

目录

基础一、python 基础元素的使用	2
1.1 python2 还是 python3	2
1.2 python 字符串表示用单引号还是双引号？	2
1.3 python 注释的方式	2
1.4 字符串的格式的表示方法	3
1.5 真和假，True 和 False	4
1.6 python 四个主要的数据结构	4
1.6.1 列表 LIST	5
1.6.2 复制数据结构 copy	8
1.6.3 for 循环	8
1.6.4 字典	9
1.6.5 集合	10
1.6.6 元组	11
基础二、函数	13
2.1 函数的定义	13
2.1.1 创建一个函数	13
2.1.2 创建一个带参数的函数并注释	13
2.1.3 包含注释的函数	14
2.1.4 函数的默认值	15
2.1.5 查看函数和注释	15
2.2 同意文件夹下的文件调用 import	16
2.3 文件打包	16

基础一、python 基础元素的使用

1.1 python2 还是 python3

- (1) Python 3 才是 Python 的未来
- (2) Python 官方都建议直接学习 Python 3
- (3) Python 2 只维护到 2020 年

Python 3.0 在设计的时候没有考虑向下相容，

Python 2 有两种字符串类型：Unicode 字符串和非 Unicode 字符串。Python 3 只有一种类型：Unicode 字符串(Unicode strings)

Python 2	Python 3
<code>u' PapayaWhip'</code>	<code>' PapayaWhip'</code>

1.2 python 字符串表示用单引号还是双引号？

- (1) 字符串变量的定义，用双引号""和单引号'都行，**建议用单引号'**

```
pyVariable1="chenstr"
```

```
pyVariable2='chenstr'
```

- (2) 两个单引号的字符串中可以包含双引号，如果用两个双引号表示，则需要转译，**推荐单引号**

```
pyVariable1 = 'ch"e"n'
```

```
pyVariable1 = "ch\"e\"n"
```

- (3) **如果刚好要描述一个单引号，这个时候建议用双引号**，比起用单引号加转译符号的方法简单哈哈！

```
pyVariable1="''"
```

```
pyVariable2='\''
```

1.3 python 注释的方式

- (1) 注释一句语句，可以用 #

```
pyVariable1 = 1 #comment of Variable
```

- (2) 文档注释，也就是 docstring（文档字符串），建议用六个双引号"""

```
def chenfun()
```

```
"""function connect, for test"""
# comment of local variable
variable1='chen'
# comment of print
print(variable1)
```

- (3) 多行注释，六个双引号和六个单引号都可以，建议双引号（陈自己建议）

```
"""
Connect1
Connect1
Connect1
"""
```

```
'''
Connect2
Connect2
Connect2
'''
```

1.4 字符串的格式表示方法

- (1) 转换组合成字符串 **str()**函数

```
numf = 10.34
chenstr = 'chstr'
strout = 'num is: '+str(numf) + ' str is ' +chenstr
print(strout)

>>> numf = 10.34
>>> chenstr = 'chstr'
>>> strout = 'num is: '+str(numf) + ' str is ' +chenstr
>>> print(strout)
num is: 10.34 str is chstr
```

- (2) 用老方式的%语法格式

```
numf = 10.34
chenstr = 'chstr'
strout = 'num is: %2.2f str is %s' % (numf,chenstr)
print(strout)

>>> numf = 10.34
>>> chenstr = 'chstr'
>>> strout = 'num is: %2.2f str is %s' % (numf, chenstr)
>>> print(strout)
num is: 10.34 str is chstr
```

- (3) 用 format 方法，**推荐采用！**

```
numf = 10.34
chenstr = 'chstr'
strout = 'num is: {} str is {}'.format(numf,chenstr)
print(strout)
```

```
>>> numf = 10.34
>>> chenstr = 'chstr'
>>> strout = 'num is: {} str is {}'.format(numf, chenstr)
>>> print(strout)
num is: 10.34 str is chstr
```

1.5 真和假，True 和 False

注意 python 用 **True** 表示真，**False** 表示假。注意首字母的大小写，因为 true 和 false 会被理解成变量

```
>>> true
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    true
NameError: name 'true' is not defined
>>> True
True
```

True 对于 1，False 对应 0，操作验证如下

```
print(True)      # True
print(False)     # False
print(1)         # 1
print(0)         # 0
```

```
print(True==1)   # True
print(True==2)   # False 只有 1 是 true，其他值不是
print(True==0)   # False
print(False==0)  # True
print(False==2)  # False 只有 0 是 false，其他值不是
```

```
print(True>False) # True
print(True>0)     # True
print(False>0)    # False
print(False<1)    # True
print(False<0)    # False
```

1.6 python 四个主要的数据结构

- ①列表 List----有序的可变对象集合 [xx1, xx1, yy1, yy2]
- ②元组 Tuple----有序的不可变对象集合 (xx1, yy1, yy2)
- ③字典 Dict----无序的键/值对集合{xx1:yy1, xx2:yy2}
- ④集合 Set----无序的唯一对象集合 {xx1, yy1, yy2}

1.6.1 列表 LIST

有序的可变对象集合。类似于 C#和 java 中的 list，理解为可以动态扩容的数组

(C 语言)。

(1) 简单定义

```
chenlist1 = [1, 2, 3]
chenlist2 = [1, 'a', 2]
chenlist3 = ['c','a','d']
chenlist4 = []
```

(2) 类型测试 type():

```
Python 3.6.8 Shell
File Edit Shell Debug Options Window
Python 3.6.8 (tags/v3.6.8:3c6b436a57,
(AMD64)] on win32
Type "help", "copyright", "credits" or
>>> chenlist1 = [1, 2, 3]
>>> type(chenlist1)
<class 'list'>
>>>
```

(3) 输出测试 print()

```
>>> chenlist2 = [1, 'a', 2]
>>> print(chenlist2)
[1, 'a', 2]
```

(4) 扩展列表 append

```
chenlist2 = [1, 'a', 2]
chenlist2.append('c')
print(chenlist2)

>>> chenlist2 = [1, 'a', 2]
>>> chenlist2.append('c')
>>> print(chenlist2)
[1, 'a', 2, 'c']
>>> |
```

(5) 列表的长度 len()

```
chenlist3 = []
print(len(chenlist3))
chenlist3.append('1')
print(len(chenlist3))
```

```
>>> chenlist3 = [ ]
>>> print(len(chenlist3))
0
>>> chenlist3.append('1')
>>> print(len(chenlist3))
1
>>>
```

(6) 删除列表 remove

注意：remove('value')，是删除对应的列表的值

```
chenlist4 = ['a','b','c',2]
chenlist4.remove('b')
print(chenlist4)

>>> chenlist4 = ['a','b','c',2]
>>> chenlist4.remove('b')
>>> print(chenlist4)
['a', 'c', 2]
```

(7) 弹出数据 pop

注意：pop(index)，删除并返回索引值（数字）的所在的列表的值，默认为空则返回最后一项

```
chenlist4 = ['a','b','c',2]
chenlist4.pop()
print(chenlist4)
```

```
chenreturn = chenlist4.pop(1)
print(chenreturn)
print(chenlist4)
```

```
chenreturn = chenlist4.pop(0)
print(chenreturn)
print(chenlist4)
```

```
>>> chenlist4 = ['a','b','c',2]
>>> chenlist4.pop()
2
>>> print(chenlist4)
['a', 'b', 'c']
>>> chenreturn = chenlist4.pop(1)
>>> print(chenreturn)
b
>>> print(chenlist4)
['a', 'c']
>>> chenreturn=chenlist4.pop(0)
>>> print(chenreturn)
a
>>> print(chenlist4)
['c']
```

(8) 对象扩展 extend

extern([xx,yy])，扩展列表，将后面的列表合并到调用它的列表中去

```
chenlist4 = ['a','b','c',2]
```

```
chenlist4.extend(['d',1])
print(chenlist4)

chenlist4.extend('e')
print(chenlist4)

>>> chenlist4 = ['a','b','c',2]
>>> chenlist4.extend(['d',1])
>>> print(chenlist4)
['a', 'b', 'c', 2, 'd', 1]
>>> chenlist4.extend('e')
>>> print(chenlist4)
['a', 'b', 'c', 2, 'd', 1, 'e']
```

(9) 对象插入 insert

insert(index,'value'), 在索引位置（数字），插入特定的列表

```
chenlist4 = ['a','b','c',2]
chenlist4.insert(1,3)
print(chenlist4)

>>> chenlist4 = ['a','b','c',2]
>>> chenlist4.insert(1,3)
>>> print(chenlist4)
['a', 3, 'b', 'c', 2]
```

(10) 列表的切片

List[start:stop:step], 起始，结束和步长，这三个值都是可选的。

- ① 如没有指定起始值，则默认为 0
- ② 如没有指定结束值，则默认为列表最大值（最右边索引）
- ③ 如没有指定步长值，则默认为 1

```
chenlist5=[1,'a',2,'b',3,'c',4,'d',5,'e']
chenlist5[0:4]
chenlist5[0:9:3]
chenlist5[5:]
chenlist5[:10]
chenlist5[::2]

>>> chenlist5=[1,'a',2,'b',3,'c',4,'d',5,'e']
>>> chenlist5[0:4]
[1, 'a', 2, 'b']
>>> chenlist5[0:9:3]
[1, 'b', 4]
>>> chenlist5[5:]
['c', 4, 'd', 5, 'e']
>>> chenlist5[:10]
[1, 'a', 2, 'b', 3, 'c', 4, 'd', 5, 'e']
>>> chenlist5[::2]
[1, 2, 3, 4, 5]
```

(11) 字符串转为列表

list () 可以将字符串转换为列表

```
chenstr='chstr'
chenlist=list(chenstr)
print(chenlist)
```

```
>>> chenstr='chstr'
>>> chenlist=list(chenstr)
>>> print(chenlist)
['c', 'h', 's', 't', 'r']
```

(12) 列表转为字符串

“`.join()`”可以将列表转换为字符串

```
chenlist=['a','b'] #注意，列表中都要为字符串
''.join(chenlist)

>>> chenlist=['a','b']
>>> ''.join(chenlist)
'ab'
```

1.6.2 复制数据结构 copy

(1) 变量的直接复制相当于引用

```
chen = [1,2]
chennew = chen
chen[1]='a'
print(chen)
print(chennew)

>>> chen = [1,2]
>>> chennew = chen
>>> chen[1]='a'
>>> print(chen)
[1, 'a']
>>> print(chennew)
[1, 'a']
```

(13) 复制数据结构

复制数据结构的基本用法为 `a = b.copy()`

```
chen = [1,2]
chennew = chen.copy()
chen[1]='a'
print(chen)
print(chennew)

>>> chen = [1,2]
>>> chennew = chen.copy()
>>> chen[1]='a'
>>> print(chen)
[1, 'a']
>>> print(chennew)
[1, 2]
```

1.6.3 for 循环

(1) 基本用法

```
for a in b
for a not in b
```

(2) 循环输出


```
for i in range(3):  
    print(i)  
>>> for i in range(3):  
        print(i)
```

```
0  
1  
2
```

(3) 输出列表中的数据

```
chenlist = ['a',1,2,'b']  
for i in chenlist:  
    print(chenlist)  
>>> chenlist=['a',1,2,'b']  
>>> for i in chenlist:  
        print(i)
```

```
a  
1  
2  
b
```

(4) 输出字符串中的数据

```
chenstr = 'string of chen'  
for ch in chenstr:  
    print(ch)  
>>> chenstr = 'string of chen'  
>>> for ch in chenstr:  
        print(ch)
```

```
s  
t  
r  
i  
n  
g  
  
o  
f  
  
c  
h  
e  
n
```

1.6.4 字典

List [值 1, 值 2, 值 3]

(1) 基本存储方式，{键 1: 值, 键 2: 值}，存储时没有顺序，字典查找速度很快

(2) 基本定义的例子

```
chendict = {'Name':'chen1', 'School':'Ch University','Age':20,'phone':13511118888}
```

```
File Edit Format Run Options Window Help
chendict = {'Name':'chen1', 'School':'Ch University','Age':20,'phone':13511118888}
print(chendict)

{'Name': 'chen1', 'School': 'Ch University', 'Age': 20, 'phone': 13511118888}
>>> |
```

(3) 字典的循环输出

for k,v in chendict.items() , 注意大家习惯上用 k,v 表示键和取值，别忘了字典后面加.items()

```
chendict = {'Name':'chen1', 'School':'Ch University','Age':20,'phone':13511118888}
for k,v in chendict.items():
    #print('key is:',k,'value is:',v)
    print('key is:{} value is:{}'.format(k,v))
key is: Name value is: chen1
key is: School value is: Ch University
key is: Age value is: 20
key is: phone value is: 13511118888
```

1.6.5 集合

List [值 1,值 2,值 3]
Dict{键 1:值, 键 2:值}, 存储时没有顺序，字典查找速度很快
Set{xx,yy,zz} 无序的!

集合不允许有重复，如果主要操作是查找，集合就比使用相应的列表要快得多。集合是
无序的! 基本定义{xx,yy,zz}

(1) 基本定义例子

```
chenset={'a',1,'b',2,'c',3}

>>> chenset={'a', 1, 'b', 2, 'c', 3}
>>> chenset
{1, 2, 3, 'a', 'b', 'c'}
```

(2) 字符串转换为集合

基本函数为 set('str')

```
chenset = set('chenchniu')
print(chenset)

>>> chenset = set('chenchniu')
>>> print(chenset)
{'u', 'e', 'c', 'h', 'i', 'n'}
```

(3) 集合的排序输出

基本用法是 sorted(iterable, key=None, reverse=False)

1) 正序输出


```
chenset = set('chenchniu')
print(chenset)
for i in sorted(chenset):
    print(i)

>>> chenset = set('chenchniu')
>>> print(chenset)
{'u', 'e', 'c', 'h', 'i', 'n'}
>>> for i in sorted(chenset):
    print(i)
```

```
c
e
h
i
n
u
```

2) 逆序输出

```
chenset = set('chenchniu')
print(chenset)
for i in sorted(chenset,reverse=True):
    print(i)
```

 1.5.5.py - E:/pythonproject/shangke/1.5.5.py (3.6.8)

File Edit Format Run Options Window Help

```
chenset = set('chenchniu')
print(chenset)
for i in sorted(chenset,reverse=True):
    print(i)
```

```
===== RESTART: E:/pythonproject/sr
{'u', 'i', 'n', 'e', 'h', 'c'}
u
n
i
h
e
c
>>>
```

1.6.6 元组

List [值 1,值 2,值 3] 有序的
Dict {键 1:值, 键 2:值}, 存储时没有顺序, 字典查找速度很快
Set {xx,yy,zz} 无序的!
tuple (xx1, yy1, yy2), 有序的一旦定义就不能修改!

元组 tuple 可以理解为常量, 也就是有序的不可变对象集合, 定义形式为 (xx1, yy1, yy2)。
元组使用前必须定义, 一旦定义就不能修改!

(1) 简单的定义

```
chentp = (1,'a',2,3)
```

```
print(chentp)
type(chentp)
print(chentp[1])

>>> chentp = (1, 'a', 2, 3)
>>> print(chentp)
(1, 'a', 2, 3)
>>> type(chentp)
<class 'tuple'>
>>> print(chentp[1])
a
```

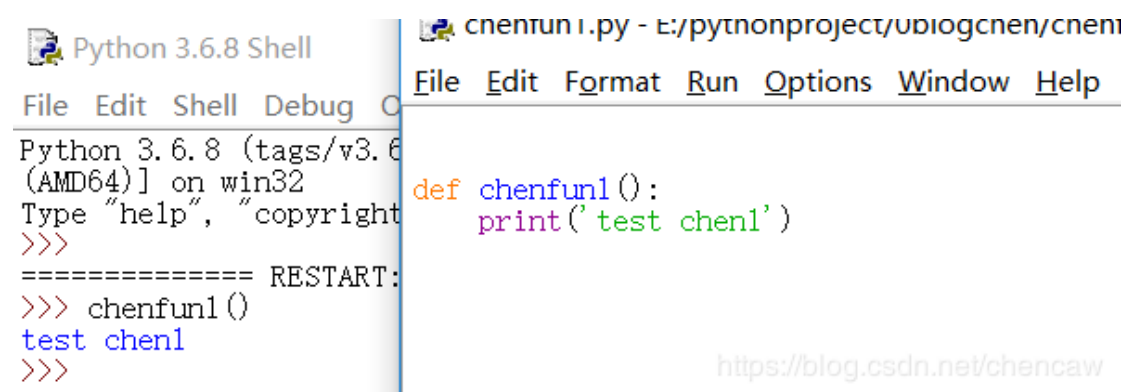
基础二、函数

2.1 函数的定义

2.1.1 创建一个函数

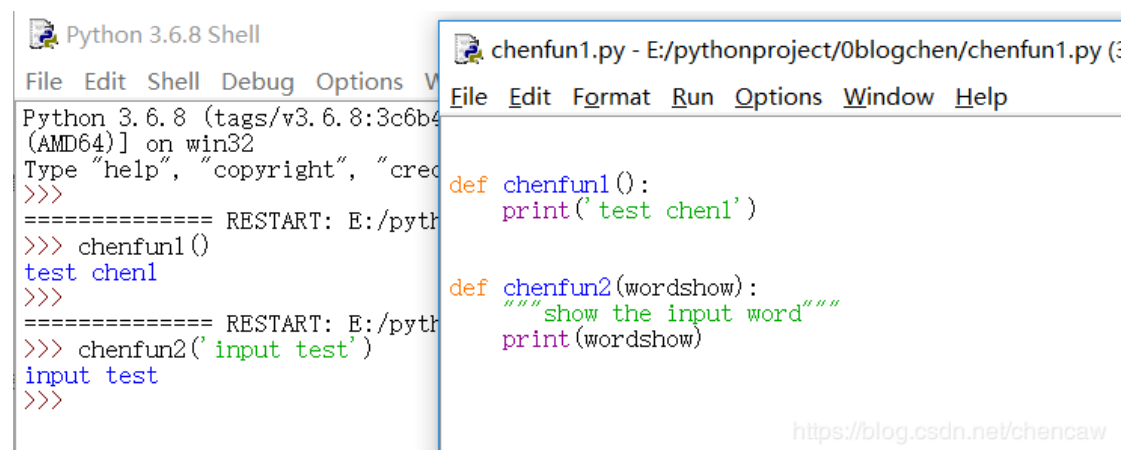
```
1
2
3 def chenfun1():
4     print('test chen1')
```

注意：记得函数的开头空两格！



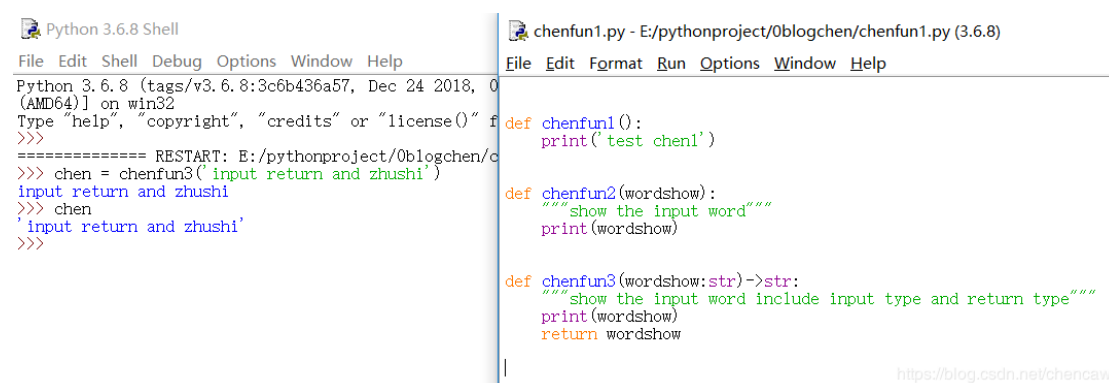
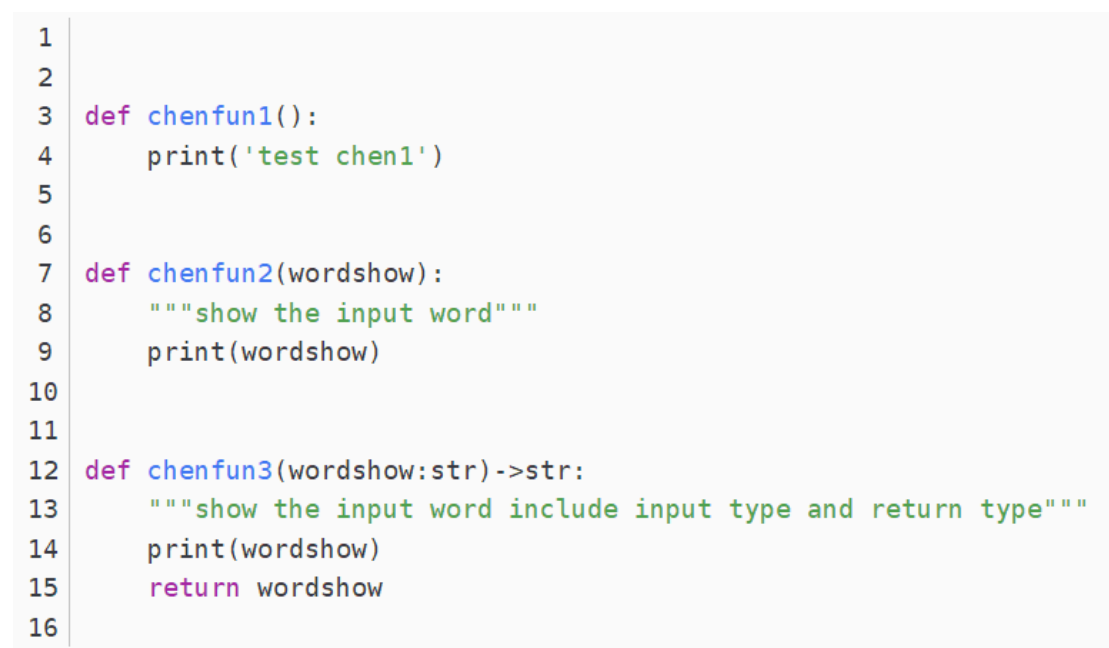
2.1.2 创建一个带参数的函数并注释

```
1
2
3 def chenfun1():
4     print('test chen1')
5
6
7 def chenfun2(wordshow):
8     """show the input word"""
9     print(wordshow)
10
```



注意：(1) 注释整段文字，docstring，建议用 6 个双引号"""XXX"""！(2) 变量字符串建议用两个单引号' (3) 如果是\则用双引号\"

2.1.3 包含注释的函数



注意：（1）参数注释是在参数后面加了:和类型（2）返回注释是函数最后加了->和类型；

（3）参数注释是给你和你的同事看的

```
def chenfun3(wordshow:str)->str:
```

2.1.4 函数的默认值

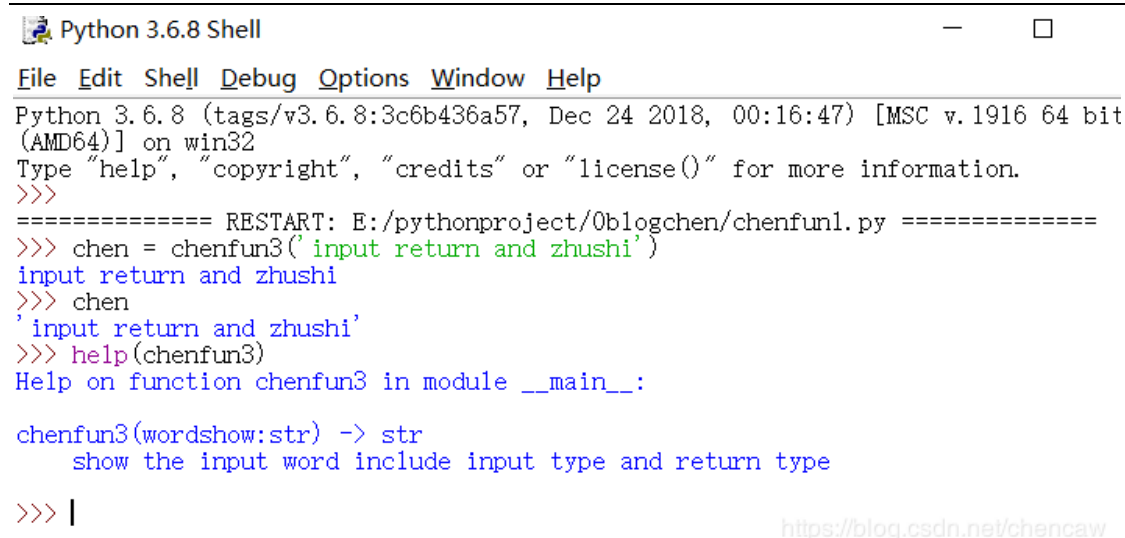
```
def chenfun1():  
    print('test chen1')  
  
def chenfun2(wordshow):  
    """show the input word"""  
    print(wordshow)  
  
def chenfun3(wordshow:str)->str:  
    """show the input word include input type and return type"""  
    print(wordshow)  
    return wordshow  
  
def chenfun4(wordshow:str='chendefault')->str:  
    """show the input word include input type and return type"""  
    print(wordshow)  
    return wordshow
```

此时如果输入为空的话就调用默认值，如果有输入的话，就是输入

```
>>> chenfun4()  
chendefault  
'chendefault'  
>>> chenfun4('shuru')  
shuru  
'shuru'
```

2.1.5 查看函数和注释

在 IDE 中输入 help(函数名)就可以



```
Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/pythonproject/0blogchen/chenfun1.py =====
>>> chen = chenfun3('input return and zhushi')
input return and zhushi
>>> chen
'input return and zhushi'
>>> help(chenfun3)
Help on function chenfun3 in module __main__:

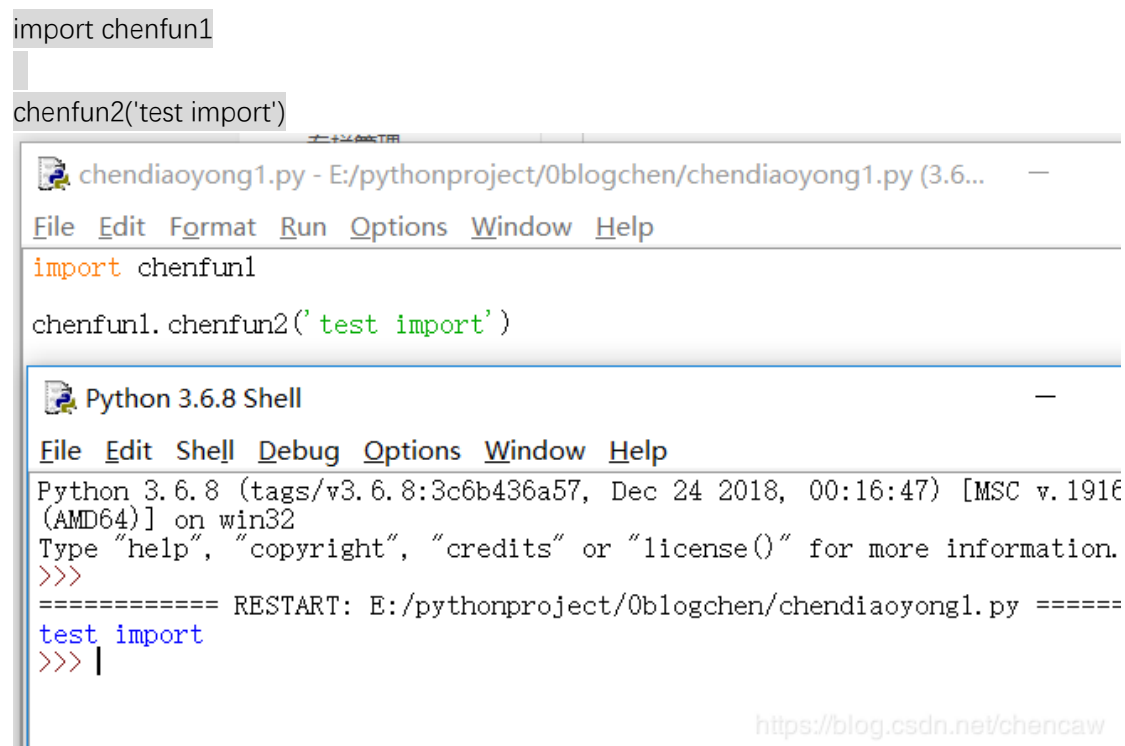
chenfun3(wordshow:str) -> str
    show the input word include input type and return type

>>> |
```

<https://blog.csdn.net/chencaw>

2.2 同一文件夹下的文件调用 import

1、在同一目录下，直接采用 import 文件名就可以调用我们的文件模块



```
import chenfun1

chenfun2('test import')
```

chendiaoyong1.py - E:/pythonproject/0blogchen/chendiaoyong1.py (3.6... —

```
File Edit Format Run Options Window Help
import chenfun1
chenfun1.chenfun2('test import')
```


```
Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/pythonproject/0blogchen/chendiaoyong1.py =====
test import
>>> |
```



<https://blog.csdn.net/chencaw>

2.3 文件打包

如果要整个 python 都能调用，则需要将其打包到 site-packages 中去

- (1) 新建一个文件夹，比如 chenpackge
- (2) 将基础 (1) 教程中的 chenfun1.py 拷贝到该目录下
- (3) 在该文件夹下新建一个空的文件，README.txt







电脑 > 本地磁盘 (E:) > pythonproject > 0blogchen > chenpackge		
名称	修改日期	类型
 chenfun1.py	2019/3/29 8:09	Python Fi
 README.txt	2019/3/29 8:23	文本文档

- (4) 在该文件夹下创建一个新的文件 setup.py，内容如下

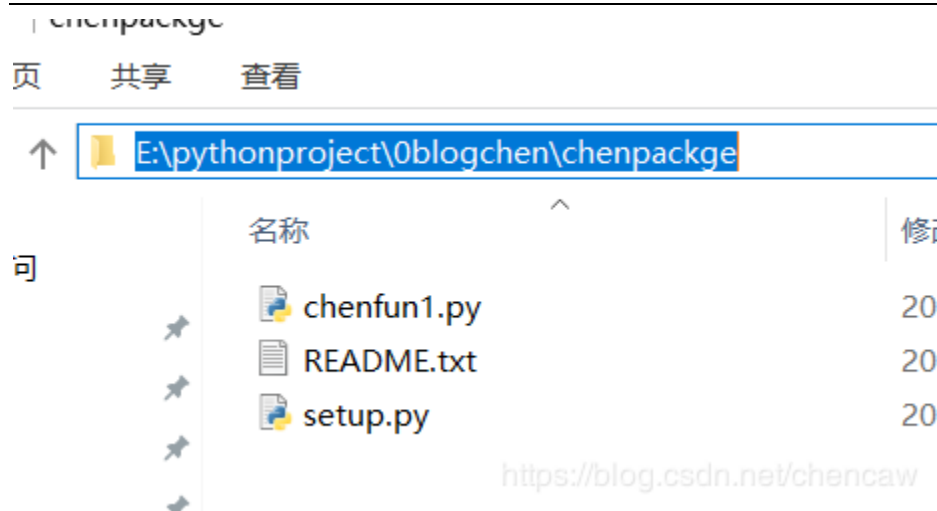
```
from setuptools import setup

setup(
    name = 'chenfun1',
    version = '1.0',
    description = 'Chen Tools Model',
    author = 'Chen',
    author_email = 'Chen@chen.com',
    url = 'chen.com',
    py_modules = ['chenfun1'],
)
```

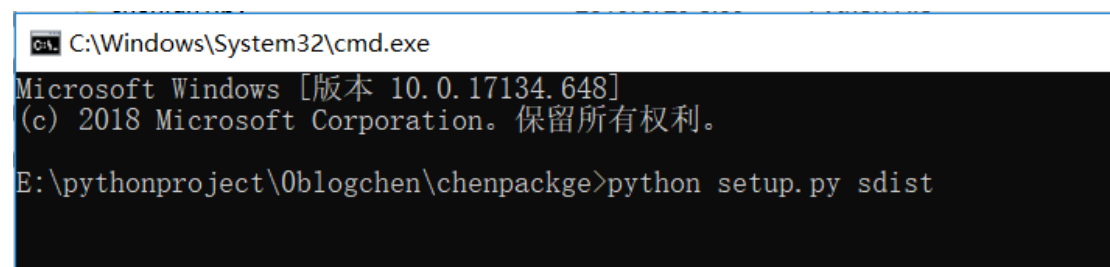


名称
 chenfun1.py
 README.txt
 setup.py

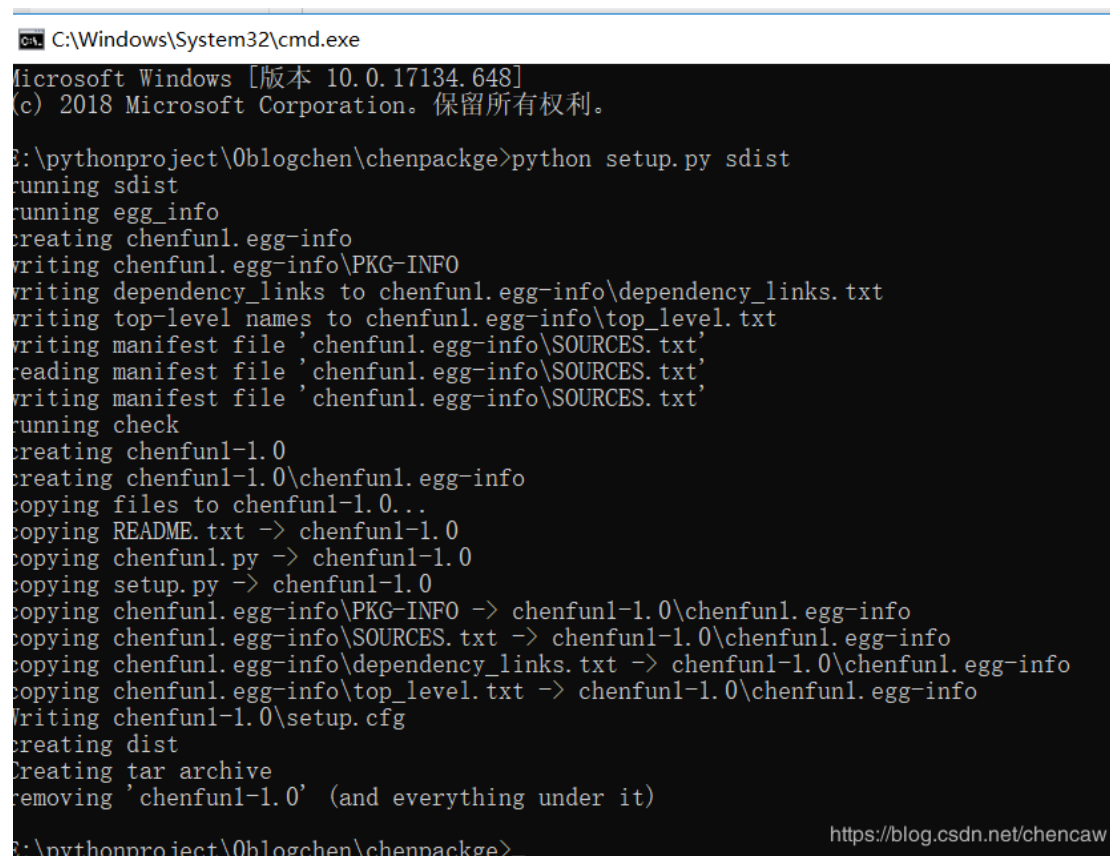
- (5) 在当前目录的路径中输入 cmd



(6) 输入 `python setup.py sdist`



(7) 编译结果如下



(8) 在该目录的 dist 文件夹下看到了打包后的压缩文件



(9) 进入该目录，然后使用 pip 安装，pip install chenfun1-1.0.tar.gz

```
E:\pythonproject\0blogchen\chenpackage>cd dist
E:\pythonproject\0blogchen\chenpackage\dist>pip install chenfun1-1.0.tar.gz

E:\pythonproject\0blogchen\chenpackage\dist>pip install chenfun1-1.0.tar.gz
Processing e:\pythonproject\0blogchen\chenpackage\dist\chenfun1-1.0.tar.gz
Building wheels for collected packages: chenfun1
  Building wheel for chenfun1 (setup.py) ... done
  Stored in directory: C:\Users\LENOVO\AppData\Local\pip\Cache\wheels\1d\53\df\63290f0579080cc94816504f81e7363a94f42f77d511e6acc5
Successfully built chenfun1
Installing collected packages: chenfun1
Successfully installed chenfun1-1.0
https://blog.csdn.net/chencaw
```

(10) 随便在其它任何路径下编写一个文件测试下引用

```
In [1]: import chenfun1

In [2]: chenfun1.chenfun1()

test chen1

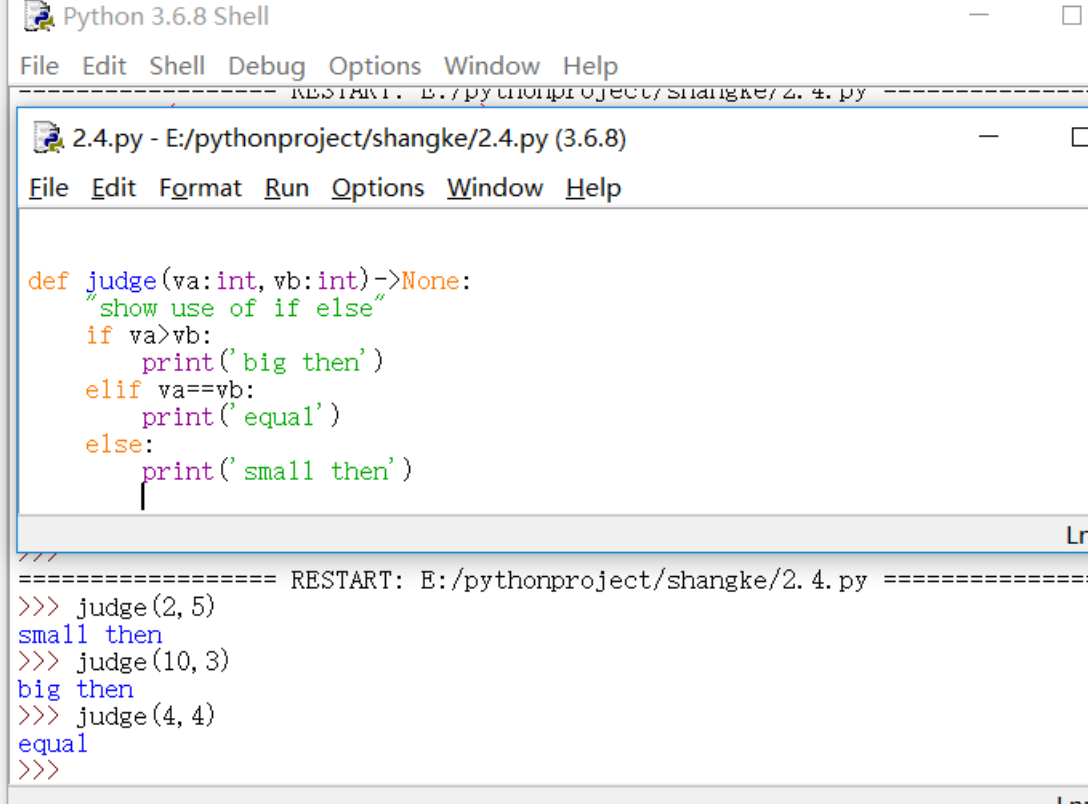
In [ ]:
```

2.4 if 和 else 和 elif

(1) 程序示例

```
def judge(va:int,vb:int)->None:
    "show use of if else"
    if va>vb:
```

```
        print('big then')
    elif va==vb:
        print('equal')
    else:
        print('small then')
```



```
Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
----- RESTART: E:/pythonproject/shangke/2.4.py -----
2.4.py - E:/pythonproject/shangke/2.4.py (3.6.8)
File Edit Format Run Options Window Help

def judge(va:int, vb:int)->None:
    "show use of if else"
    if va>vb:
        print('big then')
    elif va==vb:
        print('equal')
    else:
        print('small then')

===== RESTART: E:/pythonproject/shangke/2.4.py =====
>>> judge(2, 5)
small then
>>> judge(10, 3)
big then
>>> judge(4, 4)
equal
>>>
```