

# Python Basics

> Miranda Zhao

> click cell -> run all

## Unit 0 Preparation

- Variable, Function, Return, Conditional Statements, FizzBuzz, For Loop, While Loop

## Unit 1 List

- Get elements, Construct, Sort, Reverse

## Unit 2 Manipulation

- Lambda, Append& Insert, Remove, Pop, Modify& Switch
- Example: Insert, Binary Search, Pop

## Unit 3 Set

- Remove duplicate, Search
- Example: List2Num, Maximum Stock Gain, Pascal's Triangle

## Unit 4 String

- Replace characters, Concatenate &reverse, String &list
- Example: Palindrome, Valid Parentheses, Ngrams, All Red, Reverse Sentence, Reverse Words

## Unit 5 Boolean

- Use integers, lists, and strings as boolean, Call function, Import
- Example: Remove Punctuations, Capitalize First Letters & Letter Palindrome

## Unit 6 Dictionary

- Get key& values, Zip
- Example: Key Value Switch, Rank Keys by Values, Character Frequency, Unique Only, Same Pattern, Anagrams, Two Sum

## Unit 7 Tuple

- Format, Objects, Define& initialize a class, Self in class, Calculate using a custom class

## Unit 8 Read& Write

- Line break, Write Strings into a file, Read from a file, CSV

# unit 0 Preparation

## 0.1 Variable

```
In [1604]: #There are 200 people. Everyday, every person needs 2kg of food to survive.  
#If the capacity of a truck is 1 tons(1000kg), how many trucks do we need to s  
    ◀ ▶
```

```
In [1605]: 2 * 200 * 5 / 1000
```

```
Out[1605]: 2.0
```

```
In [1606]: food_in_kg_per_person_per_day = 2
```

```
In [1607]: num_persons = 200
```

```
In [1608]: num_days = 5
```

```
In [1609]: truck_capacity_in_kg = 1000
```

```
In [1610]: total_food_in_kg_per_day = food_in_kg_per_person_per_day * num_persons  
print(total_food_in_kg_per_day)
```

```
400
```

```
In [1611]: total_food_in_kg = total_food_in_kg_per_day * num_days  
print(total_food_in_kg)
```

```
2000
```

```
In [1612]: num_trucks = total_food_in_kg / truck_capacity_in_kg  
print(num_trucks)
```

```
2.0
```

## 0.2 Function

```
In [1613]: #If we need to send enough food for 400 people to survive for 10 days,  
#how many trucks do we need
```

```
In [1614]: food_in_kg_per_person_per_day = 2
num_persons = 400
num_days = 10
truck_capacity_in_kg = 1000
total_food_in_kg_per_day = food_in_kg_per_person_per_day * num_persons
total_food_in_kg = total_food_in_kg_per_day * num_days
num_trucks = total_food_in_kg / truck_capacity_in_kg
print(num_trucks)
#8
```

8.0

```
In [1615]: def get_num_trucks():
    food_in_kg_per_person_per_day = 2
    num_persons = 400
    num_days = 10
    truck_capacity_in_kg = 1000
    total_food_in_kg_per_day = food_in_kg_per_person_per_day * num_persons
    total_food_in_kg = total_food_in_kg_per_day * num_days
    num_trucks = total_food_in_kg / truck_capacity_in_kg
    return num_trucks
#print(num_trucks)
```

```
In [1616]: #为了显示得到结果, 可以在函数最后写@return variable; @print(va)
get_num_trucks()
#8.0
```

Out[1616]: 8.0

```
In [1617]: #也可以将变量写在小括弧中, 下次在小括弧中输入变量值即可:
def get_num_trucks(food_in_kg_per_person_per_day, num_persons, num_days, truck
    total_food_in_kg_per_day = food_in_kg_per_person_per_day * num_persons
    total_food_in_kg = total_food_in_kg_per_day * num_days
    num_trucks = total_food_in_kg / truck_capacity_in_kg
    return num_trucks
#print(num_trucks)
```

```
In [1618]: get_num_trucks(2, 400, 10, 1000)
#8.0
```

Out[1618]: 8.0

```
In [1619]: get_num_trucks(2, 200, 5, 1000)
#2.0
```

Out[1619]: 2.0

## 0.3 Return

### 0.3.1 print

如果用print，则不能assign the value to the new variable, new variable is null, 两个新的变量相加不会返回结果

```
In [1620]: def get_num_trucks(food_in_kg_per_person_per_day, num_persons, num_days, truck_capacity_in_kg):
    total_food_in_kg_per_day = food_in_kg_per_person_per_day * num_persons
    total_food_in_kg = total_food_in_kg_per_day * num_days
    num_trucks = total_food_in_kg / truck_capacity_in_kg
    print(num_trucks)
```

```
In [1621]: num_trucks_for_villiage_a = get_num_trucks(2, 400, 10, 1000)
#8.0
```

8.0

```
In [1622]: num_trucks_for_villiage_a
#不会print结果
```

```
In [1623]: num_trucks_for_villiage_b = get_num_trucks(2, 200, 5, 1000)
#2.0
```

2.0

```
In [1624]: num_trucks_for_villiage_b
#不会print结果
```

```
In [1625]: #当使用print后, 两个新的变量相加不会返回结果
#num_trucks_for_villiage_a + num_trucks_for_villiage_b

# -----
# NameError                                         Traceback (most recent call last)
# Cell In[71], line 1
# ----> 1 num_trucks_for_villiage_a + num_trucks_for_villiage_b

# NameError: name 'num_trucks_for_villiage_a' is not defined
```

### 0.3.2 return

如果用return, 而不用print, 那么就可以use the result from calling the function to do further calculations

In [1626]: #assignment: 如果使用return, 再新产生变量后, 可以得到num\_trucks作为函数结果:

```
def get_num_trucks(food_in_kg_per_person_per_day, num_persons, num_days, truck_capacity_in_kg):
    total_food_in_kg_per_day = food_in_kg_per_person_per_day * num_persons
    total_food_in_kg = total_food_in_kg_per_day * num_days
    num_trucks = total_food_in_kg / truck_capacity_in_kg
    return num_trucks
```

In [1627]: num\_trucks\_for\_villiage\_a = get\_num\_trucks(2, 400, 10, 1000)

In [1628]: num\_trucks\_for\_villiage\_a  
#8.0

Out[1628]: 8.0

In [1629]: num\_trucks\_for\_villiage\_b = get\_num\_trucks(2, 200, 5, 1000)

In [1630]: num\_trucks\_for\_villiage\_b  
#2.0

Out[1630]: 2.0

In [1631]: num\_trucks\_for\_villiage\_a + num\_trucks\_for\_villiage\_b  
#10.0

Out[1631]: 10.0

## 0.4 Python practice questions

In [1632]: def HelloWorld():
 return "Hello World"

In [1633]: def LB2KG(num\_in\_pound):
 return num\_in\_pound \* 0.4536

## 0.5 Conditional Statements

In [1634]: def Odd(num):
 if num % 2 == 0:
 return False
 if num % 2 != 0:
 return True

```
In [1635]: #improve code by else:
def Odd(num):
    if num % 2 == 0:
        return False
    else:
        return True
```

```
In [1636]: #improve code by more concise:
def Odd(num):
    return num % 2 != 0
```

```
In [1637]: #%% 是取余 remainder:
3 % 2
#1
```

Out[1637]: 1

```
In [1638]: 4 % 2
#0
```

Out[1638]: 0

```
In [1639]: 8 % 3
#2
```

Out[1639]: 2

```
In [1640]: a = 4
a
#4
```

Out[1640]: 4

```
In [1641]: #Boolean:
a == 5
#False
```

Out[1641]: False

## 0.6 FizzBuzz

```
In [1642]: #FizzBuzz
#Build a function FizzBuzz which takes a positive integer as input,
#if the number is divisible by 3, the function returns "Fizz".
#if the number is divisible by 5, the function returns "Buzz".
#if the number is divisible by both 3 and 5, the function returns "FizzBuzz".
#else, the function returns the number itself.
```

In [1643]:

```
#①注意if num % 3 == 0 and num % 5 == 0: 必须放在第一行,
#否则当一个数除以3满足后, 就不会进入 % 5 和 %3 and %5里了, 造成结果错误:
def FizzBuzz(num):
    if num % 3 == 0 and num % 5 == 0:
        return "FizzBuzz"
    if num % 3 == 0:
        return "Fizz"
    if num % 5 == 0:
        return "Buzz"
    return num
print(FizzBuzz(45))
#FizzBuzz
```

FizzBuzz

In [1644]:

```
#②打开jupyter, 学习debug:
#45满足三条中的每一条, 所以返回最后一次的结果。
#而我们希望的是如果满足一条, 就不往下跑了。
def FizzBuzz(num):
    result = num
    if num % 15 == 0:
        result = "FizzBuzz"
        print(result, "can be divided by 15")
    if num % 3 == 0:
        result = "Fizz"
        print(result, "can be divided by 3")
    if num % 5 == 0:
        result = "Buzz"
        print(result, "can be divided by 5")
    return result
print(FizzBuzz(45))
# FizzBuzz can be divided by 15
# Fizz can be divided by 3
# Buzz can be divided by 5
# Buzz
```

FizzBuzz can be divided by 15  
 Fizz can be divided by 3  
 Buzz can be divided by 5  
 Buzz

In [1645]: #学习更正, 可以加上else和tab后两条条件, 这样满足第一个条件, 就不会往下跑了。

```
def FizzBuzz(num):
    result = num
    if num % 15 == 0:
        result = "FizzBuzz"
        print(result, "can be divided by 15")
    else:
        if num % 3 == 0:
            result = "Fizz"
            print(result, "can be divided by 3")
        if num % 5 == 0:
            result = "Buzz"
            print(result, "can be divided by 5")
    return result
print(FizzBuzz(45))
# FizzBuzz can be divided by 15
# FizzBuzz
```

```
FizzBuzz can be divided by 15
FizzBuzz
```

In [1646]: #③升级版改正: 用elif, 当满足第一个条件, 就不会往下跑了, 看起来更简洁:

```
def FizzBuzz(num):
    result = num
    if num % 15 == 0:
        result = "FizzBuzz"
        print(result, "can be divided by 15")
    elif num % 3 == 0:
        result = "Fizz"
        print(result, "can be divided by 3")
    elif num % 5 == 0:
        result = "Buzz"
        print(result, "can be divided by 5")
    return result
print(FizzBuzz(45))
# FizzBuzz can be divided by 15
# FizzBuzz
```

```
FizzBuzz can be divided by 15
FizzBuzz
```

## 0.7 For Loop

```
for i in range(1, num + 1):
```

```
In [1647]: # Build a function sqrt to calculate the square root of a positive integer num
# if the result is not a integer,
# the function returns the nearest integer to the left of the result number

# sqrt(16) returns 4
# sqrt(17) returns 4
# sqrt(77) returns 8
```

```
In [1648]: #ending index > starting index:
#在起始值为1, 终止值为num的数组中,
#如果i的平方 = num, 输出i; 如果i的平方 > num, 输出i -1
def sqrt(num):
    for i in range(1, num + 1):
        if i * i == num:
            return i
        if i * i > num:
            return i - 1
```

```
In [1649]: #在jupyter中看下for Loop:
for i in range(1, 10):
    print(i)
# 1
# 2
# 3
# 4
# 5
# 6
# 7
# 8
# 9
```

1  
2  
3  
4  
5  
6  
7  
8  
9

```
In [1650]: for i in range(1, 2):
    print(i)
#1
```

1

In [1651]: #得到起始值10, 终止值3, 步长-1的数字:

```
for i in range(10, 2, -1):
    print(i)
# 10
# 9
# 8
# 7
# 6
# 5
# 4
# 3
```

```
10
9
8
7
6
5
4
3
```

In [1652]: for i in range(10, 2, -3):

```
    print(i)
# 10
# 7
# 4
```

```
10
7
4
```

In [1653]: for i in range(2, 10, 2):

```
    print(i)
# 2
# 4
# 6
# 8
```

```
2
4
6
8
```

```
In [1654]: for i in range(7):
    print(i)
# 0
# 1
# 2
# 3
# 4
# 5
# 6
```

```
0
1
2
3
4
5
6
```

## 0.8 While Loop

### 0.8.1 $a += 1 \Leftrightarrow a = a + 1$

```
In [1655]: def sqrt(num):
    i = 0
    while i * i <= num:
        return i
    if i * i > num:
        return i - 1
```

```
In [1656]: a = 1
```

```
In [1657]: a
#1
```

```
Out[1657]: 1
```

```
In [1658]: a += 1
```

```
In [1659]: a
#2
```

```
Out[1659]: 2
```

```
In [1660]: a += 3
```

```
In [1661]: a
#5
```

```
Out[1661]: 5
```

In [1662]: #放进jupyter中:

```
def sqrt(num):
    i = 0
    while i * i <= num:
        i += 1
        print(i, i * i, num)
    return i - 1
```

In [1663]: sqrt(77)

```
# 1 1 77
# 2 4 77
# 3 9 77
# 4 16 77
# 5 25 77
# 6 36 77
# 7 49 77
# 8 64 77
# 9 81 77
# 8
```

```
1 1 77
2 4 77
3 9 77
4 16 77
5 25 77
6 36 77
7 49 77
8 64 77
9 81 77
```

Out[1663]: 8

## 0.8.2 while loop和for loop的区别:

In [1664]: #while Loop和for Loop的区别:

```
#①while Loop的ending exit condition can be set in the loop itself,
#但如果用for Loop, 就需要an ending index to exit the loop, 并且ending index需要比
#因此while Loop 比 for Loop看起来更简洁,
#②但如果在while Loop中忘记设置step size步长,
#就会出现infinite Loop报错, 无休止跑下去。
```

In [1665]: # 如果将ending index设置为num,

```
# 那么当num = 1时, for Loop最后就无法到达exit condition终止条件了
# 这时用while Loop就更简洁, 因为终止条件可以设置为num本身, 而不是num + 1
# def sqrt(num):
#     for in range(1, num + 1):

# def sqrt(num):
#     i = 0
#     while i * i <= num:
```

```
In [1666]: #for Loop做法:
def sqrt(num):
    for i in range(1, num + 1):
        if i * i == num:
            return i
        if i * i > num:
            return i - 1
```

```
In [1667]: #while loop做法:
def sqrt(num):
    i = 0
    while i * i <= num:
        i += 1
    return i - 1
```

## Unit 1: List

### 1.1 List1- Introduction to List

```
In [1668]: #1.1: An empty list
```

```
In [1669]: a = []
```

```
In [1670]: print(a)
#[ ]
```

```
[]
```

```
In [1671]: #1.2 A list of numbers
```

```
In [1672]: a = [1, 2, 3]
```

```
In [1673]: print(a)
#[1, 2, 3]
```

```
[1, 2, 3]
```

```
In [1674]: #1.3 A list of strings
```

```
In [1675]: a = ['a', 'b', 'c']
```

```
In [1676]: print(a)
#[‘a’, ‘b’, ‘c’]
```

```
['a', 'b', 'c']
```

```
In [1677]: #1.4 A list of many things
```

```
In [1678]: a = [1, 'a', 'b', 2, 'c']
```

```
In [1679]: print(a)
#[1, 'a', 'b', 2, 'c']

[1, 'a', 'b', 2, 'c']
```

```
In [1680]: a = [[{}], 1, 2, ["a", "b"], "c"]
```

```
In [1681]: print(a)
#[[], 1, 2, ['a', 'b'], 'c']

[[], 1, 2, ['a', 'b'], 'c']
```

```
In [1682]: #小结:
#1.python List中可以是任何元素，可以是数字, list, dictionary。
#2.如果是list需要双括弧补齐
#3.顺序
```

## 1.2 Getting elements from a list

```
In [1683]: #2.1 看一个元素是否在list中
```

```
In [1684]: a = [1, 3, 4, 5, 9]
```

```
In [1685]: 1 in a
#True
```

```
Out[1685]: True
```

```
In [1686]: 7 in a
#False
```

```
Out[1686]: False
```

```
In [1687]: 2 in a
#False
```

```
Out[1687]: False
```

In [1688]: #2.1.1: 是否元素在list中的函数形式

```
#练习Q extra:  
# Whether an Element in the List  
# Description  
# Build a function IsIn to tell whether a target element in a list.  
  
# Examples  
# IsIn([1, 2, 3, 4, 5], 2) returns True  
# IsIn([1, 2, 3, 4, 5], 6) returns False
```

In [1689]: #法1: if k in a:

```
def IsIn(nums, target):  
    if target in nums:  
        return True  
    else:  
        return False
```

#一个值在一个列表中: if a in list, return True False

In [1690]: #法2: if target in nums:

```
def IsIn(nums, target):  
    return target in nums
```

#直接return k in a, 返回结果是True False

In [1691]: #return target in nums 是一个布尔表达式, 它会返回一个布尔值, 即 True 或 False. #

In [1692]: IsIn([1, 2, 3, 4, 5], 2)  
#True

Out[1692]: True

In [1693]: IsIn([1, 2, 3, 4, 5], 6)  
#False

Out[1693]: False

In [1694]: #2.2 用Index取需要的元素

In [1695]: a = [1, 2, 3, 4, 5, 6, 7]

In [1696]: print(a[1])  
#2

2

In [1697]: print(a[0])  
#1

1

```
In [1698]: print(a[6])
#7
```

7

```
In [1699]: #print(a[7])
#
# -----
# IndexError
# Cell In[70], line 1
# ----> 1 print(a[7])

# IndexError: List index out of range
```

Traceback (most recent call last)

```
In [1700]: #2.3 用负值index取元素
```

```
In [1701]: a = [1, 2, 3, 4, 5, 6, 7]
```

```
In [1702]: print(a[-1])
#7
```

7

```
In [1703]: print(a[-2])
#6
```

6

```
In [1704]: print(a[-7])
#1
```

1

```
In [1705]: #print(a[-8])
#
# -----
# IndexError
# Cell In[87], line 1
# ----> 1 print(a[-8])

# IndexError: List index out of range
```

Traceback (most recent call last)

```
In [1706]: #2.4 Length of a list
```

```
In [1707]: a = [1, 2, 3, 4, 5, 6, 7]
```

```
In [1708]: print(len(a))
#7
```

7

In [1709]: `a[len(a) - 1]`  
 #在a中取第六个元素, 即7  
 #7

Out[1709]: 7

In [1710]: `#a(len(a))`  
 ##在a中取第七个元素, out of list  
 #  
 #-----  
 # TypeError  
 # Cell In[103], line 1  
 # ----> 1 a(len(a))  
 #  
 # TypeError: 'list' object is not callable

Traceback (most recent call last)

In [1711]: #2.5 在list中进行slicing  
 #包括括号前位置的元素, 不包括括号后位置的元素  
 #step

In [1712]: `a = [1, 2, 3, 4, 5, 6, 7]`

In [1713]: `print(a[2:3])`  
 #[3]

[3]

In [1714]: `print(a[2:2])`  
 #[]

[]

In [1715]: `print(a[2:6])`  
 #[3, 4, 5, 6]

[3, 4, 5, 6]

In [1716]: `print(a[2:6:2])`  
 #[3, 5]

[3, 5]

In [1717]: `print(a[2:6:1])`  
 #[3, 4, 5, 6]

[3, 4, 5, 6]

In [1718]: #2.6 从负值取元素

In [1719]: `a = [1, 2, 3, 4, 5, 6, 7]`

In [1720]: `print(a[6:2]) #???`

[]

In [1721]: `#-1从后往前取  
print(a[6:2:-1])  
#[7, 6, 5, 4]`

[7, 6, 5, 4]

In [1722]: `print(a[:])  
#[1, 2, 3, 4, 5, 6, 7]`

[1, 2, 3, 4, 5, 6, 7]

In [1723]: `print(a[::-1])  
#[1, 2, 3, 4, 5, 6, 7]`

[1, 2, 3, 4, 5, 6, 7]

In [1724]: `#::-1 从后往前取  
print(a[::-1])  
#[7, 6, 5, 4, 3, 2, 1]`

[7, 6, 5, 4, 3, 2, 1]

In [1725]: `#:-1 从开始取到倒数第二个数  
print(a[:-1])  
#[1, 2, 3, 4, 5, 6]`

[1, 2, 3, 4, 5, 6]

## 1.3 Constructing a list

In [1726]: `#3.1 For Loop`

In [1727]: `#把所有值append到for Loop上  
a = []  
for i in range(10):  
 a.append(i)`

In [1728]: `print(a)  
#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`

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

In [1729]: #*i \* 2: 把所有值都乘以2*

```
a = []
for i in range(10):
    a.append(i * 2)
print(a)
#[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

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

In [1730]: #练习1: 返回一个数列

*#build a function LstConstr which takes two integers (i and j) as inputs  
#and returns a list of numbers from i to j*

```
def LstConstr(i, j):
    a = []
    for k in range(i, j+1):
        a.append(k)
    return a
```

In [1731]: LstConstr(1,5)

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

Out[1731]: [1, 2, 3, 4, 5]

In [1732]: #练习2: 返回奇数

*#build a function OddNums which takes two integers (i and j) as inputs  
#and returns a list of ODD numbers between i and j inclusively*

In [1733]: #def后; for后; if后 都有:

```
def OddNums(i, j):
    a = []
    for k in range(i, j+1):
        if k % 2 != 0:           #k是奇数
            a.append(k)
    return a
```

#注意:

#①: 先建立一个空集a, 如果是奇数, append到a中, 最后return a

#②: if k % 2 != 0 后面要加:

#③: append的写法: List.append(num)

In [1734]: OddNums(2, 6)

```
#[3, 5]
```

Out[1734]: [3, 5]

```
In [1735]: #练习3: Fibonacci List: 斐波纳契数列: 后一个数是前两个数之和
#build a function FibLst which takes an integer n as input
#and returns an n-element Fibonacci List.
#If n is less than 1, returns -1
#[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

# def Fibonacci(0, n+1):
#     a = []
#     for k in n:
#         k = k[n-1] + k[n]
#     return k
# Fibonacci(5)
```

```
In [1736]: def Fibonacci(n):
    if n < 1:
        return -1
    a = []
    for i in range(n):
        if i == 0:
            a.append(0)
        elif i == 1:
            a.append(1)
        else:
            a.append(a[i - 1] + a[i - 2])
    return a
```

#①: 建立空List a, 对于所有range(n)内的值i, 如果是0, 则赋值0, 如果是1, 则赋值1, 否则  
#②: if n < 1的edge case, 和for Loop是对齐的。  
#③: for Loop下的if, elif, else分类讨论, else后不需要再写n>1等解释; elif不能写成if。  
#④: 判断用==, 不能写成=。  
#⑤将每个数房间List用list.append(). 如果加0/1直接赋值, 就在括弧里写0/1。如果加前两数  
#⑥return与for Loop进行对齐

```
Fibonacci(1)
#i in range(1) i = 0      #[0]
Fibonacci(2)
#i in range(2) i = 0,1    #[0, 1]
Fibonacci(3)
#i in range(3) i = 0,1,2  a[1]+a[0]=1 #[0, 1, 1]
Fibonacci(4)
#i in range(4) i = 0,1,2,3 a[1]+a[0]=1 a[2]+a[1]=2 #[0, 1, 1, 2]
```

Out[1736]: [0, 1, 1, 2]

In [1737]: #3.2 List Comprehension

```
In [1738]: a = []
for i in range(10):
    a.append(i**2)
print(a)
#i: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
#i**2: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

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

In [1739]: `a = [i for i in range(10)]`

In [1740]: `print(a)`  
`#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`  
`[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`

In [1741]: `a = [i * 2 for i in range(10)]`

In [1742]: `print(a)`  
`#[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]`  
`[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]`

In [1743]: `#练习4: i j之间数值`  
`#build a function LstConstr which takes two integers(i and j) as inputs`  
`#and returns a list of numbers from i to j.`

In [1744]: `def LstConstr2(i, j):`  
 `return [k for k in range(i, j+1)]`  
`#①注意要有k作为中间变量`  
`#要有中括弧囊括return内容`

In [1745]: `LstConstr2(1,3)`  
`#[1, 2, 3]`

Out[1745]: `[1, 2, 3]`

In [1746]: `def LstConstr3(i, j):`  
 `return list(range(i, j + 1))`

In [1747]: `LstConstr3(1,3)`  
`#[1, 2, 3]`

Out[1747]: `[1, 2, 3]`

In [1748]: `#只返奇数`  
`def LstConstr4(i, j):`  
 `return [k for k in range(i, j + 1) if k % 2 != 0]`

In [1749]: `LstConstr4(1,3)`  
`#[1, 3]`

Out[1749]: `[1, 3]`

In [1750]: `#练习5: C2F:`  
`#build a function C2F which converts a list of temperatures`  
`#in degrees from Celsius to Fahrenheit`

In [1751]: #①用 `a + b` for C in Cs 写法:

```
def C2F(Cs):
    return [C * 1.8 + 32 for C in Cs]
```

#为什么必须加[]:

#在给定的代码中，方括号[]用于创建列表推导式 (List comprehension)。

#列表推导式是一种简洁的方式，通过迭代现有可迭代对象（在这种情况下是列表Cs）并对每个元素应用一个表达式，它通过迭代输入列表Cs中的每个元素C\*1.8+32，并将结果添加到新列表中。

In [1752]: #②用for Loop写法:

```
def C2F(Cs):
    a = []
    for C in Cs:
        F = C * 1.8 + 32
        a.append(F)
    return a
```

In [1753]: def C2F(Cs):

```
    result = []
    for C in Cs:
        F = C * 1.8 + 32
        result.append(F)
    return result
```

In [1754]: C2F([0, 38])

#[32.0, 100.4]

Out[1754]: [32.0, 100.4]

In [1755]: #3.3 Concatenate 不同lists

In [1756]: a = [i for i in range(10)]

```
print(a)
#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

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

In [1757]: b = [i \* 2 for i in range(10)]

```
print(b)
#[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

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

In [1758]: c = a + b

```
print(c)
#[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

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

```
In [1759]: [x + y for x, y in zip(a, b)]
#[0, 3, 6, 9, 12, 15, 18, 21, 24, 27]
```

```
Out[1759]: [0, 3, 6, 9, 12, 15, 18, 21, 24, 27]
```

```
In [1760]: #练习6: sperate numbers: 分开奇偶数
#build a function SepNum which extracts the odd numbers from a given list
#and append the odd numbers at the end of the list
```

```
In [1761]: def SepNum(nums):
    odds = []
    evens = []
    for num in nums:
        if num % 2 == 0:          #放偶数
            evens.append(num)
        else:                     #放奇数
            odds.append(num)
    return evens + odds
```

#①: 分别建立偶数空集和奇数空集, 对于在nums中的num, 如果除以2余数为0, append到偶数集。  
#②: for数字在列表中, 可以写for num in nums。  
#③: 两个list相连直接相加即可  
#④: 在if 和else后分别进行append操作, return两个list相连与 for Loop对齐。

```
In [1762]: SepNum([1, 2, 3, 4, 5, 6])
#[2, 4, 6, 1, 3, 5]
```

```
Out[1762]: [2, 4, 6, 1, 3, 5]
```

## 1.4 Sorting a list

```
In [1763]: #4.1
#.sort: 正向从小到大
#.sort(reverse = True): 反向从大到小
```

```
In [1764]: a = [1, 3, 2, 7, 5]
```

```
In [1765]: a.sort()
```

```
In [1766]: print(a)
#[1, 2, 3, 5, 7]

[1, 2, 3, 5, 7]
```

```
In [1767]: a.sort(reverse = True)
```

In [1768]:

```
print(a)
#[7, 5, 3, 2, 1]
```

[7, 5, 3, 2, 1]

In [1769]:

```
#4.2
# sorted(list)效果等同于list.sort, 区别在于list.sort后sort改变, 而sorted(list)后s
# b = sorted(a), b改变, a没有改变
```

In [1770]:

```
a = [1, 3, 2, 7, 5]
```

In [1771]:

```
sorted(a)
#[1, 2, 3, 5, 7]
```

Out[1771]:

[1, 2, 3, 5, 7]

In [1772]:

```
print(a)
#[7, 5, 3, 2, 1]
```

[1, 3, 2, 7, 5]

In [1773]:

```
b = sorted(a)
```

In [1774]:

```
print(b)
#[1, 2, 3, 5, 7]
```

[1, 2, 3, 5, 7]

In [1775]:

```
#练习7:
#build a function SortLst which sorts a list. use built-in function
def SortLst(nums):
    return sorted(nums)
SortLst([4, 2, 1, 3, 4, 5])
#[1, 2, 3, 4, 4, 5]
```

Out[1775]:

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

In [1776]:

```
#practice:
def SortLst(a):
    a = sorted(a) #为什么写a = a.sort()就不可以?
    return a
SortLst([1, 7, 3])
#[1, 3, 7]
```

Out[1776]:

[1, 3, 7]

In [1777]:

```
#练习8: 在sorted List加入一个值: append值后sorted(list)
def NumInsert(nums, target):
    nums.append(target)
    return sorted(nums)
```

In [1778]: `NumInsert([1, 2, 3, 4, 5, 6], 3)`  
`#[1, 2, 3, 3, 4, 5, 6]`

Out[1778]: `[1, 2, 3, 3, 4, 5, 6]`

In [1779]: `#练习8: merge two sorted lists`  
`#build a function mergesorted which merges two sorted lists as one sorted list`  
`def MergeSorted(Lst1, Lst2):`  
 `return sorted(Lst1 + Lst2)`

In [1780]: `MergeSorted([1, 3, 5], [2, 4, 6])`  
`#[1, 2, 3, 4, 5, 6]`

Out[1780]: `[1, 2, 3, 4, 5, 6]`

In [1781]: `#4.3 sort a list of lists`

In [1782]: `a = [[4,2], [5,1], [2,4], [3,0]]`

In [1783]: `#sorted(list) 按照第一个值sort 从小到大`  
`sorted(a)`  
`#[[2, 4], [3, 0], [4, 2], [5, 1]]`

Out[1783]: `[[2, 4], [3, 0], [4, 2], [5, 1]]`

In [1784]: `#从大到小`  
`sorted(a, reverse = True)`  
`#[[5, 1], [4, 2], [3, 0], [2, 4]]`

Out[1784]: `[[5, 1], [4, 2], [3, 0], [2, 4]]`

In [1785]: `#lambda define function`  
`#在a里面按照index = 0的元素, sort的是这个list中的第一个元素`  
`sorted(a, key = lambda x:x[0])`  
`#[[2, 4], [3, 0], [4, 2], [5, 1]]`

Out[1785]: `[[2, 4], [3, 0], [4, 2], [5, 1]]`

In [1786]: `#按第二位元素, 从小到大sort`  
`sorted(a, key = lambda x:x[1])`  
`#[[3, 0], [5, 1], [4, 2], [2, 4]]`

Out[1786]: `[[3, 0], [5, 1], [4, 2], [2, 4]]`

In [1787]: `#按第二位元素, 从大到小sort`  
`sorted(a, key = lambda x:x[1], reverse = True)`  
`#[[5, 1], [4, 2], [3, 0], [2, 4]]`

Out[1787]: `[[5, 1], [4, 2], [3, 0], [2, 4]]`

```
In [1788]: #练习9: sort list of lists 将一个包含list的list排序
#build a function SortLL which sorts a list of lists by their last elements
```

```
In [1789]: def SortLL(Lsts):
    return sorted(Lsts, key = lambda x:x[-1])
```

```
In [1790]: SortLL([[1, 3, 5], [2, 4], [9, 7], [3]])
#[[3], [2, 4], [1, 3, 5], [9, 7]]
#按最后一个元素, 3,4,5,7的顺序从小到大排序
```

```
Out[1790]: [[3], [2, 4], [1, 3, 5], [9, 7]]
```

## 1.5 Reversing a list

```
In [1791]: #完成reverse:
#①list.reverse()
#②list[::-1]
```

```
In [1792]: a = [1, 3, 2, 7, 5]
```

```
In [1793]: a.reverse()
```

```
In [1794]: print(a)
#[5, 7, 2, 3, 1]
```

[5, 7, 2, 3, 1]

```
In [1795]: a = a[::-1]
```

```
In [1796]: print(a)
#[5, 7, 2, 3, 1]
```

[1, 3, 2, 7, 5]

```
In [1797]: #练习10: 反转list
#build a function RevLst which reverse a list
```

```
In [1798]: #方法1: list.reverse()
def RevLst(nums):
    nums.reverse()
    return nums

#注意: 先写一行list.reverse(), 再写return list, 两者对齐
#为什么要写两行?
```

```
In [1799]: RevLst([3, 1, 9, 8, 4])
#[4, 8, 9, 1, 3]
```

```
Out[1799]: [4, 8, 9, 1, 3]
```

```
In [1800]: #法2: list[::-1]
def RevLst(nums):
    return nums[::-1]

#注意: 只用写一行return list[::-1]
```

```
In [1801]: RevLst([3, 1, 9, 8, 4])
#[4, 8, 9, 1, 3]
```

```
Out[1801]: [4, 8, 9, 1, 3]
```

```
In [1802]: #练习11: 移除目标元素并返回更新后的列表。

# Remove an Element
# Description
# Build a function RmEL to remove a target element from a list. If the target

# Examples
# RmEL([1, 2, 3, 4, 5], 2) returns [1, 3, 4, 5]
# RmEL([1, 2, 3, 4, 5], 6) returns [1, 2, 3, 4, 5]
# RmEL([1, 2, 3, 3, 4, 5], 3) returns [1, 2, 3, 4, 5]
```

```
In [1803]: def RmEL(nums, target):
    if target in nums:
        nums.remove(target)
    return nums

#①: 要检查target在nums中才可以进行remove操作
#②: if k in a: 直接操作a.remove(k), 再return a。不用写nums = XXX。
#③: return 应与if对齐
```

```
In [1804]: #练习12: 移除重复元素并返回更新后列表。

# Remove Duplicates from a List
# Description
# Build a function RmDuplicate which remove all the duplicates from a list.

# Examples
# RmDuplicate([3, 1, 2, 3, 4, 5, 6]) returns [1, 2, 3, 4, 5, 6]
# RmDuplicate([1, 3, 5]) returns [1, 3, 5]
```

In [1805]: #法1：如果num没在a空集中，就把num append进a中，并进行sort

```
def RmDuplicate(nums):
    a = []
    for num in nums:
        if num not in a:
            a.append(num)
    a.sort()
    return a
```

#①：只需要建立一个最后需要return的a集，

#②：for num in nums, if num not in a, a.append(num) 可以写成步步缩进格式：

#即在num在nums中的时候，如果num没在a里，就把其放进去。

#③：在append后，要进行sort。

#④：return 和 for进行对齐。

## Unit 2

### 2.1 Lambda

#### 2.1.1 Review

In [1806]: a = [[4, 3, 2], [5, 2, 1], [2, 5, 4], [3, 1, 0]]

In [1807]: #按照每个小List的第一个元素，从小到大排序

```
sorted(a)
#[[2, 5, 4], [3, 1, 0], [4, 3, 2], [5, 2, 1]]
```

Out[1807]: [[2, 5, 4], [3, 1, 0], [4, 3, 2], [5, 2, 1]]

In [1808]: #按照每个小List的最后一个元素，从小到大排序

```
sorted(a, key = lambda x:x[-1])
#[[3, 1, 0], [5, 2, 1], [4, 3, 2], [2, 5, 4]]
```

#key是根据什么来排序，默认是按照第一个元素

#x:x[-1]表示根据最后一个元素

#lambda是定义匿名函数的方式，和def用法相同，写法更简洁

#函数 = Lambda 变量: 方程

Out[1808]: [[3, 1, 0], [5, 2, 1], [4, 3, 2], [2, 5, 4]]

#### 2.1.2 Intro to Lambda

##### 2.1.2.1 Build a function

```
In [1809]: def Square(num):
    return num * num
```

```
In [1810]: Square(2)
#4
```

Out[1810]: 4

```
In [1811]: Square(3)
#9
```

Out[1811]: 9

### 2.1.2.2 Built a function with lambda

```
In [1812]: new_square = lambda num: num * num
```

```
In [1813]: new_square(2)
#4
```

Out[1813]: 4

### 2.1.3 Replace lambda with traditional function building

#### 2.1.3.1 How does lambda work in sorted

```
In [1814]: a = [[4, 3, 2], [5, 2, 1], [2, 5, 4], [3, 1, 0]]
```

```
In [1815]: #按照每个list最后一个元素排列
sorted(a, key = lambda x:x[-1])
#[[3, 1, 0], [5, 2, 1], [4, 3, 2], [2, 5, 4]]

#Lambda定义一个函数，输入x是小list，返回小list的最后一个值
```

Out[1815]: [[3, 1, 0], [5, 2, 1], [4, 3, 2], [2, 5, 4]]

```
In [1816]: #取list中最后一个元素
fun = lambda x: x[-1]
```

```
In [1817]: fun([4, 3, 2])
#2
```

Out[1817]: 2

#### 2.1.3.2 Replace lambda with a traditional function #不用lambda用def

In [1818]: #①按照每个小List中，最后一个元素，从小到大排序

```
def get_last_element(nums):
    return nums[-1]
```

In [1819]: get\_last\_element([4,3,2])

```
#2
```

Out[1819]: 2

In [1820]: get\_last\_element('abc')

```
 #'c'
```

Out[1820]: 'c'

In [1821]: sorted(a, key = get\_last\_element)

```
#[[3, 1, 0], [5, 2, 1], [4, 3, 2], [2, 5, 4]]
```

```
#效果等同于sorted(a, key = Lambda x:x[-1])
```

Out[1821]: [[3, 1, 0], [5, 2, 1], [4, 3, 2], [2, 5, 4]]

In [1822]: #②按照每个小List中，第二个元素，从小到大排序

```
def get_second_element(nums):
    return nums[1]
```

In [1823]: get\_second\_element([4,3,2])

```
#3
```

Out[1823]: 3

In [1824]: sorted(a, key = get\_second\_element)

```
#[[3, 1, 0], [5, 2, 1], [4, 3, 2], [2, 5, 4]]
```

Out[1824]: [[3, 1, 0], [5, 2, 1], [4, 3, 2], [2, 5, 4]]

In [1825]: #总结: def 和 Lambda替换写法:

```
# ①def:
# def function(nums):
#     return nums[-1]
# sorted(a, key = function)

# ②Lambda:
# sorted(a, key = Lambda x:x[-1])
```

## 2.2 Append & Insert

### 2.2.1 modify a list elementwisely

```
In [1826]: a = [1, 2, 3, 4, 5, 7, 6]
```

```
In [1827]: print(a)
#[1, 2, 3, 4, 5, 6, 7]
```

```
[1, 2, 3, 4, 5, 7, 6]
```

### 2.2.2 append an element at the end

```
In [1828]: #append将元素加在末尾
#list.append(num)对list进行操作后, 不返回任何东西; 就像.sort()不返回任何东西一样
```

```
In [1829]: a.append("a")
```

```
In [1830]: print(a)
#[1, 2, 3, 4, 5, 6, 7, 'a']
```

```
[1, 2, 3, 4, 5, 6, 7, 'a']
```

```
In [1831]: a.append("b")
```

```
In [1832]: print(a)
#[1, 2, 3, 4, 5, 6, 7, 'a', 'b']
```

```
[1, 2, 3, 4, 5, 6, 'a', 'b']
```

### 2.2.3 insert an element

```
In [1833]: #insert在i位置上插入元素k, 其余元素顺延; 即输入原list该位置元素的前面!
#list.insert(k)
```

```
In [1834]: a.insert(0, "c")
```

```
In [1835]: print(a)
#[['c', 1, 2, 3, 4, 5, 6, 7, 'a', 'b']]
```

```
[['c', 1, 2, 3, 4, 5, 6, 7, 'a', 'b']]
```

```
In [1836]: a.insert(0, "d")
```

```
In [1837]: print(a)
#[‘d’, ‘c’, 1, 2, 3, 4, 5, 6, 7, ‘a’, ‘b’]
['d', 'c', 1, 2, 3, 4, 5, 6, 7, 'a', 'b']
```

```
In [1838]: #在原list元素3的位置，插入e，其余顺延
a.insert(3, "e")
```

```
In [1839]: print(a)
#[‘d’, ‘c’, 1, ‘e’, 2, 3, 4, 5, 6, 7, ‘a’, ‘b’]
['d', 'c', 1, 'e', 2, 3, 4, 5, 6, 7, 'a', 'b']
```

```
In [1840]: #在原list最后一个元素的位置上，插入f，其余顺延
a.insert(-1, "f")
```

```
In [1841]: print(a)
#[‘d’, ‘c’, 1, ‘e’, 2, 3, 4, 5, 6, 7, ‘a’, ‘f’, ‘b’]
['d', 'c', 1, 'e', 2, 3, 4, 5, 6, 7, 'a', 'f', 'b']
```

## 2.3 Example using Insert

```
In [1842]: # 1. Insert a Number in a Sorted List

# Code
# Exercise 6.2.1: Insert a Number in a Sorted List
# Build a function NumInsert which inserts a number in a sorted list.
# Example: NumInsert([1, 2, 3, 4, 5, 6], 3) returns [1, 2, 3, 3, 4, 5, 6]
# Example: NumInsert([1, 2, 3, 4, 5, 6], 7) returns [1, 2, 3, 4, 5, 6, 7]
# Example: NumInsert([1, 2, 3, 4, 5, 6], -1) returns [-1, 1, 2, 3, 4, 5, 6]
```

```
In [1843]: #法1: append后, sort下
```

```
In [1844]: #法2: insert
#让ind = list长度, 从头至尾遍历所有元素, 当list中的元素大于插入值时, 让ind = i
#list.insert(index, target), 让index位置插入target元素

#解题思路1：从左到右逐个比较，找到合适位置
#步骤：
#1. 从左到右遍历
#2. 当元素数值大于或者等于需要插入的数字，将该数字插入到这个元素左边
#局限：当元素数值较大，需要花很多时间才能找到位置
```

```
In [1845]: def NumInsert(nums, target):
    ind = len(nums)
    for i in range(len(nums)):
        if nums[i] >= target:
            ind = i
            break
    nums.insert(ind, target)
    return nums
```

```
In [1846]: NumInsert([1, 2, 3, 4, 5, 6], 3)
#[1, 2, 3, 3, 4, 5, 6]
NumInsert([1, 2, 3, 4, 5, 6], 7)
#[1, 2, 3, 4, 5, 6, 7]
```

Out[1846]: [1, 2, 3, 4, 5, 6, 7]

## 2.4 Example using Insert with Binary Search

```
In [1847]: #二分法比遍历法快:

#解题思路2：二分法

#步骤：
# 1. 设定左右两个指针，分解为L和R
# 2. L初始值为0， R初始值为list长度
# 3. 计算中间指针的值 M = (left + right) / 2
#     3.1. 若M指针对应的数，大于需要插入的数，将M设为R
#     3.2. 若M指针对应的数，小于需要插入的数，将M设为L
# 4. 重复步骤3，直到
#     4.1 M指针对应的值等于需要插入的值
#     4.2 或者，左右指针相遇
# 5. 将目标数字插入相应的位置

#每次取一半，把另一半扔掉，比从头到尾遍历要快
```

```
In [1848]: def NumInsert(nums, target):
    left = 0
    right = len(nums)
    while right - left > 1:
        mid = (left + right) // 2 #整数除法 //: 向下取整
        if nums[mid] == target:
            nums.insert(mid, target)
            return nums
        elif nums[mid] > target:
            right = mid
        else:
            left = mid
    if target > nums[mid]:
        nums.insert(right, target)
    else:
        nums.insert(left, target)
    return nums

#https://pythontutor.com/render.html#code=def%20NumInsert%28nums,%20target%29%
```

```
In [1849]: NumInsert([1, 2, 3, 4, 5, 6], 3)
#[1, 2, 3, 3, 4, 5, 6]
```

```
Out[1849]: [1, 2, 3, 3, 4, 5, 6]
```

```
In [1850]: len([1, 2, 3, 4, 5, 6])
#6
```

```
Out[1850]: 6
```

## 2.5 Remove

### 2.5.1 remove an element

.remove(); .insert(); .sort()不返回任何东西 #.remove每次只remove一个元素，如果原list中有多个元素，只remove最左边第一个元素

```
In [1851]: a = ['d', 'c', 1, 'e', 2, 3, 4, 5, 6, 7, 'a', 'f', 'b']
```

```
In [1852]: print(a)
#[‘d’, ‘c’, 1, ‘e’, 2, 3, 4, 5, 6, 7, ‘a’, ‘f’, ‘b’]

['d', 'c', 1, 'e', 2, 3, 4, 5, 6, 7, 'a', 'f', 'b']
```

```
In [1853]: a.remove('e') #等同于a.remove("e")
```

```
In [1854]: print(a)
#[‘d’, ‘c’, 1, 2, 3, 4, 5, 6, 7, ‘a’, ‘f’, ‘b’]
['d', 'c', 1, 2, 3, 4, 5, 6, 7, 'a', 'f', 'b']
```

```
In [1855]: #a.remove('e')

# -----
# -----[REDACTED]-----#
#           Traceback (most recent call last)
# Cell In[498], line 1
# ----> 1 a.remove('e')

# ValueError: list.remove(x): x not in list
#因为list中已经没有e了，所以报错
```

## 2.5.2

避免移除后原list不存在，再run remove时报错方法 if else：当元素在原list里面时就remove，否则就说没有

```
In [1856]: if "e" in a:
            a.remove("e")
else:
    print("It is not there")

#It is not there
```

It is not there

```
In [1857]: print(a)
#[‘d’, ‘c’, 1, 2, 3, 4, 5, 6, 7, ‘a’, ‘f’, ‘b’]
['d', 'c', 1, 2, 3, 4, 5, 6, 7, 'a', 'f', 'b']
```

```
In [1858]: a.append("f")
```

```
In [1859]: print(a)
#[‘d’, ‘c’, 1, 2, 3, 4, 5, 6, 7, ‘a’, ‘f’, ‘b’, ‘f’, ‘f’]
['d', 'c', 1, 2, 3, 4, 5, 6, 7, 'a', 'f', 'b', 'f', 'f']
```

## 2.5.3

.remove每次只remove一个元素，如果原list中有多个元素，只remove最左边第一个元素

```
In [1860]: a.remove("f")
```

```
In [1861]: print(a)
#[‘d’, ‘c’, 1, 2, 3, 4, 5, 6, 7, ‘a’, ‘b’, ‘f’]
['d', 'c', 1, 2, 3, 4, 5, 6, 7, 'a', 'b', 'f']
```

```
In [1862]: a.remove("f")
```

```
In [1863]: print(a)
#[‘d’, ‘c’, 1, 2, 3, 4, 5, 6, 7, ‘a’, ‘b’]
['d', 'c', 1, 2, 3, 4, 5, 6, 7, 'a', 'b']
```

```
In [1864]: c = ["a", "a"]
```

## 2.5.4

可以一次性remove多个同样的元素，用while loop方法： #当元素在list中，就remove，直到没有为止。

```
In [1865]: while "a" in c:
    c.remove("a")
print(c)

#[]
```

```
[]
```

```
In [1866]: b = a.remove("b")
```

```
In [1867]: #.remove()操作不会返回任何值
print(b)
#None
```

```
None
```

```
In [1868]: print(a)
#[‘d’, ‘c’, 1, 2, 3, 4, 5, 6, 7, ‘a’]
['d', 'c', 1, 2, 3, 4, 5, 6, 7, 'a']
```

## 2.5.5 习题:

In [1869]: `#Exercise 6.3.1: Remove an Element: 移除重复的元素 (如有重复, 只保留一个)  
# Build a function RmEL to remove a target element from a list. If the target  
# Example: RmEL([1, 2, 3, 4, 5], 2) returns [1, 3, 4, 5]  
# Example: RmEL([1, 2, 3, 4, 5], 6) returns [1, 2, 3, 4, 5]  
# Example: RmEL([1, 2, 3, 3, 4, 5], 3) returns [1, 2, 3, 4, 5]`

In [1870]: `def RmEL(nums, target):  
 if target in nums:  
 nums.remove(target)  
 return nums`

In [1871]: `RmEL([1, 2, 3, 3, 4, 5], 3)  
#[1, 2, 3, 4, 5]`

Out[1871]: `[1, 2, 3, 4, 5]`

In [1872]: `RmEL([1, 2, 3, 3, 4, 5], 6)  
#[1, 2, 3, 3, 4, 5]`

Out[1872]: `[1, 2, 3, 3, 4, 5]`

## 2.6 Pop

### 2.6.1 pop out ana element

- i. 把最末尾的值从list中取出后返回 newList
- ii.pop和remove不同，是返回值的
- iii.pop()取出拿掉的是最后一个值
- iv.pop(i)取出拿掉的是和[i]位置上的元素

In [1873]: `a = [1, 2, 3, 4, 5, 6, 7]`

In [1874]: `print(a)  
#[1, 2, 3, 4, 5, 6, 7]`  
[1, 2, 3, 4, 5, 6, 7]

In [1875]: `b = a.pop()`

In [1876]: `print(b)`  
#[7]

7

In [1877]: `print(a)`  
#[1, 2, 3, 4, 5, 6]

[1, 2, 3, 4, 5, 6]

In [1878]: `a.pop()`  
#[6]

Out[1878]: 6

In [1879]: `print(a)`  
#[1, 2, 3, 4, 5]

[1, 2, 3, 4, 5]

In [1880]: `a.pop(1)`  
#pop在[1]这个位置上的元素: 2

Out[1880]: 2

In [1881]: `print(a)`  
#[1, 3, 4, 5]

[1, 3, 4, 5]

In [1882]: `a.pop(0)`  
#pop在[0]这个位置上的元素: 1

Out[1882]: 1

In [1883]: `print(a)`  
#[3, 4, 5]

[3, 4, 5]

In [1884]: # Exercise 6.4.1: Remove duplicates from a sorted list: 在排序号的List中移除重复  
# Build a function SortRm which removes duplicate numbers from a sorted list.  
# Example: SortRm([1, 2, 2, 3, 4, 4, 5, 6]) returns [1, 2, 3, 4, 5, 6]  
# Example: SortRm([1, 2, 3, 4, 5, 6]) returns [1, 2, 3, 4, 5, 6]

```
In [1885]: def SortRm(nums):
    if len(nums) <= 1:          #如果List只有一个元素
        return nums              #return这个元素
    tmp = nums[0]                #将List中第一位置的值赋值给tmp
    i = 1                        #指针i取值为1, 即第二位置
    while i < len(nums):        #当指针i < list长度时: 还在range中时
        print("nums = ", nums)
        print("tmp = ", tmp)
        print("i = ", i)
        print("nums[i] = ", nums[i])
        print("====")
        if nums[i] == tmp:       #指针指到的值和tmp比较相同
            nums.pop(i)          #则pop
        else:                    #如果不同
            tmp = nums[i]         #将指针指到的值赋值给tmp, 加入tmp中
            i += 1                 #i+1, 指针后移
    return nums

#https://pythontutor.com/render.html#code=def%20SortRm%28nums%29%3A%0A%20%20%20%
```

In [1886]: `SortRm([1, 2, 2, 3, 4, 4, 5, 6])  
#[1, 2, 3, 4, 5, 6]`

```
nums = [1, 2, 2, 3, 4, 4, 5, 6]
tmp = 1
i = 1
nums[i] = 2
=====
nums = [1, 2, 2, 3, 4, 4, 5, 6]
tmp = 2
i = 2
nums[i] = 2
=====
nums = [1, 2, 3, 4, 4, 5, 6]
tmp = 2
i = 2
nums[i] = 3
=====
nums = [1, 2, 3, 4, 4, 5, 6]
tmp = 3
i = 3
nums[i] = 4
=====
nums = [1, 2, 3, 4, 4, 5, 6]
tmp = 4
i = 4
nums[i] = 4
=====
nums = [1, 2, 3, 4, 5, 6]
tmp = 4
i = 4
nums[i] = 5
=====
nums = [1, 2, 3, 4, 5, 6]
tmp = 5
i = 5
nums[i] = 6
=====
```

Out[1886]: [1, 2, 3, 4, 5, 6]

## 2.7 Example using Pop

### 2.7.1 Merge Two Sorted Lists

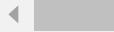
Exercise 6.4.2: Merge Two Sorted Lists Build a function MergeSorted which merges two sorted lists as one sorted list. Example: `MergeSorted([1, 3, 5], [2, 4, 6])` returns [1, 2, 3, 4, 5, 6]  
Example: `MergeSorted([1, 2, 3], [4, 5, 6])` returns [1, 2, 3, 4, 5, 6]

```
In [1887]: def MergeSorted(Lst1, Lst2):
    i = 0
    while Lst1 and i < len(Lst2):
        print("==")
        print("Lst1=", Lst1)
        print("Lst2=", Lst2)
        print("Lst1[0]=", Lst1[0])
        print("i=", i)
        print("Lst2[i]=", Lst2[i])
        if Lst1[0] <= Lst2[i]:
            num = Lst1.pop(0)
            Lst2.insert(i, num)
        i += 1
    if Lst1:
        return Lst2 + Lst1

    return Lst2
```

#什么是while Lst1 和if Lst1?

#<https://pythontutor.com/render.html#code=def%20MergeSorted%28Lst1,%20Lst2%29%>



```
In [1888]: MergeSorted([1, 3, 5], [2, 4, 6])
##[1, 2, 3, 4, 5, 6]

 ==
Lst1= [1, 3, 5]
Lst2= [2, 4, 6]
Lst1[0]= 1
i= 0
Lst2[i]= 2
 ==
Lst1= [3, 5]
Lst2= [1, 2, 4, 6]
Lst1[0]= 3
i= 1
Lst2[i]= 2
 ==
Lst1= [3, 5]
Lst2= [1, 2, 4, 6]
Lst1[0]= 3
i= 2
Lst2[i]= 4
 ==
Lst1= [5]
Lst2= [1, 2, 3, 4, 6]
Lst1[0]= 5
i= 3
Lst2[i]= 4
 ==
Lst1= [5]
Lst2= [1, 2, 3, 4, 6]
Lst1[0]= 5
i= 4
Lst2[i]= 6
```

Out[1888]: [1, 2, 3, 4, 5, 6]

## 2.8 Modify & Switch

### 2.8.1 Modify an element 修改某个位置上的元素

```
In [1889]: a = [1, 2, 3, 4, 5, 6, 7]
```

```
In [1890]: a[0] = "a"
```

```
In [1891]: print(a)
#['a', 2, 3, 4, 5, 6, 7]
```

['a', 2, 3, 4, 5, 6, 7]

In [1892]: `a[1] = "b"`

In [1893]: `print(a)  
#[ 'a', 'b', 3, 4, 5, 6, 7]`  
['a', 'b', 3, 4, 5, 6, 7]

In [1894]: `a[3] = a[0]`

In [1895]: `print(a)  
#[ 'a', 'b', 3, 'a', 5, 6, 7]`  
['a', 'b', 3, 'a', 5, 6, 7]

## 2.8.2 Switch elements 调换位置

In [1896]: `a = [1, 2, 3, 4, 5, 6, 7]`

In [1897]: `#将a[0]与a[3]进行对调, 即1与4对调:  
a[0], a[3] = a[3], a[0]`

In [1898]: `print(a)  
#[4, 2, 3, 1, 5, 6, 7]`  
[4, 2, 3, 1, 5, 6, 7]

In [1899]: `# Exercise 6.6.1: Reverse List: 翻转List: 头尾数字对调至不能对调为止 #算法系解法  
# Build a function RevLst which reverses a list.  
# Example: RevLst([3, 1, 9, 8, 4]) -> [4, 8, 9, 1, 3]`

In [1900]: `def RevLst(nums):  
 left = 0  
 right = len(nums) - 1  
 while left < right:  
 print("left0=", left)  
 print("right0=", right)  
 nums[left], nums[right] = nums[right], nums[left]  
 left += 1  
 right -= 1  
 print("left=", left)  
 print("right=", right)  
 print("====")  
 return nums`

In [1901]: RevLst([3, 1, 9, 8, 4])  
#[4, 8, 9, 1, 3]

```
left0= 0
right0= 4
left= 1
right= 3
===
left0= 1
right0= 3
left= 2
right= 2
===
```

Out[1901]: [4, 8, 9, 1, 3]

In [1902]: RevLst([3, 1, 9, 8, 4, 5])
#[5, 4, 8, 9, 1, 3]

```
left0= 0
right0= 5
left= 1
right= 4
===
left0= 1
right0= 4
left= 2
right= 3
===
left0= 2
right0= 3
left= 3
right= 2
===
```

Out[1902]: [5, 4, 8, 9, 1, 3]

In [1903]: #总结: 比较难懂的4个算法写法: 插入【遍历&二分】; 移除【pop】; 合并【pop】; 翻转【对调】

```
# Q2: Insert a Number in a Sorted List: 在list中插入一个值
# Description
# Build a function NumInsert which inserts a number in a sorted List.

#写法0: 遍历法
#遍历已排序列表, 找到合适的插入位置, 并将新的数字插入到正确的位置, 以保持列表的有序性。
#让ind = list长度, 从头至尾遍历所有元素, 当list中的元素大于插入值时, 让ind = i
#list.insert(index, target), 让index位置插入target元素
def NumInsert(nums, target):
    ind = len(nums)
    for i in range(len(nums)):
        if nums[i] >= target:
            ind = i
            break
    nums.insert(ind, target)
    return nums
```

In [1904]: #写法1: 二分法

```
#步骤:
# 1. 设定左右两个指针, 分解为L和R
# 2. L初始值为0, R初始值为list长度
# 3. 计算中间指针的值 M = (left + right) / 2
#      3.1. 若M指针对应的数, 大于需要插入的数, 将M设为R
#      3.2. 若M指针对应的数, 小于需要插入的数, 将M设为L
# 4. 重复步骤3, 直到
#      4.1M指针对应的值等于需要插入的值
#      4.2或者, 左右指针相遇
# 5. 将目标数字插入相应的位置
```

#每次取一半, 把另一半扔掉, 比从头到尾遍历要快

```
def NumInsert(nums, target):
    left = 0
    right = len(nums)
    while right - left > 1:
        mid = (left + right) // 2    #整数除法 //: 向下取整
        if nums[mid] == target:
            nums.insert(mid, target)
            return nums
        elif nums[mid] > target:
            right = mid
        else:
            left = mid
    if target > nums[mid]:
        nums.insert(right, target)
    else:
        nums.insert(left, target)
    return nums
```

```
In [1905]: # Q3: Remove duplicates from a sorted list: 从排好序的列表中移除重复值
# Description
# Build a function SortRm which removes duplicate numbers in a sorted list.

# Examples
# SortRm([1, 2, 2, 3, 4, 4, 5, 6]) returns [1, 2, 3, 4, 5, 6]
# SortRm([1, 2, 3, 4, 5, 6]) returns [1, 2, 3, 4, 5, 6]
```

1. 把两个list加起来, sort后return整个list

```
def MergeSorted(list1, list2):
    list3 = list1 + list2
```

2. 3种写法都可以:

```
list3.sort()
return list3
```

```
list3 = sorted(list3)
return list3
```

```
return sorted(list3)
```

3. 如果list1不为空, A; 为空, B

```
if list1:
    return A
return B
```

In [1906]: #写法3:

#pop1: tmp定为第一个位置元素, i指针指向第二位: 如果i指针上元素与tmp相同, 则移除; 否则  
#遍历法: 后一个数 = 前一个数则pop, 否则更新tmp, 指针i后移, 最后返回nums

```
def SortRm(nums):
    if len(nums) <= 1:          #如果nums长度是0或1
        return nums             #return nums
    tmp = nums[0]                #tmp设为nums第一个值
    i = 1                       #i设为1, 第二个值
    while i < len(nums):       #当i < nums长度
        if nums[i] == tmp:      #如果nums[i] == tmp:
            nums.pop(i)         #nums pop掉 i指针的值
        else:
            tmp = nums[i]        #tmp更新为num[i]值
            i += 1               #i+1
    return nums
```

In [1907]: # Q4: Merge Two Sorted Lists: 两个排好序的list合并

```
# Description
# Build a function MergeSorted which merges two sorted lists as one sorted list

# Examples
# MergeSorted([1, 3, 5], [2, 4, 6]) returns [1, 2, 3, 4, 5, 6]
# MergeSorted([1, 2, 3], [4, 5, 6]) returns [1, 2, 3, 4, 5, 6]
```

In [1908]: #写法3: i = 0, 如果list1[0]小于list2[i]个值, 则把list1[0] pop出来, 插入在list2[i]

```
def MergeSorted(Lst1, Lst2):
    i = 0
    while Lst1 and i < len(Lst2):
        if Lst1[0] <= Lst2[i]:
            num = Lst1.pop(0)
            Lst2.insert(i, num)
        i += 1

    if Lst1:
        return Lst2 + Lst1

    return Lst2
```



In [1909]: #Q5: Reverse List: 翻转list: 头尾数字对调至不能对调为止 #算法系解法

```
# Build a function RevLst which reverses a list.
# Example: RevLst([3, 1, 9, 8, 4]) -> [4, 8, 9, 1, 3]
```

In [1910]: def RevLst(nums):

```
    left = 0
    right = len(nums) - 1
    while left < right:
        nums[left], nums[right] = nums[right], nums[left]
        left += 1
        right -= 1
    return nums
```

In [1911]: #0714练习



```
In [1912]: # Q2: Insert a Number in a Sorted List
# Description
# Build a function NumInsert which inserts a number in a sorted list.

#1. 遍历法:
#注意限制条件用while, i < len(nums) and num[i] < target时, 就i+1
#直到nums[i] > target, 在i值前的插入target: nums.insert(i, target)
#return nums
def NumInsert(nums, target):
    i = 0
    while i < len(nums) and nums[i] < target:
        i += 1
    nums.insert(i, target)
    return nums

#2. 遍历法:
#将ind设置为最后一位元素,
#当i在len(nums)中时, 如果nums[i] >= target, 就ind = i;
#直到nums[i] < target时, nums.insert(ind, target).
#return nums
def NumInsert(nums, target):
    ind = len(nums)
    for i in range(len(nums)):
        if nums[i] >= target:
            ind = i
            break
    nums.insert(ind, target)
    return nums

#ind = 长度
#i在list内移动
#list[i]比插入值大时
#ind更新为i
#结束
#在[i]前插入target

#3. 二分法:
#左右两个指针, L和R
#L初始值0, R初始值list长度
#计算中间指针M值: M = (left + right) // 2
#如果M指针对应的值 = target, 在M前插入值
#如果M指针对应的值 > target, 将M设为R
#如果M指针对应的值 < target, 将M设为L
#重复步骤3, 直到
#M指针对应的值= target
#左右指针相遇
#将target插入相应位置

def NumInsert(nums, target):
    left = 0
    right = len(nums)
    while right - left > 1:
        mid = (left + right) // 2
        if nums[mid] == target:
            nums.insert(mid, target)
            return nums
        elif nums[mid] > target:
            right = mid
        else:
            left = mid
    if target > nums[mid]:
```

```
        nums.insert(right, target)
    else:
        nums.insert(left, target)
    return nums

#4.1 append&sort法:
# def NumInsert(nums, target):
#     nums.append(target)
#     nums.sort()
#     return nums

# 4.1 append&sort法:
# def NumInsert(nums, target):
#     nums.append(target)
#     nums = sorted(nums)
#     return nums

##-----#
#写法1:
# def NumInsert(nums, target):
#     nums.append(target)
#     nums.sort()
#     return nums
#写法2:
# def NumInsert(nums, target):
#     nums.append(target)
#     nums = sorted(nums)
#     return nums
# #写法3:
# def NumInsert(nums, target):
#     nums.append(target)
#     return sorted(nums)
```



In [1913]:

```
# Q3: Remove duplicates from a sorted list
# Description
# Build a function SortRm which removes duplicate numbers in a sorted list.

#1. append法:
#创建 newList, 对于num in nums来说
#如果num没在result中, 则append进result中
#return result

# def SortRm(nums):
#     result = []
#     for num in nums:
#         if num not in result:
#             result.append(num)
#     return result

#2. 遍历法: 后一个数 = 前一个数则pop, 否则更新tmp, 指针i后移, 最后返回nums

# def SortRm(nums):
#     if len(nums) <= 1:          #如果nums长度是0或1
#         return nums            #return nums
#     tmp = nums[0]              #tmp设为nums第一个值
#     i = 1                      #i设为1, 第二个值
#     while i < len(nums):      #当i < nums长度
#         if nums[i] == tmp:    #如果nums[i] == tmp:
#             nums.pop(i)        #nums.pop掉 i指针的值
#         else:                  #否则
#             tmp = nums[i]      #tmp更新为num[i]值
#             i += 1              #i+1
#     return nums

#错误代码:
# def SortRm(nums):
#     a = []
#     for i in range(len(nums)):
#         if nums[i] not in a:
#             a.append(nums[i])
#             nums.pop(nums[i])
#     return a

#不可以用pop(num[i]):
# nums.pop()函数的参数应该是一个索引值, 而不是列表中的元素值。因此, nums.pop(nums[i])
# 另外, 在遍历列表时, 使用nums.pop()函数会导致索引值的不一致, 因为在列表中删除元素后
# 修正后的代码应该使用一个新的列表来存储非重复的元素, 并使用一个额外的变量来迭代遍历原

#####
# def SortRm(nums):
#     result = []
#     for num in nums:
#         if num not in result:
#             result.append(num)
```

```
#     return result
```

# Unit 3

## 3.1 Example: List2Num

List to Number: Build a function Lst2Num which convert a number list into one number.

Example: Lst2Num([1, 3, 5]) returns 135

Example: Lst2Num([0, 1, 2, 0, 3]) returns 1203

```
In [1914]: #输入list, 返回list中不加空格的数字:  
#Loop遍历数字, 把数字取出来, 让res * 10 + 数字  
  
def Lst2Num(nums):  
    res = 0  
    for num in nums:  
        res = res * 10 + num  
    return res
```

```
In [1915]: Lst2Num([1, 3, 5])
```

```
Out[1915]: 135
```

```
In [1916]: def Lst2Num(nums):  
    res = 0  
    for num in nums:  
        print("res =", res)  
        print("num = ", num)  
        res = res * 10 + num  
        print("res =", res)  
        print("===")  
    return res
```

In [1917]: `Lst2Num([1, 3, 5])`

```
res = 0
num = 1
res = 1
===
res = 1
num = 3
res = 13
===
res = 13
num = 5
res = 135
===
```

Out[1917]: 135

In [1918]: `Lst2Num([0, 1, 2, 0, 3])`

```
res = 0
num = 0
res = 0
===
res = 0
num = 1
res = 1
===
res = 1
num = 2
res = 12
===
res = 12
num = 0
res = 120
===
res = 120
num = 3
res = 1203
===
```

Out[1918]: 1203

## 3.2 Maximum Stock Gain (2-loop version)

Maximum Stock Gain

Build a function MaxGain to find the maximum you can gain by buying and selling stocks. The stock prices represented as a list of numbers. You need to buy stocks before you sell them.

Example: `MaxGain([3, 4, 1, 5, 7, 2])` returns 6

Explanation: Buy stocks when the price is 1 and sell them when the price is 7.

Example: `MaxGain([5, 4, 3, 2, 1])` returns 0

```
In [1919]: #事后诸葛亮形式，看何时买何时卖能最大程度盈利
#看时间复杂度：
#所有到每一天和之前的进行比较：先把每一天遍历；再把这一天之前的每一天遍历
#如果后面的天 - 前面的天 > gain，就把gain替换上去

def MaxGain(nums):
    gain = 0
    for i in range(len(nums)):
        for j in range(i):
            if nums[i] - nums[j] > gain:
                gain = nums[i] - nums[j]
    return gain
```

```
In [2361]: #股价为1买入，股价为7卖出，max gain = 6
MaxGain([3, 4, 1, 5, 7, 2])
#6
```

```
min_price= 3
gain= 0
cur_price= 3
cur_gain= 0
=====
min_price= 3
gain= 0
cur_price= 4
cur_gain= 1
=====
min_price= 3
gain= 1
cur_price= 1
cur_gain= -2
=====
min_price= 1
gain= 1
cur_price= 5
cur_gain= 4
=====
min_price= 1
gain= 4
cur_price= 7
cur_gain= 6
=====
min_price= 1
gain= 6
cur_price= 2
cur_gain= 1
=====
```

Out[2361]: 6

In [2363]: #股价为1时买入，9时卖出, maxgain = 8

```
MaxGain([9, 10, 1, 5, 7, 2, 9])  
#8
```

```
min_price= 9  
gain= 0  
cur_price= 9  
cur_gain= 0  
=====  
min_price= 9  
gain= 0  
cur_price= 10  
cur_gain= 1  
=====  
min_price= 9  
gain= 1  
cur_price= 1  
cur_gain= -8  
=====  
min_price= 1  
gain= 1  
cur_price= 5  
cur_gain= 4  
=====  
min_price= 1  
gain= 4  
cur_price= 7  
cur_gain= 6  
=====  
min_price= 1  
gain= 6  
cur_price= 2  
cur_gain= 1  
=====  
min_price= 1  
gain= 6  
cur_price= 9  
cur_gain= 8  
=====
```

Out[2363]: 8

In [1921]: #股价后面一直降价, 不操作赚最多, max gain = 0

```
MaxGain([5, 4, 3, 2, 1])  
#0
```

Out[1921]: 0

```
In [1922]: def MaxGain(nums):
    gain = 0
    for i in range(len(nums)):
        print("====")
        print("i=", i)
        print("nums[i]=", nums[i])
        for j in range(i):
            print("nums[j] = ", nums[j])
            print("gain", gain)
            print("cur_gain", nums[i] - nums[j])
            print("-----")
            if nums[i] - nums[j] > gain:
                gain = nums[i] - nums[j]
    return gain
```

In [1923]: `MaxGain([3, 4, 1, 5, 7, 2])`

```
=====
i= 0
nums[i]= 3
=====

i= 1
nums[i]= 4
nums[j] = 3
gain 0
cur_gain 1
-----
=====

i= 2
nums[i]= 1
nums[j] = 3
gain 1
cur_gain -2
-----
nums[j] = 4
gain 1
cur_gain -3
-----
=====

i= 3
nums[i]= 5
nums[j] = 3
gain 1
cur_gain 2
-----
nums[j] = 4
gain 2
cur_gain 1
-----
nums[j] = 1
gain 2
cur_gain 4
-----
=====

i= 4
nums[i]= 7
nums[j] = 3
gain 4
cur_gain 4
-----
nums[j] = 4
gain 4
cur_gain 3
-----
nums[j] = 1
gain 4
cur_gain 6
-----
nums[j] = 5
gain 6
cur_gain 2
-----
=====

i= 5
```

```
nums[i]= 2
nums[j] = 3
gain 6
cur_gain -1
-----
nums[j] = 4
gain 6
cur_gain -2
-----
nums[j] = 1
gain 6
cur_gain 1
-----
nums[j] = 5
gain 6
cur_gain -3
-----
nums[j] = 7
gain 6
cur_gain -5
-----
```

Out[1923]: 6

In [1924]: #股价为1买入，股价为9卖出, max gain = 8  
MaxGain([9, 10, 1, 5, 7, 2, 9])

```
=====
i= 0
nums[i]= 9
=====
i= 1
nums[i]= 10
nums[j] = 9
gain 0
cur_gain 1
-----
=====
i= 2
nums[i]= 1
nums[j] = 9
gain 1
cur_gain -8
-----
nums[j] = 10
gain 1
. . ^
```

In [1925]: #后一天都比前一天价低，不操作赚的最多，所以gain = 0

```
MaxGain([5, 4, 3, 2, 1])
```

```
=====
i= 0
nums[i]= 5
=====
i= 1
nums[i]= 4
nums[j] = 5
gain 0
cur_gain -1
-----
=====
i= 2
nums[i]= 3
nums[j] = 5
gain 0
cur_gain -2
-----
nums[j] = 4
gain 0
.
```

### 3.3 Example: Maximum Stock Gain (1-loop version)

Homework 1、Maximum Stock Gain

Code Maximum Stock Gain Build a function MaxGain to find the maximum you can gain by buying and selling stocks. The stock prices represented as a list of numbers. You need to buy stocks before you sell them. Example: MaxGain([3, 4, 1, 5, 7, 2]) returns 6 --Explanation: Buy stocks when the price is 1 and sell them when the price is 7.

Example: MaxGain([5, 4, 3, 2, 1]) returns 0

In [2364]:

```
#时间复杂度为n
#1 Loop比2 Loop省时间
#1 Loop key: 把之前最低的价格记下来, 更新gain
# 遍历数字, 每到一个数字进行两次判断:
# i. 当前值 - min > gain, 就更新gain: 这天卖最赚;
# ii. 当前值 < min, 就更新min

def MaxGain(nums):
    min_price = nums[0] #第一天值
    gain = 0
    for num in nums:
        if num - min_price > gain: #※当前价 - 最低价 > gain
            gain = num - min_price #把gain值更新
        if num < min_price: #※当前价与最低价低
            min_price = num #把最低价更新
    return gain
```

In [1927]:

```
MaxGain([3, 4, 1, 5, 7, 2])
#6
```

Out[1927]:

6

In [1928]:

```
MaxGain([9, 10, 1, 5, 7, 2, 9])
#8
```

Out[1928]:

8

In [1929]:

```
MaxGain([5, 4, 3, 2, 1])
#0
```

Out[1929]:

0

In [1930]:

```
def MaxGain(nums):
    min_price = nums[0]
    gain = 0
    for num in nums:
        print("min_price=", min_price)
        print("gain=", gain)
        print("cur_price=", num)
        print("cur_gain=", num - min_price)
        print("====")
        if num - min_price > gain:
            gain = num - min_price
        if num < min_price:
            min_price = num
    return gain
```

In [1931]: MaxGain([9, 10, 1, 5, 7, 2, 9])

```
min_price= 9
gain= 0
cur_price= 9
cur_gain= 0
=====
min_price= 9
gain= 0
cur_price= 10
cur_gain= 1
=====
min_price= 9
gain= 1
cur_price= 1
cur_gain= -8
=====
min_price= 1
gain= 1
cur_price= 5
cur_gain= 4
=====
min_price= 1
gain= 4
cur_price= 7
cur_gain= 6
=====
min_price= 1
gain= 6
cur_price= 2
cur_gain= 1
=====
min_price= 1
gain= 6
cur_price= 9
cur_gain= 8
=====
```

Out[1931]: 8

In [1932]: MaxGain([5, 4, 3, 2, 1])

```
min_price= 5
gain= 0
cur_price= 5
cur_gain= 0
=====
min_price= 5
gain= 0
cur_price= 4
cur_gain= -1
=====
min_price= 4
gain= 0
cur_price= 3
cur_gain= -1
=====
min_price= 3
gain= 0
cur_price= 2
cur_gain= -1
=====
min_price= 2
gain= 0
cur_price= 1
cur_gain= -1
=====
```

Out[1932]: 0

### 3.4 Example: Pascal's Triangle (Recursion)

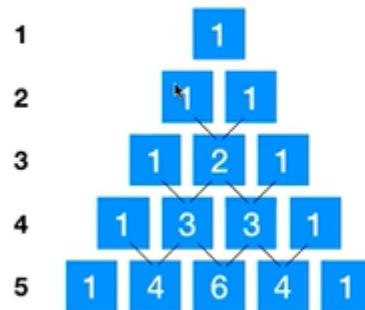
Homework 1、Pascal's Triangle

Code 7.3 Pascal's Triangle Build a function PascalT which takes an integer n as an input and generate a n layer Pascal's Triangle.

Example: PascalT(3) returns [[1], [1, 1], [1, 2, 1]]

Example: PascalT(5) returns [[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]

## 7.3 Pascal's Triangle



特点：

1. 每一层数字的数目等于层数
2. 最左边和最右边的数字都是1
3. 从第3层开始，中间第*i*个数字的值，是上一层第*i*和第*i-1*数字的和

In [1933]:

```
#杨辉三角
# 每一层数字的数目等于层数
# 最左边和最右边的数字都是1
# 从第三层开始，中间第i个数字的值，是上一层第i和第i-1数字的和
# 里面每一个元素是一个list

#每一层是上面一层元素加和出来, recursion递归: 函数内部调用函数, 互换自己

#法1:
def PascalT(n):
    if n == 1:
        return [[1]]           #返回一个list, list里只有一个数1
    elif n == 2:
        return [[1], [1, 1]]
    else:
        T = PascalT(n - 1)      #recusion: 叫自己
        last_layer = T[-1]
        new_layer = [1]          #left
        for i in range(n - 2):
            new_layer.append(last_layer[i] + last_layer[i + 1])
        new_layer.append(1)       #right
        T.append(new_layer)
    return T
```

In [1934]: PascalT(5)

Out[1934]: [[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]]

```
In [1935]: def PascalT(n):
    print("====")
    print("n=", n)
    if n == 1:
        return [[1]]
    elif n == 2:
        return [[1], [1,1]]
    else:
        T = PascalT(n - 1)
        print("T", T)
        last_layer = T[-1]
        new_layer = T[1] #left
        print("----")
        print("new_layer = ", new_layer)
        for i in range(n - 2):
            print("i = ", i)
            print("last_layer[i]", last_layer[i])
            print("last_layer[i + 1]", last_layer[i + 1])
            new_layer.append(last_layer[i] + last_layer[i + 1])
        new_layer.append(1) #right
        print("new_layer = ", new_layer)
        T.append(new_layer)
    return T
```

In [1936]: `PascalT(5)`

```
=====
n= 5
=====
n= 4
=====
n= 3
=====
n= 2
T [[1], [1, 1]]
-----
new_layer = [1, 1]
i = 0
last_layer[i] 1
last_layer[i + 1] 1
new_layer = [1, 1, 2, 1]
T [[1], [1, 1, 2, 1], [1, 1, 2, 1]]
-----
new_layer = [1, 1, 2, 1]
i = 0
last_layer[i] 1
last_layer[i + 1] 1
i = 1
last_layer[i] 1
last_layer[i + 1] 2
new_layer = [1, 1, 2, 1, 2, 3, 1]
T [[1], [1, 1, 2, 1, 2, 3, 1], [1, 1, 2, 1, 2, 3, 1], [1, 1, 2, 1, 2, 3, 1]]
-----
new_layer = [1, 1, 2, 1, 2, 3, 1]
i = 0
last_layer[i] 1
last_layer[i + 1] 1
i = 1
last_layer[i] 1
last_layer[i + 1] 2
i = 2
last_layer[i] 2
last_layer[i + 1] 1
new_layer = [1, 1, 2, 1, 2, 3, 1, 2, 3, 3, 1]
```

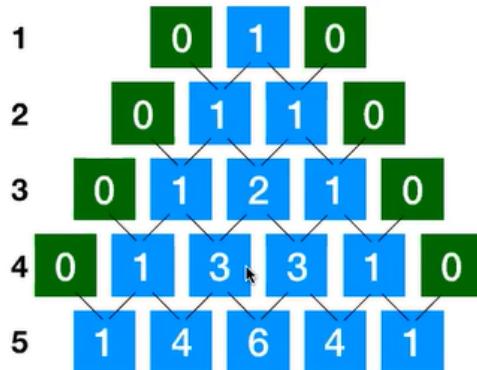
Out[1936]: `[[1], [1, 1, 2, 1, 2, 3, 1, 2, 3, 3, 1], [1, 1, 2, 1, 2, 3, 1, 2, 3, 3, 1], [1, 1, 2, 1, 2, 3, 1, 2, 3, 3, 1], [1, 1, 2, 1, 2, 3, 1, 2, 3, 3, 1]]`

### 3.5 Pascal's Triangle (LOOP) #比recursion解法更简单

Build a function `PascalT` which takes an integer `n` as an input and generate a `n` layer Pascal's Triangle.

Example: `PascalT(3)` returns `[[1], [1, 1], [1, 2, 1]]`

## 7.3 Pascal's Triangle



特点:

1. 每一层数字的数目等于层数
2. 最左边和最右边的数字都是 1
3. 从第 3 层开始，中间第  $i$  个数字的值，是上一层第  $i$  和第  $i-1$  数字的和

```
In [1937]: def PascalT(n):
    if n <= 0:
        return []
    T = [[1]]
    if n > 1:
        for i in range(1, n):
            last_layer = T[-1]
            last_layer = [0] + last_layer + [0] # 头尾加0
            new_layer = []
            for j in range(len(last_layer) - 1):
                new_layer.append(last_layer[j] + last_layer[j + 1])
            T.append(new_layer)
    return T
```

```
In [1938]: PascalT(5)
#[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]
```

Out[1938]: [[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]

```
In [1939]: def PascalT(n):
    if n <= 0:
        return []
    T = [[1]]
    if n > 1:
        for i in range(1, n):
            print("=====")
            print("cur_layer_num", i + 1)
            last_layer = T[-1]
            print("last_layer = ", last_layer)
            new_layer = []
            for j in range(len(last_layer) - 1):
                new_layer.append(last_layer[j] + last_layer[j + 1])
                print("j = ", j)
                print("new_layer", new_layer)
            T.append(new_layer)
    return T
```

```
In [1940]: PascalT(5)
# =====
# cur_layer_num 2
# last_layer = [1]
# =====
# cur_layer_num 3
# last_layer = []
# =====
# cur_layer_num 4
# last_layer = []
# =====
# cur_layer_num 5
# last_layer = []
# [[1], [], [], [], []]
```

```
=====
cur_layer_num 2
last_layer = [1]
=====
cur_layer_num 3
last_layer = []
=====
cur_layer_num 4
last_layer = []
=====
cur_layer_num 5
last_layer = []
```

Out[1940]: [[1], [], [], [], []]

## 3.6 Intro to Set

### 3.6.1 Set

a set is denoted by a pair of curly braces

#### 3.6.1.1 An empty set

```
In [1941]: #花括号定义set  
a = {}
```

```
In [1942]: print(a)  
#{}  
  
{}
```

#### 3.6.1.2 Sets of numbers

```
In [1943]: a = {1, 2, 3, 4}
```

```
In [1944]: print(a)  
#{1, 2, 3, 4}  
  
{1, 2, 3, 4}
```

#### 3.6.1.3 A set of strings

```
In [1945]: a = {"a", "b", "cd", "e"}
```

```
In [1946]: print(a)  
#{'b', 'e', 'cd', 'a'}  
  
{'e', 'cd', 'a', 'b'}
```

#### 3.6.1.4 A set of numbers and strings

```
In [1947]: a = {1, 2, "a", "bc"}
```

```
In [1948]: print(a)  
#{1, 2, 'a', 'bc'}  
  
{1, 'bc', 2, 'a'}
```

#### 3.6.1.5 No list or set can be in a set

```
In [1949]: #list里可以有list, set里不能有list
#[[]]

# -----
# TypeError
# Cell In[88], line 1
# ----> 1 [[]]

# TypeError: unhashable type: 'List'
```

Traceback (most recent call last)

```
In [1950]: #set里也不能有set
#[{}]

# -----
# TypeError
# Cell In[90], line 1
# ----> 1 {}

# TypeError: unhashable type: 'dict'
```

Traceback (most recent call last)

### 3.6.1.6 A set contains no duplicates

set中没有重复值，类似于select distinct

```
In [1951]: a = {1, 1, 2, 3, 4, 4, 5}
```

```
In [1952]: print(a)
#{1, 2, 3, 4, 5}
```

{1, 2, 3, 4, 5}

```
In [1953]: a = {"a", 1, 3, "a", 1, 3}
```

```
In [1954]: print(a)
#{1, 3, 'a'}
```

{1, 3, 'a'}

## 3.7 Using set to remove duplicates

```
In [1955]: #Transformation between set and list
```

### 3.7.1 List to set

```
In [1956]: a = [1, 2, 3, 4]
```

```
In [1957]: #将这个list转换成set: set(list)
set(a)
#{1, 2, 3, 4}
```

Out[1957]: {1, 2, 3, 4}

### 3.7.2 Set to list

```
In [1958]: a = {1, 2, 3, 4}
```

```
In [1959]: #将set转换成list: list(set)
list(a)
#[1, 2, 3, 4]
```

Out[1959]: [1, 2, 3, 4]

### 3.7.3 Exercise 2.2.1: Remove Duplicates from a List

Build a function RmDuplicate which remove all the duplicates from a list. Example:

RmDuplicates([3, 1, 2, 3, 4, 5, 6]) returns [1, 2, 3, 4, 5, 6] Example: RmDuplicates([1, 3, 5]) returns [1, 3, 5]

```
In [1960]: def RmDuplicates(nums):
    return list(set(nums))
```

```
In [1961]: RmDuplicates([3, 1, 2, 3, 4, 5, 6])
#[1, 2, 3, 4, 5, 6]
```

Out[1961]: [1, 2, 3, 4, 5, 6]

## 3.8 Using Set for Search

### 3.8.1 Elements in a set

#### 3.8.1.1 We can't get an element from a set using index

```
In [1962]: a = {1, 2, 3, 4}
```

```
In [1963]: print(a)
```

```
#{1, 2, 3, 4}
```

{1, 2, 3, 4}

In [1964]: #不能用index去叫set中的数:

```
#print(a[1])
# -----
# Type Error
# Cell In[135], line 1
# ----> 1 print(a[1])

# TypeError: 'set' object is not subscriptable
```

### 3.8.1.2 Checking whether an element in a set is fast

总结: set可以去重; 如果找是否有一个元素在set中, 速度特别快

In [1965]: #a是list, b是set, set中找东西很方便  
#set找东西的速度很快

```
a = [i for i in range(1000)]
b = set(a)
```

In [1966]: print(5 in a)

True

In [1967]: print(5 in b)

True

## Unit 4: String

### 4.1 Introduction to String

In [1968]: #list用[]; set用{}; string用' ' or " " or """"""

#### 4.1.1 An empty string

In [1969]: a = ""

In [1970]: print(a)

In [1971]: a = ''

```
In [1972]: print(a)
```

```
In [1973]: a = "*****"
```

```
In [1974]: print(a)
```

```
In [1975]: a = '*****'
```

```
In [1976]: print(a)
```

```
In [1977]: #但引号不能混用: eg. 一个单引号 一个双引号  
#a = ''
```

```
#SyntaxError: unterminated string literal (detected at line 1)
```

## 4.1.2 A string with letters

```
In [1978]: #红色是字符串  
a = "abcde"
```

```
In [1979]: print(a)
```

```
abcde
```

```
In [1980]: #绿色是数字  
a = 1234
```

```
In [1981]: print(a)
```

```
1234
```

```
In [1982]: a = "1234"
```

```
In [1983]: print(a)
```

```
1234
```

In [1984]: #数字相加:

```
a = 1234  
b = 4321  
print(a + b)  
#5555
```

5555

In [1985]: #字符串相加: 将数字引号起

```
a = "1234"  
b = "4321"  
print(a + b)  
#12344321
```

12344321

### 4.1.3 Strings with quotation marks

In [1986]: #双引号, 单引号, 三引号, 可以组合出现; 只要最外层和内里的引号种类不同, 就可以打印内里

In [1987]: a = "ab'c'de"

In [1988]: print(a)  
#ab'c'de

ab'c'de

In [1989]: a = 'a"bcd"e'

In [1990]: print(a)  
#a"bcd"e

a"bcd"e

In [1991]: a = """a"b'c'de"""

In [1992]: print(a)  
#a"b'c'de

a"b'c'de

## 4.2 Example: Palindrome

### 4.2.1 Get characters from a string

In [1993]: `a = "abcde"`

In [1994]: `a[0]`  
`#'a'`

Out[1994]: `'a'`

In [1995]: `a[1]`  
`#'b'`

Out[1995]: `'b'`

In [1996]: `a[-1]`  
`#'e'`

Out[1996]: `'e'`

### 4.2.2 Exercise 2.1.1: Palindrome

Build a function `isPalindrome` which takes a string as an input and returns boolin value telling whether the string is a palindrome. Example: `isPalindrome("abcba")` returns True Example: `isPalindrome("abba")` returns True Example: `isPalindrome("a")` returns True Example: `isPalindrome("")` returns True Example: `isPalindrome("abcd")` returns False

回文：看是否对称

In [1997]: `def isPalindrome(s):`  
 `left = 0`  
 `right = len(s) - 1`  
 `while left < right:`  
 `if s[left] == s[right]:`  
 `left += 1`  
 `right -= 1`  
 `else:`  
 `return False`  
 `return True`

In [1998]: `#左从第一个指针，右从最后一个元素指针，开始(最后一个元素的数，[]是元素Length - 1)`  
`#当 左 小于 右：看左右指针是否相等`  
`#如果相等，就把左右 都往内 移一格；`  
`#当奇数个数字时，左右指针相遇相等时结束；当偶数个数字时，左右指针在最近的地方挨着时结`  
`#一旦不等，就return False；否则走完所有指针都相等 return True`

```
In [1999]: isPalindrome("abcde")
#False
```

Out[1999]: False

```
In [2000]: isPalindrome("abcd")
#False
```

Out[2000]: False

```
In [2001]: def isPalindrome(s):
    left = 0
    right = len(s) - 1
    while left < right:
        print("left = ", left)
        print("right = ", right)
        print("s[left]=", s[left])
        print("s[right]=", s[right])
        print("====")
        if s[left] == s[right]:
            left += 1
            right -= 1
        else:
            return False
    return True
```

```
In [2002]: isPalindrome("abcba")
```

```
left = 0
right = 4
s[left]= a
s[right]= a
=====
left = 1
right = 3
s[left]= b
s[right]= b
=====
```

Out[2002]: True

```
In [2003]: isPalindrome("abcd")
```

```
left = 0
right = 3
s[left]= a
s[right]= d
=====
```

Out[2003]: False

## 4.3 Example: Valid Parentheses

### 4.3.1 Exercise 2.1.2: Valid Parenthesis

Build a function ValidParenthesis to determine the parenthesis in a string are valid. Example:  
 ValidParenthesis('((())())') returns True Example: ValidParenthesis('()()()') returns True Example:  
 ValidParenthesis(')()()') returns False Example: ValidParenthesis('(((())')) returns False

```
In [2004]: #list 和 string的综合应用 => stack
#消消乐:
# i. 遇到(：就将其存入tmp
# ii. 遇到)：如果之前有存入的(，就将其消掉；如果没有，就false
# iii. 走完全程时，tmp不为空，还有剩的(，也false
```

```
In [2005]: def ValidParenthesis(s):
    tmp = []
    for c in s:
        if c == '(':
            tmp.append(c)
        if c == ')':
            if tmp:
                tmp.pop()
            else:
                return False
        if tmp:
            return False
    else:
        return True
```

```
In [2006]: ValidParenthesis('((())())')
#True
```

Out[2006]: True

```
In [2007]: ValidParenthesis(')()()()')
#False
```

Out[2007]: False

```
In [2008]: def ValidParenthesis(s):
    tmp = []
    for c in s:
        print("tmp=", tmp)
        print("c=", c)
        print("====")
        if c == '(':
            tmp.append(c)
        if c == ')':
            if tmp:
                tmp.pop()
            else:
                return False
    print("final tmp=", tmp)
    if tmp:
        return False
    else:
        return True
```

```
In [2009]: ValidParenthesis('((())())')
```

```
tmp= []
c= (
=====
tmp= ['(']
c= (
=====
tmp= ['(', '(']
c= (
=====
tmp= ['(', '(', '(']
c= )
=====
tmp= ['(', '(']
c= )
=====
tmp= ['(']
c= )
=====
tmp= []
c= (
=====
tmp= ['(']
c= )
=====
final tmp= []
```

```
Out[2009]: True
```

In [2010]: `ValidParenthesis('(((()))())')`

```
tmp= []
c= (
=====
tmp= ['(']
c= (
=====
tmp= ['(', '(']
c= (
=====
tmp= ['(', '(', '(']
c= (
=====
tmp= ['(', '(', '(', '(']
c= )
=====
tmp= ['(', '(', '(', ')
c= )
=====
tmp= ['(', '(', '(', ')
c= )
=====
tmp= ['(']
c= (
=====
tmp= ['(', '(']
c= )
=====
```

final tmp= ['(']

Out[2010]: False

```
In [2011]: ValidParenthesis('((())())')
#           if c == ')':
#               if tmp:
#                   tmp.pop()
#               else:
#                   return False
#如果是退无可退的), 直接return False

tmp= []
c= (
=====
tmp= ['(']
c= (
=====
tmp= ['(', '(']
c= )
=====
tmp= ['(']
c= )
=====
tmp= []
c= )
=====
```

Out[2011]: False

## 4.4 Example: Ngrams

### 4.4.1 Get a substring

In [2012]: a = 'abcde'

In [2013]: a[2:4]  
 #'cd'

Out[2013]: 'cd'

In [2014]: a[2:]  
 #'cde'

Out[2014]: 'cde'

In [2015]: a[:2]  
 #'ab'

Out[2015]: 'ab'

### 4.4.2 Exercise 2.2.1: N Grams

Build a function NGram which takes a string s and an integer n as inputs, returns a list of n grams of s.

Example: NGram("techlent", 2) returns ["te", "ec", "ch", "hl", "le", "en", "nt"]

Example: NGram("abc", 3) returns ["abc"]

Example: NGram("abc", 4) returns []

Example: NGram("abc", 1) returns ["a", "b", "c"]

```
In [2016]: #s代表字符串, n是需要几个字符:  
#len(s) - n + 1的作用, 是i运行到 元素总数 - n 个元素时停止:  
def NGram(s, n):  
    res = []  
    for i in range(len(s) - n + 1):  
        res.append(s[i: i + n])  
    return res
```

```
In [2017]: NGram("techlent", 1)  
#[ 't', 'e', 'c', 'h', 'l', 'e', 'n', 't']  
https://pythontutor.com/render.html#code=def%20NGram%28s,%20n%29%3A%0A%20%20%20
```

Out[2017]: ['t', 'e', 'c', 'h', 'l', 'e', 'n', 't']

```
In [2018]: NGram("techlent", 2)  
https://pythontutor.com/render.html#code=def%20NGram%28s,%20n%29%3A%0A%20%20%20
```

Out[2018]: ['te', 'ec', 'ch', 'hl', 'le', 'en', 'nt']

```
In [2019]: NGram("techlent", 3)  
https://pythontutor.com/render.html#code=def%20NGram%28s,%20n%29%3A%0A%20%20%20
```

Out[2019]: ['tec', 'ech', 'chl', 'hle', 'len', 'ent']

```
In [2020]: NGram("techlent", 10)
```

Out[2020]: []

```
In [2021]: def NGram(s, n):  
    res = []  
    print("len(s) - n + 1 =", len(s) - n + 1)  
    for i in range(len(s) - n + 1):  
        print("i=", i)  
        print("i + n =", i + n)  
        print("s[i : i + n] = ", s[i : i + n])  
        print("====")  
        res.append(s[i: i + n])  
    return res
```

In [2022]: `NGram("techelent", 1)`

```
len(s) - n + 1 = 9
i= 0
i + n = 1
s[i : i + n] =  t
=====
i= 1
i + n = 2
s[i : i + n] =  e
=====
i= 2
i + n = 3
s[i : i + n] =  c
=====
i= 3
i + n = 4
s[i : i + n] =  h
=====
i= 4
i + n = 5
s[i : i + n] =  e
=====
i= 5
i + n = 6
s[i : i + n] =  l
=====
i= 6
i + n = 7
s[i : i + n] =  e
=====
i= 7
i + n = 8
s[i : i + n] =  n
=====
i= 8
i + n = 9
s[i : i + n] =  t
=====
```

Out[2022]: `['t', 'e', 'c', 'h', 'e', 'l', 'e', 'n', 't']`

In [2023]: `NGram("techlent", 2)`

```
len(s) - n + 1 = 7
i= 0
i + n = 2
s[i : i + n] =  te
=====
i= 1
i + n = 3
s[i : i + n] =  ec
=====
i= 2
i + n = 4
s[i : i + n] =  ch
=====
i= 3
i + n = 5
s[i : i + n] =  hl
=====
i= 4
i + n = 6
s[i : i + n] =  le
=====
i= 5
i + n = 7
s[i : i + n] =  en
=====
i= 6
i + n = 8
s[i : i + n] =  nt
=====
```

Out[2023]: `['te', 'ec', 'ch', 'hl', 'le', 'en', 'nt']`

In [2024]: `NGram("techlent", 3)`

```

len(s) - n + 1 = 6
i= 0
i + n = 3
s[i : i + n] = tec
=====
i= 1
i + n = 4
s[i : i + n] = ech
=====
i= 2
i + n = 5
s[i : i + n] = chl
=====
i= 3
i + n = 6
s[i : i + n] = hle
=====
i= 4
i + n = 7
s[i : i + n] = len
=====
i= 5
i + n = 8
s[i : i + n] = ent
=====
```

Out[2024]: `['tec', 'ech', 'chl', 'hle', 'len', 'ent']`

In [2025]: `NGram("techlent", 10)`

```
len(s) - n + 1 = -1
```

Out[2025]: `[]`

## 4.5 Example: All Red

In [2026]: `a = "abcde"`

```

print("a" in a)
#True

print("bcd" in a)
#True

print("acd" in a)
#False
```

```
True
True
False
```

## 4.5.1 Exercise 2.3.1: All Red

Build a function AllRed that takes a list of strings as input, returns a boolean indicating whether all the strings containing word "red".

Example: AllRed(["red hat", "a pair of red shoes", "three red apples"]) returns True

Example: AllRed(["red hat", "white shirt", "black eyes"]) returns False

Example: AllRed([]) returns False

```
In [2360]: def AllRed(Lst):
    if not Lst:
        return False
    for s in Lst:
        if 'red' not in s:
            return False
    return True
```

```
In [2028]: AllRed(["red hat", "a pair of red shoes", "three red apples"])
#True
```

Out[2028]: True

```
In [2029]: AllRed(["red hat", "white shirt", "black eyes"])
#False
```

Out[2029]: False

```
In [2030]: def AllRed(Lst):
    if not Lst:
        return False
    for s in Lst:
        print("s=", s)
        if 'red' not in s:
            return False
    return True
```

```
In [2031]: AllRed(["red hat", "a pair of red shoes", "three red apples"])

s= red hat
s= a pair of red shoes
s= three red apples
```

Out[2031]: True

```
In [2032]: AllRed(["red hat", "white shirt", "black eyes"])

s= red hat
s= white shirt
```

Out[2032]: False

In [2033]: `AllRed([])`

Out[2033]: `False`

## 4.6 Modifying a string: Replacing characters in a string

### 4.6.1 Strings are immutable

List可以修改，string不可修改 一定要改，用.replace function

In [2034]: `a = "abcde"`

In [2035]: `a[0]`  
`#'a'`

Out[2035]: `'a'`

In [2036]: `#string is immutable: string不可修改`

```
#a[0] = "f"

# -----
# TypeError
# Cell In[58], line 1
# ----> 1 a[0] = "f"

# TypeError: 'str' object does not support item assignment
```

*Traceback (most recent call last)*

### 4.6.2 Replace

1. `a.replace("要替代的string", "想替成的string")`
2. `a.replace("要替代的string", "想替成的string", 2)`: 把前两个要提到的string换成新的
3. `"想替成的string" + a[1:]`: 新string和从第二个元素开始结合
4. `a[:m] + "想替成的string" + a[n:]`: 前m-1个元素字符 + 新String + n个元素后的字符

In [2037]: `#之前学到remove没有返回值, pop有返回值`  
`#replace有返回值，并没有改变a本身`  
`#把新a赋值给a时，才改变a本身`

In [2038]: `a = 'abcde'`

```
In [2039]: #直接进行replace操作, 不会改变原a
a.replace("a", "f")
#'fbcd'e
print(a)
#abcde
```

abcde

```
In [2040]: #赋值给a后进行replace操作, 会改变原a
a = a.replace("a", "f")
print(a)
#fbcd'e
```

fbcd'e

```
In [2041]: a = "aaaaaa"
print(a)
#aaaaaa
```

aaaaaa

```
In [2042]: a.replace("a", "f")
#'fffff'
print(a)
#aaaaaa
```

aaaaaa

```
In [2043]: a.replace("a", "f", 2)
#'ffaaaa'
```

Out[2043]: 'ffaaaa'

```
In [2044]: "f" + a[1:]
#'faaaaa'
```

Out[2044]: 'faaaaa'

```
In [2045]: a[:3] + "f" + a[4:]
#'aaafaa'
```

Out[2045]: 'aaafaa'

## 4.7 Modifying a string: Concatenating strings & reversing strings

### 4.7.1 Homework

- 1、Exercise 3.4.1: Reverse String
- 2、Exercise 3.4.2: Palindrome

## 4.7.2 Concatenate Strings

```
In [2046]: a = "abcde"
b = "12345"
c = a + b
print(c)
#abcde12345
```

abcde12345

## 4.7.3 Reverse a string

```
In [2047]: a = "abcde"
```

```
In [2048]: a[::-1]
#'edcba'
```

Out[2048]: 'edcba'

```
In [2049]: a[::-1]
#'edcba'
```

Out[2049]: 'edcba'

```
In [2050]: a[::2]
#'ace'
```

Out[2050]: 'ace'

## 4.7.4 Exercise 3.4.1: Reverse String

Build a function RevStr to reverse a string. Example: RevStr("abcd") returns "dcba"

```
In [2051]: def RevStr(s):
    return s[::-1]
```

```
In [2052]: RevStr("abcd")
#'dcba'
```

Out[2052]: 'dcba'

## 4.7.5 Exercise 3.4.2: Palindrome

Build a function isPalindrome which takes a string as an input and returns boolin value telling whether the string is a palindrome. Example: isPalindrome("abcba") returns True Example: isPalindrome("abba") returns True Example: isPalindrome("a") returns True Example: isPalindrome("") returns True Example: isPalindrome("abcd") returns False

```
In [2053]: def isPalindrome(s):
    return s == s[::-1]
```

```
In [2054]: isPalindrome("abcba")
#True
```

Out[2054]: True

```
In [2055]: isPalindrome("abcd")
#False
```

Out[2055]: False

## 4.8 Transforming a string: String to List & List to String

### 4.8.1 String to list

①使用sorted(string) ② a = list(string); a.sort()

```
In [2056]: a = 'baedc'
```

```
In [2057]: #a.sort()
```

```
# -----
# AttributeError                                Traceback (most recent call last)
# Cell In[117], line 2
#      1 a = 'baedc'
#      2 a.sort()
#
# AttributeError: 'str' object has no attribute 'sort'
```

```
In [2058]: #sort string: ①使用sorted(string)
sorted(a)
#[ 'a', 'b', 'c', 'd', 'e']
```

Out[2058]: ['a', 'b', 'c', 'd', 'e']

```
In [2059]: #sort string: ② a = list(string); a.sort()
a = list(a)
print(a)
#[ 'b', 'a', 'e', 'd', 'c']
a.sort()
print(a)
#[ 'a', 'b', 'c', 'd', 'e']
```

```
['b', 'a', 'e', 'd', 'c']
['a', 'b', 'c', 'd', 'e']
```

## 4.8.2 List to string

①使用string = "分隔符或为空".join(list) ②string = str(list)

```
In [2060]: a = ["a", "b", "c", "d", "e"]
print(a)
#[ 'a', 'b', 'c', 'd', 'e' ]
['a', 'b', 'c', 'd', 'e']
```

```
In [2061]: a = "".join(a)
print(a)
#abcde
```

abcde

```
In [2062]: a = ["a", "b", "c", "d", "e"]
print(a)
#[ 'a', 'b', 'c', 'd', 'e' ]
['a', 'b', 'c', 'd', 'e']
```

```
In [2063]: b = str(a)
print(b)
#[ 'a', 'b', 'c', 'd', 'e' ]
['a', 'b', 'c', 'd', 'e']
```

```
In [2064]: b[0]
#[ '['
```

Out[2064]: '['

```
In [2065]: b[1]
#"'"
```

Out[2065]: "''"

```
In [2066]: str(1)
#'1'
```

Out[2066]: '1'

```
In [2067]: a = ["a", "b", "c", "d", "e"]
print(a)
#[ 'a', 'b', 'c', 'd', 'e' ]
['a', 'b', 'c', 'd', 'e']
```

```
In [2068]: a = ",".join(a)
print(a)
#a,b,c,d,e

a,b,c,d,e
```

```
In [2069]: a = ["a", "b", "c", "d", "e"]
"".join(a)
#'abcde'
```

Out[2069]: 'abcde'

```
In [2070]: ";" .join(a)
#'a;b;c;d;e'
```

Out[2070]: 'a;b;c;d;e'

### 4.8.3 Split a string

```
In [2071]: a = 'a, b, cd, e'
a.split(",")
#[['a', 'b', 'cd', 'e']]
```

Out[2071]: ['a', 'b', 'cd', 'e']

```
In [2072]: a = 'a, b, cd,, e'
a.split(",")
#[['a', 'b', 'cd', '', 'e']]
```

Out[2072]: ['a', 'b', 'cd', '', 'e']

```
In [2073]: a = 'abcde'
a.split()
#[['abcde']]
```

Out[2073]: ['abcde']

## 4.9 Example: Reverse Sentence

### 4.9.1 Exercise 4.3.1: Reverse Sentence

Build a function RevSen to reverse a sentence. Example: RevSen("we are all friends") returns "friends all are we"

①list = string.split(): string分开为list

②list\_2 = list[::-1]: list全部返回来

③string = "".join(list): join一起成为新string

```
In [2074]: def RevSen(sen):
    words = sen.split()
    return " ".join(words[::-1])
```

```
In [2075]: RevSen("we are all friends")
# 'friends all are we'
```

Out[2075]: 'friends all are we'

```
In [2076]: def RevSen(sen):
    words = sen.split()
    print("words = ", words)

    new_words = words[::-1]
    print("new_words", new_words)

    res = " ".join(new_words)
    print("res", res)

    return res
```

```
In [2077]: RevSen("we are all friends")
# words = ['we', 'are', 'all', 'friends']
# new_words ['friends', 'all', 'are', 'we']
# res friends all are we
# 'friends all are we'
```

```
words = ['we', 'are', 'all', 'friends']
new_words ['friends', 'all', 'are', 'we']
res friends all are we
```

Out[2077]: 'friends all are we'

## 4.10 Example: Reverse Words

### 4.10.1 Exercise 4.3.2: Reverse Words

Build a function RevWords to reverse each word of a sentence. Example: RevWords("we are all friends") returns "ew era lla sdneirf"

①法1：句split成词; 词倒过来；放进res ②法2：句split成词; 每个单词倒过来加到新的list; 再join[更简单]

In [2078]: #法1：句split成词；词倒过来；放进res

```
def RevWords(sen):
    words = sen.split()
    res = ""
    for word in words:
        if res:
            res += " "
        res += word[::-1]
    return res
```

In [2079]: RevWords("we are all friends")
#'ew era lla sdneirf'

Out[2079]: 'ew era lla sdneirf'

In [2080]: def RevWords(sen):
 words = sen.split()
 print("words = ", words)
 res = ""
 for word in words:
 print("word = ", word)
 print("word[::-1]=", word[::-1])
 if res:
 res += " "
 res += word[::-1]
 print("res", res)
 print("====")
 return res

In [2081]: RevWords("we are all friends")

```
words = ['we', 'are', 'all', 'friends']
word = we
word[::-1]= ew
res ew
=====
word = are
word[::-1]= era
res ew era
=====
word = all
word[::-1]= lla
res ew era lla
=====
word = friends
word[::-1]= sdneirf
res ew era lla sdneirf
=====
```

Out[2081]: 'ew era lla sdneirf'

```
In [2082]: #法2：句split成词；每个单词倒过来加到新的list；再join
def RevWords(sen):
    words = sen.split()
    new_words = []
    for word in words:
        new_words.append(word[::-1])
    return " ".join(new_words)
```

```
In [2083]: RevWords("we are all friends")
# 'ew era lla sdneirf'
```

```
Out[2083]: 'ew era lla sdneirf'
```

## Unit 5: Boolean

### 5.1 Boolean: Review of Boolean

True and False

```
In [2084]: True
```

```
Out[2084]: True
```

```
In [2085]: False
```

```
Out[2085]: False
```

```
In [2086]: 1 == 2
#False
```

```
Out[2086]: False
```

```
In [2087]: (1 == 2) == False
#True
```

```
Out[2087]: True
```

```
In [2088]: not True
#False
```

```
Out[2088]: False
```

```
In [2089]: not False
#True
```

```
Out[2089]: True
```

In [2090]: `#Boolean function:`

```
def boolean_exmaple(boolean):
    if boolean:
        return "It is true"
    else:
        return "It is false"
```

In [2091]: `boolean_exmaple(True)`  
 `#'It is true'`

Out[2091]: 'It is true'

In [2092]: `boolean_exmaple(False)`  
 `#'It is false'`

Out[2092]: 'It is false'

In [2093]: `boolean_exmaple(2 == 2)`  
 `#'It is true'`

Out[2093]: 'It is true'

## 5.2 Boolean: Using Integers as Booleans

Code

Number

In [2094]: `#0就是false, 1就是true`  
`#python中其他整数值, 即使不直接等于true(e.g. -1, 2), 但是可以当做true来用`

In [2095]: `def number_example(num):`  
 `if num:`  
 `return "It is true"`  
 `else:`  
 `return "It is false"`

In [2096]: `number_example(1)`  
 `#'It is true'`

Out[2096]: 'It is true'

In [2097]: `#对整数integer来讲, 0就是false。`  
`number_example(0)`  
 `#'It is false'`

Out[2097]: 'It is false'

```
In [2098]: number_example(2)
#It is true
# 2 本身不等于true, 但number_example(2) 可以当做 true来用
```

Out[2098]: 'It is true'

```
In [2099]: number_example(-1)
#It is true'
```

Out[2099]: 'It is true'

```
In [2100]: 1 == True
#True
```

Out[2100]: True

```
In [2101]: 2 == True
#False
```

Out[2101]: False

```
In [2102]: 0 == False
#True
```

Out[2102]: True

```
In [2103]: -1 == True
#False
```

Out[2103]: False

```
In [2104]: #应用0会成为false的特殊性质, 当num = 0时会退出, 所以显示的是num 到 0 , 递减为1的序
def count_down(num):
    while num:
        print(num)
        num -= 1
```

```
In [2105]: count_down(4)
# 4
# 3
# 2
# 1
```

4  
3  
2  
1

## 5.3 Boolean: Using Lists as Booleans

List

```
In [2106]: #一个list里，只要有东西（包括0），就返回True；如果为空，返回False
def list_example(li):
    if li:
        return "It is true"
    else:
        return "It is false"
```

```
In [2107]: list_example([1, 2, 3, 4])
#'It is true'
```

Out[2107]: 'It is true'

```
In [2108]: list_example([0])
#'It is true'
```

Out[2108]: 'It is true'

```
In [2109]: list_example(["a", "b", "c", "d"])
#'It is true'
```

Out[2109]: 'It is true'

```
In [2110]: list_example([])
#'It is false'
```

Out[2110]: 'It is false'

```
In [2111]: [1, 2, 3, 4] == True
#False
```

Out[2111]: False

```
In [2112]: [] == False
#False
```

Out[2112]: False

```
In [2113]: #应用性质，如果list里有东西，就会打印 + pop
def pop_out(li):
    while li:
        print(li)
        li.pop()
```

```
In [2114]: pop_out([1, 2, 3, 4])
#[1, 2, 3, 4]
#[1, 2, 3]
#[1, 2]
#[1]
```

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

## 5.4 Boolean: Using Strings as Booleans

"": echo特效

```
In [2115]: def string_example(string):
    if string:
        return "It is true"
    else:
        return "It is False"
```

```
In [2116]: string_example("Techlent")
#'It is true'
```

```
Out[2116]: 'It is true'
```

```
In [2117]: string_example("")
#'It is False'
```

```
Out[2117]: 'It is False'
```

```
In [2118]: string_example(" ")
#'It is true'
```

```
Out[2118]: 'It is true'
```

```
In [2119]: "1234" == True
#False
```

```
Out[2119]: False
```

```
In [2120]: "" == False
#False
```

```
Out[2120]: False
```

```
In [2121]: def shrink(string):
    while string:
        print(string)
        string = string[1:]
```

```
In [2122]: shrink("Techlent")
# Techlent
# echlent
# chlent
# hlen
# lent
# ent
# nt
# t
```

```
Techlent
echlent
chlent
hlen
lent
ent
nt
t
```

```
In [2123]: shrink("Samoyed")
```

```
Samoyed
amoyed
moyed
oyed
yed
ed
d
```

```
In [2124]: def echo(string):
    while string:
        print(string)
        string = string[2:]

echo("要 ~ 开 ~ 心 ~ 吆 ~")
```

```
要 ~ 开 ~ 心 ~ 吆 ~
~ 开 ~ 心 ~ 吆 ~
开 ~ 心 ~ 吆 ~
~ 心 ~ 吆 ~
心 ~ 吆 ~
~ 吆 ~
邬 ~
~
```

## 5.5 Built-in functions: Calling A Function

Code

Function determining whether a number is odd

```
In [2125]: def is_odd(num):
    """
        This function returns whether a num is odd.
    """
    if num % 2 == 0: #数字除以2
        return False #如余数 == 0, 则为偶数, 故false
    else:
        return True #如余数 != 0, 则为奇数, 故true
```

```
In [2126]: is_odd(2)
#False
```

Out[2126]: False

```
In [2127]: is_odd(3)
#True
```

Out[2127]: True

### 5.5.1 Using the function in another function

类似于cte table, 使用上方已产生的变量和表名 写一个collect所有奇数的函数, 将判断奇数的函数, 放在里面

```
In [2128]: def collect_odds(nums):
    """
        This function returns a list of all the odd numbers in nums
    """
    res = []
    for num in nums:
        if is_odd(num):
            res.append(num)
    return res
```

```
In [2129]: collect_odds([1, 2, 3, 4, 5])
#[1, 3, 5]
```

Out[2129]: [1, 3, 5]

## 5.6 Import

### 5.6.1 Import is\_even function

##先下载even.py到所在文件夹, 就可以调用even.py中写好的函数, 即类似调用numpy, pandas中的函数

```
In [2130]: #even.py中内容:
# def is_even(num):
#     if num % 2 == 0:
#         return True
#     else:
#         return False
```

```
In [2131]: from even import is_even
```

```
In [2132]: is_even(2)
#True
```

```
Out[2132]: True
```

```
In [2133]: is_even(3)
#False
```

```
Out[2133]: False
```

## 5.6.2 Using the imported function in another function

```
In [2134]: def collect_evens(nums):
    """
    This function returns a list of all the even numbers in nums.
    """
    res = []
    for num in nums:
        if is_even(num):
            res.append(num)
    return res
```

```
In [2135]: collect_evens([1, 2, 3, 4, 5])
#[2, 4]
```

```
Out[2135]: [2, 4]
```

## 5.7 Example: Remove Punctuations

### 5.7.1 String Module -Built-in functions

```
In [2136]: import string
```

```
In [2137]: string.ascii_letters
#'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
Out[2137]: 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
In [2138]: string.ascii_lowercase
#'abcdefghijklmnopqrstuvwxyz'
```

```
Out[2138]: 'abcdefghijklmnopqrstuvwxyz'
```

```
In [2139]: string.digits
#'0123456789'
```

```
Out[2139]: '0123456789'
```

```
In [2140]: string.punctuation
#!#$%&\'()*+,.-./;<=>?@[\\]^_`{|}~'
```

```
Out[2140]: '!#$%&\'()*+,.-./;<=>?@[\\]^_`{|}~'
```

## 5.7.2 Example: Remove Punctuations

Build a function RmPunct to remove punctuations from a string.

Example: RmPunct("He said, 'that is great!!") returns "He said that is great"

```
In [2141]:
```

```
import string
def RmPunct(s):
    res = ""
    punts = set(string.punctuation)
    for c in s:
        if c not in punts:           #如果不在punctuation中
            res += c                #就加到res中
    return res
```

```
In [2142]: RmPunct("He said, 'that is great!!")
#'He said that is great'
```

```
Out[2142]: 'He said that is great'
```

```
In [2143]: RmPunct("10001 He said, 'that is great!'")
#'10001 He said that is great'
```

```
Out[2143]: '10001 He said that is great'
```

## 5.8 Examples: Capitalize First Letters & Letter Palindrom

### 5.8.1 Code: Capitalize

```
In [2144]: #大写词: 首字母大写
"apple".capitalize()
#'Apple'
```

Out[2144]: 'Apple'

```
In [2145]: #大写句子: 首字母大写
>this is an apple".capitalize()
#'This is an apple'
```

Out[2145]: 'This is an apple'

```
In [2146]: #小写:
"APPLE".lower()
#'apple'
```

Out[2146]: 'apple'

```
In [2147]: "Apple".lower()
#'apple'
```

Out[2147]: 'apple'

### 5.8.2 Exercise 6.2.1: Capitalize First Letters

Build a function CapLett to capitalize the first letter of each word in a sentence. Example:  
CapLett("we are all friends") returns "We Are All Friends"

```
In [2148]: def CapLett(sen):
    res = ""
    for word in sen.split():
        if res:
            res += " "
        res += word.capitalize()
    return res
```

```
In [2149]: CapLett("we are all friends")
#'We Are All Friends'
```

Out[2149]: 'We Are All Friends'

### 5.8.3 Lower

```
In [2150]: "Techlent".lower()
# 'techlent'
```

```
Out[2150]: 'techlent'
```

```
In [2151]: "TECHLENT".lower()
# 'techlent'
```

```
Out[2151]: 'techlent'
```

### 5.8.4 Exercise 6.3.1: Letter Palindrome

Build a function LettPalind to determine a string is a palindrome or not. We only consider letters this time.

Example: LettPalind("909@..") returns True

Example: LettPalind("He is Sieh!") returns True

Example: LettPalind("His name is Sieh.") returns False

判断一个句子是否是回文

```
In [2152]: import string
def LettPalind(s):
    letts = set(string.ascii_lowercase)
    s = s.lower()
    left = 0
    right = len(s) - 1
    while left < right:
        if s[left] not in letts:
            left += 1
        elif s[right] not in letts:
            right -= 1
        elif s[left] == s[right]:
            left += 1
            right -= 1
        else:
            return False
    return True
```

```
In [2153]: LettPalind("He is Sieh!")
#True
```

```
Out[2153]: True
```

```
In [2154]: LettPalind("His name is Sieh!")
#False
```

```
Out[2154]: False
```

## Unit 6: Dictionary

### 6.1 Introduction to Dict

set也是花括号，但set是没有value的dictionary

```
In [2155]: a = {"a": 1, "b": 2, "c": 3}
print(a)
#{'a': 1, 'b': 2, 'c': 3}

{'a': 1, 'b': 2, 'c': 3}
```

#### 6.1.1 Get the value with key

调用dictionary的key

```
a["key"] = value
```

```
In [2156]: a["a"]
#1
```

```
Out[2156]: 1
```

```
In [2157]: a["b"]
#2
```

```
Out[2157]: 2
```

```
In [2158]: #a["d"]
```

```
# -----
# KeyError
# Cell In[176], line 1
# ----> 1 a["d"]

# KeyError: 'd'
```

*Traceback (most recent call last)*

#### 6.1.2 Keys have to be unique

重复命名的key，将采用最新一次赋值的value

```
In [2159]: a = {"a": 1, "a": 2, "b": 3}
```

```
In [2160]: print(a)
#{'a': 2, 'b': 3}

{'a': 2, 'b': 3}
```

### 6.1.3 Values don't have to be unique

values不用唯一值

可以把一个东西装在两个不同的盒子里面，但是不能一个盒子装两样东西

```
In [2161]: a = {"a": 1, "b": 1, "c": 2}
```

```
In [2162]: print(a)
#{'a': 1, 'b': 1, 'c': 2}

{'a': 1, 'b': 1, 'c': 2}
```

### 6.1.4 Check whether a key is in a dictionary

看元素是否在dictionary里：

```
print("key" in dict)
```

```
In [2163]: a = {"a": 1, "b": 2, "c": 3}
```

```
In [2164]: print("c" in a)
#True
```

True

```
In [2165]: print("d" in a)
#False
```

False

### 6.1.5 Keys can be either numbers or strings

`print(a[key])`: 返回key上的value

```
In [2166]: a = {2: "a", 1: "b", "c": 3}
```

```
In [2167]: #print(a[key]): 返回key上的value
print(a[2])
#a
```

a

```
In [2168]: #print(a[key]): 返回key上的value
print(a["c"])
#3
```

3

```
In [2169]: a = {2: "a", 1: "b", "c": 2}
```

```
In [2170]: print(a)
#{2: 'a', 1: 'b', 'c': 2}
{2: 'a', 1: 'b', 'c': 2}
```

## 6.2 Getting Keys & Values from Dict

### 6.2.1 Keys and values

```
In [2171]: a = {"a":1, "b":2, "c":3}
```

```
In [2172]: print(a)
#{'a': 1, 'b': 2, 'c': 3}
{'a': 1, 'b': 2, 'c': 3}
```

### 6.2.2 Get all the keys

a.keys() 返回dict\_keys(['a', 'b', 'c'])

list(a.keys()) 返回['a', 'b', 'c']

list(a) 返回['a', 'b', 'c']

set(a) 返回{'a', 'b', 'c'}

```
In [2173]: a.keys()
#dict_keys(['a', 'b', 'c'])
```

```
Out[2173]: dict_keys(['a', 'b', 'c'])
```

```
In [2174]: list(a.keys())
#[‘a’, ‘b’, ‘c’]
```

```
Out[2174]: [‘a’, ‘b’, ‘c’]
```

```
In [2175]: list(a)
#[‘a’, ‘b’, ‘c’]
```

```
Out[2175]: [‘a’, ‘b’, ‘c’]
```

```
In [2176]: set(a)
#{‘a’, ‘b’, ‘c’}
```

```
Out[2176]: {‘a’, ‘b’, ‘c’}
```

### 6.2.3 Get all the values

a.values()返回[1, 2, 3]

list(a.values())返回[1, 2, 3]

set(a.values())返回{1, 2, 3}

```
In [2177]: a = {"a": 1, "b": 2, "c": 3}
```

```
In [2178]: a.values()
#dict_values([1, 2, 3])
```

```
Out[2178]: dict_values([1, 2, 3])
```

```
In [2179]: list(a.values())
#[1, 2, 3]
```

```
Out[2179]: [1, 2, 3]
```

```
In [2180]: set(a.values())
#{1, 2, 3}
```

```
Out[2180]: {1, 2, 3}
```

### 6.2.4 Get key-value pairs

a.items()返回dict\_items([('a', 1), ('b', 2), ('c', 3)])

list(a.items())返回[('a', 1), ('b', 2), ('c', 3)]

sorted(a.items()), 根据key排序, 返回[('a', 1), ('b', 2), ('c', 3)]

sorted(a.items(), key = lambda x:x[-1], reverse = True), 根据value排序, 返回[('c', 3), ('b', 2), ('a', 1)]

```
In [2181]: a.items()
#dict_items([('a', 1), ('b', 2), ('c', 3)])
```

```
Out[2181]: dict_items([('a', 1), ('b', 2), ('c', 3)])
```

```
In [2182]: list(a.items())
#[('a', 1), ('b', 2), ('c', 3)]
```

```
Out[2182]: [('a', 1), ('b', 2), ('c', 3)]
```

```
In [2183]: sorted(a.items())
#[('a', 1), ('b', 2), ('c', 3)]
```

```
Out[2183]: [('a', 1), ('b', 2), ('c', 3)]
```

```
In [2184]: sorted(a.items(), key = lambda x:x[-1], reverse = True)
#[('c', 3), ('b', 2), ('a', 1)]
```

```
Out[2184]: [('c', 3), ('b', 2), ('a', 1)]
```

## 6.3 Exercise 2.3.1: Key Value Switch

key和value互换： Build a function kvSwitch to switch the keys and values of a dictionary.

Example: kvSwitch({"a": 1, "b": 2, "c": 3}) returns {1: "a", 2: "b", 3: "c"}

```
In [2185]: def kvSwitch(d):
    return {v:k for k,v in d.items()}
```

```
In [2186]: kvSwitch({"a": 1, "b": 2, "c": 3})
#{1: 'a', 2: 'b', 3: 'c'}
```

```
Out[2186]: {1: 'a', 2: 'b', 3: 'c'}
```

```
In [2187]: def kvSwitch(d):
    new_dict = {} #新字典为空
    for k, v in d.items(): #在字典key value pair中, for key, value
        new_dict[v] = k #新字典key = v, value = k
    return new_dict
```

```
In [2188]: kvSwitch({"a": 1, "b": 2, "c": 3})
#{1: 'a', 2: 'b', 3: 'c'}
```

```
Out[2188]: {1: 'a', 2: 'b', 3: 'c'}
```

In [2189]: #演示过程:

```
def kvSwitch(d):
    new_dict = {}
    for k, v in d.items():
        new_dict[v] = k
        print("k=", k)
        print("v=", v)
        print("new_dict = ", new_dict)
        print("==")
    return new_dict
```

In [2190]: kvSwitch({"a": 1, "b": 2, "c": 3})

```
k= a
v= 1
new_dict = {1: 'a'}
==
k= b
v= 2
new_dict = {1: 'a', 2: 'b'}
==
k= c
v= 3
new_dict = {1: 'a', 2: 'b', 3: 'c'}
==
```

Out[2190]: {1: 'a', 2: 'b', 3: 'c'}

## 6.4 Exercise 2.3.2: Rank Keys by Values

按照value大小，排key-value pair，用random forest里有Feature importance, 通过value sort keys; NLP根据重复次数，找到关键词

Build a function RankK which takes a dictionary as input returns a list of keys which sorted by their values. Example: RankK({"a": 7, "b": 5, "c": 9}) returns ["b", "a", "c"]

In [2191]: def RankK(d):

```
    items = sorted(d.items(), key = lambda x:x[-1])      #key = value
    res = []                                              #key返回到list中
    for item in items:
        res.append(item[0])                                #拿出key填到list中
    return res
```

In [2192]: RankK({"a": 7, "b": 5, "c": 9})  
#[‘b’, ‘a’, ‘c’]

Out[2192]: ['b', 'a', 'c']

```
In [2193]: def RankK(d):
    items = d.items()
    print("items=", items)
    sorted_items = sorted(items, key = lambda x:x[-1])
    print("sorted_items=", sorted_items)
    res = []
    for item in sorted_items:
        res.append(item[0])
    return res
```

```
In [2194]: RankK({"a": 7, "b": 5, "c": 9})
# items= dict_items([('a', 7), ('b', 5), ('c', 9)])
# sorted_items= [('b', 5), ('a', 7), ('c', 9)]
# ['b', 'a', 'c']

items= dict_items([('a', 7), ('b', 5), ('c', 9)])
sorted_items= [('b', 5), ('a', 7), ('c', 9)]
```

Out[2194]: ['b', 'a', 'c']

## 6.5 Zip

### 6.5.1 Exercise 2.3.2: Rank Keys by Values

Build a function RankK which takes a dictionary as input returns a list of keys which sorted by their values. Example: RankK({"a": 7, "b": 5, "c": 9}) returns ["b", "a", "c"]

```
In [2195]: def RankK(d):
    items = sorted(d.items(), key = lambda x:x[-1]) #items拿出, 根据value去排序
    return list(zip(*items))[0]
```

```
In [2196]: RankK({"a": 7, "b": 5, "c": 9})
#['b', 'a', 'c']
```

Out[2196]: ('b', 'a', 'c')

```
In [2197]: a = {"a":7, "b":5, "c":9}
```

```
In [2198]: print(a)
#{'a': 7, 'b': 5, 'c': 9}
```

{'a': 7, 'b': 5, 'c': 9}

```
In [2199]: items = sorted(a.items(), key = lambda x:x[-1])
```

```
In [2200]: print(items)
#[('b', 5), ('a', 7), ('c', 9)]
```

[('b', 5), ('a', 7), ('c', 9)]

```
In [2201]: # zip(*): 把里面的内容拆出来了
zip(*items)
#<zip at 0x22c68e4c700>
```

```
Out[2201]: <zip at 0x164ba8dfa00>
```

```
In [2202]: list(zip(*items))
#[('b', 'a', 'c'), (5, 7, 9)]
```

```
Out[2202]: [('b', 'a', 'c'), (5, 7, 9)]
```

```
In [2203]: #拿出zip(*items)中的key:
list(zip(*items))[0]
#('b', 'a', 'c')
```

```
Out[2203]: ('b', 'a', 'c')
```

```
In [2204]: a = [1, 2, 3]
b = [4, 5, 6]
c = list(zip(a, b))
```

```
In [2205]: print(c)
#[(1, 4), (2, 5), (3, 6)]
```

```
[(1, 4), (2, 5), (3, 6)]
```

```
In [2206]: list(zip(*c))
#[(1, 2, 3), (4, 5, 6)]
```

```
Out[2206]: [(1, 2, 3), (4, 5, 6)]
```

```
In [2207]: for i in zip(a, b):
    print(i)
# (1, 4)
# (2, 5)
# (3, 6)
```

```
(1, 4)
(2, 5)
(3, 6)
```

## 6.5.2 zip(\*)的实例

```
In [2208]: # a = [x, y, z]
# b = [θ, γ, Ω]
# ①c = [(x, θ), (y, γ), (z, Ω)]
# ②list(zip(*c)) = [(x, y, z), (θ, γ, Ω)]
# ③for i in zip(a, b):
#     print(i)
# (x, θ), (y, γ), (z, Ω)
```

## 6.6 Solving problems with dictionary

### 6.6.1 Example: 3.1 Character Frequency

属下每个字符出现的频率，算一段话，word出现了多少次（除了stop word等），出现越多的词，说明越重要。此处考虑字符频率。

Build a function ChaFreq to calculate the frequencies of characters in a string. Store the results in a dictionary. Example: ChaFreq("techlent") returns {'c': 1, 'e': 2, 'h': 1, 'l': 1, 'n': 1, 't': 2}

```
In [2209]: def ChaFreq(s):
    res = {}
    for c in s:
        if c in res:
            res[c] += 1
        else:
            res[c] = 1
    return res
```

```
In [2210]: ChaFreq("techlent")
#{'t': 2, 'e': 2, 'c': 1, 'h': 1, 'l': 1, 'n': 1}
```

```
Out[2210]: {'t': 2, 'e': 2, 'c': 1, 'h': 1, 'l': 1, 'n': 1}
```

```
In [2211]: def ChaFreq(s):
    res = {}
    for c in s:
        print("c =", c)
        if c in res:
            res[c] += 1
        else:
            res[c] = 1
        print("res", res)
    return res
```

In [2212]: `ChaFreq("techlent")`

```
c = t
res {'t': 1}
c = e
res {'t': 1, 'e': 1}
c = c
res {'t': 1, 'e': 1, 'c': 1}
c = h
res {'t': 1, 'e': 1, 'c': 1, 'h': 1}
c = l
res {'t': 1, 'e': 1, 'c': 1, 'h': 1, 'l': 1}
c = e
res {'t': 1, 'e': 2, 'c': 1, 'h': 1, 'l': 1}
c = n
res {'t': 1, 'e': 2, 'c': 1, 'h': 1, 'l': 1, 'n': 1}
c = t
res {'t': 2, 'e': 2, 'c': 1, 'h': 1, 'l': 1, 'n': 1}
```

Out[2212]: `{'t': 2, 'e': 2, 'c': 1, 'h': 1, 'l': 1, 'n': 1}`

## 6.7 Example: 3.2 Unique Only

写一个函数，看其中是否只包含独立的字符： Build a function UniqueOnly to detect whether a string only contains unique characters. Example: UniqueOnly("techlent") returns False  
Example: UniqueOnly("abcde") returns True

In [2213]: `def UniqueOnly(s):
 tmp = {}
 for c in s:
 if c in tmp:
 return False
 else:
 tmp[c] = 1
 return True`

In [2214]: `UniqueOnly("techlent")
#False`

Out[2214]: `False`

In [2215]: `UniqueOnly("abcde")
#True`

Out[2215]: `True`

```
In [2216]: def UniqueOnly(s):
    tmp = {}
    for c in s:
        print("c =", c)
        if c in tmp:
            return False
        else:
            tmp[c] = 1
    print("tmp = ", tmp)
    return True
```

```
In [2217]: UniqueOnly("techlent")
```

```
c = t
tmp = {'t': 1}
c = e
tmp = {'t': 1, 'e': 1}
c = c
tmp = {'t': 1, 'e': 1, 'c': 1}
c = h
tmp = {'t': 1, 'e': 1, 'c': 1, 'h': 1}
c = l
tmp = {'t': 1, 'e': 1, 'c': 1, 'h': 1, 'l': 1}
c = e
```

```
Out[2217]: False
```

```
In [2218]: #法2: [更简单]
# set是没有value的dict, 如有重复值, 则自动去掉
# 如果string长度 = set长度, 说明字符都是unique

def UniqueOnly(s):
    return len(s) == len(set(s))
```

```
In [2219]: UniqueOnly("techlent")
#False
```

```
Out[2219]: False
```

```
In [2220]: UniqueOnly("abcde")
#True
```

```
Out[2220]: True
```

## 6.8 Example: Same Pattern

Build a function SamePattern takes two strings as inputs and determines whether they are in the same pattern.

Example: SamePattern('egg', 'zoo') returns True

Example: SamePattern('abb', 'ab') returns False

Example: SamePattern('hello', 'agree') returns False

Example: SamePattern('aaa', 'abc') returns False

Example: SamePattern('abc', 'aaa') returns False

In [2221]: #用dict, 第一个单词, 每个字符做一个key, 第二个单词, 每个字符做一个value, 去对应:  
#如果g已经用过了, 发现对不上

```
def SamePattern(s1, s2):
    if len(s1) != len(s2):
        return False
    d1 = {}
    d2 = {}
    for i in range(len(s1)):
        if s1[i] in d1:
            if d1[s1[i]] != s2[i]:
                return False
        else:
            d1[s1[i]] = s2[i]

        if s2[i] in d2:
            if d2[s2[i]] != s1[i]:
                return False
        else:
            d2[s2[i]] = s1[i]
    return True
```

In [2222]: SamePattern('egg', 'zoo')
#True

Out[2222]: True

In [2223]: SamePattern('abb', 'ab')
#False

Out[2223]: False

In [2224]: SamePattern('abb', 'aba')
#False

Out[2224]: False

In [2225]: SamePattern('abb', 'bba')
#False

Out[2225]: False

```
In [2226]: def SamePattern(s1, s2):
    if len(s1) != len(s2):                      #string1和string 2长度不同, 直接pass
        return False
    else:
        d1 = {}
        d2 = {}

        for i in range(len(s1)):
            print("====")
            print("i=", i)
            print("s1[i]=", s1[i])
            print("s2[i]=", s2[i])
            print("d1=", d1)
            print("d2=", d2)
            if s1[i] in d1:
                if d1[s1[i]] != s2[i]:
                    return False
            else:
                d1[s1[i]] = s2[i]

            if s2[i] in d2:
                if d2[s2[i]] != s1[i]:
                    return False
            else:
                d2[s2[i]] = s1[i]
        print("new_d1=", d1)
        print("new_d2=", d2)
    return True
```

```
In [2227]: SamePattern('egg', 'zoo')
#i= 0
#先开始为空, 找到string1, string2第一个元素, 存入d1, d2
#d1中e对应z, d2中z对应e
#再找到string1, string2第2个元素, 存入d1, d2
#i = 1
#d1中g对应o, d2中o对应g
#再找到string1, string2第3个元素, 存入d1, d2
#d1中g对应o, d2中o对应g
#i = 2:
#此时s1[i]=g在d1中, 看在d1中g所对应的字符, 是否是现在的s2[i]即o; 因为相等, 所以不报错
#此时s2[i]=o在d2中, 看在d2中o对应的字符, 是否是现在的s1[i]即g; 因为相等, 所以不报错
```

```
=====
i= 0
s1[i]= e
s2[i]= z
d1= {}
d2= {}
new_d1= {'e': 'z'}
new_d2= {'z': 'e'}
=====
i= 1
s1[i]= g
s2[i]= o
d1= {'e': 'z'}
d2= {'z': 'e'}
new_d1= {'e': 'z', 'g': 'o'}
new_d2= {'z': 'e', 'o': 'g'}
=====
i= 2
s1[i]= g
.....
```

```
In [2228]: SamePattern('egt', 'zoo')
#i = 2
#s2[i] = o时, s1[i] = t了, 与'o':'g'中的g 对应不上, 则报错

=====
i= 0
s1[i]= e
s2[i]= z
d1= {}
d2= {}
new_d1= {'e': 'z'}
new_d2= {'z': 'e'}
=====
i= 1
s1[i]= g
s2[i]= o
d1= {'e': 'z'}
d2= {'z': 'e'}
new_d1= {'e': 'z', 'g': 'o'}
new_d2= {'z': 'e', 'o': 'g'}
=====
i= 2
s1[i]= t
s2[i]= o
d1= {'e': 'z', 'g': 'o'}
d2= {'z': 'e', 'o': 'g'}
```

Out[2228]: False

## 6.9 Example: Anagrams

字符出现的频率相同

Build a function `anagrams` to determine whether two given strings are anagrams.

Example: `anagrams('abc', 'cba')` returns True

Example: `anagrams('ab', 'abc')` returns False

Example: `anagrams('abccba', 'aabbcc')` returns False

Example: `anagrams('abccba', 'aabbcc')` returns True

```
In [2229]: def anagrams(s1, s2):
    d1 = {}
        #算s1中每个字符出现的频率，如果已有则+1，负责赋值为1
    for c in s1:
        if c in d1:
            d1[c] += 1
        else:
            d1[c] = 1
    for c in s2:
        #算s2字符出现频率时用减法
        if c in d1 and d1[c] > 0:
            d1[c] -= 1
        else:
            return False
    return True
```

```
In [2230]: anagrams('abc', 'cba')
#True
```

Out[2230]: True

```
In [2231]: anagrams('ab', 'abc')
#False
```

Out[2231]: False

```
In [2232]: anagrams('abccba', 'aabbbc')
#False
```

Out[2232]: False

```
In [2233]: anagrams('abccba', 'aabbcc')
#True
```

Out[2233]: True

```
In [2234]: def anagrams(s1, s2):
    d1 = {}

    for c in s1:
        print("c=", c)
        if c in d1:
            d1[c] += 1
        else:
            d1[c] = 1
        print("d1=", d1)

    print("==")

    for c in s2:
        print("c=", c)
        if c in d1 and d1[c] > 0:
            d1[c] -= 1
        else:
            return False
        print("d1=", d1)

    return True
```

```
In [2235]: anagrams('abccba', 'aabbcc')
```

```
c= a
d1= {'a': 1}
c= b
d1= {'a': 1, 'b': 1}
c= c
d1= {'a': 1, 'b': 1, 'c': 1}
c= c
d1= {'a': 1, 'b': 1, 'c': 2}
c= b
d1= {'a': 1, 'b': 2, 'c': 2}
c= a
d1= {'a': 2, 'b': 2, 'c': 2}
==
c= a
d1= {'a': 1, 'b': 2, 'c': 2}
c= a
d1= {'a': 0, 'b': 2, 'c': 2}
c= b
d1= {'a': 0, 'b': 1, 'c': 2}
c= b
d1= {'a': 0, 'b': 0, 'c': 2}
c= c
d1= {'a': 0, 'b': 0, 'c': 1}
c= c
d1= {'a': 0, 'b': 0, 'c': 0}
```

```
Out[2235]: True
```

In [2236]: `anagrams('abccba', 'aabbcce')`  
#当e在s2, 但没在s1中, 无法减时, 就报错了

```
c= a
d1= {'a': 1}
c= b
d1= {'a': 1, 'b': 1}
c= c
d1= {'a': 1, 'b': 1, 'c': 1}
c= c
d1= {'a': 1, 'b': 1, 'c': 2}
c= b
d1= {'a': 1, 'b': 2, 'c': 2}
c= a
d1= {'a': 2, 'b': 2, 'c': 2}
==
c= a
d1= {'a': 1, 'b': 2, 'c': 2}
c= a
d1= {'a': 0, 'b': 2, 'c': 2}
c= b
d1= {'a': 0, 'b': 1, 'c': 2}
c= b
d1= {'a': 0, 'b': 0, 'c': 2}
c= c
d1= {'a': 0, 'b': 0, 'c': 1}
c= c
d1= {'a': 0, 'b': 0, 'c': 0}
c= e
```

Out[2236]: False

In [2237]:

```
anagrams('abccba', 'aabbcca')
#当读取到s2最后一个a时, 因为前面都减完没有可减的了, 所以报错
```

```
c= a
d1= {'a': 1}
c= b
d1= {'a': 1, 'b': 1}
c= c
d1= {'a': 1, 'b': 1, 'c': 1}
c= c
d1= {'a': 1, 'b': 1, 'c': 2}
c= b
d1= {'a': 1, 'b': 2, 'c': 2}
c= a
d1= {'a': 2, 'b': 2, 'c': 2}
==
c= a
d1= {'a': 1, 'b': 2, 'c': 2}
c= a
d1= {'a': 0, 'b': 2, 'c': 2}
c= b
d1= {'a': 0, 'b': 1, 'c': 2}
c= b
d1= {'a': 0, 'b': 0, 'c': 2}
c= c
d1= {'a': 0, 'b': 0, 'c': 1}
c= c
d1= {'a': 0, 'b': 0, 'c': 0}
c= a
```

Out[2237]: False

In [2238]: #法2: 当sorted(s1) = sorted(s2)时, 则两个string, 字符出现频率相同

```
def anagrams(s1, s2):
    return sorted(s1) == sorted(s2)
```

In [2239]:

```
anagrams('abc', 'cba')
#True
```

Out[2239]: True

In [2240]:

```
anagrams('ab', 'abc')
#False
```

Out[2240]: False

## 6.10 Example: Two Sum

Leetcode第一题：判断给出的list中，有没有两个数相加等于target

Build a function TargetSum which takes a list and a target number as inputs. If there are two numbers in the list whose sum equals to the target number, the function returns the indice of the two numbers, otherwise returns -1.

Example: TargetDiff([1, 2, 5, 9, 8, 11], 6) returns 0, 2 [1+5=6]

Example: TargetDiff([1, 2, 5, 9, 8, 11], 11) returns 1, 3

Example: TargetDiff([1, 2, 5, 9, 8, 11], 5) returns -1

Example: TargetDiff([1, 2, 5, 9, 8, 11], 14) returns -1

In [2241]: #法1：暴力解法

#时间复杂度:  $n * n = n^2$

```
def TargetDiff(nums, target):
    for i in range(len(nums)):
        for j in range(i):
            print("i=", i)
            print("j=", j)
            if (nums[i] + nums[j]) == target:
                return j, i
    return -1
```

In [2242]: TargetDiff([1, 2, 5, 9, 8, 11], 6)

```
# i= 1
# j= 0
# i= 2
# j= 0
# (0, 2)
```

```
i= 1
j= 0
i= 2
j= 0
```

Out[2242]: (0, 2)

In [2243]: `TargetDiff([1, 2, 5, 9, 8, 11], 5)`

```
i= 1
j= 0
i= 2
j= 0
i= 2
j= 1
i= 3
j= 0
i= 3
j= 1
i= 3
j= 2
i= 4
j= 0
i= 4
j= 1
i= 4
j= 2
i= 4
j= 3
i= 5
j= 0
i= 5
j= 1
i= 5
j= 2
i= 5
j= 3
i= 5
j= 4
```

Out[2243]: -1

In [2244]: `def TargetDiff(nums, target):
 tmp = {}
 for i in range(len(nums)):
 print("tmp=", tmp)
 if nums[i] in tmp:
 return tmp[nums[i]], i
 else:
 tmp[target - nums[i]] = i
 return -1`

In [2245]: `#当target value - nums[i], 放入index中
# 当跑到第三个元素时, 5已经在dict中了, 因为之前6 - 1 = 5,
# 那么久把存入的5的index0, 和现在5的index2拿出来即可`

`TargetDiff([1, 2, 5, 9, 8, 11], 6)`

```
tmp= {}
tmp= {5: 0}
tmp= {5: 0, 4: 1}
```

Out[2245]: (0, 2)

In [2246]: `TargetDiff([1, 2, 5, 9, 8, 11], 11)`

```
tmp= {}
tmp= {10: 0}
tmp= {10: 0, 9: 1}
tmp= {10: 0, 9: 1, 6: 2}
```

Out[2246]: (1, 3)

In [2247]: `TargetDiff([1, 2, 5, 9, 8, 11], 5)`

```
tmp= {}
tmp= {4: 0}
tmp= {4: 0, 3: 1}
tmp= {4: 0, 3: 1, 0: 2}
tmp= {4: 0, 3: 1, 0: 2, -4: 3}
tmp= {4: 0, 3: 1, 0: 2, -4: 3, -3: 4}
```

Out[2247]: -1

## Unit 7 Tuple

### 7.1 Tuple

#### 7.1.1 What is this?

In [2248]: `def tmp_fun(num1, num2):  
 return num1, num2`

In [2249]: `tmp_fun(1, 2)  
#(1, 2)`

Out[2249]: (1, 2)

In [2250]: `#返回的是tuple, 不是返回list  
tmp_fun(1, 2) == [1, 2]  
#False`

Out[2250]: False

In [2251]: `#返回的是tuple  
tmp_fun(1, 2) == (1, 2)  
#True`

Out[2251]: True

```
In [2252]: tmp_fun("a", [1, 2])
#('a', [1, 2])
```

```
Out[2252]: ('a', [1, 2])
```

## 7.1.2 Defining a Tuple

```
In [2253]: a = (1, 2, 2, 3)
```

```
In [2254]: print(a)
#[1, 2, 2, 3]

(1, 2, 2, 3)
```

```
In [2255]: list(a)
#[1, 2, 2, 3]
```

```
Out[2255]: [1, 2, 2, 3]
```

```
In [2256]: set(a)
#{1, 2, 3}
```

```
Out[2256]: {1, 2, 3}
```

```
In [2257]: str(a)
#'(1, 2, 2, 3)'
```

```
Out[2257]: '(1, 2, 2, 3)'
```

## 7.1.3 Similarity with List

### 7.1.3.1 Getting the length

```
In [2258]: a = (1, 2, 2, 3)
```

```
In [2259]: #与list, set, string 得到长度的方式相同:
len(a)
#4
```

```
Out[2259]: 4
```

### 7.1.3.2 Getting element(s)

```
In [2260]: a = (1, 2, 2, 3)
```

In [2261]: a[0]  
#1

Out[2261]: 1

In [2262]: a[1]  
#2

Out[2262]: 2

In [2263]: a[-1]  
#3

Out[2263]: 3

In [2264]: a[1:]  
#(2, 2, 3)

Out[2264]: (2, 2, 3)

In [2265]: a[1:-1]  
#(2, 2)

Out[2265]: (2, 2)

### 7.1.3.3 Concatinating tuples

In [2266]: #与连接两个list的方式相同  
a = (1, 2, 2, 3)  
b = ("a", "b", "c", "d")  
a + b  
#(1, 2, 2, 3, 'a', 'b', 'c', 'd')

Out[2266]: (1, 2, 2, 3, 'a', 'b', 'c', 'd')

### 7.1.4 Difference from List

#### 7.1.4.1 Lack of many functions

In [2267]: #tuple不能append  
#a.append(4)

# -----  
# AttributeError  
# Cell In[68], line 1  
# ----> 1 a.append(4)

Traceback (most recent call last)

# AttributeError: 'tuple' object has no attribute 'append'

```
In [2268]: #tuple不能remove
#a.remove(1)

# -----
# AttributeError
# Cell In[70], line 1
# ----> 1 a.remove(1)

# AttributeError: 'tuple' object has no attribute 'remove'
```

```
In [2269]: #tuple不能sort
#(1, 2, 3).sort()

# -----
# AttributeError
# Cell In[72], line 1
# ----> 1 (1, 2, 3).sort()

# AttributeError: 'tuple' object has no attribute 'sort'
```

```
In [2270]: #tuple使用sorted()后变成list
sorted((3, 2, 3))
#[2, 3, 3]
```

Out[2270]: [2, 3, 3]

#### 7.1.4.2 Immutable

```
In [2271]: a = (1, 2, 2, 3)
```

```
In [2272]: #a[0] = 4

# -----
# TypeError
# Cell In[78], line 2
#      1 a = (1, 2, 2, 3)
# ----> 2 a[0] = 4

# TypeError: 'tuple' object does not support item assignment
```

```
In [2273]: #tuple可以放在set中
{(1, 2), (3, 4)}
#{(1, 2), (3, 4)}
```

Out[2273]: {(1, 2), (3, 4)}

```
In [2274]: #tuple可以放在set中是去重的
{(1, 2), (1, 2)}
# {(1, 2)}
```

Out[2274]: {(1, 2)}

In [2275]: #list不能放在tuple中

```
# {[1, 2], [3, 4]}

# -----
# TypeError
# Cell In[88], line 1
# ----> 1 {[1, 2], [3, 4]}

# TypeError: unhashable type: 'List'
```

Traceback (most recent call last)

In [2276]: #set不能放在tuple中

```
#set是没有value的dictionary
#{[1, 2], {3, 4}}
```

```
# -----
# TypeError
# Cell In[91], line 1
# ----> 1 {[1, 2], {3, 4}}
```

Traceback (most recent call last)

```
# TypeError: unhashable type: 'set'
```

In [2277]: #在dictionary中, tuple可以做key

```
{(1, 2):3, ("a", "b", "c"):"d"}
#{(1, 2): 3, ('a', 'b', 'c'): 'd'}
```

Out[2277]: {(1, 2): 3, ('a', 'b', 'c'): 'd'}

In [2278]: #在dictionary中, list不可以做key

```
#{[1, 2]:3, ["a", "b", "c"]:"d"}
```

```
# -----
# TypeError
# Cell In[97], line 1
# ----> 1 {[1, 2]:3, ["a", "b", "c"]:"d"}
```

Traceback (most recent call last)

```
# TypeError: unhashable type: 'List'
```

## 7.2 Format

### 7.2.1 Using format to output pretty strings

#### 7.2.2 Example 1:

Write a function which takes a name string as the input and outputs a self-introduction string  
“Hello, my name is {name}.”

```
In [2279]: #返回句子+变量
#return "{}".format(变量)
def introduction(name):
    return "Hello, my name is {}".format(name)
```

```
In [2280]: introduction("ICBer")
```

```
Out[2280]: 'Hello, my name is ICBer.'
```

### 7.2.3 Example 2:

Write a function which outputs a text string to remind about an appointment. The text string includes the name of the customer and the meeting time.

```
In [2281]: #按顺序多个变量放入句子中
#return "{} {} {}".format(变量1, 变量2, 变量3)
def appointment(customer_name, hour, minute):
    return "{} will meet you at {}:{}".format(customer_name, hour, minute)
```

```
In [2282]: appointment("ICBer", 12, 12)
```

```
Out[2282]: 'ICBer will meet you at 12:12.'
```

```
In [2283]: #在写报错Log中, 常用此种形式
```

### 7.2.4 Example 3:

Write a function which outputs division statement

```
In [2284]: #equals {} 返回一连串的数字
def divide(num1, num2):
    result = num1 / num2
    return "{} divided by {} equals {}".format(num1, num2, result)
```

```
In [2285]: divide(10, 2)
#'10 divided by 2 equals 5.0'
```

```
Out[2285]: '10 divided by 2 equals 5.0'
```

```
In [2286]: divide(7, 3)
#'7 divided by 3 equals 2.3333333333333335'
```

```
Out[2286]: '7 divided by 3 equals 2.3333333333333335'
```

```
In [2287]: #f表示float number, 2表示小数点后面, 只要2个两位数
>equals {:.2f} 返回保留两位小数
def divide_float(num1, num2):
    result = num1 / num2
    return "{} divided by {} equals {:.2f}".format(num1, num2, result)
```

```
In [2288]: divide_float(10, 2)
```

```
Out[2288]: '10 divided by 2 equals 5.00'
```

```
In [2289]: divide_float(7, 3)
```

```
Out[2289]: '7 divided by 3 equals 2.33'
```

```
In [2290]: def divide_float(num1, num2):
    result = num1 / num2
    return "{0} divided by {1} equals {2:.2f}".format(num1, num2, result)
```

```
In [2291]: divide_float(10, 2)
#'10 divided by 2 equals 5.00'
```

```
Out[2291]: '10 divided by 2 equals 5.00'
```

```
In [2292]: def divide_float(num1, num2):
    result = num1 / num2
    return "{2} divided by {1} equals {0:.2f}".format(num1, num2, result)
```

```
In [2293]: # {2}第三个元素 = 10/2 = 5
# {1}第二个元素 = 2
# {0}第一个元素 = 10

divide_float(10, 2)
#'5.0 divided by 2 equals 10.00'
```

```
Out[2293]: '5.0 divided by 2 equals 10.00'
```

```
In [2294]: def divide_float(num1, num2):
    result = num1 / num2
    return "{0} divided by {1} equals {2:.2f}. {0} is the number being divided
```

```
In [2295]: # {0}第一个元素 = 10
# {1}第二个元素 = 2
# {2}第三个元素 = 10/2 = 5

divide_float(10, 2)
#'10 divided by 2 equals 5.00. 10 is the number being divided.'
```

```
Out[2295]: '10 divided by 2 equals 5.00. 10 is the number being divided.'
```

## 7.3 Objects in Python

每一个数据结构，都是一个对象

```
In [2296]: #a是一个对象  
a = [1, 2, 3]
```

```
In [2297]: a  
#[1, 2, 3]
```

```
Out[2297]: [1, 2, 3]
```

```
In [2298]: len(a)  
#3
```

```
Out[2298]: 3
```

```
In [2299]: a.pop()  
#3
```

```
Out[2299]: 3
```

```
In [2300]: a  
#[1, 2]
```

```
Out[2300]: [1, 2]
```

```
In [2301]: import string
```

```
In [2302]: string.ascii_letters  
#'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
Out[2302]: 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

## 7.4 Define & Initialize a Class

### 7.4.1 Defining a class

In [2303]: #矩阵, 三维矢量

```
class Vector():
    """
    This class creates a 3 dimentional vector.
    """

    def __init__(self, elements):  #将(1,2,3)输入elements
        """
        Initialize the object.
        Input: a 3 element tuple
        """

        self.els = elements
        self.x = elements[0]
        self.y = elements[1]
        self.z = elements[2]

    def __repr__(self):          #represent, 全局变量: 变量本身被函数记住(self.)
        return "<{}, {}, {}>".format(self.x, self.y, self.z)

    def length(self):
        return (self.x ** 2 + self.y ** 2 + self.z ** 2) ** 0.5

    def add(self, another_vector):
        new_x = self.x + another_vector.x
        new_y = self.y + another_vector.y
        new_z = self.z + another_vector.z
        new_tuple = (new_x, new_y, new_z)
        return Vector(new_tuple)
```

### 7.4.2 Inititalize a object

In [2304]: v1 = Vector((1, 2, 3))

In [2305]: v1  
#<1, 2, 3>

Out[2305]: <1, 2, 3>

## 7.5 Self in Class

### 7.5.1 Defining a class

```
In [2306]: #如果没有self, 变量只在每个def下的函数里存在,
#当存在后调用其他函数def时, 跑完是不会被记的。
#如果有self, 可以在调用其他函数时, 使用被记住的变量

class Vector():
    """
    This class creates a 3 dimentional vector.
    """

    def __init__(self, elements):
        """
        Initialize the object.
        Input: a 3 element tuple
        """
        self.els = elements
        self.x = elements[0]
        self.y = elements[1]
        self.z = elements[2]

    def __repr__(self):
        return "<{}, {}, {}>".format(self.x, self.y, self.z)

    def length(self):
        return (self.x ** 2 + self.y ** 2 + self.z ** 2) ** 0.5

    def add(self, another_vector):
        new_x = self.x + another_vector.x
        new_y = self.y + another_vector.y
        new_z = self.z + another_vector.z
        new_tuple = (new_x, new_y, new_z)
        return Vector(new_tuple)
```

### 7.5.2 Initialize a object

```
In [2307]: v1 = Vector((1, 2, 3))
```

```
In [2308]: v1
#<1, 2, 3>
```

```
Out[2308]: <1, 2, 3>
```

### 7.5.3 Getting info from the object

```
In [2309]: #因为全局变量只叫了self.els, 没有elements, 所以没有记忆会报错
#v1.elements

# -----
# AttributeError
# Cell In[201], line 1
# ----> 1 v1.elements

# AttributeError: 'Vector' object has no attribute 'elements'
```

```
In [2310]: #定义过全局变量self.els = elements, 所以可以被叫到

v1.els
#(1, 2, 3)
```

```
Out[2310]: (1, 2, 3)
```

```
In [2311]: #定义过全局变量self.x = elements[0], 所以可以被叫到

v1.x
#1
```

```
Out[2311]: 1
```

```
In [2312]: v1.y
#2
```

```
Out[2312]: 2
```

```
In [2313]: v1.z
#3
```

```
Out[2313]: 3
```

```
In [2314]: v1.length()
#3.7416573867739413

#(self.x ** 2 + self.y ** 2 + self.z ** 2) ** 0.5
#(1^2 + 2^2 + 3^2)^2 = 3.74
```

```
Out[2314]: 3.7416573867739413
```

## 7.6 Calculations using a Custom Class

### 7.6.1 Defining a class¶

```
In [2315]: class Vector():
    """
        This class creates a 3 dimensional vector.
    """
    def __init__(self, elements):
        """
            Initialize the object.
            Input: a 3 element tuple
        """
        self.els = elements
        self.x = elements[0]
        self.y = elements[1]
        self.z = elements[2]

    def __repr__(self):
        return "<{}, {}, {}>".format(self.x, self.y, self.z)

    def length(self):
        return (self.x ** 2 + self.y ** 2 + self.z ** 2) ** 0.5

    def add(self, another_vector):
        new_x = self.x + another_vector.x
        new_y = self.y + another_vector.y
        new_z = self.z + another_vector.z
        new_tuple = (new_x, new_y, new_z)
        return Vector(new_tuple)
```

### 7.6.2 Initialize a object

```
In [2316]: v1 = Vector((1, 2, 3))
```

```
In [2317]: v1
#<1, 2, 3>
```

```
Out[2317]: <1, 2, 3>
```

### 7.6.3 Getting info from the object

```
In [2318]: #v1.elements  
  
# -----  
# AttributeError  
# Cell In[223], line 1  
# ----> 1 v1.elements  
  
# AttributeError: 'Vector' object has no attribute 'elements'
```

Traceback (most recent call last)

```
In [2319]: v1.els  
#(1, 2, 3)
```

Out[2319]: (1, 2, 3)

```
In [2320]: v1.x  
#1
```

Out[2320]: 1

```
In [2321]: v1.y  
#2
```

Out[2321]: 2

```
In [2322]: v1.z  
#3
```

Out[2322]: 3

```
In [2323]: v1.length()  
#3.7416573867739413
```

Out[2323]: 3.7416573867739413

### 7.6.4 Calculations with custom objects

```
In [2324]: v1 = Vector((1, 2, 3))
```

```
In [2325]: v1  
#<1, 2, 3>
```

Out[2325]: <1, 2, 3>

```
In [2326]: v2 = Vector((3, 4, 5))
```

In [2327]: v2  
#<3, 4, 5>

Out[2327]: <3, 4, 5>

In [2328]: v3 = v1.add(v2)

In [2329]: v3  
#<4, 6, 8>

Out[2329]: <4, 6, 8>

In [2330]: v3.els  
#(4, 6, 8)

Out[2330]: (4, 6, 8)

In [2331]: v3.length()  
#10.770329614269007

Out[2331]: 10.770329614269007

## Unit 8

### 8.1 Line Break

In [2332]: #自带module  
import string

In [2333]: #写一个string, 把字母小写打印  
string.ascii\_lowercase  
#'abcdefghijklmnopqrstuvwxyz'

Out[2333]: 'abcdefghijklmnopqrstuvwxyz'

In [2334]: ##写一个string, 把字母大写打印  
string.ascii\_uppercase  
#'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

Out[2334]: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

In [2335]: #小写, 大写, 顺序, 放在{}里  
txt = ""  
for i in range(len(string.ascii\_lowercase)):  
 lower = string.ascii\_lowercase[i]  
 upper = string.ascii\_uppercase[i]  
 txt += "{}{},{}".format(i, lower, upper)

In [2336]: #顺序1和A连起来了

```
print(txt)
```

```
0,a,A1,b,B2,c,C3,d,D4,e,E5,f,F6,g,G7,h,H8,i,I9,j,J10,k,K11,l,L12,m,M13,n,N14,
o,O15,p,P16,q,Q17,r,R18,s,S19,t,T20,u,U21,v,V22,w,W23,x,X24,y,Y25,z,Z
```

In [2337]: #使用\n断行符切断

```
txt = ""
for i in range(len(string.ascii_lowercase)):
    lower = string.ascii_lowercase[i]
    upper = string.ascii_uppercase[i]
    txt += "{}{},{}{}\n".format(i, lower, upper, upper)
```

In [2338]: print(txt)

```
0,a,A
1,b,B
2,c,C
3,d,D
4,e,E
5,f,F
6,g,G
7,h,H
8,i,I
9,j,J
10,k,K
11,l,L
12,m,M
13,n,N
14,o,O
15,p,P
16,q,Q
17,r,R
18,s,S
19,t,T
20,u,U
21,v,V
22,w,W
23,x,X
24,y,Y
25,z,Z
```

In [2339]: #看原始

```
txt
```

Out[2339]: '0,a,A\n1,b,B\n2,c,C\n3,d,D\n4,e,E\n5,f,F\n6,g,G\n7,h,H\n8,i,I\n9,j,J\n10,k,K
\n11,l,L\n12,m,M\n13,n,N\n14,o,O\n15,p,P\n16,q,Q\n17,r,R\n18,s,S\n19,t,T\n20,
u,U\n21,v,V\n22,w,W\n23,x,X\n24,y,Y\n25,z,Z\n'

## 8.2 Write Strings into A File

#打开，文件名，写入 as file

```
with open("letters.txt", "w") as f: f.write(txt)
```

```
In [2340]: import string
```

```
In [2341]: #将没有换行符的txt写进file
txt = ""
for i in range(len(string.ascii_lowercase)):
    lower = string.ascii_lowercase[i]
    upper = string.ascii_uppercase[i]
    txt += "{}{},{}{}".format(i, lower, upper)
```

```
In [2342]: print(txt)
#0,a,A1,b,B2,c,C3,d,D4,e,E5,f,F6,g,G7,h,H8,i,I9,j,J10,k,K11,l,L12,m,M13,n,N14,
◀          ▶
```

```
0,a,A1,b,B2,c,C3,d,D4,e,E5,f,F6,g,G7,h,H8,i,I9,j,J10,k,K11,l,L12,m,M13,n,N14,
o,O15,p,P16,q,Q17,r,R18,s,S19,t,T20,u,U21,v,V22,w,W23,x,X24,y,Y25,z,Z
```

```
In [2343]: with open("letters.txt", "w") as f:
    f.write(txt)
```

```
In [2344]: #将有换行符的txt写进file
txt = ""
for i in range(len(string.ascii_lowercase)):
    lower = string.ascii_lowercase[i]
    upper = string.ascii_uppercase[i]
    txt += "{}{},{}{}\n".format(i, lower, upper)
```

In [2345]: `print(txt)`

```
0,a,A  
1,b,B  
2,c,C  
3,d,D  
4,e,E  
5,f,F  
6,g,G  
7,h,H  
8,i,I  
9,j,J  
10,k,K  
11,l,L  
12,m,M  
13,n,N  
14,o,O  
15,p,P  
16,q,Q  
17,r,R  
18,s,S  
19,t,T  
20,u,U  
21,v,V  
22,w,W  
23,x,X  
24,y,Y  
25,z,Z
```

In [2346]: `with open("letters.txt", "w") as f:  
 f.write(txt)`

In [2347]: `## 一行一行写  
#0, a, A1, b, B2, c, C3  
with open("letters2.txt", "w") as f:  
 for i in range(len(string.ascii_lowercase)):  
 lower = string.ascii_lowercase[i]  
 upper = string.ascii_uppercase[i]  
 line = "{}, {}, {}".format(i, lower, upper)  
 f.write(line)`

In [2348]: `# 0, a, A  
# 1, b, B  
# 2, c, C  
with open("letters2.txt", "w") as f:  
 for i in range(len(string.ascii_lowercase)):  
 lower = string.ascii_lowercase[i]  
 upper = string.ascii_uppercase[i]  
 line = "{}, {}, {}\n".format(i, lower, upper)  
 f.write(line)`

## 8.3 Read From A File

```
with open("letters.txt", "r") as f:
```

```
    lines = f.readlines()
```

```
    for line in lines:
```

```
        print(line)
```

```
        print(line[:-1])
```

In [2349]: `import string`

In [2350]: `#分隔符\n要写清`

```
txt = ""
for i in range(len(string.ascii_lowercase)):
    lower = string.ascii_lowercase[i]
    upper = string.ascii_uppercase[i]
    txt += "{}, {}, {}{}\n".format(i, lower, upper, upper)
```

In [2351]: `#写入`

```
with open("letters.txt", "w") as f:
    f.write(txt)
```

In [2352]: `#读取filte: open("name.txt", "r"): 用分隔符读取`

```
with open("letters.txt", "r") as f:
    lines = f.readlines()
```

In [2353]: `#读取list`

```
print(lines)

['0, a, A\n', '1, b, B\n', '2, c, C\n', '3, d, D\n', '4, e, E\n', '5, f, F\n',
 '6, g, G\n', '7, h, H\n', '8, i, I\n', '9, j, J\n', '10, k, K\n', '11,
 l, L\n', '12, m, M\n', '13, n, N\n', '14, o, O\n', '15, p, P\n', '16, q, Q
 \n', '17, r, R\n', '18, s, S\n', '19, t, T\n', '20, u, U\n', '21, v, V\n', '2
 2, w, W\n', '23, x, X\n', '24, y, Y\n', '25, z, Z\n']
```

```
In [2354]: for line in lines:  
    print(line)
```

0, a, A

1, b, B

2, c, C

3, d, D

4, e, E

5, f, F

6, g, G

7, h, H

8, i, I

9, j, J

10, k, K

11, l, L

12, m, M

13, n, N

14, o, O

15, p, P

16, q, Q

17, r, R

18, s, S

19, t, T

20, u, U

21, v, V

22, w, W

23, x, X

24, y, Y

25, z, Z

In [2355]: #末尾分隔符去掉:

```
for line in lines:
    print(line[:-1])
```

```
0, a, A
1, b, B
2, c, C
3, d, D
4, e, E
5, f, F
6, g, G
7, h, H
8, i, I
9, j, J
10, k, K
11, l, L
12, m, M
13, n, N
14, o, O
15, p, P
16, q, Q
17, r, R
18, s, S
19, t, T
20, u, U
21, v, V
22, w, W
23, x, X
24, y, Y
25, z, Z
```

## 8.4 CSV

①with open("letters.csv", "w") as f:

```
f.write("rank, lower, upper\n")
```

②import pandas as pd

```
df = pd.read_csv("letters.csv")
```

In [2356]: import string

In [2357]: #把txt写进csv

```
with open("letters.csv", "w") as f:
    f.write("rank, lower, upper\n")
    for i in range(len(string.ascii_lowercase)):
        lower = string.ascii_lowercase[i]
        upper = string.ascii_uppercase[i]
        line = "{}, {}, {}\n".format(i, lower, upper)
        f.write(line)
```

```
In [2358]: #用pandas读csv变成date frame
import pandas as pd
df = pd.read_csv("letters.csv")
df
```

Out[2358]:

|    | rank | lower | upper |
|----|------|-------|-------|
| 0  | 0    | a     | A     |
| 1  | 1    | b     | B     |
| 2  | 2    | c     | C     |
| 3  | 3    | d     | D     |
| 4  | 4    | e     | E     |
| 5  | 5    | f     | F     |
| 6  | 6    | g     | G     |
| 7  | 7    | h     | H     |
| 8  | 8    | i     | I     |
| 9  | 9    | j     | J     |
| 10 | 10   | k     | K     |
| 11 | 11   | l     | L     |
| 12 | 12   | m     | M     |
| 13 | 13   | n     | N     |
| 14 | 14   | o     | O     |
| 15 | 15   | p     | P     |
| 16 | 16   | q     | Q     |
| 17 | 17   | r     | R     |
| 18 | 18   | s     | S     |
| 19 | 19   | t     | T     |
| 20 | 20   | u     | U     |
| 21 | 21   | v     | V     |
| 22 | 22   | w     | W     |
| 23 | 23   | x     | X     |
| 24 | 24   | y     | Y     |
| 25 | 25   | z     | Z     |