Introduction to Computer Science and Programming Using Python(3)

# Week three: Structured Types

## 1. TUPLES 元组

* an ordered sequence of elements, can mix element types.
* **immutable**, cannot change element values
* represented with parentheses ()
* >>> te = ()  
  >>> t = (2, "one", 3)  
  >>> t  
  (2, 'one', 3)  
  >>> t[0]  
  2  
  >>> t + (5,6)  
  (2, 'one', 3, 5, 6)  
  >>> t[1:2] #不包括后括号里面的参数  
  ('one',)  
  >>> t[1:3] #必须增加一位来包括前面的参数  
  ('one', 3)  
  >>> ('one',) #，代表这是一个元组  
  ('one',)  
  >>> ('one') #无，代表这是个字符串  
  'one'

## 2. List 列表

* ordered sequence
* square brackets: []

#### mutable!!!

## 3. Operations on Lists

* **ADD---.append/.extend**（*objectname.dosomething*()）
* >>> l = [2,1,3]  
  >>> print(l)  
  [2, 1, 3]  
  >>> l.append(5) #添加单个对象至列表最后  
  >>> print(l)  
  [2, 1, 3, 5]  
  >>> L1 = [2,1,3]  
  >>> L2 =[4,5,6]  
  >>> L3 = L1 + L2 #列表合并  
  >>> print(L3)  
  [2, 1, 3, 4, 5, 6]  
  >>> L1.extend([0,10]) #添加列表  
  >>> print(L1)  
  [2, 1, 3, 0, 10]

##### Remove---del(List[])/L.pop()/L.remove

* >>> L  
  [1, 3, 6, 3, 7, 0]  
  >>> L.remove(3)  
  >>> L  
  [1, 6, 3, 7, 0]  
  >>> del(L[1])  
  >>> L  
  [1, 3, 7, 0]  
  >>> L.pop()  
  0  
  >>> L  
  [1, 3, 7]  
  >>> L.pop(0)  
  1  
  >>> L  
  [3, 7]

##### Convert Lists to Strings and back---list()/''.join()

* >>> s = 'I < 3 cs'  
  >>> list(s) #字符串变列表  
  ['I', ' ', '<', ' ', '3', ' ', 'c', 's']  
  >>> s.split('<') #分割字符串  
  ['I ', ' 3 cs']  
    
  >>> L = ['a','b','c']  
  >>> ''.join(L) #列表变字符串  
  'abc'  
  >>> L  
  ['a', 'b', 'c']  
  >>> '\_'.join(L)  
  'a\_b\_c'

##### Other List operations---sorted()/.sort()/.reverse()

* >>> L = [9, 6, 0, 3]  
  >>> sorted(L)  
  [0, 3, 6, 9]  
  >>> L  
  [9, 6, 0, 3]  
  >>> L.sort  
  <built-in method sort of list object at 0x000001C63CA297C8>  
  >>> L.sort()  
  >>> L  
  [0, 3, 6, 9]  
  >>> L.reverse()  
  >>> L  
  [9, 6, 3, 0]
* MORE: https://docs.python.org/3/tutorial/datastructures.html

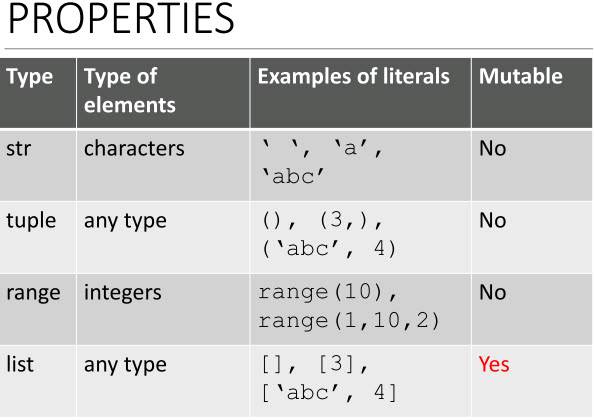
## 3. Mutation

* If two lists print the same thing, does not mean they are the same structure.
* >>> cool  
  ['blue', 'green', 'grey']  
  >>> chill  
  ['blue', 'green', 'grey'] #列表中对象一样，不一定列表相同  
  >>> chill[2] = 'blue'  
  >>> chill  
  ['blue', 'green', 'blue']  
  >>> cool  
  ['blue', 'green', 'grey']
* **Cloning a list**: create a new list and **copy every element using** chill = cool[:]
* >>> cool = ['blue','green','grey']  
  >>> chill = cool[:] #克隆列表，生成新的列表，不指向原列表。  
  >>> chill.append('black')  
    
    
  >>> chill  
  ['blue', 'green', 'grey', 'black']  
  >>> cool  
  ['blue', 'green', 'grey']
* calling.sort()**mutates** the list,returns nothing
* calling sorted() does not mutate list, must assign result to a variable.
* **Lists of Lists of Lists**
  + can have nested lists
  + side effects still possible after muation
  + >>> warm = ['yellow', 'orange']  
    >>> hot = ['red']  
    >>> brigtcolor = [warm]  
    >>> brigtcolor  
    [['yellow', 'orange']]  
    >>> brigtcolor.append(hot) #添加另一个列表  
    >>> brigtcolor  
    [['yellow', 'orange'], ['red']]  
    >>> hot.append('pink') #原列表变化  
    >>> hot  
    ['red', 'pink']  
    >>> bright.color  
    >>> brigtcolor #所有指向列表也跟着变化  
    [['yellow', 'orange'], ['red', 'pink']]
* **AVIOD** mutating a list as you are iterating over it!!!
* def remove\_dups (L1, L2):  
   for e in L1:  
   if e in L2:  
   L1.remove(e)   
     
  >>> L1 = [1,2,3,4]  
  >>> L2 = [1,2,5,6]  
  >>> remove\_dups(L1,L2)  
  >>> L1 #重复项[2]并未被剔除，因为代码中列表被改变，程序终止。  
  [2, 3, 4]  
  >>> L2  
  [1, 2, 5, 6]  
    
  #改进方法，给列表重新赋值建立新的列表  
  def remove\_dups\_new(L1,L2):  
   L1\_copy = L1[:] #对新列表进行运算，不涉及原列表改变  
   for e in L1\_copy:  
   if e in L2:  
   L1.remove(e)  
    
  >>> L1 = [1,2,3,4]  
  >>> L2 = [1,2,5,6]  
  >>> remove\_dups\_new(L1,L2)  
  >>> L1  
  [3, 4]  
  >>> L2  
  [1, 2, 5, 6]
* **Map Function** *Return an iterator that applies function to every item of iterable, yielding the results.*
* >>> list(map(abs, [1,-2,3,-4,3.5, 6.8]))  
  [1, 2, 3, 4, 3.5, 6.8]  
    
  >>> list(map(int, [1,-2,3,-4,3.5, 6.8]))  
  [1, -2, 3, -4, 3, 6]

## 4. Quick Review

* STRINGS, TUPLES, RANGES, LISTS
  + Common operations
  + seq[i]---i th element of sequence
  + len(seq)---length of sequence
  + seq1 + seq2---concatenation of sequences (not range)
  + n\*seq---sequence that repeats seq n times (not range)
  + seq[start:end] ---slice of sequence
  + e in seq---True if e contained in sequence
  + e not in seq---True if e contained in sequence
  + for e in seq---iterates over elements of sequence

##### Properties

* 

## 5. Dictionary 字典

* my\_dict = {}
* >>> grades = {'Ann':'B', 'John':'A+','Denise':'A','Katy':'A'}  
  >>> grades  
  {'Ann': 'B', 'John': 'A+', 'Denise': 'A', 'Katy': 'A'}  
  >>> grades['John']  
  'A+'  
  >>> grades['A']  
  Traceback (most recent call last):  
   File "<pyshell#77>", line 1, in <module>  
   grades['A']  
  KeyError: 'A'
* **values**
  + any type(immutable and mutable)
  + can be **duplicates**
  + dictionary values can be lists, even other dictionaries
* **keys**
  + must be **unique**
  + **immutable** type: int,float,string,tuple,bool
  + careful with float type as akey
* **no order** to keys or values!!!
* >>> d = {4:{1:0},(1,3):'twelve', 'const':[3.14, 2.7, 8.44]}  
  >>> d  
  {4: {1: 0}, (1, 3): 'twelve', 'const': [3.14, 2.7, 8.44]}

##### LIST VS. DICTIONARIES

* 