Part I. Basic

1. Python Data Types

- Text: str【'a'】
- Numeric: int【1】, float【1.2】, complex
- Sequence: list【[1, 2, 3]】, tuple【(1, 2, 3)】, range【range(5)】【range(5, 8)】
- Mapping: dict 【{'A': 1, 'B': 'a'}】
- Set: set 【{}】, frozenset
- Boolean: bool 【Ture】【False】
- Binary: bytes, bytearray, memoryview

2. Range

- range(10) --> range(0, 10) 【0, 1, 2, 3, 4, 5, 6, 7, 8, 9】
- range(2, 10) --> range(2, 10) 【2, 3, 4, 5, 6, 7, 8, 9】
- range(0, 10, 2) --> range(0, 10, 2) 【0, 2, 4, 6, 8】
- range(5, -1, -1) --> range(5, -1, -1) 【5, 4, 3, 2, 1, 0】 【从5到-1前一位即0，by-1】

3. Indexing

- string[] 【找到list里面index对应的元素，string indexing】
- list.index(element, start, end) 【找到list里面对应数字或string或其他的index】

```
In [1]: x = 'abcdef'
        x[0]

In [2]: x[2:4] # 注意index2-4不包含4

In [3]: x[2:] # 取index 2开始后面所有

In [4]: x[-3:] # 取最后3位数

In [5]: y = [1, 2, 3, 4]
        y[-1]  # 取倒数第一位，即最后一位

In [6]: y.index(1)

In [7]: #y.index(5)  # ValueError, the element is not in the list
```

4. Print

- x = 0 y = 1 print(x, y) ≣ 0 1 • list = [0, 1, 2, 3] print(list, end = ',') ≣ 0,1,2,3 •

```
In [8]: x = 0
        y = 1
        print(x, y)

In [9]: l = [0, 1, 2, 3]
        print(l, end = ',')
```

5. Sort

5.1 Two Types of Sorting

- list.sort() v.s. sorted(list)

```
In [10]: A = [2, 3, 10, 6, 1, 2]
         sorted(A)  # 注意：sort结果储存在新的list里没有改变 list A

In [11]: A

In [12]: A.sort()  # 注意：改变了 list A，不会 print out
         A

Out[12]: [1, 2, 2, 3, 6, 10]
```

5.2 Descending Order

- list.sort(reverse = True) 【降序】
- sorted(list, reverse=True) 【降序】

```
In [13]: A.sort(reverse = True)
         A

Out[13]: [10, 6, 3, 2, 2, 1]
```

5.3 Sort by Function Key

- list.sort(key = len) 【按长度排序】
- sorted(list, key = len) 【按长度排序】
- list.sort(key = takeSecond) 【指定第二个元素排序】
- sorted(list, key = takeSecond) 【指定第二个元素排序】

```
In [14]: B = ["abc", "ghijk", "def", "lmno"]
         B.sort(key = len)
         B

Out[14]: ['abc', 'def', 'lmno', 'ghijk']

In [15]: # Self-Defined Function: take second element for sort
         def takeSecond(elem):
             return elem[1]

         C = [(2, 2), (3, 4), (4, 1), (1, 3)]
         C.sort(key = takeSecond)
         C

Out[15]: [(4, 1), (2, 2), (1, 3), (3, 4)]
```

6. Basic Calculation Operation

- A // B 【商】
- A % B 【余数】
- A ** B 【A的B次方】

```
In [16]: 7//2

Out[16]: 3

In [17]: 7%2

Out[17]: 1

In [18]: 2**3

Out[18]: 8
```

7. Print

```
In [19]: print("ABC", "DE")

         ABC DE

In [20]: print("ABC" + "DE")

         ABCDE

In [21]: #print(1 + "ABC" + "DE")  # TypeError 注意：不能混杂 int 和 string
```

8. Integer

- int()

```
In [22]: int("100")

Out[22]: 100
```

- int("string decimal") v.s. int(decimal)

```
In [23]: #int("9.9")  # ValueError
         int(9.9)

Out[23]: 9
```

9. List Basics

- [list] * number
- list.count(item) 【count】
- list(range(number))

```
In [24]: [0]*3

Out[24]: [0, 0, 0]

In [25]: ["abc"] * 3

Out[25]: ['abc', 'abc', 'abc']

In [26]: list(range(10))

Out[26]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

10. List – Append, Extend, Insert, Pop, Remove

- Append
  - list.append(item) 【O(1)】
- Append v.s. Extend:

```
In [27]: l = [1, 2, 3]
         new = [4, 5]
         l.append(new)
         l

Out[27]: [1, 2, 3, [4, 5]]

In [28]: l = [1, 2, 3]
         new = [4, 5]
         l.extend(new)
         l

Out[28]: [1, 2, 3, 4, 5]
```

- Insert
  - list.insert(index, item) 【在对应index位置插入item, O(n)】
- Pop
  - list.pop() 【默认去掉list里最后一个item, O(1)】
  - list.pop(index) 【去掉对应index的item, O(n)】

```
In [29]: l = [10, 20, 30, 40, 50]
         x = l.pop(2)  # 注意可以赋值记录被去掉的那个item!
         print(x)
         print(l)

         30
         [10, 20, 40, 50]

In [30]: l = [1, 2, 3]
         print(l.pop(-1))  # 负数index用法
         print(l)

         3
         [1, 2]
```

- Remove
  - list.remove(item) 【去掉对应第一个item（如有重复），注意remove不像pop，remove必须要argument】

11. String

- ''.join(set(“string”)) 【不保证character的排列顺序】

```
In [31]: 'abcde'[0]

Out[31]: 'a'

In [32]: ''.join(set('stttringg'))

Out[32]: 'trnsig'
```

- 'string'.isnumeric() --> return boolean
- 'letter'.isalpha() --> return boolean 【是否是字母，不论大小写】

```
In [33]: '100'.isnumeric()  # 是否是数字

Out[33]: True

In [34]: 'a'.isalpha()

Out[34]: True

In [35]: 'A'.isalpha()

Out[35]: True
```

- 'string'.count('xxx') 【数某个character/substring出现了几次】
- '%s other_content_1 %s other_content_2 %s' % (a, b, c) 【a, b, c分别代替三个%s】
- Floating String:
  - “%.1f” % x 【x保留1位小数】

```
In [36]: print("%s 1 %s 2 %s" % ("A", "B", "C"))

         A 1 B 2 C

In [37]: print("%s1%s2%s" % ("A", "B", "C"))  # 注意空格区别

         A1B2C
```

12. Dictionary

- dict = {'A': 100, 'B': 'abcde', ...}【字典，key可以是数字或者string】
- dict.keys() 【一个list包含所有keys】
- dict.values() 【一个list包含所有values】
- dict.items() 【一个list包含所有[(key_1, value_1), (key_2, value_2), ...]，tuple形式】
- dict.update()
  - dict.update('C': 10) 【多加了一个key 'C'】
  - dict.update(A = 1000) 【把key 'A'的value换成了1000，注意这里key不用加引号！】
  - dict.update([('C', 10), ('D', 20)]) 【增加多个keys，注意这里里面是tuple格式】
- dict_1.update(dict_2) 【把两个dictionaries合并】
- Index: dictionary 不能用index用key！
- Sort Dictionary 【注意output是tuple格式！】
  - (1) sorted(dict.items(), key = lambda x: x[1], reverse = True) 【x[0]: keys, x[1]: values; descending order: reverse = True，这里只sort by values】
  - (2) sorted(dict.items(), key = lambda x: (x[1], x[0])) 【sort by values then keys】
  - (3) sorted(dict.items(), key = lambda x: (-x[1], x[0])) 【sort by descending values then ascDending keys】

```
In [38]: dic = {'A': 100, 'B': 'abcde'}
         dic.keys()

Out[38]: dict_keys(['A', 'B'])

In [39]: dic.values()

Out[39]: dict_values([100, 'abcde'])

In [40]: dic.items()

Out[40]: dict_items([('A', 100), ('B', 'abcde')])

In [41]: dic.update({'C': 10})  #多加了一个key 'C'
         dic

Out[41]: {'A': 100, 'B': 'abcde', 'C': 10}

In [42]: dic.update(A = 1000)  #把key 'A'的value换成了1000，注意这里key不用加引号！
         dic

Out[42]: {'A': 1000, 'B': 'abcde', 'C': 10}

In [43]: dic.update([('C', 10), ('D', 20)])  #增加多个keys，注意这里里面是tuple格式
         dic

Out[43]: {'A': 1000, 'B': 'abcde', 'C': 10, 'D': 20}
```

13. Set

- An unordered collection of distinct hashable objects 【注意顺序可能打乱】
- {element1, element2, ...}
- set([list])

```
In [44]: set([1, 2, 3])

Out[44]: {1, 2, 3}

In [45]: A = set([1, 2, 2, 2, 3, 4, 5])  # 去掉重复并且变成set格式{}
         A

Out[45]: {1, 2, 3, 4, 5}

In [46]: A.add(5)  # add，只添加不重复的元素，重复的element不会加上去
         A

Out[46]: {1, 2, 3, 4, 5}

In [47]: A.add(6)
         A

Out[47]: {1, 2, 3, 4, 5, 6}

In [48]: B = list(A)
         B

Out[48]: [1, 2, 3, 4, 5, 6]
```

14. Tuple

- () 【括号】

```
In [49]: l = [('A', 100), ('B', 'abcde')]
         l[0][0]

Out[49]: 'A'

In [50]: l.append(('C', 200))
         l

Out[50]: [('A', 100), ('B', 'abcde'), ('C', 200)]
```

15. Tensor 张量

- 多层arrays
- Rank: number of dimension
- Shape: number of rows and columns
- Type: data type of tensor’s elements
- Python:
  - (1) 0阶 s = 123
  - (2) 1阶 v = [1.1, 2.2, 3.3] 【vector】
  - (3) 2阶 m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] 【matrix】
  - (4) 3阶 t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]] 【tensor】

16. Type Change

- Integer
  - int('1000') 1000
  - int('9.99') 【value error】
- Numeric
- String
  - str() 【变成string】
  - list("string") 【type 转化成 list】
  - ''.join(list) 【list 转化成 string】
  - combine string: 【string也可以用+=】

```
In [51]: list("abcd")

Out[51]: ['a', 'b', 'c', 'd']

In [52]: A = 'abc'
         A += 'def'
         A

Out[52]: 'abcdef'
```

- set("string")
- list(set("string"))

```
In [53]: set('Sstttringg8')

Out[53]: {'8', 'S', 'g', 'i', 'n', 'r', 's', 't'}

In [54]: list(set('Sstttringg8'))

Out[54]: ['t', 'r', 'S', 'n', '8', 's', 'i', 'g']
```

- List --> Tuple ([list])

```
In [55]: l = [1, 2, 3]
         tuple(l)  # 注意:是tuple([…])

Out[55]: (1, 2, 3)
```

17. Excape Character 转义字符

- \a - BEL 响铃
- \b - BS 退格
- \f - FF 换页
- \n - LF 换行
- \r - CR 回车
- \t - HT 水平制表（跳到下一个tab位置）
- \v - VT 垂直制表
- \ - Backslash 反斜线
- \' - 单引
- \" - 双引
- \0 - NULL 空字符

18. List Comprehension

```
In [58]: x = [i for i in range(2, 10)]
         print(x)

         [2, 3, 4, 5, 6, 7, 8, 9]
```

19. Recursive Function 【递归算法：公式套用】

- 求和1, 2, 3, ..., n

```
In [65]: def recurSum(n):
             if n <= 1:
                 return n
             return n + recurSum(n - 1)
         print(recurSum(5))

         15
```

20. Break, Continue, Pass

- break – terminates CURRENT loop and resume execution 【while/for loop】
- continue – rejects ALL REMAINING statements in the CURRENT iteration loop and moves control back to the top of the loop 【跳过那一条去到下一个，如果还有loop就继续，没有就下一个环节】
- pass – a null operation, nothing happens when it executes

21. Deep Copy v.s. Shallow Copy

- 例子1:

```
In [70]: import copy
         origin = [1, 2, [3, 4]]
         #origin 里边有三个元素：1,  2, [3, 4]
         cop1 = copy.copy(origin)  # shallow copy
         cop2 = copy.deepcopy(origin)  # deep copy
         cop1 == cop2
         cop1 is cop2
         #cop1 和 cop2 看上去相同，但已不再是同一个object

Out[70]: False

In [71]: origin[2][0] = "hey!"
         origin

Out[71]: [1, 2, ['hey!', 4]]

In [72]: cop1

Out[72]: [1, 2, ['hey!', 4]]

In [73]: cop2

Out[73]: [1, 2, [3, 4]]
```

把origin内的子list [3, 4] 改掉了一个元素，观察 cop1 和 cop2
浅拷贝会跟着变，深拷贝不会

- 例子2:

```
In [76]: a = [1, 2, 3]
         b = a
         a = [4, 5, 6] # 赋新的值给 a
         a

Out[76]: [4, 5, 6]

In [77]: b
         # a 的值改变后，b 并没有随着 a 变

Out[77]: [1, 2, 3]

In [80]: a = [1, 2, 3]
         b = a
         a[0], a[1], a[2] = 4, 5, 6  # 改变原来 list 中的元素
         a

Out[80]: [4, 5, 6]

In [81]: b
         # a 的值改变后，b 随着 a 变了

Out[81]: [4, 5, 6]
```