# 2018.8.11

### 转义字符”\”

列：

Print( ‘Let\’s go’ )

解：

斜杠后转换成字符串

### 换行显示：

在print中”\n”可以换行显示

### 取消斜杠转义功能

Print r”c:/a/b”

### 保留换行显示，可以注释

“””

Aaaaa

bbbbb

“””

# 2018.8.12

### 加括号会优先执行

(20 > 4) and 5 < 2

### 数字类型

#### 关于平方数

1.5e4 = 1.5E4 = 15000.0

如果使用e记法, ”整数类型” 会变成 ”浮点数类型“。

#### 关于布尔类型的小细节

布尔类型可以计算

True + False = 1

因为，True = 1 ，False = 0

#### 数据类型判断

type()可以显示数据类型,不过不推荐使用

isinstance() 返回布尔值

a = 5

isinstance(a,int) #如果a符号要求的数据类型,则返回"True"否则返回"False"

# 2018.8.13

### 计算符

A = 2

A = /= 3 相当于 A = 2 / 3

加减乘除类似

#### % ，\*\* ，//

5 % 2 结果取余数

5 \*\* 2 表示5的二次方

5 // 2 结果保留整数

### 优先级



### 3元操作符

x = 1  
y = 2  
a = x if x > y else y #如果条件: 为"真"则返回x否则返回y  
print (a)

### 自爆代码

assert 3>5 #如果条件为假则,程序主动崩溃

# 2018.8.15

### 计算参数的长度

each = '123456'

print (each , len(each)) #len显示参数长度

### range（）生成数字序列

常用于for中

For i in range(1 , 5): #生成从1开始5结束( 不包括5 )

List( range(1 , 10 , 2) ) #生成列表，每隔2个单位生成一个数字

### 关于列表

#### 增加元素

test = [ 'A','B',12,21.2,'c' ]

test.append('D') #增加单个元素

test.extend(['d','e']) #增加多个元素

test.insert(0,'S') #自定义位置注入元素

#### 删除元素

test = ['A', 'B', 12, 21.2, 'c']

test.remove('c') # 必须知列表元素值才能删除

del test[0] # 删除元素

cut = test.pop(1) # 取出元素

#### 复制列表方法

test = ['A', 'B', 12, 21.2, 'c']

test2 = test[ : ] # 全复制

test3 = test[ 0:3 ] # 切片复制

#### 查看列表元素方法

list = [1,2,3,4,5,6,1,1,3,3,44,5]

list.count(1) # 查看列表中相同元素数量

list.index(44) # 查看元素在列表中的序列

#### 列表排序方法

list.reverse() # 列表序列元素翻转

list.sort() # 从小到大排序

list.sort(reverse=True) # 先排序后翻转( sort可以添加一些参数来进行排序 )

#### 列表的正确复制方法

List = [ 1,2,3,4 ]

List2 = list[ : ]

不能通过赋值的方式复制列表, 那只是增加一个指引而已, 并没有真正的复制。

# 208.8.19

错误：count() 计算列表函数 ， sort()是从小到大排序函数



### 请用一句话描述什么是列表？再用一句话描述什么是元组？

列表可包含各种类型数据并允许进行修改，元组被创建后则相当于无法修改的列表

元组可以使用切片功能 如: temp1 = (1,2,3,4) temp2 =temp1[:2]

#### 元组创建单个元素

Temp = （1，）关键点在逗号

### 字符串的小秘密

# 索引字符串数据

A = 'abcdefg'

print ( A[0], A[3] ) # 通过这种方法直接可以索引出字符串数据

print ( type( A[0]))

字符串可以使用切片功能

A = 'abcdefg'

B = A[ :2 ]

# 2018.8.20

## 字符串方法大全

#### # 字符串方法,位置更替format()

a = "{0} love {1}".format("I" , "you")

b = "{a} love {b}".format(a="I" , b = "you")

print (a,b)

# # 字符串格式修改函数  
  
 # 字符串开头字母大写  
 a1 = 'zhao teng'.capitalize()  
 print (a1)  
  
 # 字符串全部小写  
 a2 = 'Zhao'.casefold()  
 print (a2)  
  
 # 字符串剧中  
 a3 = 'a'.center(15)  
 print (a3)  
  
 # 设置tab空格大小  
 a6 = 's\tpace'.expandtabs(tabsize=4)  
 print(a6)  
  
 # 以字符串作为分隔符，插入到111中所有的字符之间  
 a19 = 'oo7'.join('111')  
 print (a19)  
  
 # 大写转小写字母  
 a20 = 'oosdASD'.lower()  
 print (a20)  
  
 # 去掉字符串左边空格  
 a21 = ' oo77 '.lstrip()  
 print (a21)  
  
 # 替换(原内容,新内容,替换次数)  
 a22 = 'abcdeasdfasdcas'.replace('a','1',3)  
 print (a22)  
  
 # 去掉末尾空格  
 a24 = 'oo7a '.rstrip()  
 print (a24)  
  
 # 查看是否以指定字符开头(可自定义开始位置)  
 a25 = 'zhaosdsd'.startswith('zhao',0,4)  
 print (a25)  
  
 # 删除前后空格  
 a26 = ' asdasdaadss '.strip()  
 print (a26)  
  
 # 大小写翻转  
 a27 = 'asdsASD'.swapcase()  
 print (a27)  
  
 # 强制开头大写其他小写  
 a28 = 'zhao'.title()  
 print (a28)  
  
 # 所有小写转化为大写  
 a29 = 'asdasd'.upper()  
 print (a29)  
  
  
# # 部分字符串内容判断函数  
  
 # 统计指定字符串数量(可自定义位置),然后返回数量  
 a4 = 'jkhaksdnbfkbnavnboabdofe'.count('a',0,4)  
 print (a4)  
  
 # 判断是否以指定字符串结尾(可自定义位置),是则返回True否则返回False  
 a5 = 'asdasoo7'  
 test = a5.endswith('oo7')  
 print (test)  
  
 # 查看字符串中是否有指定字符串,有则返回索引值,否则返回-1  
 a7 = 'asdasdasdzhaoasfloo7asds'.find('oo7')  
 print (a7)  
  
 # 与find功能一样,从右开始查找  
 a23 = 'zhaoteng'.rfind('en')  
 print(a23)  
  
# # 整体字符串类型判断函数  
  
 # 如果字符串只有字母数字则返回True  
 a8 = 'asdasdasd'.isalnum()  
 a9 = '123123123'.isalnum()  
 a10 = 'asdas/23123asd'.isalnum()  
 print (a8,a9,a10)  
  
 # 如果字符串全部是字母则返回True  
 a11 = 'aaa22'.isalpha()  
 print (a11)  
  
 # 如果字符串全为10进制则返回True  
 a12 = '123123'.isdecimal()  
 print (a12)  
  
 # 如果字符串全为数字则返回True  
 a13 = '12312a333'.isdigit()  
 print (a13)  
  
 # 如果字符串全为小写字母则返回True  
 a14 = 'asds'.islower()  
 print (a14)  
  
 # 如果字符串全为数字则返回True  
 a15 = '12312a333'.isnumeric()  
 print (a15)  
  
 # 如果字符串全为空格则返回True  
 a16 = ' '.isspace()  
 print (a16)  
  
 # 如果字符串开头大写,其余小写则为True  
 a17 = 'Aasd'.istitle()  
 print (a17)  
  
 # 如果字符串字母全部为大写则返回True  
 a18 = 'ASDDa'.isupper()  
 print (a18)

### 格式化方法

其实就是给数据划一个“标准的规范”用各种函数制定规范

# 转化为Ascall码  
 a = '%c %c' % (97,98)  
 print (a)  
  
 # 格式化字符串类型  
 b = 90  
 a1 = '%s' % b  
 print (a1)  
 print (type(b))  
 print (type(a1))  
  
 # 格式化整数类型  
 a2 = '%d' % 20.56  
 print (a2)  
  
 # 格式化8进制  
 a3 = '%o' % 123  
 print (a3)  
  
 # 格式化为16进制  
 a4 = '%x' % 1232323395946  
 print (a4)  
  
 # 格式化为16进制大写  
 a5 = '%X' % 123213123123547788769  
 print (a5)  
  
 # 格式化为浮点数  
 a6 = '{0:.1f}'.format(23.68)  
 a7 = '%f' % 123  
 print (a6)  
 print (a7)  
  
 # 格式化e单位  
 print (1e3)  
 a8 = '%e' % 1213123124324.21313  
 print (a8)  
  
 # 合理化格式化单位选择 %f or %e  
 a9 = '%g' % 12123123.123  
 print (a9)  
  
# # 格式化辅助命令  
  
 # m( 占用格式大小 ).n( 保留几位小数 )  
 a10 = '%10.1f' % 273.233  
 print (a10)  
  
 # -左对齐  
 a11 = '%-10.1f' % 123.2  
 print (a11)  
  
 # 显示正号  
 a12 = '%+d' % 12  
 print (a12)  
  
 # 加入#可以识别0o为8进制, 0x为16进制  
 a13 = '%#o' % 1231412  
 a14 = '%#x' % 123120987  
 print (a13 + '\n'+ a14)  
  
 # 0表示空格替换成0  
 a15 = '%010d' % 123  
 print(a15)

# 2018.8.22

### 序列

# 关于list()与len()  
a = 'asdasd'  
b = list(a) # 关于list()将,字符串转化成列表的元素  
c = len(b) # 关于len(),返回元素个数  
print (b)  
print (c)  
  
# 关于max(), 返回最大值.min()相反;在比较字母时实际比较是ascll码,但是字母和数字不可以混合比较  
e = (1,23,4,3425,32)  
e1 = [123,123,2134,4,523,21]  
e = max(e)  
e1 = max(e1)  
print (e,e1)  
  
# 关于sum() , 元素相加返回和  
f = [1,23,4,5,3]  
f1 = (12,23,2,43)  
print (sum(f) + sum(f1))  
  
# 关于sorted()与reversed()与sort()和reverse()没有什么区别  
  
# 关于enumerate(), 将引导值与元素值组成元组  
g = [23,34,5,6,23,2]  
g1 = list(enumerate(g))  
print (g1)  
  
# 关于zip() ,二个列表合并对应元素位置,组合成元组  
h = [1,2,3,4,5,6,7]  
h1 = [7,6,54,3,21,]  
print(list(zip(h,h1)))

# 十进制转换为二进制

Bin()

# 2018.8.29

### 关于函数参数问题

# 给于参数默认值,可以保证在没有参数情况下不出错  
def test (name = 0 , score = 0):  
 print (name,':',score)  
  
# 防止参数传递顺序错误的问题  
def test2 (name = 0 , score = 0):  
 print(name, ':', score)  
  
# 收集参数,可以传入多个参数( 保存为元组 )  
def test3 (\*name , score = 0): # "\*name"可以收集多个参数,并保存为元祖.  
 print(name[:], ':', score)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 test()  
 test2(score=88 , name ='赵飞')  
 test3('赵',1,2,3,4,'a',score='5000000000')

# 关于全局变量与局部变量

函数内是可以访问全局变量。

如果全局变量名称与局部变量名称相同时，则互不影响。

## 然后在函数内部修改全局变量

def test():

global x # 加入global , 就相当于全局变量的x , 如果没有它就只是局部变量.

x = 5

return x

## 非全局外部变量引用

def test2():

s = 20

def test3():

nonlocal s # 在使用" 非全局外部变量 "时加入nonlocal可防止错误 , 可以有效的将变量引用.

s \*= 2

return s

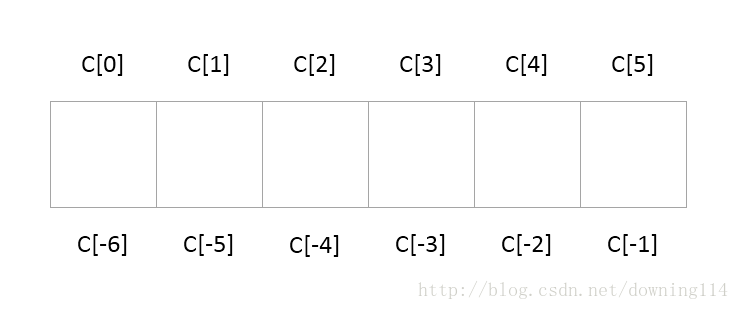
return test3()

## 关于end=‘’

print默认是打印一行，结尾加换行。[end](https://www.baidu.com/s?wd=end&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank)=' '意思是末尾不换行

## 关于切片

切片a[ start , end , step ] start开始，end结束(不包含结束值位置)，step跳跃步数



此时，如果我们将步长设为-1，那么就可以很方便的得到一个反序的序列了：

>>> C[::-1]

['F', 'E', 'D', 'C', 'B', 'A']

# 正则表达式提前小知识

## 生成和提取

Re.compile()用法

pattern = re.compile('[a-zA-Z]') #生成字母

findall()用法

print( pattern.findall( 'asdasd234234asd123213' ) ) #进符合条件的提取出来返回成列表

pattern2 = re.compile(' ')

print( pattern2.findall(x) ) # findall()只接受pattern对象,并以列表的形式返回相等的元素

# 2018.9.6

### Lambda的用法

一些简洁的def语句可以用lambda代替更加方便

g = lambda x : x\*\*2

print(g(5))

add1 = lambda x,y : x+y

print(add1(5,6))

### Filter()函数过滤器的用法及与lambda写法结合

Filter()其实是过滤器,只返回“真”，可以传入2个参数,一个函数一个加工对象

加工结果为真则返回出

print(list(filter(None,[ 1,0,True,False ]))) **# 返回真**

test = filter( lambda x : x % 2 , range(9) ) **# 返回偶数**

print(list(test))

### Map()用法

Map()可以传入2个参数，一个函数，一个加工对象，并且返回一个可迭代结果( 可以以列表的形式返回结果 )

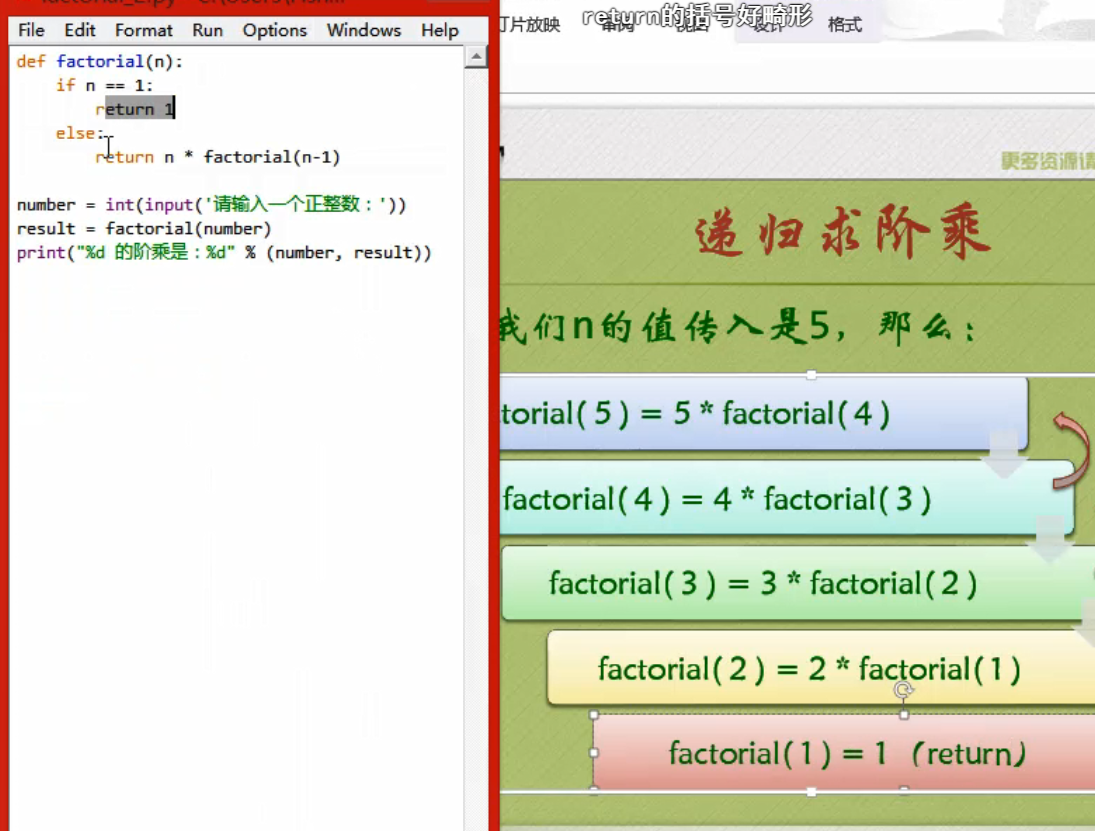
test2 = map( lambda x : x\*\*2 , range(9) ) **# 便捷式加工**

print( list(test2) )

## 递规（用到恰到好处才能体现水平）

什么才是递规?

1. 调用自身
2. 不是死循环



# 2018.9.10

## 字典创建与调用

### # # 创建字典

#### # 标准创建字典方法

dict1 = { 1 : ' one ', 2 : 'two' }

#### # 利用dict()方法创建字典

dict2 = dict(三='three', 四='four' ) # 此方法"键值"有问题!

dict3 = dict((( 5,' five ' ), (6,' six ')))

#### # 利用fromkeys()创建字典( 主要用来创建键值 )

dict4 = {}

dict5 = {}

dict4 = dict4.fromkeys((range(5)),'seven')

dict5 = dict5.fromkeys((1,2,3),'eight')

print(dict1,dict2,dict3,dict4,dict5)

### # # 字典的修改/创建元素

a={1:'what' , 2:'fuck'}

print(a)

a[1] = 'hello' # 通过键值修改对应元素

a[3] = '!' # 添加元素

print(a)

### 字典值迭代需要用的方法keys(),values(),items()

# keys()引用字典的键值

# values()引用字典的内容

# items()以元组的形式索引出字典元素

For eachKey in dict1.keys():

Print(eachKey)

### 用正确的姿势引导字典元素

#### Get()索引字典元素用法

因为使用

Dict1 = {}

Dict1 = dict1.fromkeys( (range(31)),’hello’ )

Dict1[32] # 因为没有32元素所以会报错

Dict1.get(32 , ’ 没有 ’) # 则不会报错,并提示“没有”( 默认返回None )

#### 利用关系符判断是否存在字典

继上方dict1字典

判断有无元素

32 in dict1 # 有则返回True否则返回False

#### 清空字典

# # 清空字典

c = {}

c = c.fromkeys((range(10)),'hello') # 生成字典

print(c)

c.clear() # 执行清空函数

print(c)

#### 拷贝字典

D = {}

D = d.fromkeys(fromkeys(),’hello’)

E = d.copy() # 复制字典

d.clear() # 删除被复制的字典

print(e) # e字典依然有自己的元素

#### 取出字典元素

# # 取出字典元素

# pop() 取出字典单个元素

# popitem() 随机返回并删除字典中的一对键和值(一般删除末尾对)

f = {}

f = f.fromkeys((range(5)),'hello')

f.pop(1) # 取出对应字典元素

f.popitem() # 取出最后一位字典元素

print(f)

#### 添加字典元素

添加字典元素有很多方法

方法1：

a[3] = '!' # 添加元素

方法2：

Setdefault()方法添加元素

# # 通过setdefault() 添加字典元素

g = {}

g = g.fromkeys((range(5)),'wokao!')

print(g)

g.setdefault(5,'2333')

print(g)

# 通过便捷方法添加字典元素

print( 6 in g ) # 用关系符检查一下字典中是否有无键值

g[6] = '55555' # 添加元素

print(g)

#### 更新字典方法

说白喽其实将一个字典元素添加到另外一个字典

# # 更新字典

h = {}

h = h.fromkeys((range(5)),'++++')

j = {666:'---'}

print(h,j)

h.update(j) # 更新 ( 将j字典中的元素加入到h字典中 )

print(h)

### 列表转化为字典

Test = dict([('two', 2), ('one', 1), ('three', 3)])

可以将其转化为字典

### 恢复记忆字符分割split(',')

split(',')以逗号为分割点转化为列表

data = "1000,小甲鱼,男"  
MyDict = {}  
**# 还记得字符串的分割方法吧，别学过就忘啦^\_^**  
(MyDict['id'], MyDict['name'], MyDict['sex']) = **data.split(',')**  
  
print("ID: " + MyDict['id'])  
print("Name: " + MyDict['name'])  
print("Sex " + MyDict['sex'])

### 集合set()

Set1 = {1,2,3}

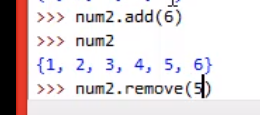
list = [1,2,3]

set1 = set(list) 转化成集合

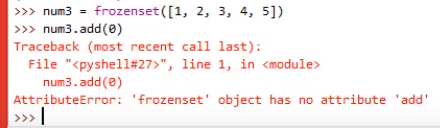
1.集合中没有重复的元素，并且不可索引

2.可以用in等关系符判断是否存在元素

#### 修改集合



不可变集合



# 2018.9.13

关于字典底层原理通过哈希方法存储数据请参考：

<https://fishc.com.cn/thread-45016-1-1.html>

### 集合set()

1.集合不支持索引

2.元素不重复

3.也可以通过关系符判断是否有对应元素

#### 集合常用的内置函数

set1 = {1,2,3,4}

print(set1)

set1.add(233) # 添加元素

set1.remove(2) # 移除元素

print(set1)

# 创建不可变合集

set2 = frozenset([1,2,3,4])

# 常用内置函数

set1.pop() # 移除并返回一个任意元素

set3 = {5,6}

set1.update(set3) # 更新函数

print(set1)

# 2018.9.17

### 文件操作

#/usr/bin/python3

#-\*- coding:utf-8 -\*-

if \_\_name\_\_ == "\_\_main\_\_":

# # 文件的读取 , 查看 , 写入

# # 读取

# 读写模式很重要

# 'a' : 追加模式

# 'w' : 覆盖写入模式

# 't' : 文本模式查看( 默认 )

# 'r' : 只读打开方式( 默认 )

# 'b' : 以二进制模式打开

# '+' 可读写模式（可添加到其他模式中使用）

# 'x' 这里要注意的是'x'和'w'均是以“可写入”的模式打开文件，但以'x'模式打开的时候，如果路径下已经存在相同的文件名，会抛出异常，而'w'模式的话会直接覆盖同名文件。

# 因此，'w'模式打开文件会比较危险，容易导致此前的内容遗失，因此使用'w'模式打开文件前先检查该文件名是否已经存在显得非常重要！

f= open('G:test.txt','at+',encoding='utf-8') # 使用open函数( 路径，读写模式，文件编码 )

# # 查看

f.read() # 查看全部 , 如果有值则查看一部分

f.readline() # 查看一行, 如果有值则查看一部分

f.tell() # 查看当前标签位置

# # 写入

# 二个文件内容复制时 f2.write(f1.read())

f.write( 'str' ) # 写入字符串

f.writelines(列表) # 直接写入一个列表或者序列内容

f.seek(x , y) # y设置起点(0代表起点,1代表当前位置,2代表尾部),x设置标签前进字节大小

f.tell() # 返回当前标签位置

f.close() # 关闭文件

#### 关于换行问题直接在即将写入的字符串中加入”\n”

# open()更多信息请查看: <https://fishc.com.cn/forum.php?mod=viewthread&tid=45279&extra=page%3D1%26filter%3Dtypeid%26typeid%3D403>

### 迭代器

import sys

list = [1,2,3,4]

it = iter(list) # iter()创建一个迭代器

while True:

try:

print(next(it),end="") # next(it)输出迭代器下一个元素

except StopIteration: # StopIteration防止迭代器出现死循环

sys.exit()

#### 关于iter（）函数多行输入小技巧

stopword = ':w'

str = ''

for line in iter(input, stopword): # 此时iter有二个参数，而stopword是设置停止条件

str += line + '\n' # 在每一行尾加入换行符号

### 关于try提前的探索

如果try发生异常，则执行except语句

try:

f = open('D:test3.txt','rt')

except:

f = 0

print(f)

# 2018.9.23

### os模块更多信息：(https://fishc.com.cn/forum.php?mod=viewthread&tid=45512&extra=page%3D1%26filter%3Dtypeid%26typeid%3D403 )

#### Os.path分支



#### 对于路径以及文件的判断,采用函数

import os

# # 判断路径

os.path.exists('E://test.txt') # 判断文件/目录是否存在

os.path.isabs() # 判断是否是绝对路径

os.path.isdir() # 判断是否存在目录

os.path.isfile() # 判断文件是否存在

os.path.islink() # 判断是否是一个符号链接

os.path.ismount() # 判断是否是挂载点( 硬盘分区 )

os.paht.samefile('路径1','路径2') # 判断2个路径是否指定的是一个文件

# # 路径处理

根据全路径获取文件名称的方法os.path.basename(path)

获取文件所在路径的方法os.path.dirname(path)

路径拼接把目录和文件名合成一个路径os.path.join()

# 2018.9.30

### Pickle存储模块

具体使用方法

# 存储

test = [1,2,34,5,123,12,324,4,234,324,234,234]

pickle\_file = open('test.pkl','wb') # 后缀名随便写,不过最好要有提示性( 注意读取模式使用二进制写入 )

pickle.dump( test , pickle\_file ) # test列表保存到test.pkl文件

pickle\_file.close()

# 读取

pickle\_file = open('test.pkl','rb') # 注意读取模式使用二进制读取

test = pickle.load( pickle\_file ) # 读取pickle文件

print(test)

#### 关于os,walk下的使用详解

#!/usr/bin/python3

import os

road = input('输入: ')

# # 关于os.walk

# 接受:

# 接受3个数据参数,开始路径road,onerror错误报告默认为None可不写,反向目录获取topdown=False

# 返回:

# 一个元祖( road(所有文件夹路径 - 字符串类型) , dirs(所有文件夹名称 - 列表类型) , file(所有文件名称 - 列表类型) )

for road, dirs, files in os.walk(road, topdown=False):

# 显示所有文件( 带路径位置 )

for i in files:

print( road+os.sep+i ) # os.sep是本系统路径分隔符

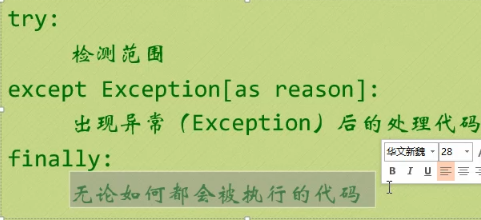
# 2018.10.9

### 异常处理

Test = [1,2,3]

Assert len(test) > 0 # 断言验证,为假则抛出异常

#### 关于try….except异常处理



关于TRY有二种使用方法，1，具体报错处理，2，全异常处理

错误类型的更多信息在: <https://fishc.com.cn/forum.php?mod=viewthread&tid=45814&extra=page%3D1%26filter%3Dtypeid%26typeid%3D403>

方法1: 推荐使用具体错误信息处理：

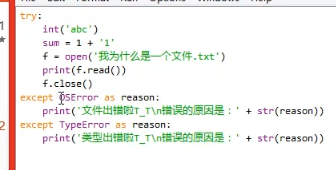
关于Except OSError(指定检测错误的类型) as reason(具体报错信息变量) :

try:

int('abc')

except ValueError as x:

print('错误ValueError: '+str(x))





方法2: 全异常处理



#### Raise故意引出异常

故意引出数字报错异常



s = 0

try:

int('12')

raise ZeroDivisionError # 抛出异常

except (ValueError, ZeroDivisionError) as x:

print('错误: '+str(x))

### 关于else用法

if else , while else , try except else，for else

#### While…..else….

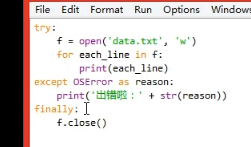
While else如果while被完整的执行,中间没有break那么会执行else的语句,否则不会执行else语句

#### Try……except……else

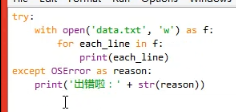
如果try出现异常则不会执行else，相反如果try没有异常则执行else

### With自动关闭文件用法

普通使用try方法：



使用with方法：



try:

with open('e://test.txt','wt+') as file:

for i in file:

print(i)

except NameError as x:

print('NameError'+str(x))

else:

print('成功读取')

#### with打开多文件方法

1. with A() as a, B() as b:
2. suite

with open(file1) as f1 , open(file2) as f2:

print(‘OK’)

### 导入模块的3种方法

第一种: import easygui

第二种: from eaysui import \* ( 这种方法，在打包文件时减少大小有非常重要的作用 )

第三种: import easygui as g ( 匿名模块,写起来轻松 )



### Os.system下多条语句执行方法

os.system('第一条命令 && 第二条命令')多条命令执行方式

b=os.popen('ls') # 功能与system相同，不过可以读取执行后的所显示的内容

b.read()

### 使用subprocess调用cmd命令，并且等待子程序完成才执行其它命令

cmd = "%s && %s" % ( setpath,ffmpegset ) # 调用多条语句

a = subprocess.Popen(cmd,shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT) # 调用cmd命令

a.wait() # 等待子程序( cmd命令 )运行结束,才开始执行新的命令

# 2018.12.29

### 强大的eval()函数

这个eval()函数可以将字符串转换成可执行代码,强大的过分！！！

# 2019.1.13

### 进制转换

二进制转10进制

Int( ‘1010100’,2 ) # 接受字符串类型的二进制，参数为2，即可转换为十进制

Bin() 十进制转二进制

Oct() 十进制转八进制

Hex() 十进制转16进制

# 2019.3.26

### 替换字符串

Test = “abads\nasd”

Test.replace(‘\n’,’$’) # 将’\n’替换成’$’

# 2019.3.26

### 逻辑运算符

“and” 与C的 “&&” 对应

“ or ” 与C的 “||” 对应

“not” 与C的” ! ”非 对应

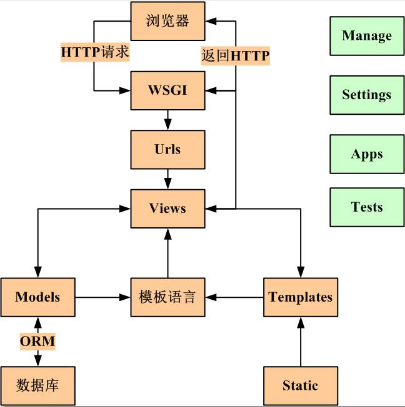
# 2019.5.15

## Django采用mvc设计模式

**模型(Model)**：用于封装与应用程序的业务逻辑相关的数据及对数据的处理方法，是Web应用程序中用于处理应用程序的数据逻辑的部分，Model只提供功能性的接口，通过这些接口可以获取Model的所有功能。白话说，这个模块就是Web框架和数据库的交互层。

**视图(View)**：负责数据的显示和呈现，是对用户的直接输出。

**控制器(Controller)**：负责从用户端收集用户的输入，可以看成提供View的反向功能。



# 2019.5.16

### 关于pip3安装调试在谷歌后端收藏夹中有博客链接

## virtualenv虚拟环境，python3虚拟环境操作

他能有效的防止各种版本冲突问题，可以为每一个python项目单独设置空间和环境

### 安装：

1. pip3 install virtualenv
2. 建立软链接：ln -s /usr/local/python3/bin/virtualenv /usr/bin/virtualenv（ 特别重要 ）
3. 建立软链接的目的是，让python3认识到此库的安装

#### 创建目录:

1. Mkdir test\_vir
2. cd test\_vir

#### 创建虚拟环境:

1. 进入test\_vir目录内
2. Virtualenv - -no-site- -packages vens ( 创建干净的python3环境，vens是环境变量名称,可修改 )
3. Virtualenv - -python=/usr/bin/python3 vens( 创建默认python3安装环境 )

#### 进入虚假环境:

1. Source venv/bin/activate
2. 进入成功则显示

#### 安装第三方库:

1. Pip3 install Django==2.2

#### 退出虚拟环境:

1. Deactivate

#### 删除虚拟环境:

1. 直接删除虚拟环境所在目录即

#### 查看使用pip已安装项目:

1. Pip3 list

#### 虚拟环境交接：

1. 方案一：

连同虚拟环境和项目一起拷贝给他人

1. 方案二：

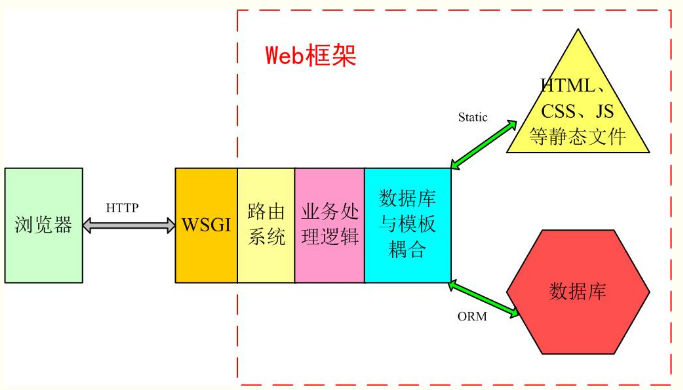
在虚拟环境中，冻结依赖需求文本

把项目和依赖需求文本给他人

他人在本地创建一个新的虚拟环境，并根据依赖需求文本安装相关库

# 2019.5.18

### Web框架：



### MVC/MTV简介

#### MVC:

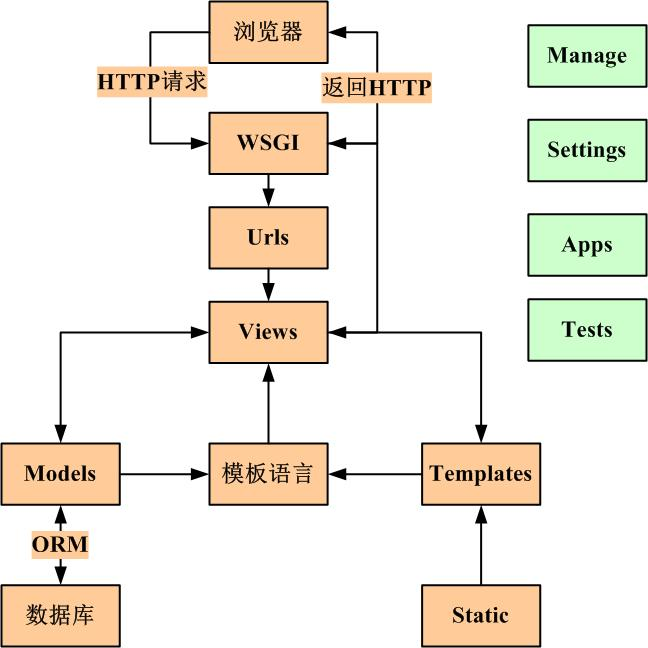
1. M( 模型 : model ): 定义数据库相关内容
2. V( 视图:view ): 与前端东西相关
3. C( 控制器: controller ): 与业务逻辑有关联

#### MTV：

他其实是DJANGO下的模式，但是与传统的MVC换烫不换药

1. M( 模型:model.py ): 操纵数据库相关内容
2. T( 模板:Templates ): 放前端的东西
3. V( 控制器: view.py ): 操作业务逻辑

#### Django框架模型组



## Django实战记录,创建登陆系统

### Django安装

* 1. 过程参考上放virtualenv虚拟环境篇目

Pip3 install Django==2.1.8

### 验证django是否可用

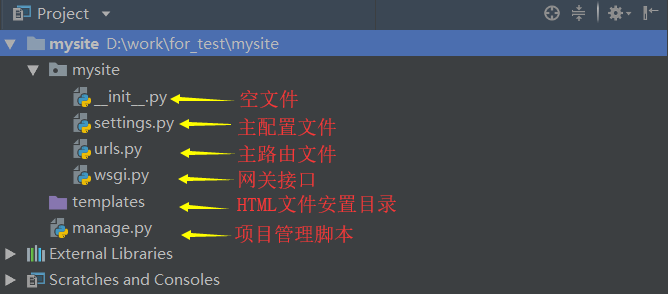
Django-admin –help # 出现内容则OK

### 创建django项目

Django-admin startproject mysit # mysit也可以是其它名称

如果创建成功:

注: templates文件夹与创建的mysite同级,并且templates是手动创建,用于存放html模板文件



### 创建app

Python manage.py startapp login # 注意: 此app非鄙app

#### Django中，说白了就是功能模块

* + 1. App实现一个功能,如投票系统
    2. 一个project可以配置多个app
    3. 一个app也可以属于多个project

### 编写路由

路由就是浏览器输入访问的路径

在mysit/urls.py中

From login import views # 导入login,app中views.py文件

urlpatterns = [

path('admin/', admin.site.urls),

path('index/', views.index), # 执行views类下的index函数,并定义路径名称为index/

]

### 编写login/views.py操作app业务逻辑

From Django.shortcuts import HttpResponse # 导入http传输模块

Def index( request ): # request是必须的,行业规则

Return HttpResponse(‘hello word!!!’) # 给html渲染输出hello word

### 运行django自带的轻量级web服务,生产模式不可用,开发模式使用

Python manage.py runserver # 浏览器输入 127.0.0.1:8000

python manage.py runserver 0:8000 # 指定ip和端口访问, 0.0.0.0:8000可访问( 局域网都可以访问 )

python manage.py runserver 8080 # 指定端口访问 127.0.0.1：8080访问

# 运行服务后访问: 127.0.0.1:8000/index/

### 返回html文件

#### 在templates文件夹下创建index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>test</title>

</head>

<body>

<h1>Hello World!</h1>

</body>

</html>

#### 在login/view.py业务逻辑模块设置

From Django.shortcuts import render

Def index(request):

# render( request, '模板文件' )此函数用于渲染模板,第一个参数request是必须项,第二个是模板文件

Render( request, ‘index.html’ )

#### 确定好业务逻辑,我们告诉mysit/setting.py我们的模板文件在那

# 模板文件设定

# os.path.join( )合并路径函数,基本共

# BASE\_DIR为django自己的路径是一个变量,默认就好

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [os.path.join(BASE\_DIR, 'templates')], # 设定模板所在路径

'APP\_DIRS': True,

……

#### 然后就可以启动服务啦

Python manage.py runserver

### 静态文件配置,如js，css，图片什么的,存放在static文件夹中

#### 建立static文件夹

Static文件夹要与,templates同级

建立static/js/jquery.3.2.1.min.js

#### 老规矩配置mysit/setting.py告诉django前端素材在那

# 素材与插件路径

STATIC\_URL = '/static/' # 访问静态文件加的前缀 如:www.oo7.fun/static/xx.jpg

STATICFILES\_DIRS = [

os.path.join( BASE\_DIR, 'static' ) # 和并路径

]

#### 接着设置下templates/index.html目的测试下能不能正常访问static/js/jquery.3.2.1.min.js文件

# Index.html文件中加入:

<script type=”text/javascript” src=”/static/js/jquery.3.2.1.min.js” ><script>

### 接受用户发送的数据

#### 在index.html创建一个表单

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>首页</title>

</head>

<body>

<h1>用户输入：</h1>

<form action="/index/" method="post"> <!—以post传输给index业务函数 -->

{% csrf\_token %} <!--因为django有跨站保护机制,所以要加入这行代码 -->

用户名：<input type="text" name="username" /><br /> <!—name名称很重要 -->

密码：<input type="password" name="password" /><br />

<input type="submit" value="提交" />

</form>

</body>

</html>

#### 在login/view.py中配置业务逻辑,来应付提交的表单

from django.shortcuts import render

from django.shortcuts import HttpResponse

# Create your views here.

def index(request):

if request.method == 'POST': # 检测是否有post数据传输

username = request.POST.get('username') # 获取标签name属性username传输来的数据

password = request.POST.get('password')

print(username, password)

return render(request, 'index.html')

#### 启动服务

访问127.0.0.1:8000/index.html 提交表单，即可在后台看到返回信息

### 返回动态页面,将数据存储到字典中,并返回显示在html中

#### 改进login/view.py业务逻辑

from django.shortcuts import render

from django.shortcuts import HttpResponse

# Create your views here.

user\_list = [] # 创建用户列表

def index(request):

if request.method == 'POST':

username = request.POST.get('username')

password = request.POST.get('password')

print(username, password)

temp = {'user': username, 'pwd': password} # 以字典的形式存储二个变量

user\_list.append(temp) # temp字典加入列表中

return render(request, 'index.html', {'userdata': user\_list}) # 将列表内容加入字典，并以字典的形式返回到页面

#### 改进html文档

<!doctype html>

<html>

<head>

<title>登录测试</title>

<meta charset="utf-8">

<script type="text/javascript" src="/static/js/jquery-3.4.1.min.js" ></script>

</head>

<body>

<h1>Hello Word!!!</h1>

<form action="/index/" method="post" >

{% csrf\_token %} <!-- 因为django有跨站保护,所以要加入这行代码 -->

User: <input type="text" name="username"> <br/>

PassWord: <input type="password" name="password"> <br/>

<input type="submit" value="SUBMIT">

</form>

<!-- 展示区域 -->

<h1>展示</h1>

<dl>

{% for item in userdata %} <!-- 推测: 模拟python的for in迭代循环 -->

<dt> {{ item.user }} </dt> <!-- 推测: 双大括号,可以使python变量值实体化 -->

<dd> {{ item.pwd }} </dd>

{% endfor %}

</dl>

</body>

</html>

### 与数据库关联

#### 没有数据库，我们提交的密码将无法保存

#### 我们先在mysit/setting.py中注册下功能app,这样数据库就能找到app项目

# mysit/setting.py注册app

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'login', # 填入我们创建的app名称( 不注册,数据库就不知道给那个app创建表 )

]

#### 指定数据库类型

#默认数据路sqlite3，当然也可以配置使用其它数据库

# 指定数据库类型

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3', # 指定数据库类型

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

#### 既然玩数据库，那么设置models.py就必不可少，架构摆在那呢

# 编辑login/models.py，创建2个字段分别保存用户的名字和密码

from django.db import models

class UserInfo( models.Model ): # 固定写法,必须要有'models.Model'

user = models.CharField( max\_length=32 ) # 创建2个字段,最大长度不超32个

pwd = models.CharField( max\_length=32 )

#### 创建迁移记录文件

在login目录中的migrations目录中生成一个0001\_initial.py迁移记录文件

执行:

Python manage.py makemigrations # 生成一个0001\_initial.py迁移记录文件

#### 创建数据库表

执行:

# 这样，我们就在数据库中将所有app的数据表都创建好了。我们可以看到项目根目录下出现了一个db.sqlite3文件：

Python manage.py migrate

#### 改进业务逻辑，既然我们要输出数据库内容，这一步必不可少

# 编辑 login/view.py

from django.shortcuts import render

from django.shortcuts import HttpResponse

from login import models # 导入models文件

user\_list = [] # 用户列表

def index(request): # 潜规则,第一个参数必须是request

if request.method == 'POST':

username = request.POST.get('username')

password = request.POST.get('password')

# 将传送来的变量值保存到数据库中( 'UserInfo'与models.py中设定的表对应 )

models.UserInfo.objects.create( user=username, pwd=password )

# 读取数据库中全部数据

user\_list = models.UserInfo.objects.all()

return render(request, 'index.html', {'userdata':user\_list} )

#### 开启服务器测试吧！！！

# 2019.5.21

Join()函数