values(['North Carolina', 'South Carolina', 'California'])
```

In [24]:
```
# loop through dictionary.
for st in states:
    print(st, states[st])
```
```
NC North Carolina
SC South Carolina
CA California
```

```
In [25]: # dictionary is orderless
         # states[0] will give a KeyError
```

modify dict

```
In [26]: states.update({'NY':'New York'})
         print(states)

         {'NC': 'North Carolina', 'SC': 'South Carolina', 'CA': 'California', 'NY': 'N
         ew York'}
```

```
In [27]: states.update({'NC':'Tarheel'})
         print(states)

         {'NC': 'Tarheel', 'SC': 'South Carolina', 'CA': 'California', 'NY': 'New Yor
         k'}
```

why dict?

quick lookup

How to lookup? dict[key] or dict.get(key,

```
In [28]: states.get('NC','')

Out[28]: 'Tarheel'
```

```
In [29]: states.get('TN','')

Out[29]: ''
```

```
In [30]: # this give an error!
         # states['TN']
```

```
In [31]: for name in ['NC', 'SC', 'CA', 'TN', 'NY']:
             print(name, states.get(name, 'no information found'))

         NC Tarheel
         SC South Carolina
         CA California
         TN no information found
         NY New York
```