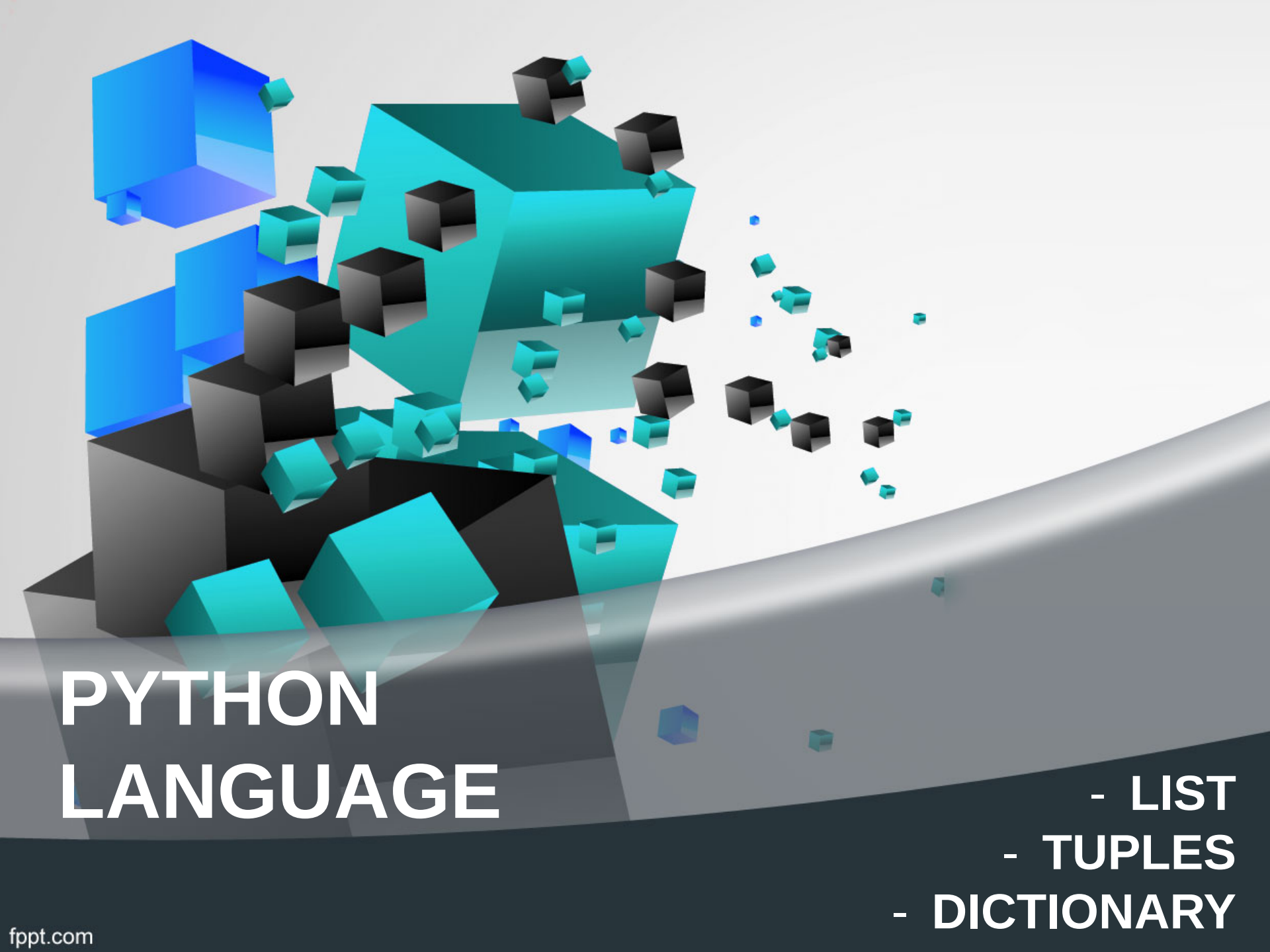

PYTHON LANGUAGE

- LIST
- TUPLES
- DICTIONARY

LIST

ลิสต์ คือชนิดข้อมูลที่หลากหลายมากที่สุดที่มีอยู่ในไพธอน ซึ่งสามารถเขียนเป็นลิสต์ของค่า นั้นๆคั่นด้วยเครื่องหมายจุลภาค (,) และอยู่ภายในวงเล็บ [ ]มีค่าเริ่มต้นที่ดัชนีเท่ากับ 0 ข้อดีของลิสต์ในไพธอนคือในลิสต์หนึ่งนั้นไม่จำเป็นต้องมีข้อมูลที่เป็นชนิดเดียวกัน

ตัวอย่าง

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, "a", "b"];
```

LIST

ในการเข้าถึง ค่าในลิสต์ใช้วงเล็บ [] พร้อมกับดัชนี เพื่อให้ได้ค่าที่มีอยู่ในลิสต์ ณ ดัชนีนั้น

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, "a", "b"];
print "list1[0]: ", list1[0];
```

```
list1[0]: physics
```

เราสามารถใช้เครื่องหมายบวก(+)ในการนำสายอักขระมาต่อกันก่อนแสดงผลและใช้เครื่องหมายดอกจันทร์ (*) ในการทำซ้ำข้อมูลตามจำนวนครั้งที่กำหนด

```
print list1 + list2;
Print list2 * 2;
```

```
['physics', 'chemistry', 1997,2000,1, 2, 3, 4, 5, "a", "b"]
[1, 2, 3, 4, 5, "a", "b", 1, 2, 3, 4, 5, "a", "b"]
```

LIST

การแสดงผลแบบอื่นๆ

```
list2 = [1, 2, 3, 4, 5, "a", "b"];

print "list2[1:5]: ", list2[1:5];
list2[1:5]: [2, 3, 4, 5]


print "list2[2:]: ", list2[2:];
list2[2:]: [3, 4, 5, "a", "b"];


print "list2[-2]: ", lis1[-2];
list2[-2]: a
```

LIST

เราสามารถแก้ไขข้อมูลและลบข้อมูลภายในลิสต์ได้โดยวิธีง่ายๆ โดยการลบนั้นจะสามรถเรียกใช้ฟังก์ชัน del แล้วตามด้วยค่าดัชนีในลิสต์นั้นที่เราจะลบ

ตัวอย่าง

```
list1 = ['physics', 'chemistry', 1997, 2000];
print "Value available at index 2 : ", list1[2];
list1[2] = 2001;
print "New value available at index 2 : ",list1[2];
del list1[2];
print list1;
```

ผลลัพธ์

```
Value available at index 2 : 1997
New value available at index 2 : 2001
['physics', 'chemistry', 2000]
```

LIST

Built-in List Functions & Methods

1. cmp(list1, list2) เป็นฟังก์ชันใช้สำหรับเปรียบเทียบ องค์ประกอบของทั้งสองลิสต์ โดยเปรียบเทียบแบบตัวต่อตัว คือ นำตัวแรกของทั้ง 2 ลิสต์มาเปรียบเทียบกัน ถ้าค่าเท่ากันก็จะใช้ตัวถัดไป มาเปรียบเทียบจนกว่าจะเจอค่าที่ต่างกันก็จะส่งค่ากลับมา

ถ้า List1 > List2 จะส่งค่ากลับมาเป็น 1

ถ้า List 1 < List2 จะส่งค่ากลับมาเป็น -1

ถ้า List1 == List2 จะส่งค่ากลับมาเป็น 0

โดยที่ตัวเลขจะมีค่าน้อยกว่าตัวอักษร และ a จะมีค่าน้อยกว่า b-z เรียงตามลำดับไป ถึงแม้ว่าเป็น String แล้วจำนวนตัวอักษรนั้นเยอะกว่าก็ไม่มีผล เช่น “z” จะมีค่ามากกว่า “aaaaa” เป็นต้น และ อักษรพิมพ์เล็กมีค่ามากกว่าอักษรพิมพ์ใหญ่ ดังตัวอย่าง

```
list1 = ['z','x','y'];
list2 = ['aaaa','z','x','y'];
print cmp(list1,list2)          ##  1
```

```
list1 = [0,687,9];
list2 = [1];
print cmp(list1,list2);         ##  -1
```

```
list1 = ['a',0,687,9];
list2 = ['A',0,687,9];
print cmp(list1,list2)          ##  1
```

```
list1 = ['a',0,687,9];
list2 = [‘a',0,687,9];
print cmp(list1,list2)          ##  0
```

LIST

2. len(list) เป็นฟังก์ชันใช้สำหรับนับจำนวนค่าที่อยู่ในลิสต์ ซึ่งฟังก์ชันจะส่งค่ากลับมาเป็นจำนวนของค่าที่มีอยู่ในลิสต์ ดังตัวอย่าง

```
list1 = [123, 'xyz', 'zara']
list2 = [456, 'abc']
print "First list length : ", len(list1);
print "Second list length : ", len(list2);
```

```
First list length : 3
Second list length : 2
```

LIST

3. max(list) , min(list) เป็นฟังก์ชันสำหรับหาค่าที่มากที่สุดและน้อยที่สุดที่อยู่ในลิสต์นั้น แล้วฟังก์ชันจะส่งค่านั้นกลับมา

```
list1 = [123, 'xyz', 'zara', 'abc'] ;
list2 = [456, 700, 200];
print "Max value element : ", max(list1);
print "Max value element : ", max(list2);
print "Min value element : ", min(list1);
print "Min value element : ", min(list2);
```

```
Max value element : zara
Max value element : 700
Min value element : 123
Min value element : 200
```

LIST

4. list(seq) เป็นฟังก์ชันสำหรับแปลง Tuples ให้กลายเป็นลิสต์

```
aTuple = (123, 'xyz', 'zara', 'abc');
aList = list(aTuple);
print "List elements : ", aList;
```

```
List elements : [123, 'xyz', 'zara', 'abc']
```

LIST

นอกจากนี้ยังมีเมธอดอื่นๆของลิสต์ที่เราสามารถเรียกใช้ได้อีกคือ

1. **list.append(obj)** เป็นเมธอดที่ใช้สำหรับเพิ่มวัตถุเข้าไปต่อท้ายในลิสต์นั้น ดังตัวอย่าง

```
list1 = ["physics", 1997, 2000]
list2 = [0, 2, 3, 0, 5, "a", "b"]
list1.append(list2)
print list1
```

```
['physics', 1997, 2000, [0, 2, 3, 0, 5, 'a', 'b']]
```

LIST

2. list.extend(seq) เป็นเมธอดที่ใช้สำหรับเพิ่มค่าเข้าไปต่อท้ายในลิสต์นั้นคล้ายๆกับ list.append(obj) แต่ต่างกันที่ extend ไม่ได้มองข้อมูลที่เพิ่มเข้าไปเป็นแบบวัตถุ แต่จะเพิ่มเข้าไปแบบเป็นสมาชิกในลิสต์นั้น ดังตัวอย่าง

```
list1 = ["physics", 1997, 2000]
list2 = [0, 2, 5, "a", "b"]
list1.append(list2)
list1.extend(list2)
print list1
```

```
['physics', 1997, 2000,[0, 2, 5, 'a', 'b'], 0, 2, 5,'a','b']
```

LIST

3. list.count(obj) เป็นเมธอดที่ใช้สำหรับนับว่ามีค่าที่เท่ากับวัตถุนี้อยู่กี่ค่า ดังตัวอย่าง

```
list1 = ["physics", 'chemistry', 1997, 2000]
list2 = [0, 2, 3, 0, 5, "a", "b"]
print list2.count(0)
print list1.count(list2)
```

```
2
0
```

LIST

4. list.index(obj) เป็นเมธอดที่ใช้สำหรับหาค่าตำแหน่งของวัตถุในลิสต์นั้นๆโดยจะส่งค่ากลับมาเป็นค่าตำแหน่งที่วัตถุนั้นอยู่โดยเริ่มที่ตำแหน่งแรกเป็น 0 ถ้ามีวัตถุที่หามากกว่า 1 วัตถุในลิสต์ เมธอดจะส่งค่ากลับมาโดยเป็นค่าตำแหน่งที่เจอตำแหน่งแรก แต่ถ้าไม่พบวัตถุนั้นในลิสต์ โ ปรแกรมจะเกิด error ดังตัวอย่าง

```
list1 = ["physics", 'chemistry', 1997, 2000]
list2 = [0, 2, 3, 0, 5, "a", "b"]
print list1.index("chemistry")          ##  1
print list2.index(5)                    ##  4
print list2.index(0)                    ##  0
print list2.index(10)       ## ValueError: 10 is not in list
```

LIST

5. list.insert(index, obj) เป็นเมธอดสำหรับใส่วัตถุลงไปในลิสต์นั้น โดยใส่ค่าตำแหน่งที่ต้องการเพิ่มกับวัตถุที่ต้องการเพิ่มลงไป โดยวัตถุนั้นสามารถเป็นลิสต์ได้ ดังตัวอย่าง

```
list2 = [0, 2, 3, 4, 5, "a", "b"]
list1 = [1,5,7,9,11]
list2.insert(1,list1)
print list2
```

```
[0, [1, 5, 7, 9, 11], 2, 3, 4, 5, 'a', 'b']
```

LIST

6. list.pop(obj=list[-1]) เป็นเมธอดสำหรับลบค่าทางขวาสุดของลิสต์ออก คือ ถ้ามองเป็น stack นั่นก็คือการลบค่าบนสุดออกนั้นเอง ดังตัวอย่าง

```
list2 = [0, 2, 3, 4, 5, "a", "b"]
print list2
list2.pop()
print list2
```

```
[0, 2, 3, 4, 5, 'a', 'b']
[0, 2, 3, 4, 5, 'a']
```

LIST

7. list.remove(obj) เป็นเมธอดที่ใช้ลบค่าในลิสต์ โดยการรับค่าของวัตถุนั้นแล้วนำไปหาภายในลิสต์ถ้าเจอค่าที่ตรงกันก็จะลบค่านั้นออก ถ้าไม่เจอวัตถุนั้น จะเกิด error ขึ้น ดังตัวอย่าง

```
list2 = [0, 2, 3, 4, 5, "a", "b"]
list2.remove(0)
print list2
list2.remove(1)
```

```
[2, 3, 4, 5, 'a', 'b']
ValueError: list.remove(x): x not in list
```

LIST

8. list.reverse() เป็นเมธอดที่ใช้กลับลิสต์จากตำแหน่งแรกเป็นตำแหน่งสุดท้าย ตำแหน่งสุดท้ายมาเป็นตำแหน่งแรก โดยเรียงลำดับกันไป ดังตัวอย่าง

```
list2 = [0, 2, 3, 4, 5, "a", "b"]
print list2
list2.reverse()
print list2
```

```
[0, 2, 3, 4, 5, 'a', 'b']
['b', 'a', 5, 4, 3, 2, 0]
```

LIST

9. list.sort([func]) เป็นเมธอดสำหรับเรียงค่าในลิสต์จากน้อยไปมากโดยที่ค่าตัวเลขจะมาก่อนตัวอักษร แต่ถ้าเป็นวัตถุจะอยู่หลังตัวเลขแต่ก่อนตัวอักษร และ ตัวอักษรพิมพ์เล็กมีค่ามากกว่าตัวอักษรพิมพ์ใหญ่ ดังตัวอย่าง

```
list1 = [0, 2, "A", "B", 'chemistry', 1997, 2000,[5,"5555"]]
print list1
list1.sort()
print list1
```

```
[0, 2,'A', 'B', 'chemistry', 1997, 2000, [5, '5555']]
[0, 2, 1997, 2000, [5, '5555'], 'A', 'B','chemistry']
```

TUPLES

ทูเพิล คือชนิดข้อมูลที่มีความคล้ายคลึงกับลิสต์ จะแตกต่างกันที่ลิสต์อยู่ภายใน [ ] และสามารถเพิ่มเติมหรือแก้ไขข้อมูลหรือขนาดได้ แต่ทูเพิลมีการประกาศค่าอยู่ในวงเล็บ () และไม่สามารถเปลี่ยนแปลงข้อมูลภายในได้

ตัวอย่าง

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, "a", "b");
```

TUPLES

ในการเข้าถึง ค่าในทูเพิล ใช้วงเล็บ [] พร้อมกับดัชนี เพื่อให้ได้ค่าที่มีอยู่ในทูเพิล ณ ดัชนีนั้น

```
tup1 = (12, 34.56,789,10);
tup2 = ('abc', 'xyz','pl','su');
```

ตัวอย่าง

```
print tup1[0];
print tup1[1:5];
print tup2[2:];
print tup1[-2];
```

ผลลัพธ์

```
12
(34.56, 789, 10)
('pl', 'su')
789
```

TUPLES

นอกจากนี้ เราสามารถใช้เครื่องหมายบวก(+)ในการนำสายอักขระมาต่อกันก่อนแสดงผลและใช้เครื่องหมายดอกจันทร์ (*) ในการทำซ้ำข้อมูลตามจำนวนครั้งที่กำหนดในการแสดงผล

```
tup1 = (12, 34.56,789,10);
tup2 = ('abc', 'xyz','pl','su');
```

ตัวอย่าง

```
print tup1 * 2;
print tup1+tup2
```

ผลลัพธ์

```
(12, 34.56, 789, 10, 12, 34.56, 789, 10)
(12, 34.56, 789, 10, 'abc', 'xyz', 'pl', 'su')
```

TUPLES

ทูเพิลไม่สามารถแก้ไขข้อมูลภายในได้เหมือนกับลิสต์ ถ้าจะแก้ไขข้อมูลได้มีทางเดียวคือต้องประกาศ ทูเพิลใหม่มารองรับ

ตัวอย่าง

```
tup1 = (12, 34.56,789,10);
tup2 = ('abc', 'xyz','pl','su');
tup3 = tup1 + tup2;
print tup3;
```

ผลลัพธ์

```
(12, 34.56, 789, 10, 'abc', 'xyz', 'pl', 'su')
```

และทูเพิลไม่สามารถลบข้อมูลในทูเพิลแบบระบุตำแหน่งได้ จะลบได้ก็คือต้องลบทั้งทูเพิล

TUPLES

Built-in Tuple Functions

มีฟังก์ชัน **cmp(tup1,tup2)** , **len(tuple)** , **max(tuple)** , **min(tuple)** เหมือนในลิสต์แต่จะมีฟังก์ชัน ที่ไม่เหมือนลิสต์คือ **tuple(seq)** เป็นฟังก์ชันสำหรับแปลงลิสต์ให้กลายเป็นทูลเพิล ดังตัวอย่าง

```
aList = [123, 'xyz', 'zara', 'abc'];
aTuple = tuple(aList)
print "Tuple elements : ", aTuple
```

```
uple elements : (123, 'xyz', 'zara', 'abc')
```

DICTIONARY

ตัวแปร dictionary หรือบางครั้งเราเรียกสั้นๆ ว่าตัวแปร dict เป็นตัวแปร ที่อ้างอิงข้อมูลของสมาชิกย่อยด้วยชื่อ (associate array) ตัวแปร dict เป็นตัวแปรที่ค่าภายในของมัน สามารถเปลี่ยนแปลงได้ (ซึ่งตรงกันข้ามกับตัวแปร tuple)

การประกาศตัวแปร dict และการอ้างถึงสมาชิกในตัวแปร dict

ตัวอย่าง

```
mydict = {}
mydict = {'nickName':'ProLang','age':99
print mydict['age']
print mydict['nickName']
print type(mydict)
```

ผลลัพธ์

```
99
ProLang
<type 'dict'>
```

DICTIONARY

<u>การอัพเดทค่า และการเพิ่มค่าลงในตัวแปร dict</u>

ตัวแปร dict เป็นตัวแปรที่เราสามารถเพิ่มค่าสมาชิกได้ แต่ชื่อที่อ้างถึงต้องไม่ซ้ำกับชื่อเก่าที่มีอยู่แล้ว ไม่เช่นนั้นจะเป็นการเปลี่ยนแปลงค่า แทนที่จะเป็นการเพิ่ม สังเกตจากตัวอย่างต่อไปนี้

ตัวอย่าง

```
mydict = {'nickName':'ProLang','age':99}
print mydict
mydict['phone'] = '0888888888'
print mydict
```

ผลลัพธ์

```
{'age': 99, 'nickName': 'ProLang'}
{'phone': '0888888888', 'age': 99, 'nickName': 'ProLang'}
```

DICTIONARY

การลบค่าในตัวแปร dict

การลบค่าในตัวแปร dict เราสามารถทำได้สองวิธี

วิธีแรก เราก็ใช้คำสั่ง del ค่าในตัวแปร dict ที่จับคู่กันอยู่ได้เลย

ตัวอย่าง

```
myDict = {'name':'hello','lastname':'world','age':36,'mobile':'081099999'}
>>> del myDict['mobile']
>>> print myDict
{'lastname': 'world', 'age': 36, 'name': 'hello'}
```

DICTIONARY

วิธีที่สอง เราใช้เมธอด clear ในการลบค่าในตัวแปร dict ตัวนั้นทั้งหมด

ตัวอย่าง

```
myDict = {'name':'hello','lastname':'world','age':36,'mobile':'081099999'}
>>> myDict.clear()
>>> myDict
```

DICTIONARY

นอกจากนี้ ยังมีเมธอดที่สำคัญๆที่มาพร้อมกับตัวแปร dict ที่มักใช้งานบ่อยๆ ดังนี้

1. dict.keys() ใช้สำหรับดึงค่า key ทีอยู่ในตัวแปร dict

```
myDict = {'age': 36, 'lastname': 'world', 'mobile': '081099999', 'name': 'hello'}

>>> myDict.keys()

['mobile', 'lastname', 'age', 'name']
```

2. dict.get(key,default=None) ใช้สำหรับดึงค่าจาก key ที่เรากำหนด ถ้าไม่มีอยู่จริง เราสามารถกำหนดเป็นอย่างอื่นได้ เช่น

```
>>> myDict.get('name',None)                    ##   'hello'

>>> myDict.get('sex',None)

>>> myDict.get('sex','No value')                ##  No value'
```

DICTIONARY

3. dict.has_key(key) ใช้สำหรับเช็คว่ามี key ที่กำหนดหรือไม่ จะให้ค่า true หรือ false กลับมา แล้วแต่กรณี เช่น

```
myDict = {'age': 36, 'lastname': 'world', 'mobile': '081099999', 'name': 'hello'}

>>> myDict.has_key('name')                ##  True

>>> myDict.has_key('sex')                 ##  False
```

4. dict.values() ดึงเฉพาะค่าทั้งหมดออกมาจากตัวแปร dict ไม่เอา key เช่น

```
>>> myDict.values()

['081099999', 'world', 36, 'hello']
```

DICTIONARY

5. dict.items() ดึง key และค่าของ key นั้นๆ จับคู่กันออกมาในรูปแบบตัวแปร list ซึ่งเป็นค่า tuple ซ้อนอยู่อีกที่ ตัวอย่างเช่น

```
myDict = {'age': 36, 'lastname': 'world', 'mobile': '081099999', 'name': 'hello'}

>>> myDict.items()

[('mobile', '081099999'),  ('lastname', 'world'),  ('age', 36),  ('name', 'hello')]

>>> mylist =  myDict.items()      #  สร้างตัวแปร มารับค่า

>>> mylist

[('mobile', '081099999'),  ('lastname', 'world') , ('age', 36),  ('name', 'hello')]

>>> mylist[0][0]  #แสดงค่าที่อยู่ในตัวแปร list

'mobile'
```

DICTIONARY

6. dict.fromkeys(seq[,value]) เราสามารถสร้างตัวแปร dict โดยกำหนดค่า Key จากข้อมูลที่อยู่ในรูป sequence (list , tuple) ได้ เช่น

```
>>> seq = ['url','domain','server']

>>> newDict = dict.fromkeys(seq)

>>> newDict

{'domain': None, 'server': None, 'url': None}
```

โดยเราสามารถกำหนดค่า value ลงไปเลย หรือจะไม่กำหนดลงไปก็ได้ แต่ถ้าไม่กำหนดลงไป มันจะให้มีค่าเป็น None ไว้ก่อน แต่ถ้าหากกำหนดลงไป เราสามารถทำได้ ดังนี้

```
>>> newDict2 = dict.fromkeys(seq,'xxx')

>>> newDict2

{'domain': 'xxx', 'server': 'xxx', 'url': 'xxx'}
```

DICTIONARY

7. dict.copy() ช่วยให้เรา copy ค่าตัวแปร dict ให้มีค่าเหมือนกันได้ เช่น

```
>>> dict1 = {'name':'xxx','value':10}
>>> dict2 = dict1.copy()
>>> dict2                          # {'name': 'xxx', 'value': 10}
```

มันต่างจากวิธี dict2 = dict1 ตรงที่ การเปลี่ยนแปลงที่ dict1 จะไม่ส่งผลให้ dict2 เปลี่ยนแปลงตามไปด้วย เรามาดูว่าจะเกิดไรขึ้น หากเรา ไม่ใช้ dict.copy()

```
>>> dict3 = dict1
>>> dict3                          #  {'name': 'xxx', 'value': 10}
>>> dict1['name'] = 'zzzz'
>>> dict3['name']                  # ‘zzzz‘  จะเห็นว่าค่าใน key 'name' ของ dict3 ถูกเปลี่ยนแปลง
```

DICTIONARY

8. dict.update(dict2) เราสามารถอัพเดทค่าที่อยู่ใน dict ได้ เมื่อ key ที่อยู่ใน dict2 ตรงกับ key ที่อยู่ใน dict ตรงกัน เช่น

```
>>> dict1                          # {'name': 'xxxx', 'value': 10}
>>> dict2                          # {'name': 'kkkkk', 'value': 10}
>>> dict1.update(dict2)
>>> dict1                          # {'name': 'kkkkk', 'value': 10}
```

แต่ถ้าค่า key ไม่ตรงกัน จะเป็นการเพิ่มค่าคู่ใหม่ลงไปใน dict เช่น

```
>>> dict1                          # {'name': 'kkkkk', 'value': 10}
>>> dict4 = {'option': 200}
>>> dict4.update(dict1)
>>> dict4                          # {'name': 'kkkkk', 'option': 200, 'value': 10
```

DICTIONARY

9. dict.setdefault(key, default=None) เหมือนๆ กับคำสั่ง dict.get(key,default=None) ต่างกันตรงที่ ถ้ามันหา key ที่เราต้องการไม่เจอ มันจะเติม key นั้นให้ และกำหนดค่า default ที่เราต้องการไปให้ทันที สังเกตดูจากตัวอย่าง

```
>>> dict4                          {'name': 'kkkkk', 'option': 200, 'value': 10}
>>> dict4.setdefault('sex','Male')  'Male'
>>> dict4                          {'name': 'kkkkk', 'option': 200, 'sex': 'Male', 'value':
10}
```

ถ้าเราใช้คำสั่ง dict4.get('sex','Male') จะไม่ปรากฏคู่ของ sex เพิ่มใน dict4

```
>>> dict4                          {'name': 'kkkkk', 'option': 200, 'value': 10}
>>> dict4.get('sex','Male')         'Male'
>>> dict4                          {'name': 'kkkkk', 'option': 200, 'value': 10}
```

จัดทำโดย

1. นาย ธีรวัฒน์ สาลีคำ รหัสนักศึกษา 07550445 (Dict.)
2. นาย ปกรณ์ นุชเจริญ รหัสนักศึกษา 07550450 (List , Tuple)
3. นาย วรพงษ์ ภู่ระย้า รหัสนักศึกษา 07550465 (List,Tuple)