# 常规操作

**Python程序书写规范**：<https://zh-google-styleguide.readthedocs.io/en/latest/google-python-styleguide/python_style_rules/>

解决中文注释无法编译的问题

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

Ctrl+[ 或 crtl+] 集体缩进

Alt+n 输入上一句语法

dir(\_\_builtins\_\_) #查看内置函数 str() int() float() abs()等等

dir(list) #查看某个模块的内置函数，以list为例

help(内置函数名) #查询内置函数说明

python的字符串可以使用单引号’或者双引号”,但要成对出现，不能混合使用；

需要使用’或者”怎么办？使用转义字符 **\**.

1）输入路径 str=‘C:\\folder\\ff’

2）使用原始字符串 str=r‘C:\folder\ff’

三重引号字符串，输出很长的字符串：

Str=”””

白日梦想家，

天天算数据，

佘山索菲特。

“””

# 判断变量的类型：

type(变量) #输出类型

isinstance(变量，类型) #输出bool值

# 操作符

运算操作符： + - \* / // \*\*

// floor整除,如 5/2=2.5 5//2=2

\*\* 幂指数，相当于^或pow(x,n)

逻辑操作符：and not or

not False //输出True

**x>y>z == x>y and y>z**

多元操作符：

x,y=4,5.0

small = x if x<y else y //类型C small=x<y?x : y;

assert x>y 类型与C: Error(“adadsa”);

print(i,end='') //循环里不换行输出值

# while循环和for循环

内置函数range()与for

range(5) // 0,1,2,3,4

range(1,5) // 1,2,3,4 [1,5)

range(1,5,2) // 1,3 （步长在最后面）注意与matlab中的linspace(1:2:5)差别（步长在中间）

# 列表

**列表**，**中括号**，相当于C#中的ArraylList,动态的，但元素不确定

mix=['sss',123,[1,2,3]]

len(mix) //确定长度

mix.append('haluo') //末尾增加一个元素

mix.extend(['sdsd',222]) //末尾增加多个元素

mix.insert(1,'1111') //列表索引处插入一个元素

mix[0],mix[1]=mix[1],mix[0] //列表元素调换，python中可以使用a,b=b,a实现元素交换

mix.remove(‘sdsd’) //列表中删除元素（根据元素值实现删除）

del mix[1] //列表中删除元素（根据下标实现删除）

del mix //删除整个列表

mix.pop() //删除列表末尾处的元素

mix.pop(index) //删除列表索引处的元素

mix[1:3] mix[:3] mix[1:] mix[:] //跟matlab类似的索引功能,**其中mix[:]可用于列表拷贝！**

列表的比较

List1=[1]

List2=[2]

List1>List2 //返回False

List1=[1,3]

List2=[2,1]

List1>List2 //返回False,与字符串的比较方式一样，只比较第一个元素

List1+List2 //返回[1, 3, 2, 1],类似extend函数操作

List1\*2 //返回[1, 3, 1, 3]

1 in List1 //返回True

1 not in List1 //返回False

列表其它内置函数：

.count(123) //列表中元素123出现的次数

.index(123) //列表中元素123首次出现的索引

.reverse() //列表元素翻转

.sort() //列表元素从小到大排序（默认），从大到小.sort(reverse=True)

列表拷贝

List3=List2[:] //深拷贝，更改List2不影响List3

List3=List2 //浅拷贝，更改List2会影响List3，相当于C中多了一个标签

# 元组

**元组**：受限制的列表（相当于C中的数组），**逗号和小括号**

元组元素不能随意改变！！！

>>> tuple1=(1,2,3,4,5,6,7,8)

>>> tuple1

(1, 2, 3, 4, 5, 6, 7, 8)

>>> tuple1[3]

4

>>> tuple1[:6]

(1, 2, 3, 4, 5, 6)

>>> tuple2=tuple1[:]

>>> tuple2

(1, 2, 3, 4, 5, 6, 7, 8)

>>> temp=(1) //元组的关键是加逗号，有没有小括号无所谓

>>> temp

1

>>> type(temp)

<class 'int'>

>>> temp2=2,3,4

>>> type(temp2)

<class 'tuple'>

>>> temp=(1,)

>>> type(temp)

<class 'tuple'>

>>> temp=1,

>>> type(temp)

<class 'tuple'>

>>> temp=()

>>> type(temp)

<class 'tuple'>

>>> 8\*(8)

64

>>> 8\*(8,)

(8, 8, 8, 8, 8, 8, 8, 8)

temp(‘sss’,’ddd’)

temp=temp[:1]+(‘love’,)+temp[1:]

>>> (‘sss’,’love’,’ddd’)

# 字符串

**字符串:操作与列表类似，内置函数的功能与C的差不多**

>>> str1='I like Luo YiNi'

>>> str1

'I like Luo YiNi'

>>> str1[:6]

'I like'

>>> str1[0]

'I'

>>> str1[:6]+'very' + str1[6:]

'I likevery Luo YiNi'

>>> str1[:6]+'very ' + str1[6:]

'I likevery Luo YiNi'

>>> str1[:6]+' very' + str1[6:]

'I like very Luo YiNi'

>>> str2='xiaoxie'

>>> str2.capitalize()

'Xiaoxie'

>>> str2.center(20)

' xiaoxie '

>>> str1.count(like)

Traceback (most recent call last):

File "<pyshell#10>", line 1, in <module>

str1.count(like)

NameError: name 'like' is not defined

>>> str1.count('like')

1

>>> str3='I\tlobe\tluoyini!'

>>> str3

'I\tlobe\tluoyini!'

>>> str3.expandtabs()

'I lobe luoyini!'

>>> str3.find(be)

Traceback (most recent call last):

File "<pyshell#15>", line 1, in <module>

str3.find(be)

NameError: name 'be' is not defined

>>> str3.find('be')

4

>>> str3.find("be")

# 格式化输出

**格式化：主要用于输出等格式化**

>>> "{0} love {1} {2}".format("I","Luo","Silong") //有点类似C#的输出

'I love Luo Silong'

>>> '{0:.1f}{1}'.format(25.325,'GB') //：.1f（保留一位小数）

'25.3GB'

>>> '%c' % 97 // %c格式化ASCII码

'a'

>>> '%c %c %c' % (97,98,99)

'a b c'

各类转义字符等等

>>> **print('%s love %s in %d' % ('luo','yini',520))**

luo love yini in 520

# 序列的常用内置函数

**序列的常用内置函数**：列表、元组和字符串

**list(x), tuple(x), str(x), int(x), max(x), min(x), len(x), sum(x), zip(a,b)… //可使用help(list)查看内置函数的用法**

>>> a=list()

>>> a

>>> b='I LOVE YOU'

>>> b=list(b)

>>> b

['I', ' ', 'L', 'O', 'V', 'E', ' ', 'Y', 'O', 'U']

>>> c=(1,1,2,3,5,8,13,21)

>>> c=list(c)

>>> c

[1, 1, 2, 3, 5, 8, 13, 21]

# python函数的注意点

## 1）**函数的文档**

def Myfunc():

'函数的定义233'

print('第一个函数')

>>> Myfunc.\_\_doc\_\_ //方法1查看

'函数的定义233'

>>> help(Myfunc) //方法2查看

Help on function Myfunc in module \_\_main\_\_:

Myfunc()

函数的定义233

## 2）关键字参数强行匹配

def Myfunc2(name1,name2):

print(name1+' love you '+name2)

>>> Myfunc2('luo','yini')

luo love you yini

>>> Myfunc2(name2='luo',name1='yini') //采用形成指定参数强行匹配

yini love you luo

## 3）默认参数（与C++类似）

def Myfunc2(name1,name2=‘YiNi’):

print(name1+' love you '+name2)

## 4) 搜集参数（函数参数数目不确定）

def test(\*param):

print('参数的长度是：',len(param),exp);

print('第二个参数是：',param[1]);

>>> test(1,'luo',3.14)

参数的长度是： 3

第二个参数是： luo

如果有某些参数是确定的怎么和搜集参数配合使用？

def test(\*param,exp): //方法1，已知参数指定

print('参数的长度是：',len(param),exp);

print('第二个参数是：',param[1]);

>>> test(1,'luo',3.14,exp=5)

参数的长度是： 3 5

第二个参数是： luo

def test(\*param,exp=5): //方法2，已知参数设置为默认参数

print('参数的长度是：',len(param),exp);

print('第二个参数是：',param[1]);

>>> test(1,'luo',3.14)

参数的长度是： 3 5

第二个参数是： luo

# 局部变量和全局变量

与C++的作用域一样（不要在函数中修改全局变量的值，上跳出函数后，是无效的（实质：函数在函数内部**创建了一个与全局变量名字一样的局部变量**））

a=10

>>> def func():

a=5

print(a)

>>> a

10

>>> func()

5

>>> a

10

一定要在函数内部修改全局变量的值怎么办？ 使用global关键字

>>> def func():

global a=5

print(a)

# 内嵌函数和闭包

def Myfunc():

'函数的定义'

print('第1个函数')

def Myfunc2(): //内嵌函数

print('第2个函数')

Myfunc2() //内嵌函数只能在上一层函数中调用

>>> Myfunc()

第1个函数

第2个函数

>>> Myfunc2() //内嵌函数只能在上一层函数中调用

Traceback (most recent call last):

File "<pyshell#30>", line 1, in <module>

Myfunc2()

NameError: name 'Myfunc2' is not defined

**闭包（有点类似于C#中的函数委托！函数名做参数）**

>>> def funx(x):

def funy(y):

return x\*y

**return funy** //**返回函数名**

>>> i=funx(8)

>>> i //i为函数名

<function funx.<locals>.funy at 0x0000000002EB8828>

>>> i(5) //i相当于funy函数的委托

40

>>> funx(8)(5) //复合写法

40

>>> def fun1():

x=5 //fun1()函数的局部变量x

def fun2():

x\*=x //这个x是fun2()函数的局部变量，不是上面的x

return x

**return fun2()**  //**返回函数，即fun2函数的返回值！注意与上面的区别！**

>>> fun1() //报错

解决办法：**nonlocal关键字，弱化的global**

>>> def fun1():

x=5 //fun1()函数的局部变量x

def fun2():

nonlocal x //声明这个x不是fun2()函数的局部变量，而是上面的x

x\*=x

return x

return fun2()

>>> fun1() //正确，输出25

# Lambda表达式和匿名函数

**常规函数：**

**>>> def ds(x):**

**return 2\*x+1**

**>>> de(5)**

**11**

**匿名函数：**

**>>> g=lambda x:2\*x+1 //优点：间洁，不需要命名，定义和调用在一块**

**>>> g(5)**

**11**

# 两个牛逼的内置函数：filter()和map()

**1)filter():过滤函数**

>>> filter(None,[1,0,False,True]) //从后面过滤出非真的值，返回为真的值

<filter object at 0x0000000002E92C88>

>>> list(filter(None,[1,0,False,True]))

[1, True]

>>> def odd(x):

return x % 2

>>> temp=range(10)

>>> show=filter(odd,temp) //从temp中过滤出odd函数中的偶数，得到奇数

>>> list(show)

[1, 3, 5, 7, 9]

>>> list(filter(lambda x: x % 2,range(10))) //一行实现上面的代码

[1, 3, 5, 7, 9]

>>> list(map(lambda x: x \* 2,range(10))) //代入迭代数据，进行输出并储存

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

# 递归函数及其应用

**递归**：消耗内存，尽量不使用（求和，阶乘等等）

>>> def sum1(N):

if N <= 0:

return 0

else:

return N+sum1(N-1)

**Fiboncci数列**

---------常规方法----------

>>> def fun2(n):

n1=1

n2=1

n3=1

if n<1:

print("error")

return -1

while (n-2) > 0:

n3 = n2 + n1

n1 = n2

n2 = n3

n -= 1

return n3

--------------递归方法----------------

>>> def fun1(N):

s=1;

if N==1 or N==2:

return 1

else:

return fun1(N-1)+fun2(N-2)

**汉诺塔递归算法：**

def hanoi(n,x,y,z):

if n == 1:

print(x,'-->',z)

else:

hanoi(n-1,x,z,y)#将前n-1个盘子从x移动到y上

print(x,'-->',z) #将最底下的最后一个盘子从x移动上y上

hanoi(n-1,y,x,z)# 将y的n-1个盘子移动到z上

n=int(input('输入汉诺塔的层数：'))

hanoi(n,'x','y','z')

**字典（键key--值value）**：Hash映射，**大括号**

>>> brand = ['LiNing','Nike','Adias','Luo']

>>> slogan=['1111','2222','3333','4444']

>>> print('Luo的口号是:',slogan[brand.index('Luo')])

Luo的口号是: 4444

//----------使用字典进行操作-----------------------------

>>> dict1={'LiNing':'1111','Nike':'2222','Adias':'3333','Luo':'4444'}

>>> print('Luo的口号是:',dict1['Luo'])

Luo的口号是: 4444

>>> dict2={1:'one',2:'two',3:'three'}

>>> dict2[1]

'one'

>>> dict3=dict((('F',70),('i',105),('C',67)))

>>> dict3

{'F': 70, 'i': 105, 'C': 67}

>>> dict4=dict(我='luo',你='Yini')

>>> dict4

{'我': 'luo', '你': 'Yini'}

>>> dict4['爱']='love' //增加键值对

>>> dict4

{'我': 'luo', '你': 'Yini', '爱': 'love'}

dict1={}

>>> dict1

{}

>>> dict1.fromkeys((1,2,3))

{1: None, 2: None, 3: None}

>>> dict1.fromkeys((1,2,3),'bumber')

{1: 'bumber', 2: 'bumber', 3: 'bumber'}

>>> dict1.fromkeys((1,2,3),('one','two','three'))

{1: ('one', 'two', 'three'), 2: ('one', 'two', 'three'), 3: ('one', 'two', 'three')}

>>> dict1.fromkeys((1,3),'number')

{1: 'number', 3: 'number'}

>>> dict1 = dict1.fromkeys(range(32),'赞')

>>> dict1

{0: '赞', 1: '赞', 2: '赞', 3: '赞', 4: '赞', 5: '赞', 6: '赞', 7: '赞', 8: '赞', 9: '赞', 10: '赞', 11: '赞', 12: '赞', 13: '赞', 14: '赞', 15: '赞', 16: '赞', 17: '赞', 18: '赞', 19: '赞', 20: '赞', 21: '赞', 22: '赞', 23: '赞', 24: '赞', 25: '赞', 26: '赞', 27: '赞', 28: '赞', 29: '赞', 30: '赞', 31: '赞'}

>>> for eachkey in dict1.keys():

print(eachkey) //打印所有的键

>>> for eachvalue in dict1.values():

print(eachvalue) //打印所有的值

>>> for eachitem in dict1.item2():

print(eachitem) //打印所有的键-值对

字典的内置函数：如.pop(),.update(),.clear(),.get()等需要时再查看

# 集合

**集合，跟字典类型，但没有键值对**

>>> num={}

>>> type(num)

<class 'dict'>

>>> num2={1,2,3,4,5,6} //大括号初始化，不加键值对

>>> type(num2)

<class 'set'>

>>> num2={4,5,6,7,8,9}

>>> num2

{4, 5, 6, 7, 8, 9}

>>> num2={4,5,6,7,8,9,1,0,1} //集合中元素会自动从小到大排序，且不存在相同的元素

>>> num2

{0, 1, 4, 5, 6, 7, 8, 9}

>>> num2[1] //集合中的元素不得调用

Traceback (most recent call last):

File "<pyshell#8>", line 1, in <module>

num2[1]

TypeError: 'set' object is not subscriptable

>>> set1=set([1,1,2,2,3,4,5,5,6]) //使用set()进行初始化

>>> set1

{1, 2, 3, 4, 5, 6}

>>> num1=[0,1,1,2,3,4,4,5,6,6]

>>> num1

[0, 1, 1, 2, 3, 4, 4, 5, 6, 6]

>>> temp=[]

>>> for each in num1: //迭代法消除列表中的重复元素

if each not in temp:

temp.append(each)

>>> temp

[0, 1, 2, 3, 4, 5, 6]

>>> num1=list(set(num1)) //集合法消除重复元素

>>> num1

[0, 1, 2, 3, 4, 5, 6] //**注意：集合会自动排序**

>>> 1 in num2

True

>>> num2.add(6) //增加集合元素

>>> num2.add(16)

>>> num2

{0, 1, 4, 5, 6, 7, 8, 9, 16}

>>> num2.remove(7) //删除集合元素

>>> num2

{0, 1, 4, 5, 6, 8, 9, 16}

>>> num3=frozenset([1,2,3,5,5,6]) //**冻结集合**（不允许对几何元素进行增减操作）

>>> num3

frozenset({1, 2, 3, 5, 6})

>>> num3.add(7)

# 文件流

**文件流 (跟C的差不多)**

>>> f=open(r'C:\Users\Administrator\Desktop\p1.txt','r') //读写模式：‘r’, ‘w’, ‘a’等等

>>> **f.read() //.read()函数，一次性全部读取**

'1 2 222 33,55123,,,32221\n你说的什么鬼？\nhelloworld!'

>>> f.close() **//.close()函数，操作完关闭**

>>> f=open(r'C:\Users\Administrator\Desktop\p1.txt','r')

>>> f.read(5) **//.read(m)函数，读取m个字符**

'1 2 2'

>>> f.tell() **//.tell()函数，当前指针读取的位置**

5

>>> f.seek(10,0) **//.seek()函数，修改当前文件指针读取的位置**

10

>>> f.readline()  **//.readline()函数，按行读取**

',55123,,,32221\n'

>>> list(f) **//读取的内容存放至列表中**

['你说的什么鬼？\n', 'helloworld!']

>>> f.seek(0,0) **//.seek(0,0)函数，当前文件指针读取的位置至于初始**

0

>>> for str1 in list(f): **//用列表来存储读取的内容：效率慢**

print(str1)

1 2 222 33,55123,,,32221

你说的什么鬼？

helloworld!

>>> f.seek(0,0)

0

>>> for each in f: **//直接存储内容：推荐**

print(each)

1 2 222 33,55123,,,32221

你说的什么鬼？

helloworld!

>>> f.close()

>>> f=open(r'C:\Users\Administrator\Desktop\pw.txt','w')

>>> f.write('I love you!') //.write()，.writeline()函数，写操作

11

>>> f.close()

# Os模块目录操作

**Os模块进行文件目录操作：获取目录及其下面的文件**

>>> import os //导入os模块

>>> os.getcwd()

'E:\\ofPython\\ofInstallPython'

>>> os.listdir("E:\\") //**E盘的目录（即E盘内的文件夹名）**

['$RECYCLE.BIN', '01-PS-CC2017-WIN-安装包', '01-PS-CC2017-WIN-安装包.rar', '1．03．150107．STC.exe', '20180509MGEX数据下载', '360Downloads', '360MoveData', '360手机相册备份', 'adcfg.json', 'Adobe Reader V9.4.0 简体中文版', 'ADSafe', 'baidu download', 'BaiduYunDownload', 'BaiduYunGuanjia', 'c3ddesigner.dle', 'CAJViewer 7.2.self(1).exe.dl', 'CAJViewer 7.2.self(2).exe.dl', 'CAJViewer 7.2.self.exe', 'caj格式到pdf', 'coreftplite.exe', 'coreftplite\_baidu.zip', 'c代码', 'DCSM', 'debug', 'dlcache', 'Download', 'GIS导论课件资源', 'GMT', 'GMT.rar', 'gnsss', 'Google', 'GouWoGames', 'image001.gif', 'kankan', 'kinggsoft', 'ksafe', 'KwDownload', 'maxToCult3d.rar', 'Microsoft Office', 'MSOCache', 'MYVS', 'of dream', 'of wps', 'ofAnaconda', 'ofArcGIS', 'ofBerbese', 'ofGamit', 'ofGMT5', 'ofPS', 'ofPython', 'ofQt', 'ofQt5VSAddin', 'ofTEQC', 'ofVS', 'OriginPro 9.1', 'OriginPro 9.1.zip', 'pagefile.sys', 'pingcha', 'Program Files', 'Program Files (x86)', 'psiphon3.exe.orig', 'QQGame', 'QQMusicCache', 'QT5ANZHUANGBAO', 'QtProjects', 'Qt\_MyPractices', 'RTKLIB-rtklib\_2.4.3\_Qt\_Dr.Liu', 'RTKLIB-rtklib\_2.4.3\_Qt\_Dr.Liu(20171124).zip', 'STK.rar', 'STK11.0', 'STK9.2', 'STK9.2安装及破解.zip', 'stkDocuments1', 'STK\_LiXingxing', 'System Volume Information', 'Tencent', 'Thunder Network', 'tudouClient', 'TuRui\_PPP-RTK', 'UE', 'Users', 'Wangyiyun', '典型要素更新系统', '南方平差易2002解密版', '杂项文件', '深度学习视频', '科傻1.0正式版', '腾讯游戏', '说明.txt', '课设', '课设 - 副本', '迅雷下载', '选星优化算例数据集']

>>> os.mkdir('E:\\AAA') //新建目录（文件夹）

>>> os.rmdir("E:\\AAA") //删除目录（文件夹）

>>> os.system("cmd") //调用系统控制台

-1073741510

>>> os.system("calc") //计算器

0

>>> os.path.basename('E:\\A\\B\\sex.avi') //获取路径下的文件名，不含路径

'sex.avi'

>>> os.path.dirname('E:\\A\\B\\sex.avi') //获取路径，不含文件名

'E:\\A\\B'

>>> os.path.join('C:\\','A','B') //路径组合

'C:\\A\\B'

>>> os.path.split('C:\\A\\B\\sex.avi') //路径切割：目录+文件

('C:\\A\\B', 'sex.avi')

>>> os.path.splitext('C:\\A\\B\\sex.avi') //获取文件的后缀，类型

('C:\\A\\B\\sex', '.avi')

# pickle（泡菜）模块

**数据存储与读取，pickle（泡菜）模块**

**用途：数据打包！将大部分数据(列表数据，元组，字典，几何等)保存到pickle二进制文件中(.pkl)，便于调用（类似于之前C++做的正态分布表文件.dat）**

>>> import pickle //载入pickel模块

>>> list1=[123,3.14,'luosilong',['luo','yini']]

>>> pickle\_file=open('list1.pkl','wb') //’wb’:二进制写入文件

>>> pickle.dump(list1,pickle\_file) //将数据倒入二进制文件

>>> pickle\_file.close() //弄完关闭

>>> pickle\_file=open('list1.pkl','rb') //’rb’:二进制文件读取文件

>>> list2=pickle.load(pickle\_file) //读取数据

>>> print(list2)

[123, 3.14, 'luosilong', ['luo', 'yini']]

# 检测异常

try:

检测范围

except Exception[as reason]:

出现异常的处理代码

finally:

必须要处理的语句块

-------------------------------------------------

try:

n=int('abc')

s = 1 + '1'

f = open("fffilename.txt")

print('需要处理的语句块—前')

print(f.read())

f.close()

except OSError as reason:

print("fileerror")

print("错误的原因是:",str(reason))

except TypeError as reason:

print("fileerror")

print("错误的原因是:",str(reason))

except (OSError,TypeError):

print("fileerror22222!!!")

except: #提醒所有可能出现的错误，但不会显示原因，不推荐

print("fileerror11111!!!")

finally:

print('需要处理的语句块—后')

f.close()

# 丰富的else语句

在python中,else语句除了与if搭配，还能和循环、try异常语句搭配，实现：

1）干完了能怎样，干不完就别想怎样

2）没有问题，那怎样 的功能。如

try:

n=int('abc')

except ValueError as reason:

print("fileerror")

print("错误的原因是:",str(reason))

else:

print("no Error") // 出现异常就不会进入else,正常就会进行else

i=0

for i in range(7): // for循环，如果触发break语句，else不会执行，如果不触发，执行else语句，while循环一样

print("i=",i)

if i == 8:

print("break")

break

else:

print('全部执行完')

print('结束')

# with语句

**with语句**（**用于读写文件，但不需要f.close()，with会自动关闭**）

try:

with open(r'C:\Users\Administrator\Desktop\ddata.txt','w') as f:

for each in f:

print(each)

except OSError as reason:

print("Error:"+str(reason))

# 图形用户界面

**图形用户界面：EasyGUI （需要使用时再去看）**

先下载该模块：pip install easygui

>>> import easygui

>>> easygui.msgbox('hello world!')

'OK'

>>> from easygui import \*

>>> easygui.msgbox('hi!lsl')

>>> import easygui as g //推荐的导入方法

>>> g.msgbox("你好！美女！")

# 面向对象编程class

class Wugui: //定义类，**封装**

#attributions //属性

color ='green'

legs=4

#method //方法

def climb(self): // **self 相当于 C中的 this 指针！**

print("I am climbing")

def run(self):

print("I am running")

tt=Wugui() //实例化

tt.run() //调用

class Xiaowugui(Wugui): //类的**继承**

#attributions //属性

color ='green'

legs=4

#method //方法

def climb(self):

print("I am climbing")

def run(self):

print("I am running")

t1=Wugui() //实例化

t1.run() //调用

t2=Xiaowugui() //实例化

t2.run() //调用 //不同的run()函数，**多态**

class Ball:

def \_\_init\_\_(self, name): //\_\_init\_\_默认构造函数，前后两个下划线

self.name = name

def kick(self):

print("I am %s!" % self.name)

a=Ball('A') //构造函数进行初始化

b=Ball('B')

a.kick()

b.kick()

# 公有与私有

**c**lass Ball:

typeball = 'basketball' //一般都是公有成员

\_\_ower = 'Luo silong' //双下划线：声明私有成员

def \_\_init\_\_(self, name):

self.name = name

def kick(self):

print("I am %s!" % self.name)

def getower(self): //通过成员函数获得私有属性

return self.\_\_ower

a=Ball('A')

b=Ball('B')

a.kick()

b.kick()

print(a.typeball) //公有成员外部随便使用

#a.\_\_ower //私有成员外部无法使用

print("%s" % a.getower()) //通过公有成员函数访问私有成员

print(a.\_Ball\_\_ower) //通过”类的实例.\_类名\_\_私有属性”方法访问私有属性

------------**可以看见，python不是严格封装的！！！！这与C有很大区别**----------------------

# 类的继承

class SmallBall(Ball): //继承

colorball = 'yellow'

def hello(self):

print("hello world")

def kick(self): //创造与父类一样的函数

print("I am SON method")

c=SmallBall('C') //可以跟父类一样初始化

print(c.colorball)

print(c.typeball) //也可以使用父类的属性

c.hello()

c.kick() //父类的同名成员函数被屏蔽，调用了自己的成员函数

class VerySmallBall(Ball): //继承

pass //**pass用于纯粹继承**父类，没有新的属性

# super关键字

**super关键字：用于继承（多重继承）**

class Ball: //基类（父类）

typeball = 'basketball'

\_\_ower = 'Luo silong'

def \_\_init\_\_(self, name):

self.name = name

def kick(self):

print("I am %s!" % self.name)

def getower(self):

return self.\_\_ower

a=Ball('A')

b=Ball('B')

a.kick()

b.kick()

print(a.typeball)

#a.\_\_ower

print("%s" % a.getower())

print(a.\_Ball\_\_ower)

class SmallBall(Ball): **//子类**

def \_\_init\_\_(self,name,sizeball): //子类中增加了一些私有属性

#super(SmallBall,self).\_\_init\_\_(name) //使用super使子类可以使用父类的私有属性

super().\_\_init\_\_(name) //**python 3.x版本用法**

self.sizeball = sizeball

colorball = 'yellow'

def hello(self):

print("hello world")

def kick(self): //子类使用父类的同名函数，但增加了功能

#super(SmallBall,self).kick() //使用super使子类可以使用父类的同名函数

**super().kick()** //**python 3.x版本用法**

print("I am SON method")

c=SmallBall('C',10)

print(c.colorball)

print(c.typeball)

c.hello()

c.kick()

print(c.sizeball)

print("%s" % c.name)

# 多重继承

class Base1:

def foo1(self):

print("I am Base1!")

class Base2:

def foo2(self):

print("I am Base2!")

class C(Base1,Base2): //多重继承

pass

c = C() //子类可以实现两个父类的方法

c.foo1()

c.foo2()

# 类的组合

**类的组合：横向关系的类**

class Turtl:

def \_\_init\_\_(self,x):

self.num=x

class Fish:

def \_\_init\_\_(self,x):

self.num=x

class Pool:

def \_\_init\_\_(self,x,y):

self.turtle = Turtl(x)

self.fish = Fish(y)

def printnum(self):

print('There are %d in the pool' % (self.turtle.num+self.fish.num))

#t=Turtl(5) //不需要实例化

#f=Fish(6)

p=Pool(5,6) //实例化Pool时，会在之后实例化Turtl和Fish

p.printnum()

# 类、类对象和实例对象

>>> class C:

def x(self):

print('X-man!')

>>> c=C()

>>> c.x() //调用x()函数

X-man!

>>> c.x = 1 //更改x为属性

>>> c.x //操作可行！

1

>>> c.x() //**属性会覆盖方法！因此，属性名和方法名不要相同！**

Traceback (most recent call last):

File "<pyshell#8>", line 1, in <module>

c.x()

TypeError: 'int' object is not callable

# 绑定

>>> class BB:

def printBB(): // 注意：这里没有self参数！

print('no zuo no die!')

>>> BB.printBB() //类对象！可以调用自己的函数

no zuo no die!

>>> bb=BB() //实例对象，不可以调用函数

>>> bb.printBB()

Traceback (most recent call last):

File "<pyshell#15>", line 1, in <module>

bb.printBB()

TypeError: printBB() takes 0 positional arguments but 1 was given

>>> class CC:

def setxy(self,x,y):

self.x=x

self.y=y

def printxy(self):

print(self.x,self.y)

>>> dd=CC() //实例对象不进行初始化

>>> dd.\_\_dict\_\_ //实例对象没有属性

{}

>>> CC.\_\_dict\_\_ //类对象有属性

mappingproxy({'\_\_module\_\_': '\_\_main\_\_', 'setxy': <function CC.setxy at 0x0000000002EF05E8>, 'printxy': <function CC.printxy at 0x0000000002EF0678>, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'CC' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'CC' objects>, '\_\_doc\_\_': None})

>>>

>>> dd.setxy(4,5) //实例对象初始化后就有对象了

>>> dd.\_\_dict\_\_

{'x': 4, 'y': 5}

>>> CC.\_\_dict\_\_

mappingproxy({'\_\_module\_\_': '\_\_main\_\_', 'setxy': <function CC.setxy at 0x0000000002EF05E8>, 'printxy': <function CC.printxy at 0x0000000002EF0678>, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'CC' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'CC' objects>, '\_\_doc\_\_': None})

# 类与对象相关的BIF(内置函数)

**issubclass(B,A)**  //检查B类是否是A类的子类，是的返回True

**isinstance(obj,A)**  //检查实例obj是否是A类的实例，是返回True

**hasattr(obj,’attr’)**  //检查实例obj是否有属性attr

>>> class C:

def \_\_init\_\_(self, x = 0):

self.x = x

>>> c1=C()

>>> hasattr(c1,'x')

True

**getattr(obj,’attr’)**

**setattr(obj,’new\_attr’)**

**delattr(obj,’attr’)**

>>> class C:

def \_\_init\_\_(self, x = 0):

self.x = x

>>> c1=C()

>>> hasattr(c1,'x')

True

>>> getattr(c1,'x')

0

>>> getattr(c1,'y')

Traceback (most recent call last):

File "<pyshell#44>", line 1, in <module>

getattr(c1,'y')

AttributeError: 'C' object has no attribute 'y'

>>> setattr(c1,'y')

Traceback (most recent call last):

File "<pyshell#45>", line 1, in <module>

setattr(c1,'y')

TypeError: setattr expected 3 arguments, got 2

>>> setattr(c1,'y','fish')

>>> c1.y

'fish'

>>> delattr(c1,'y')

>>> c1.y

Traceback (most recent call last):

File "<pyshell#49>", line 1, in <module>

c1.y

AttributeError: 'C' object has no attribute 'y'

# property()描述符

>>> class C:

def \_\_init\_\_(self,x = 10):

self.x = x

def getx(self):

return self.x

def setx(self,v):

self.x = v

def delx(self):

del self.x

**s = property(getx,setx,delx) //通过属性去设置属性、修改属性和删除属性，作用：中转，便于向用户提供稳定的接口（即函数可以随意修改，但提供的s不需要变化）**

>>> c=C()

>>> c.getx()

10

>>> c.s //从**property（）函数中调用属性**

10

>>> c.s = 20 //从**property（）函数中修改属性**

>>> c.getx()

20

>>> del c.s //从**property（）函数中删除属性**

>>> c.s

Traceback (most recent call last):

File "<pyshell#87>", line 1, in <module>

c.s

File "<pyshell#80>", line 5, in getx

return self.x

AttributeError: 'C' object has no attribute 'x'

**构造和析构**

\_\_init\_\_()

\_\_new\_\_() //构造器

\_\_del\_\_() //析构器

>>> class C:

def \_\_init\_\_(self):

print("\_\_init\_\_")

def \_\_del\_\_(self):

print("\_\_del\_\_")

# 工厂函数(魔法方法)

**以双下划线开始的函数（python的内置函数！），一切习以为常的东西都可以进行修改，如运算符，., =, getattr()的功能**

运算符重载？？？

>>> class New\_int(int):

def \_\_add\_\_(self,other): //重载加法，**比C++简洁很多**！

return int.\_\_sub\_\_(self,other)

def \_\_sub\_\_(self,other):

return int.\_\_add\_\_(self,other) //重载减法

>>> a=New\_int(3)

>>> b=New\_int(5)

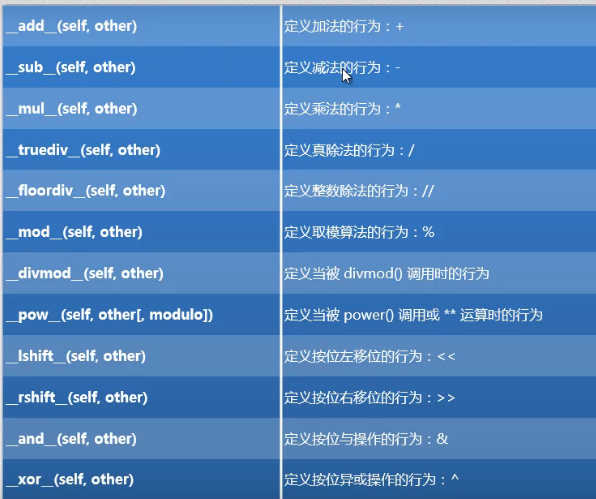
>>> a+b

-2

>>> a-b

8

//-----------------------------即操作符**+，-，\*，/, %**的操作都可以自己进行设计------------------



（**还有很多工厂函数，百度**）

**描述符（魔法方法）**

**递归迭代的bug:**

class Rec:

def \_\_init\_\_(self,width = 0,height = 0): //构造函数赋值

self.width = width

self.height = height

def \_\_setattr\_\_(self,name,value):

if name == 'square':

self.width = value

self.height = value

else:

#self.name = value //设置属性赋值，会调用构造函数赋值，构造函数又会调用设置属性，从而递归迭代出错！

super().\_\_setattr\_\_(name,value) //使用其基类进行赋值，避免重复调用自身的错误！

def getA(self):

return self.width \* self.height

//-----------------------------即**getattr，setattr和delattr**的操作都可以自己进行设计------------------

# 描述符

将一些类的实例对象作为其他类的属性

\_\_get\_\_(self,instance,owner) //等价于访问号“.”，如ob.x, 使用时就会访问该属性，返回属性的值

\_\_set\_\_(self,instance,value) //等价于赋值好“=”，如ob.x=10,在属性分配操作中调用，不返回任何内容

\_\_delete\_\_(self,instance) //等价于删除号，如del ob.x，删除操作，无返回值

//-----------------------------即属性**赋值，取值和删除**的操作都可以自己进行设计------------------

>>> class MyDec: //**描述符类：含有\_\_set\_\_,\_\_get\_\_,\_\_delete\_\_的一个或几个函数**

def \_\_get\_\_(self,instance,owner):

print("getting...",self,instance,owner)

def \_\_set\_\_(self,instance,value):

print("setting...",self,instance,value)

def \_\_delete\_\_(self,instance):

print("deleting...",self,instance)

>>> class Test:

x=Mydec() //x为MyDec描述符类的实例，并且其实现了\_\_set\_\_,\_\_get\_\_,\_\_delete\_\_的一个或几个，因此x就可以作为Test类的描述符。即：

test = Test()

test.x //调用描述符x的get操作

test.x=’luo’ //调用描述符x的set操作

del test.x //调用描述符x的delete操作

**property()函数就是使用描述符实现的！x=property(setx,getx,delx),其中“=”的功能被重新设计！**

**定制容器的协议（修改容器的内置函数）：协议类似于C#中的接口**

**容器的内置函数，如：**

\_\_len\_\_() len(obj)

\_\_getitem\_\_() ob[i]

\_\_setitem\_\_() ob[i] = value

\_\_delitem\_\_() del ob[i]

**迭代器**

**1) iter() //生成一个指向元素的“指针”**

**2) next() //提取元素的值**

>>> string = "Luo SILONG"

>>> it=iter(string)

>>> next(it)

'L'

>>> next(it)

'u'

>>> next(it)

'o'

>>> next(it)

' '

>>> next(it)

'S'

>>> next(it)

'I'

>>> next(it)

'L'

>>> next(it)

'O'

>>> next(it)

'N'

>>> next(it)

'G'

>>> next(it)

Traceback (most recent call last):

File "<pyshell#24>", line 1, in <module>

next(it)

StopIteration

For循环的while改写：

>>> while True:

try:

each = next(it)

except StopIteration:

break

print(each)

迭代器的魔法方法：

**\_\_iter\_\_()**

**\_\_next\_\_()**

>>> class Fibs:

def \_\_init\_\_(self,n=10):

self.a=0

self.b=1

self.n=n

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self): //改写迭代方法：使用Fabicc数列

self.a, self.b = self.b,self.a+self.b

if self.a > self.n:

raise StopIteration

return self.a

>>> fib=Fibs()

>>> for i in fib:

print(i)

# 生成器即和推导表达式

作用于协同程序：函数可以暂停或者挂起，并在需要的时候继续或者重新开始！

作用：能够产生一个序列的函数输出。例如，确定range(10000000000000000000000000000000000000000000000000000000000000000000000000000)里面的素数？怎么办？

**常规思想：**通过迭代的方法一次性计算出所有的素数，然后保存至内存起来，调用时通过遍历的方式！

**问题：**数据量太大，计算太花时间，占用大量内存！

**思路：**通过生成器将求素数函数挂机，当需要是输出一个即可！极大地减少了内存的占有量！

总结：

* generator 是用来产生一系列值的
* yield 则像是 generator 函数的返回结果
* yield 唯一所做的另一件事就是保存一个 generator 函数的状态
* generator 就是一个特殊类型的迭代器（iterator）
* 和迭代器相似，我们可以通过使用 next() 来从 generator 中获取下一个值
* 通过隐式地调用 next() 来忽略一些值

>>> def myGen():

print("生成器被执行！")

yield 1 //**生成器yield相当于一个return,但它可以在当此执行时挂起，并在下一次调用时执行后面的语句**

yield 2

>>> myG = myGen()

>>> next(myG) //返回yield 1

生成器被执行！

1

>>> next(myG) //直接返回yiled 2

2

>>>

>>> next(myG) //不存在yield了，出错

Traceback (most recent call last):

File "<pyshell#9>", line 1, in <module>

next(myG)

StopIteration

>>>

**生成器生成斐波那契数列：**

>>> def libs():

a=0

b=1

while True: //不用担心无限循环，因为生成器会挂起

a,b=b,a+b

yield a

>>> for i in libs():

if i > 100:

break

print(i,end=" ")

1 1 2 3 5 8 13 21 34 55 89

**列表推导式：**

>>> a = [i for i in range(100) if not (i % 2) and i % 3] //100以内能被2整除但不能被3整除的数

>>> a

[2, 4, 8, 10, 14, 16, 20, 22, 26, 28, 32, 34, 38, 40, 44, 46, 50, 52, 56, 58, 62, 64, 68, 70, 74, 76, 80, 82, 86, 88, 92, 94, 98]

**字典推导式：**

>>> b={i:i%2 == 0 for i in range(10)} //判断10以内数的奇偶性

>>> b

{0: True, 1: False, 2: True, 3: False, 4: True, 5: False, 6: True, 7: False, 8: True, 9: False}

**集合推到式：**

>>> c = {i for i in [1,1,2,3,4,5,6,7,8,3,2,1]}

>>> c

{1, 2, 3, 4, 5, 6, 7, 8}

**字符串推导式？不存在的！**

>>> d = 'i for i in "I love you!"'

>>> d

'i for i in "I love you!"'

**元组推导式：以生成器方式存在**

>>> e = (i for i in range(10))

>>> e

<generator object <genexpr> at 0x00000000030FB948>

>>> next(e)

0

>>> next(e)

1

>>> for each in e:

print(each)

2

3

4

5

6

7

8

9

>>> sum((i for i in range(100) if i % 2)) //**100以内的奇数的和**

2500

>>> sum((i for i in range(100) if i % 2)) //**简化括号**

2500

# 模块

* **容器：数据的封装**
* **函数：语句的封装**
* **类：方法和属性的封装**
* **模块：模块，就是程序，及.py文件**

注意：当程序想作为模块，那么**程序文件.py和python.exe**在一个目录内！或者sys.path.append()增加搜索路径！

## 1.模块的导入方法：

#Method import mokuai

#method 1 //不推荐方法1，失去了模块命名空间的优势

#from test1 import hello //从模块中导入指定函数

#from test1 import \* //从模块中导入所有函数

#hello()

#method 2 //直接导入模块

#import test1

#test1.hello()

#method 3 //导入模块，设置为别名

import test1 as tt

tt.hello()

## 2.模块的封装注意

**if \_\_name\_\_== "\_\_main\_\_" 的用法**

***编写私有化部分 ，这句代码以上的部分，可以被其它的调用，以下的部分只有这个文件自己可以看见，如果文件被调用了，其他人是无法看见并调用私有化部分的***

**假设test1.py为设计的模块**

def hello(): #Modules

x=10

print("-----------hello world!--------------")

return x

def test\_hello(): #Modules test

print("Test:hello():")

print(hello())

if \_\_name\_\_== "\_\_main\_\_": //**当模块test1.py被直接运行时，以下代码块将被运行，当模块是被导入时，代码块不被运行。**

test\_hello()

**test2.py为需要test1.py模块的函数**

import test1 as tt

tt.hello()

## 3.模块的搜索路径

**sys.path.append(r"E:\ofAnaconda\Lib\site-packages")**

**//---增加Anaconda下载的安装包！！！**

前面已经介绍了一种模块的调用前提，即：**程序文件.py和python.exe**在一个目录内！但是，也可以通过**增加python的搜索路径**实现模块的自定义存放位置，例如导入所下载的anaconda包！

import sys //导入系统模块

print(sys.path) //查看当前系统搜索的路径

sys.path.append(r"E:\ofPython\ofInstallPython\myModels") //**增加搜索路径，例如将上上述写的test1.py存放在myModels文件夹下**

import test1 as tt //可以导入模块了

tt.hello()

# 4.包 package

将同一类的模块或者为实现一个功能的模块函数放到一个文件夹内，作为一个包。**例如将上上述写的test1.py存放在myModels文件夹下的M1文件夹内。**

1.创建一个文件夹

2.在文件夹里创建一个**\_\_init\_\_.py**文件（**可以是空文件**，即告知python这个文件夹是一个模块包！）

3.导入包的模块 **’import 包名.模块名 as 别名’**

import M1.test1 as tt //从包里导入模块

tt.hello()

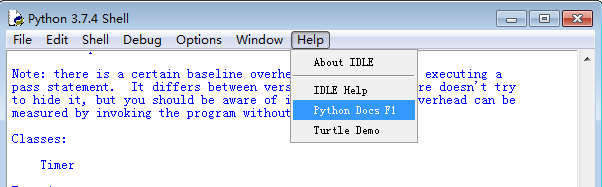
# 像极客一样思考

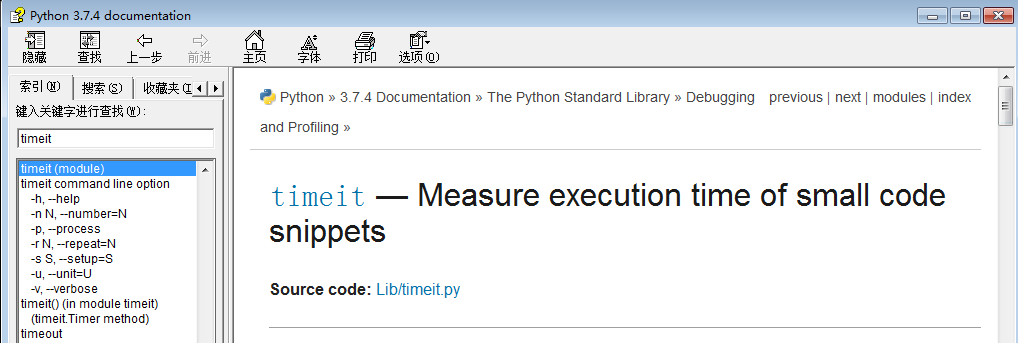
啥都要自己写？不存在的？获取别人已经完成的模块和功能！Pypi模块社区和Github！

<https://pypi.org/>

怎么使用别人模块：

**方法1：去Shell的官方文档搜索**





**方法2：**

help（模块名）

**方法3：**

>>> import timeit //**导入模块后，查看模块的信息和使用方法**

>>> print(timeit.\_\_doc\_\_)

>>> dir(timeit) //查看模块包含的内容

>>> timeit.\_\_all\_\_ //过滤掉不需要的属性，**展示模块作者希望外界调用的函数或类**

['Timer', 'timeit', 'repeat', 'default\_timer']

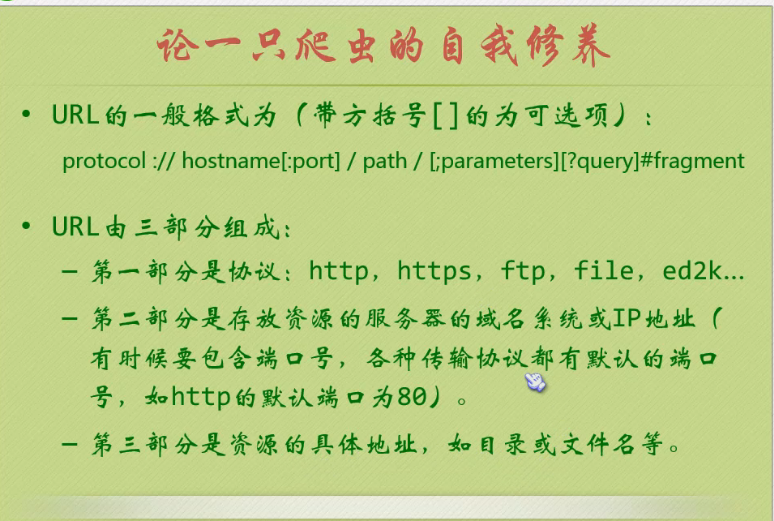
>>> import timeit

>>> timeit.\_\_file\_\_ //寻找源代码的路径

'E:\\ofPython\\ofInstallPython\\lib\\timeit.py'

# Python爬虫

**做百度引擎？不存在的！使用python提取网页上的各种所需数据！（爬取美女图片！）**



**爬虫所需包：urllib**

>>> import urllib.request

>>> respond = urllib.request.urlopen(r'https://fishc.com.cn/')

>>> html = respond.read()

>>> print(html) //打印出网页html代码（二进制数据），但格式比较乱

**以下位置查看网页的数据编码：**



>>> html = html.decode('gbk') //**按其元数据格式进行解码**

>>> print(html)

**下载一只猫**

import urllib.request

方法1：

response = urllib.request.urlopen('https://placekitten.com/500/600')

cat\_img = response.read()

with open('cat\_500\_600.jpg','wb') as f:

f.write(cat\_img)

方法2：

req = urllib.request.Request('https://placekitten.com/300/200')

response = urllib.request.urlopen(req)

cat\_img = response.read():

with open('cat\_300\_200.jpg','wb') as f:

f.write(cat\_img)

>>> response.geturl()

'https://placekitten.com/300/200'

>>> response.info()

<http.client.HTTPMessage object at 0x0000000003278508>

>>> print(response.info())

Date: Wed, 31 Jul 2019 06:10:53 GMT

Content-Type: image/jpeg

Transfer-Encoding: chunked

Connection: close

Set-Cookie: \_\_cfduid=ded33c6f6db78e59370611a641d15a5891564553453; expires=Thu, 30-Jul-20 06:10:53 GMT; path=/; domain=.placekitten.com; HttpOnly

Access-Control-Allow-Origin: \*

Cache-Control: public, max-age=86400

Expires: Thu, 01 Aug 2019 06:10:53 GMT

CF-Cache-Status: HIT

Age: 14448

Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"

Vary: Accept-Encoding

Server: cloudflare

CF-RAY: 4fed696d6e866cec-SJC

# 正则表达式

## 1. 匹配IP地址

从一大串繁乱的字符串里面找到具有一定规则的子字符串。

>>> import re

>>> re.search(r'FishC','i LOVE FishC.com!')

<re.Match object; span=(7, 12), match='FishC'>

>>> 'i LOVE FishC.com!'.find('FishC')

7

>>> re.search(r'.','i LOVE FishC.com!')

<re.Match object; span=(0, 1), match='i'>

>>> re.search(r'FishC.','i LOVE FishC.com!')

<re.Match object; span=(7, 13), match='FishC.'>

>>> re.search(r'\.','i LOVE FishC.com!')

<re.Match object; span=(12, 13), match='.'>

>>> re.search(r'\d','i LOVE FishC.com 123!')

<re.Match object; span=(17, 18), match='1'>

>>> re.search(r'\d\d','i LOVE FishC.com 123!')

<re.Match object; span=(17, 19), match='12'>

>>> re.search(r'\d\d\d\.\d\d\d\.\d\d\d\.\d\d\d','i LOVE 1.125.45.12 FishC.com!')

>>> re.search(r'\d\d\d\.\d\d\d\.\d\d\d\.\d\d\d','i LOVE 192.125.145.102 FishC.com!')

<re.Match object; span=(7, 22), match='192.125.145.102'>

>>> re.search(r'[aeiou]','I love you very much!124')

<re.Match object; span=(3, 4), match='o'>

>>> re.search(r'[aeiouAEIUO]','I love you very much!124')

<re.Match object; span=(0, 1), match='I'>

>>> re.search(r'[a-z]','I love you very much!124')

<re.Match object; span=(2, 3), match='l'>

>>> re.search(r'[0-9]','I love you very much!124')

<re.Match object; span=(21, 22), match='1'>

>>> re.search(r'[3-9]','I love you very much!124')

<re.Match object; span=(23, 24), match='4'>

>>> re.search(r'ab{3}c','abbbc')

<re.Match object; span=(0, 5), match='abbbc'>

>>> re.search(r'ab{3,10}c','abbbbbbbbc45')

<re.Match object; span=(0, 10), match='abbbbbbbbc'>

>>> re.search(r'[0-255]','188')

<re.Match object; span=(0, 1), match='1'>

>>> re.search(r'[0-2][0-5][0-5]','188')

>>> re.search(r'[01]\d\d|2[0-4]\d|25[0-5]','235')

<re.Match object; span=(0, 3), match='235'>

>>> re.search(r'(([01]\d\d|2[0-4]\d|25[0-5])\.){3}([01]\d\d|2[0-4]\d|25[0-5])','i LOVE 1.125.45.12 FishC.com!')

>>> re.search(r'(([01]{0,1}\d{0,1}\d|2[0-4]\d|25[0-5])\.){3}([01]{0,1}\d\{0,1}d|2[0-4]\d|25[0-5])','i LOVE 1.125.45.12 FishC.com!')

>>> re.search(r'(([01]{0,1}\d{0,1}\d|2[0-4]\d|25[0-5])\.){3}([01]{0,1}\d{0,1}\d|2[0-4]\d|25[0-5])','i LOVE 1.125.45.12 FishC.com!')

<re.Match object; span=(7, 18), match='1.125.45.12'>

# GUI的终极选择——tkinter

import tkinter as tk

class APP:

def \_\_init\_\_(self,master):

frame = tk.Frame(master)

frame.pack(side=tk.LEFT,padx=10,pady=10)

self.hi\_there = tk.Button(frame,text='Hi!Nice to meet you.',fg = 'blue',bg='black',command=self.say\_hi)

self.hi\_there.pack(side=tk.LEFT,padx=50,pady=50)

def say\_hi(self):

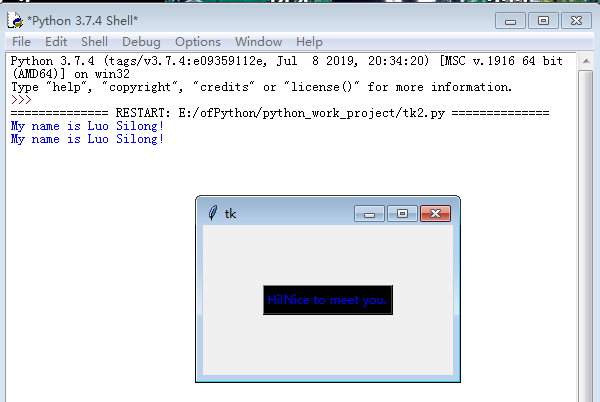
print('My name is Luo Silong!')

if \_\_name\_\_ == '\_\_main\_\_':

root = tk.Tk()

app = APP(root)

root.mainloop()



from tkinter import \*

root = Tk()

photo = PhotoImage(file = r'C:\Users\Administrator\Desktop\VSlogo.gif')

theLabel = Label(root,

text = "Learn Python\n to FishC",

justify = LEFT,

image = photo,

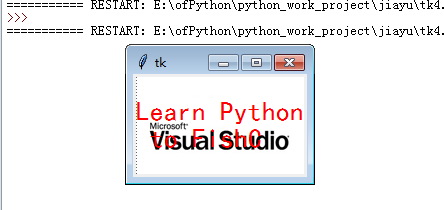
compound = CENTER,

font = ('楷体',20),

fg = 'red')

theLabel.pack()

mainloop()



from tkinter import \*

def callback():

var.set("I don't believe you!EXIT!")

root = Tk()

frame1 = Frame(root)

frame2 = Frame(root)

var = StringVar()

var.set("This is a label of textLabel!\nPlease leave right now!")

textLabel = Label(frame1,

textvariable=var,

justify=LEFT,

padx=10,

pady=20)

textLabel.pack(side=LEFT)

photo = PhotoImage(file = r'C:\Users\Administrator\Desktop\VSlogo.gif') #support .gif

imgLabel = Label(root,image=photo)

imgLabel.pack(side=RIGHT)

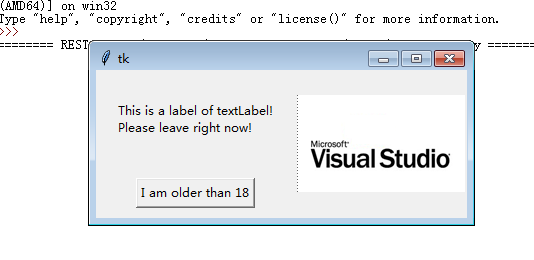
theButton = Button(frame2,text='I am older than 18',command=callback)

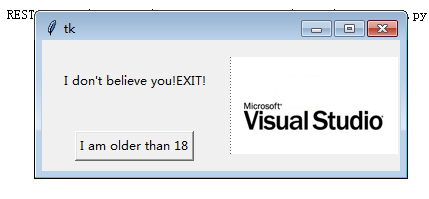
theButton.pack()

frame1.pack(padx=10,pady=10)

frame2.pack(padx=10,pady=10)

mainloop()





from tkinter import \*

root = Tk()

GIRLS = ['西施','貂蝉','王昭君','Luosilong']

v = []

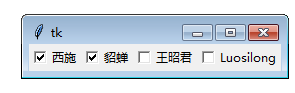
for girl in GIRLS:

v.append(IntVar())

b = Checkbutton(root,text= girl,variable = v[-1])

b.pack(side=LEFT)

mainloop()



from tkinter import \*

root = Tk()

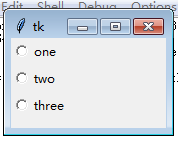
v = IntVar()

Radiobutton(root,text= 'one',variable = v,value=1).pack(anchor=W)

Radiobutton(root,text= 'two',variable = v,value=2).pack(anchor=W)

Radiobutton(root,text= 'three',variable = v,value=3).pack(anchor=W)

mainloop()



from tkinter import \*

root = Tk()

LANGS = [

('Python',1),

('C++',2),

('JAVE',3),

('PHP',4)

]

v = IntVar()

v.set(1)

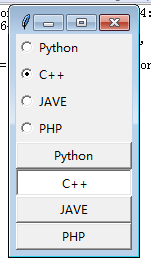
for lang,num in LANGS:

b = Radiobutton(root,text= lang,variable = v,value=num).pack(anchor=W)

for lang,num in LANGS:

c = Radiobutton(root,text= lang,variable = v,value=num,indicatoron=False).pack(fill=X)

mainloop()



from tkinter import \*

root = Tk()

group = LabelFrame(root,text="最好的脚本语言是？",padx=5,pady=5)

group.pack(padx=10,pady=10)

LANGS = [

('Python',1),

('C++',2),

('JAVE',3),

('PHP',4)

]

v = IntVar()

v.set(1)

for lang,num in LANGS:

c = Radiobutton(group,text= lang,variable = v,value=num,indicatoron=False).pack(fill=X)

mainloop()

