Python

## Python的发展史：

### Python之父的发展史：Guido van Rossum

Guido van Rossum：名副其实的荷兰人,龟叔

1982年从阿姆斯特丹大学取得了数学和计算机硕士学位。

20世纪80年代中期，Python之父Guido van Rossum还在CWI(数学和理论计算机科学领域的研究中心，位于阿姆斯特丹)，那个时候他Guido构思了一门致力于解决问题的编程语言。他觉得现有的编程语言对非计算机专业的人十分不友好。

1989年12月份，在媳妇回娘家休假的期间，为了打发无聊 的圣诞节假期，Guido开始写Python的第一个版本。经过四周的研究终于研究出了python的第一个版本。（值得一提的是Python这个名字的由来，Python有蟒蛇的意思，但Guido起这个名字完全和蟒蛇没有关系。当Guido在实现Python的时候，他还阅读了Monty Python's Flying Circus《巨蟒剧团之飞翔的马戏团》的剧本，这是来自一部来自20世纪70年代的BBC喜剧。Guido认为他需要一个简短、独特且略显神秘的名字，因此他决定将该语言称为Python。

）





|  |  |
| --- | --- |
| 版本 | 特性 |
| 1994年1月发布Python1.0版本。 | 主要新功能是lambda, map, filter和reduce |
| 2000年10月，Python2.0发布 | 主要新功能是内存管理和循环检测垃圾收集器以及对Unicode的支持 |
| 2008年的12月份，Python3.0发布 | Python3.x不向后兼容Python2.x，这意味着Python3.x可能无法运行Python2.x的代码。Python3代表着Python语言的未来。 |
| 官网 | [www.python.org](http://www.python.org)  com:国际域名  cn  .net 网络服务商  .gov.cn  Org组织 |

## Python的应用领域

### 1.Web开发（facebook,twitter,知乎）

尽管今天PHP依然是Web开发的流行语言，但Python上升势头更劲。随着Python的Web开发框架逐渐成熟，比如耳熟能详的Django和flask你可以快速地开发功能强大的Web应用。小编我首推Django，强烈建议所以有志于从事Python Web开发的人掌握这门框架。无论是建大型网站，开发OA或Web API，Django都可以轻松胜任。

### 2.网络爬虫（python专业）

多亏有了Python，一个小白用几行代码就可以写个爬虫爬段子爬福利图了。然而爬虫的真正作用是从网络上获取有用的数据或信息，可以节省大量人工时间。能够编写网络爬虫的编程语言有不少，但Python绝对是其中的主流之一。Python自带的urllib库，第三方的requests库和Scrappy框架让开发爬虫变得非常容易。

### 3.计算与数据分析（python专业）

随着NumPy，SciPy，Matplotlib等众多程序库的开发和完善，Python越来越适合于做科学计算和数据分析了。它不仅支持各种数学运算，还可以绘制高质量的2D和3D图像。和科学计算领域最流行的商业软件Matlab相比，Python比Matlab所采用的脚本语言的应用范围更广泛，可以处理更多类型的文件和数据。

### 4.人工智能（AI：python）

当前最热工资最高的IT工作就是从事人工智能领域的工程师了。Python在人工智能大范畴领域内的机器学习、神经网络、深度学习等方面都是主流的编程语言，得到广泛的支持和应用。最流行的神经网络框架如Facebook的PyTorch和Google的TensorFlow都采用了Python语言。你不学Python,你会用那些框架吗?

### 5.自动化运维

这几乎是Python应用的自留地，作为运维工程师首选的编程语言。在很多操作系统里，Python是标准的系统组件。大多数Linux发行版和MacOSX都集成了Python，可以在终端下直接运行Python。Python标准库包含了多个调用操作系统功能的库。通过pywin32这个第三方软件包，Python能够访问Windows的COM服务及其它WindowsAPI。使用IronPython，Python程序能够直接调用.NetFramework。一般说来，Python编写的系统管理脚本在可读性、性能、代码重用度、扩展性几方面都优于普通的shell脚本。

### 6.云计算

Python的最强大之处在于模块化和灵活性，而构建云计算的平台的IasS服务的OpenStack就是采用Python的，云计算的其他服务也都是在IasS服务之上的。

### 7.网络编程

Python提供了丰富的模块支持sockets编程，能方便快速地开发分布式应用程序。很多大规模软件开发计划例如Zope，Mnet, BitTorrent和Google都在广泛地使用它。

### 8.游戏开发（LOL,炉石传..c++，c,python）

很多游戏使用C++编写图形显示等高性能模块，而使用Python或者Lua编写游戏的逻辑、服务器。相较于Python，Lua的功能更简单、体积更小，然而Python则支持更多的特性和数据类型。Python的PyGame库也可用于直接开发一些简单游戏。

以上就是Python八大主要应用领域，你都知道吗？总之，Python是一门新手友好、功能强大、高效灵活的编程语言，学会之后无论是想进入数据分析、人工智能、网站开发这些领域，还是希望掌握第一门编程语言，都可以用Python来开启未来无限可能！

## Python的就业方向

**发展前景一：Linux运维**

**发展前景二：Python Web网站工程师**

**发展前景三：Python自动化测试**

**发展前景四：数据分析**

**发展前景五：人工智能**

语言之间的区别：

解释型语言：python

编译型语言：java 🡪 编程机器可以识别的语言🡪执行语言

## Python的编程环境的安装

1. 安装python开发工具包
2. 安装IDEA

安装注意：

1. 不要有中文：极有可能编程乱码
2. 不要有空格：防止解析路径只解析到空格处停止。

## Python入门而谈：

1. 我们学python2 or 3呢？
   1. 2.x版本是过去式，3.x是未来的一种语言，一种去向。
   2. 2.x不会在有大的变化，3.x的库的更新，只会在3.x上更新。
      1. 因为：要清理和整理2.x。所以3.x和2.x版本就不会进行兼容。就是说2.x版本开发的程序3.x的解释器不能解释。
   3. 3.x能支持中文编程。----字符编码的问题
   4. 2.x的Twisted的模块3.x不支持的。

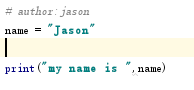
## Python的安装：

* 1. <http://www.python.org/downloads/>
  2. 默认安装路径：c:\python
  3. 然后配置环境变量：
     1. 直接在path变量里新增
     2. 或者间接的新建python\_home变量来存储python安装路径。
  4. Linux、mac系统有自带的python环境。自带的是2.6，请更新至2.7版本。

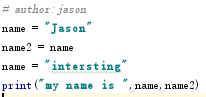
1. 第一个helloworld程序
   1. 在python的交互器里输入(python)
   2. >>>print(“hello world”)
   3. 然后将该命令保存到一个.py格式的文件里。
   4. 用python hello.py执行。
   5. 第一个helloworld程序与其他程序语言的对比。

## 变量、字符编码

* 1. 何为变量：存储数据的变化的量叫变量



* 1. 但是要注意一点:



Name与name2在内存里都指向了Jason,当name2指向了intersting的时候，name却没有改变指向。

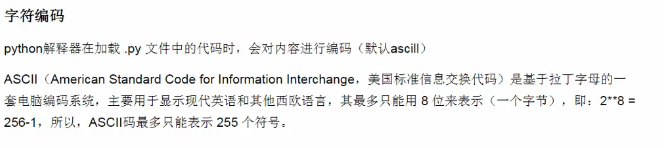
所以打印的结果是：Jason,intersting

1. 变量的命名规则：

\_name

* 1. 多个单词组成用下划线连接
  2. 常量：python没有常量的概念
     1. 常量单词全部大写

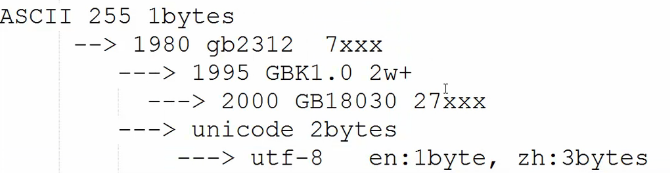
1. 字符编码与二进制
   1. 字符编码





1. 比如日本的Shift\_JIS编码：

image5.png



所以在python2里用中文的话，就在脚本里加一行：

# -\*- coding:utf-8 -\*-即可

就是告诉解释器用utf-8编码来执行源代码。

## 注释

# : 单行注释

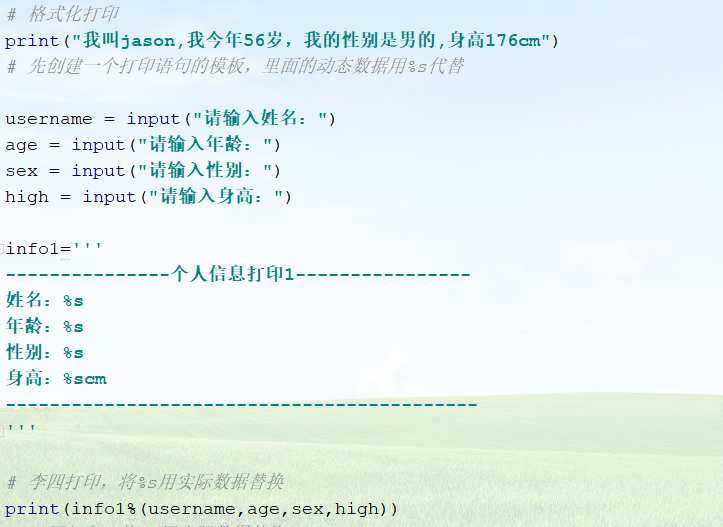
’’’ ’’’:多行注释，三引号

### 用户的交互程序

如何实现键盘输入：input()

Int（）:强制类型转换为整型

Float():强制类型转换为浮点类型



## 格式化打印

### 一、%s格式化打印

|  |
| --- |
| info1=**''' ---------------个人信息打印1---------------- 姓名：%s 年龄：%s 性别：%s 身高：%scm ------------------------------------------- '''** *# 李四打印，将%s用实际数据替换* print(info1%(username,age,sex,high)) |

### 二、format格式化方式

|  |
| --- |
| print(**"-----------------------2--------------------------------"**) info2 = **''' -----------------个人信息打印2---------------------  年龄：{age}，  性别：{sex}，  身高：{high}cm,  姓名：{username}, '''** print(info2.format(username=**"张三"**,age=45,sex=**"男"**,high=178)) |

### 三、{n}格式化打印

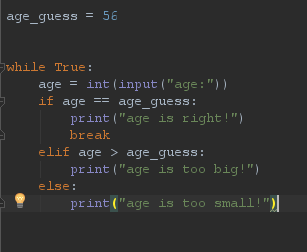
|  |
| --- |
| info2 = **''' -----------------个人信息打印3---------------------    姓名：{0},  年龄：{1}，  性别：{2}，  身高：{3}cm,  姓名：{0},   '''** print(info2.format(**"张三"**,45,**"男"**,178)) |

## If else 流程判断

### 一个简单的if…else

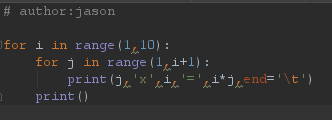
|  |
| --- |
| *# -\*- coding:utf-8 -\*- # author:jason  '''  程序结构：  循序执行：  选择结构：if...else...  单分支：if...  双分支：  多分支：  循环结构： '''* age = 20 if age >= 18:  print(**"可以去网吧玩LOL！"**) else:  print(**"小屁孩，回家去，别在这！！！！"**)  print(**"--------------------分数管理系统--------------------"**) score = int(input(**"请录入学生的分数："**)) **'''  90~100:优秀！！！  80~90：良好！  70~80：良！  60~70：很危险！  0~60：小伙子，对不起，您的分数在路上！  其他情况：输入非法！ '''** *# if 的第一种* **''' if score >= 90 and score <= 100:  print("优秀！！！") elif score >= 80 and score < 90:  print("良好！") elif score >= 70 and score < 80:  print("良！！！") elif score >= 60 and score < 70:  print("很危险！！！！") elif score >= 0 and score < 60:  print("小伙子，对不起，您的分数在路上么！!!!") else:  print("输入非法！！！！") '''** *# 简写方式* if score in range(90,101): *# 90~100* print(**"优秀！！！"**) elif score in range(80,90):  print(**"良好！"**) elif score in range(70,80):  print(**"良！！！"**) elif score in range(60,70):  print(**"很危险！！！！"**) elif score in range(0,60):  print(**"小伙子，对不起，您的分数在路上么！!!!"**) else:  print(**"输入非法！！！！"**) |

### 一个简答的while循环：



|  |
| --- |
| *# author:jason* import random **''' 猜数字游戏：  系统会产生一个随机数。  由用户一直去猜，一直猜中为止。  若用户猜的数字过大，提示，过大。  若用户猜的过小，打印，过小。  1.技术选型：  while  input  if  print  random:随机数模块  1.导入模块:import random  2.调用random(),系统会返回一个随机数 '''** *# 系统产生的一个随机* a=int(random.random() \* 100)*# 产生 0~100以内的数* i=0 while True:*# 死循环* i = i + 1 *# 加入了计数器的功能* num=int(input(**"请输入您要猜的数："**))  if num > a:  print(**"过大！"**)  elif num < a:  print(**"过小！"**)  else:  print(**"恭喜你，猜中了！您本次使用"**,i ,**"次！"**)  break *# 猜中后，跳出循环* |

## For



作业：

1. 一：将学的知识写到博客。
2. 二：写一个登录接口
   1. 输入用户名密码
   2. 认证成功后显示欢迎信息
   3. 输错三次后锁定

# 模块初识

## 模块：python的强大之处在于他有非常丰富和强大的标准库和第三方库（需要安装才能使用的像django框架），以后会更深入的讲到各种常用的库，ok,现在我们先来学两个简单的：

### sys库

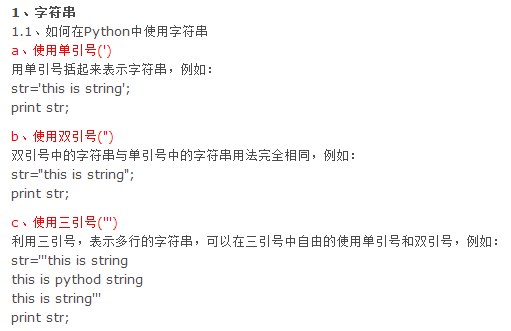
### sys模块的作用：

* sys是system的缩写，用来获取操作系统和编译器的一些配置，设置及操作。

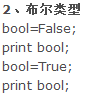
# Python数据类型

## 数据类型

1、字符串



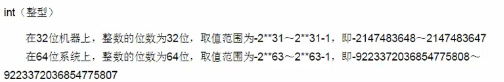
2、布尔类型



3、整数：byte short int long

image15.png

Int （整型）：



Long（长整型）：



严重注意：python3.x版本已将long类型和int类型整合了，当int类型表示不过来了会自动提升为long类型。

4、浮点数:0.36

4.1布尔：bool

比较特殊：False True

5、数字

6、列表:[jason,”陈兴安”,”淘秀琴“]

数据可以修改，替换，删除，增加

7、元组:（jason,”陈兴安”,”陶秀琴”)

数据写死后，不能进行任何修改操作

8、字典：{010:”北京”,020:“上海”,030:”广州”}

格式：新华字典={键1：值1，键2：值2}

新华字典[“键2”]

Float（浮点型）：

带有小数点的数据为浮点类型

## 数据运算：

### 基本运算：

**+ ，-，\* ,/ % // \*\***

**%:取模**

**x\*\*y次幂：x的y次幂**

**//取整除，：取商的整数部分。9//2，整数部分未4.0**

### 运算符

==判断是否等于。比较对象是否相等

!=不等于，比较两个对象是否不相等

<>不等于，比较两个对象是否不相等

>大于，

<小于

>=大于等于

<=小于等于

### 赋值运算

=：简单的赋值运算

+=：加法赋值运算。C+=a 与 c = c +a同效 a += 1 🡪 a = a + 1

-=：减法赋值运算。C-=a 与c= c – a同效

\*=：乘法赋值运算。C\*=a 与 c = c\* a同效

/=：除法赋值运算。c/=a 与 c = c/a同效

\*\*=：幂赋值运算。C\*\*=a 与 c = c \*\*a 同效

//=：取整商赋值运算：c//=a 与 c = c//a同效

### 逻辑运算

(a>6) and (b <= 9)

And 就相当于&& (规则：运算符两边有一个不成立，整体都不成立。)

If(a > 15 and b<20) if(a>15 && b<20)

Or 就相当于||

If(a>12 or a<52) 与其他语言的 if(a>12 || a<52)

Not 就相当于 ！

If( not(a>12 || a< 52)) 与其他语言的 If( ！(a>12 || a< 52))

a = 6 ,b = 7

If (a >=6) and (b <= 7): # True

### 成员运算

in:判断某个值在某个字符序列是否存在

not in:

### 身份运算

is :判断两个标识符是引用自一个对象（可以用id(a) 来查看对象的身份标识号）

not is :

### 位运算符

&：与运算，3 & 5:0011 & 0101 = 0001 = 1 (0b0011 & 0b0101)

| ：或运算，3 | 5 ：0011 | 0101 = 0111 = 7 (0b0011 | 0b0101)

^ :异或运算, 3 ^ 5 : 0011 ^ 0101 = 0110 = 6 (0b0011 ^ 0b0101),相同为0，不同为1

~：取反运算： ~5 ： ~ 0101 = -6

>>:右移运算符： 5 >> 1 : 0101 >> 1 =0010 =2

<<:左移运算符: 5<< 1 : 0101 << 1 = 1010 = 10

### Bytes数据类型

三元运算符：

a,b,c = 5,4,8d = a if a > c else c print(d)

a,b,c = 8,96,52d = a if a>b else (b + 1 if a +5 > b else b) print(d)

二进制：

0b0010110

八进制

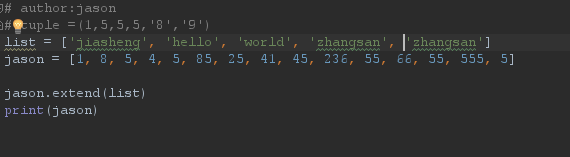
0o01076(是零o开头，不是0开头了)

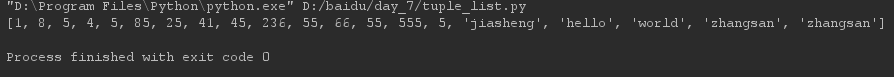
十六进制

0xd0dfd1

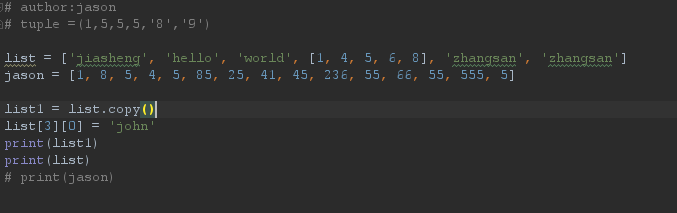
### 元组和列表

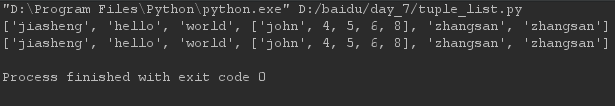
* 元组与列表：
  + 就相当于一个数组。
  + Name\_tuple=(1,”hello”,6355,’intersting’,True)
  + Name\_list = [1,”hello”,6355,’intersting’,True]
* 取值:
  + Name\_tuple[0]
  + Name\_list[0]
* 修改值
  + Name\_tuple是无法修改元组的值
  + Name\_list是可以修改该列表内的值。Name\_list[2] = 63;
* 添加值
  + Name\_tuple：元组不能添加值，初始为多少那么元组里就是多少，后期不能进行添加值。
  + Name\_list:列表可以添加值，需要调用append方法，name\_list.append()，添加的位置是在列表的结尾。
  + Name\_list:列表可以插入值，需要调用insert方法，insert(index,value);需要提供要插入的角标值和实际值
  + 删除值：
    - 对于元组无法删除值
    - 列表的删除值：
      * 删除某个值：name\_list.remove(“张三”);
        + 注意：如果该列表里存在相同的元素，remove方法只删除其中一个并不是删除全部的。
      * 删除某个角标的值：
        + Name\_list.pop();若不提供角标值则默认删除的是该列表的最后一个值。
        + Name\_list.pop(int index); 删除指定角标对应的值。
      * 删除全部列表
        + Name\_list.clear();将列表的元素全部清除。
        + Del name\_list;
  + 列表元素的查找：
    - name\_list.index(‘zhangsan’) 查找元素的角标：
  + 列表元素的统计
    - Name\_\_list.count(“zhangsan”);统计zhangsan在列表中出现的次数。Count()
  + 排序
    - Sort（）方法，默认从小到大排序
    - 注意：排序方法只能排序列表中元素为相同类型的列表。
  + 列表的扩展
    - 如果想让其他列表中的元素想添加到此列表中，用扩展即可
    - Extends
    - List1.extend(list)
    - 比如：



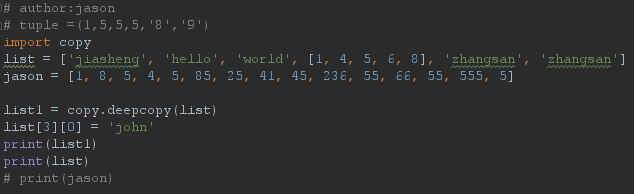


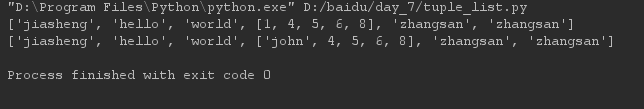
* + 列表的复制：(浅复制和深度复制)
    - 浅copy;对于列表中没有子列表的列表可以用copy。
    - 但是如果有对于列表中有子列表的列表就不能用copy来复制了。





* + - 所以我们用深度copy。
      * import copy
      * 用copy的deepcopy方法
      * List1 = list.deepcopy(list)
      * Print(list1)





所以对于浅copy我们基本上不用。

* + - 列表的切片
      * List = [4,8,7,5,6,9,2,1]
      * List[1:5];表示：从角标为1的元素开始打印一直打印到角标为5-1的元素。
      * 若切片的前面的起始角标不写默认从0开始。
        + List[:5]与list[0:5]是一样的效果。
      * 负数的情况。
        + List[-5:-1],从列表的后面开始往前数分别为-1，-2…….

所以打印结果为：5,6,9,2

* + - * 如果在打印切片的时候中间隔一个打印一个呢？
        + 那么就在切片的后面再加一个切片：
        + List[0:-2:2]：表示从0角标开始到倒数第三个数结束。中间隔一个数打印一下。

#步长切片 for i in range(0,len(list),2): print(list[i])

浅copy的适用场景：用于创建多人的共同的共享数据

water = ['cup', ['升', 60]]person = water[:] person1 = water[:] person[0] = 'kitty' person1[0] = '史努比'person[1][1] -= 3person1[1][1] -= 6 print(person) print(person1)

就好比，两个人有各自的水杯，但是他俩都用的是同一个饮水机。

那么浅copy就是将杯子的型号copy了一份，但是饮水机还是哪个饮水机。

总结：

* 所以对于元组，是一种只读的列表序列，只有count和index方法。
* 尽管有好几种表现方式，但这是列表和元组唯一的“技术差异”。比如：列表有一个 append() 的方法来添加更多的元素，而元组却没有这个方法：
* 元组并不需要一个 append() 方法，因为元组不能修改。

一个练习：购物车

# author:jason shop = [ ['phone', 6000], ['MAC phone', 5600], ['iphone 8x', 10000], ['watch', 1200], ['lenovo pc', 5200] ] while True: salary = input("input your salary please:") if salary.isdigit(): salary = int(salary)break else: print("please input salary again!") print("-------------------welcome to my shopping store!-----------------\n") my\_cart = [] while True: for index, item in enumerate(shop): print(index, item) choice = input(">:") if choice.isdigit(): choice = int(choice) if choice < len(shop): if salary >= shop[choice][1]: my\_cart.append(shop[choice]) salary -= shop[choice][1] print("\033[32;20;1m Added %s you current salary is %s \033[0m" % (shop[choice], salary)) else: print('\033[41;20;1m sorry you salary is not enough! please charge and shopping!s\033[0m') else: print("\033[42;20;1m Current shopping product is not exists!please input again!\033[0m") elif choice == 'q': break else: print("you input is not illegal ,please input again!") print('-----------------bye !----------------------------------!')print('your cart has next product:') for i in my\_cart: print(i) print("your balance is ", salary)

## 字符串常用操作：

1. 首字母大写：capitalize()
   1. Name = ‘jason’
   2. Name.capitalize()
   3. Print(name) 结果为Jason
2. 统计出现的次数
   1. Name.count(‘a’) 单独统计某个字符出现的次数
   2. print(\_\_author\_\_.count("ja")) #统计某个字符出现的次数,也可以统计出现的字符串
3. 字符串的自动居中并填充
   1. Center(length,char),将字符串居中，两边少的空白部分用字符来填充。
   2. name = ‘jason’
   3. name.center(20,’a’).

image24.png

结果：

1. 判断是否以某个字符串结尾的。
   1. Endswith();
   2. Name.endswith(“so”).判断该字符串是否是以so结尾的。

## 字典操作

### 字典是一种key-value的数据类型，使用就像我门上面用字典。

比如：info = {

'student001': 'jason',

'student002': 'john',

'student003': 'alice',

'student004': 'albert'

}

其中的学生号与学生名称是一一对应的。

### 字典的特性：

1. Dict是无序性的。
2. Key必须保持唯一性，也就是在字典里不存在两个相同的键。也就保证了唯一的键值对。
3. 字典的键不能使用可变数据类型来做键。

### 字典的输出

* Print(info[‘student001’]),就把其对应的值打印出来了。

### 字典的修改：

Info[‘student001’] = ‘hello’

### 字典的删除

* Del info
* Del info[‘key’] 删除字典中某一个值
* Info.clear(),清除字典里的所有键值对，
* Infor.pop(‘key’),删除某个一个键值对

### 字典的添加

* 直接在字典里写:info[‘key’] = value;

### 字典的值的存在判断

* 用in 关键字来判断是否在dict里是否存在。
* ‘student001’ in info;
* 该句意思是判断student001在info字典里是否存在。（只判断键是否在字典里存在）
  + 在python2.7版本中通常是用has\_key来写的。Info.has\_key(‘student001’);

### 三级菜单

|  |
| --- |
| # author:jason  data = {  "北京":{  "昌平":{  "天通苑":['海底捞','呷哺'],  '龙泽':['永辉超市']  },  "海淀":{  '公主坟':['军事博物馆','中华世纪园'],  '科普场馆':['中国科技馆','北京天文馆'],  '高校':['北京大学','清华大学']  },  "朝阳":{  '龙城':['鸟化石国家地质公园','朝阳南北塔'],  '双塔':['朝阳凌河公园','朝阳凤凰山']  }  }  }  ########打印城市的函数  def print\_place(choice):  for i in choice:  print('\t\t',i)  ########城市  for con in data:  print(con)  choice1 = input("请选择城市：choice1:>>>")  flag = False  while not flag :  if choice1 in data:  if choice1 == 'q':  break  print\_place(data[choice1])  choice2 = input("choice2>>>")  if choice2== 'q':  break  if choice2 in data[choice1]:  print\_place(data[choice1][choice2])  choice3 = input("choice3>>>")  if choice3 == 'q':  break  if choice3 in data[choice1][choice2]:  print\_place(data[choice1][choice2][choice3]) |

## 集合的使用

### 1、将列表转换为集合

List=[1,4,489,6,8,2,9,63,5,5,6,68555,522]

列表可以转换为集合

s=set(list) //【可以将列表转换为集合】

print(s,type(s))

### 2、直接申明集合

S=set（[5,8,85,235,52,565,423,3544,5]）

或者

S=set（{5,8,85,235,52,565,423,3544,5}）

或者

S=set（(5,8,85,235,52,565,423,3544,5)）

### 集合的一个特点：

1. 无序性：集合里的值显示都是无序排序的。
2. 唯一性：集合里的值都是唯一的，不存在重复的值。

### 集合的一些操作：

1. 两个集合取交集：
   1. s1 = set({2,6,45,2,5})
   2. s2 = set({6,9,8,2,3})
   3. print(s1.intersection(s2))
   4. 结果为：{2,6}
2. 两个集合取并集：其实就是将两个集合合并然后去除重复的数据

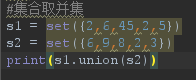


image26.png

1. 两个集合取差集

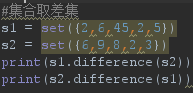
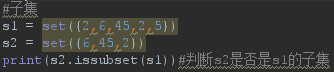


image28.png

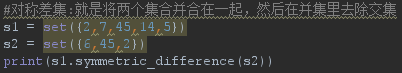
* 1. 结果：

1. 判断某个集合是否是某个集合的子集

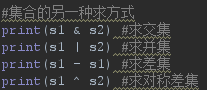


* 1. 结果为True

1. 对称子集



* 1. {7, 5, 6, 14}



### 基本操作

1. 集合元素的添加

s1 = set([5,6,9,22,10,5,20,65,5,45])s1.add(9456) print(s1)

* 1. 结果为：{65, 5, 6, 9, 10, 45, 9456, 20, 22}

1. 集合元素的移除
   1. s1.remove(22)就是将元素22从集合中移除。
2. 向集合中添加其他集合元素,添加多项
   1. s1.update([56,32,75,14,95,63]);
3. 求长度：
   1. Len(s1);求s1集合的长度。
4. 测试
   1. X in s1
   2. 判断x是否是s1集合中的一个元素
5. discard(project obj1)
   1. 如果要移除的元素是该集合中的一部分，则删除，如果不是该集合中的元素则什么也不做。
   2. S=set([25,6,5,89,2145,55])
   3. S.discard(6),若6是s集合的一个成员则删除，若不是则什么都不做。

## 面向对象（抽象思维）

**面向过程(C,C++) VS 面向对象(Java**

**【詹姆斯.高斯林(绿色计划🡪java[oak])】,Python)**

面向过程的程序设计的核心是过程（流水线式思维），过程即解决问题的步骤，面向过程的设计就好比精心设计好一条流水线，考虑周全什么时候处理什么东西。

**优点是：极大的降低了写程序的复杂度，只需要顺着要执行的步骤，堆叠代码即可。**

**缺点是：一套流水线或者流程就是用来解决一个问题，代码牵一发而动全身。**

应用场景：一旦完成基本很少改变的场景，著名的例子有Linux內核，git，以及Apache HTTP Server等。

C++,C（只适合写底层的一些驱动：USB驱动，无线网卡驱动，各种驱动99%都是C,c++语言编写）

Class 洗衣机{

具有一些功能

App(){

}

}

Class 微波炉{

具有一些功能

App(){

}

}

面向对象的程序设计的核心是对象（上帝式思维），要理解对象为何物，必须把自己当成上帝，上帝眼里世间存在的**万物皆为对象**，不存在的也可以创造出来。面向对象的程序设计好比如来设计西游记，如来要解决的问题是把经书传给东土大唐，如来想了想解决这个问题需要四个人：唐僧，沙和尚，猪八戒，孙悟空，每个人都有各自的特征和技能（这就是对象的概念，特征和技能分别对应对象的属性和方法），然而这并不好玩，于是如来又安排了一群妖魔鬼怪，为了防止师徒四人在取经路上被搞死，又安排了一群神仙保驾护航，这些都是对象。然后取经开始，师徒四人与妖魔鬼怪神仙互相缠斗着直到最后取得真经。如来根本不会管师徒四人按照什么流程去取。

面向对象的程序设计的

优点：更符合现代人的思考习惯，能大大的提高生产效率。

**优点是：解决了程序的扩展性。对某一个对象单独修改，会立刻反映到整个体系中，如对游戏中一个人物参数的特征和技能修改都很容易。**

**缺点：可控性差，无法向面向过程的程序设计流水线式的可以很精准的预测问题的处理流程与结果，面向对象的程序一旦开始就由对象之间的交互解决问题**，**即便是上帝也无法预测最终结果。于是我们经常看到一个游戏人某一参数的修改极有可能导致阴霸的技能出现，一刀砍死3个人，这个游戏就失去平衡。**

应用场景：需求经常变化的软件，一般需求的变化都集中在用户层，互联网应用，企业内部软件，游戏等都是面向对象的程序设计大显身手的好地方。

在python 中面向对象的程序设计并不是全部。

面向对象编程可以使程序的维护和扩展变得更简单，并且可以大大提高程序开发效率 ，另外，基于面向对象的程序可以使它人更加容易理解你的代码逻辑，从而使团队开发变得更从容。

了解一些名词：类、对象、实例、实例化

类：具有相同特征的一类事物(人、狗、老虎)

对象／实例：具体的某一个事物（隔壁阿花、楼下旺财）

实例化：类——>对象的过程（这在生活中表现的不明显，我们在后面再慢慢解释）

### 如何去创建一个对象？

1. 先画一个模板，图纸。（类）

车：

属性：

轮子：4

颜色：红色

座位数：5

品牌 : BMW

功能（方法）：

跑

class [类名]：

类体代码

1. 创建对象。（实体）  
   c = car() *# 车造完了*c.num = 4  
   c.color = **"宝色蓝"**c.brand = **"BMW"**c.run() *# 跑起来*print(**"我的车有"**,c.num,**"个轮胎，车身是"**,c.color,**"颜色，总之我的车是"**,c.brand,**"牌子的车！"**)

### 类与对象

类：就是构造一个对象的图纸、蓝图

用自然语言描述一辆车：

事务的属性、变量：

品牌：BWM A6

轮数：4个

颜色：蓝色

发动机的引擎杠数：8

功能，行为：跑

Python来描述这辆车：

class 类名：

属性

方法

对象：

类名() # 创建一个对象

|  |
| --- |
| class Car:  color = **"蓝色"** num = 4  brand = **"BWM A6"** start = 8   def run(self):  print(self.color , **"颜色的"** , self.brand , **"车在大路跑来跑去，溜了溜了！"**)  *# 创建对象* c = Car() *# 跑* c.run() print(c.num) print(c.color) print(c.start) |

### 类属性的私有化（ \_ \_ ）

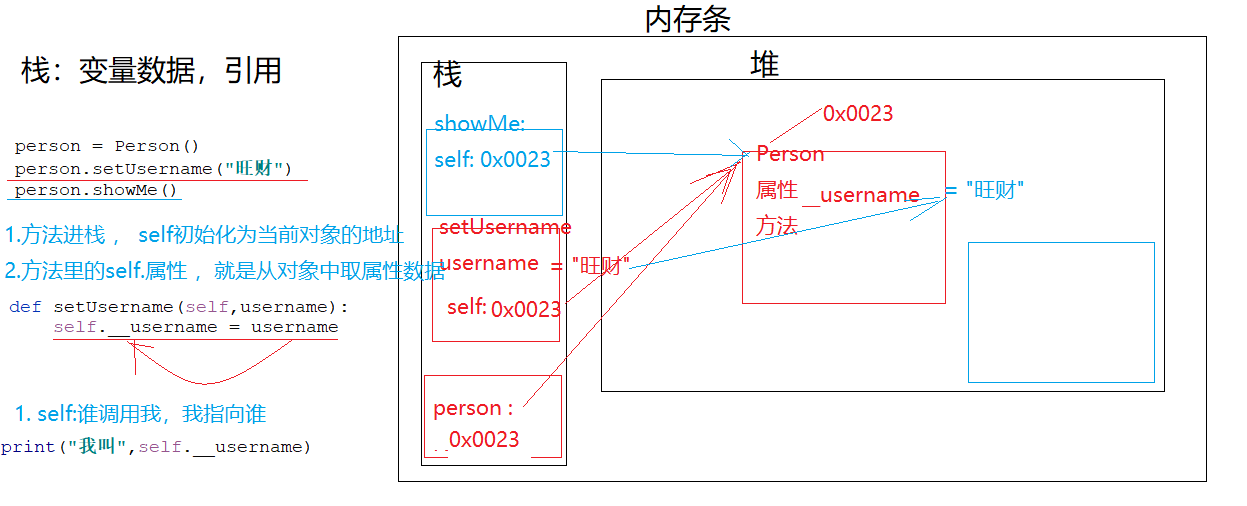
*'''  
将类的属性进行私有化：在类外任何对象都不能直接对属性进行赋值  
1.在属性前面加“\_\_”  
'''*

*# author:jason*class Person:  
 *# 避免这些属性在类外进行赋值  
 '''  
 将类的属性进行私有化：在类外任何对象都不能直接对属性进行赋值  
 1.在属性前面加“\_\_”  
 '''* \_\_username = None  
 \_\_age = None  
 \_\_sex = None  
 def setAge(self,age):  
 if age > 100 or age < 0:  
 print(**"年龄不合法！"**)  
 else:  
 self.\_\_age = age  
  
 def getAge(self):  
 return self.\_\_age  
  
person = Person()  
  
person.setAge(-56)  
  
print(person.getAge())

### Self关键字

Self关键作用：指向当前对

Self一般存在与方法内，谁调用这个方法，self就指向谁



*# author:jason*class Person:  
 \_\_username = **"jason"** def setUsername(self,username): *# this* self.\_\_username = username  
  
 def getUsername(self):  
 return self.\_\_username  
  
 def showMe(self):  
 print(**"我叫"**,self.\_\_username)  
  
 def compare(self,p1):  
 if self.\_\_username == p1.getUsername():  
 print(**"是同一个人！"**)  
 else:  
 print(**"你是间谍！"**)  
  
person = Person()  
person.setUsername(**"旺财"**)  
person.showMe()  
  
*# 方法通过self关键字来指向的*person1 = Person()  
person1.setUsername(**"王二麻子"**)  
person1.showMe()  
  
person.compare(person1)

### \_\_init\_\_方法(初始化)：构造

作用：用于在构建对象的时候进行赋值。

好处：避免了大量调用setXxxxx来进行，若有N个属性，调用setXxxx方法进行赋值这是繁琐的过程。就可以在构建对象时直接把数据传入并进行初始化。

|  |
| --- |
| *# author:jason ''' 普通方法：  需要对象.方法 方式去调用 构造方法：  创建对象时，就是立刻调用，若不写，系统会提供一个空的\_\_init\_\_方法。  用来初始化对象的   '''* class Person:  \_\_username = None  \_\_password = None   def setPassword(self,password):  self.\_\_password=password   def setUsername(self,username):  self.\_\_username = username   def getUsername(self):  return self.\_\_username   def getPassword(self):  return self.\_\_password   def \_\_init\_\_(self,username,password):  self.\_\_username = username  self.\_\_password = password  print(**"嘿，对象构建完成！"**)   p = Person(**"郑树航"**,**"admin"**) |

### 继承

什么是继承？

是一个类是另一个类的其中的一种。

好处：提高了代码的可复用性，可扩展性。

按键手机 🡪 翻盖 🡪 智能手机

老手机（父类、超类）来电显示：铃声、电话号码、姓名

新手机（子类、派生类）来电显示：铃声、电话号码、姓名、地区、大头贴

格式：

Class 类名(要继承的类)：

属性

方法

### 关键字

super 🡪 指向的就是父类的一片空间

|  |
| --- |
| *'''  c++:支持这种写法  java:不支持多个父类：不支持多继承(老头子)  python:支持多继承:好处，程序的可复用性更强（年轻人）  '''* class Animal: *# 父类，超类* name = **"旺财"** age = None  sex = None  def eat(self):  print(**"吃食物！"**)  class Bird:  name = **"八哥"** age = 56  def eat(self):  print(**"吃虫子！"**)  class Chicken(Bird,Animal):  def show(self):  age = super().age  print(age)  super().eat()  class Cameral(Animal): *# 子类* pass  class Dog(Animal): *# 子类* def show(self):  super().eat()  class Cat(Animal): *# 子类* pass  dog = Dog() dog.show() |

## 文件操作

读取文件的基本操作：

1. 先打开文件得到文件的一个引用。
2. 通过引用对文件进行操作
3. 关闭文件（关闭流）

|  |  |  |  |
| --- | --- | --- | --- |
| 模式 | 可做操作 | 若文件不存在 | 是否覆盖 |
| R | 只能读 | 报错 | - |
| R+ | 可读可写 | 报错 | 是 |
| W | 只能写 | 创建 | 是 |
| W+ | 可读可写 | 创建 | 是 |
| a | 只能写 | 创建 | 否，追加内容 |
| A+ | 可读可写 | 创建 | 否，追加内容 |

B: 字节读写

#### 读取方法：

Readline():读取一行

Readlines()：读取所有行，并将每一行存储到列表里

Read():将全文数据，读取到字符串里（用的不多）

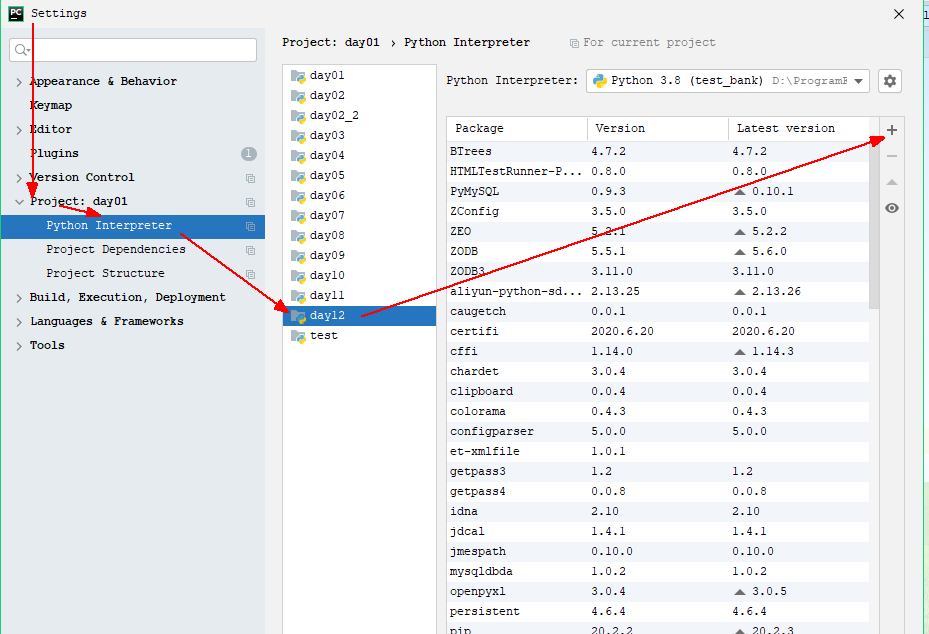
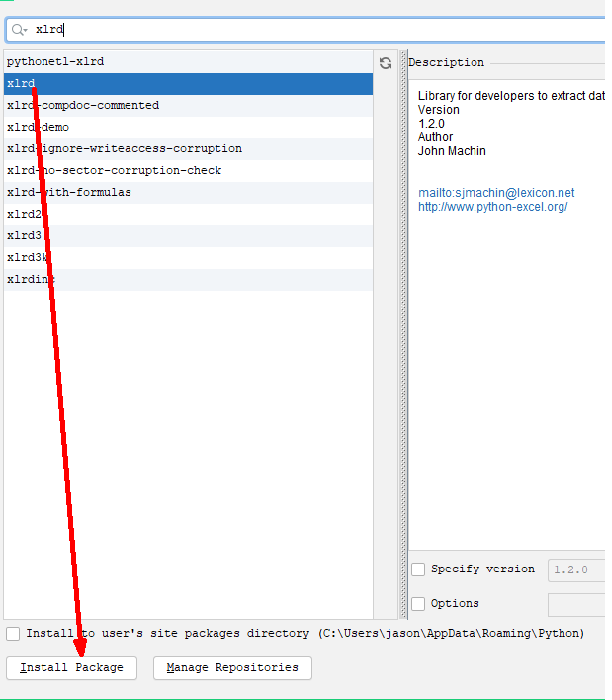
### 文件读写之注册系统

需求：输入用户名和密码，判断文件中是否有改用户，没有则将用户信息存储到文件中

技术选型：文件读取，文件写入，字符串的操作的API

|  |
| --- |
| *# 1.将所有数据行提取，存储到字典里（字典：缓冲区。便于快速的修改）* db={} *# 开始读取db.txt文件* f = open(**"db.txt"**,**"r+"**,encoding=**"utf-8"**) data = f.readlines() *# ["张三:root","李四:admin"]* for i in data:  line = i.split(**":"**) *# ["张三","root"] # 通过：号将前后进行切割。* db[line[0]] = line[1].replace(**"**\n**"**,**""**) *# 替换所有密码后面的\n改成""* f.close() **'''  注册：  输入用户名（非空），密码（非空）。  之后存到db,然后将db写入db.txt（用户:密码） '''** while True:  name = input(**"请输入用户名："**).strip()  password = input(**"请输入密码："**).strip()  if len(name) != 0 and len(password) != 0:  if name in db:  print(**"该用户已经存在！"**)  else:  *# 正常注册* db[name] = password  break  else:  print(**"输入不能为空！"**) *# 更新db 到db.txt # 1.遍历db* w = open(**"db.txt"**,**"w+"**,encoding=**"utf-8"**) for key in db:  w.write(**"{name}:{password}**\n**"**.format(name=key,password=db[key])) w.close() |

## Excel表的读取

1. 导入xlrd
2. 
3. 
4. 

二：安装方式

1. Cmd 🡪 pip install xlrd

### Excel表的操作

|  |
| --- |
| *# author:jason* import xlrd *# 是当前文档的句柄* data = xlrd.open\_workbook(**"F:**\\**测试开发-python2**\\**day12**\\**a.xlsx"**)  *# 查看工作表* sheets = data.sheet\_names()  *# data.sheet\_by\_index(0)* sheet = data.sheet\_by\_name(**"用户管理"**)   *# 读取数据* num = sheet.nrows *# 获取所有行* cols = sheet.ncols *# 获取所有列* for i in range(num):  for j in range(cols):  print(sheet.cell\_value(i,j),end=**"**\t**"**)  print() |

## 异常

什么是异常？

异常是一个事件。在程序运行当中出现不该出现的情况，即为异常。

一般情况下，在Python无法正常处理程序时就会发生一个异常。

异常是Python对象，表示一个错误。

当Python脚本发生异常时我们需要捕获处理它，否则程序会终止执行。

异常处理：

* **向上抛：异常出现 🡪 python解释器 🡪 打印控制台**
  + **Raise：手动向上抛**
  + **本质问题：这个异常始终都没有处理**
* **捕获并进行处理：try…catch…finally**

### Python常见异常：exception

IndexError：角标异常

ZeroDivisionError：被除数为0异常。

|  |  |
| --- | --- |
| 异常名称 | 描述 |
| BaseException | 所有异常的基类 |
| SystemExit | 解释器请求退出 |
| KeyboardInterrupt | 用户中断执行(通常是输入^C) |
| Exception | 常规错误的基类 |
| StopIteration | 迭代器没有更多的值 |
| GeneratorExit | 生成器(generator)发生异常来通知退出 |
| StandardError | 所有的内建标准异常的基类 |
| ArithmeticError | 所有数值计算错误的基类 |
| FloatingPointError | 浮点计算错误 |
| OverflowError | 数值运算超出最大限制 |
| ZeroDivisionError | 除(或取模)零 (所有数据类型) |
| AssertionError | 断言语句失败 |
| AttributeError | 对象没有这个属性 |
| EOFError | 没有内建输入,到达EOF 标记 |
| EnvironmentError | 操作系统错误的基类 |
| IOError | 输入/输出操作失败 |
| OSError | 操作系统错误 |
| WindowsError | 系统调用失败 |
| ImportError | 导入模块/对象失败 |
| LookupError | 无效数据查询的基类 |
| IndexError | 序列中没有此索引(index) |
| KeyError | 映射中没有这个键 |
| MemoryError | 内存溢出错误(对于Python 解释器不是致命的) |
| NameError | 未声明/初始化对象 (没有属性) |
| UnboundLocalError | 访问未初始化的本地变量 |
| ReferenceError | 弱引用(Weak reference)试图访问已经垃圾回收了的对象 |
| RuntimeError | 一般的运行时错误 |
| NotImplementedError | 尚未实现的方法 |
| SyntaxError | Python 语法错误 |
| IndentationError | 缩进错误 |
| TabError | Tab 和空格混用 |
| SystemError | 一般的解释器系统错误 |
| TypeError | 对类型无效的操作 |
| ValueError | 传入无效的参数 |
| UnicodeError | Unicode 相关的错误 |
| UnicodeDecodeError | Unicode 解码时的错误 |
| UnicodeEncodeError | Unicode 编码时错误 |
| UnicodeTranslateError | Unicode 转换时错误 |
| Warning | 警告的基类 |
| DeprecationWarning | 关于被弃用的特征的警告 |
| FutureWarning | 关于构造将来语义会有改变的警告 |
| OverflowWarning | 旧的关于自动提升为长整型(long)的警告 |
| PendingDeprecationWarning | 关于特性将会被废弃的警告 |
| RuntimeWarning | 可疑的运行时行为(runtime behavior)的警告 |
| SyntaxWarning | 可疑的语法的警告 |
| UserWarning | 用户代码生成的警告 |

### 异常和错误

异常（丐帮）：可以被捕捉和解决

错误（斧头帮）：不能被解决

### 异常的解决

1. 采用raise关键字来抛出异常
   1. 弊端：没有进行实质性的解决
   2. 作用：用来警告调用的人我可能会产生异常，让调用的人来进行处理。
2. Try …..execpt….finally:能进行实质性的解决
   1. 好处：能真正的捕捉和解决改异常
   2. Finally代码块，不管上面有没有捕捉异常，finally都必定会执行
   3. Try只能写一个，except可以写多个，finally只能写一个
3. try:  
    *# 放入可能出现的异常代码* 习大大()  
     
   except IndexError:  
    *# 异常的处理代码* print(**"角标异常了！，不要瞎弄了"**)  
   except ZeroDivisionError:  
    print(**"除数不能为0，别瞎弄！"**)  
   except Exception:  
    print(**"我捕获了一个你们解决不了的异常！"**)

### 自定义异常

自己定义一个异常：UserNotExistsError

只要继承异常体系中的一员即可。

*# 自定义异常*class UserNotExistsError(Exception):  
 def \_\_init\_\_(self,msg):  
 self.msg = msg  
  
try:  
 raise UserNotExistsError(**"用户不存在！！！！！"**)  
except UserNotExistsError:  
 print(**"用户不存在！系统已退出！"**)

### 异常使用案例：

|  |
| --- |
| *# author:jason* from cn.jason.prictice.UserException import UserAuthorticationException class Person:  \_\_username = None  \_\_age = None   def \_\_init\_\_(self,username,age):  self.\_\_username = username  self.\_\_age = age   def setUsername(self,username):  self.\_\_username = username   def getUsername(self):  return self.\_\_username   def setAge(self,age):  self.\_\_age = age   def getAge(self):  return self.\_\_age   def compare(self,person):  if self.\_\_username != person.getUsername() or self.\_\_age != person.getAge():  raise UserAuthorticationException(**"不是同一个人！"**)  else:  print(**"是同一个人！"**) |
| class UserAuthorticationException(Exception):  def \_\_init\_\_(self,msg):  self.msg = msg |
| from cn.jason.prictice.User import Person from cn.jason.prictice.UserException import UserAuthorticationException *# from 包.包....文件 import 类* p = Person(**"jason"**,57) p1 = Person(**"jason"**,56)  try:  p.compare(p1) except UserAuthorticationException as error:  print(error.args) |

## 单元测试（unittest测试模块）

1. 创建被测试类

|  |
| --- |
| class Calc:  def add(self,a,b):  return a + b   def incre(self,a,b):  return a - b   def multi(self,a,b):  return a \* b   def devide(self,a,b):  return a // b |

1. 创建测试类

* 测试类：测试类必须是Test开头，比如TestXxxxx,是标准规范。
* 测试方法：测试方法必须是test开头，否则测试框架无法识别测试方法。比如testXsss,否则测试框架无法识别。尤其是test必须写对，首字母不能大写，否则报错。
* 类方法：
  + setUpClass方法:是类加载时就执行的方法。一般用作资源的初始化
  + tearDownClass方法：是类在进行释放前的执行的方法。用于类释放之前的资源的销毁。
  + 上述的两个方法都必须用@classmethod关键字标注才能进行使用。注意方法内部的cls关键字是指当前类。Self是指当前对象。

|  |
| --- |
| *# author:jason # descrip:测试入口* import unittest from cn.jason.calc.calc import Calc **'''  1.导入unittest模块才能进行使用  2.自定义类继承unittest.TestCase即可  3.静态方法（类方法）与普通对象方法  静态方法是随着类的加载而执行  对象方法是对象调用而执行    先加载类，然后创建对象  类加载时创建该类对象，后面的所有对象都可以直接使用这个对象 '''** class TestCalc(unittest.TestCase):  calc = None *# 全局变量 1.先置位空* @classmethod *# 表示是类方法,什么时运行这个类，这个随着一起运行* def setUpClass(cls) -> None:*# 是做类资源的初始化，加载操作* print(**"小小的资源初始化完成！"**)  cls.calc = Calc()   @classmethod  def tearDownClass(cls) -> None: *# 销毁方法 --> 类才进行释放* print(**"小小的清洁操作！"**)  calc = None  *# chromeDriver.quit() # 浏览器关闭* def testAdd(self): *# 测试计算器的加法运算  # chromeDriver.open("www.baidu.com")  # chromeDriver.catiptical()  # chromeDriver.xpath(xxxxxxx).sendKey("如何学习python！")  # chromeDriver.clickButton()* a = 56 *# 参数1* b = 44 *# 参数2* c = 100 *# 期望值* sum = self.calc.add(a,b) *# 调用被测程序得到实际值  # 断言* print(**"测试方法1 运行！"**)  self.assertEqual(c,sum)  def testIncre(self):  a = 6  b = 4  c = 2  s = self.calc.incre(a,b)  print(**"测试方法2 运行！"**)  self.assertEqual(c,s) |

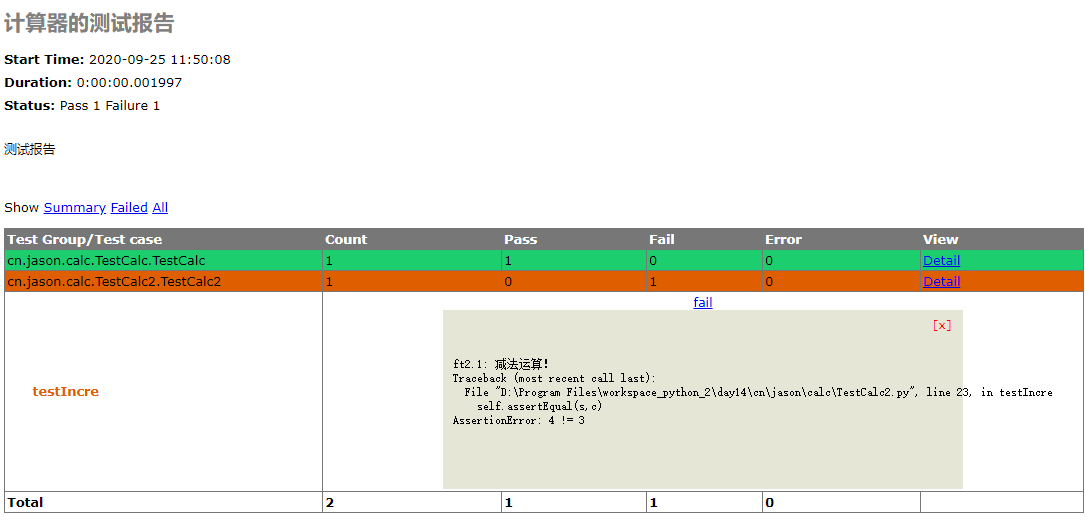
### 通过测试集来集中进行测试

|  |
| --- |
| 1. 通过文本测试集测试   import unittest from cn.jason.calc.TestCalc import TestCalc from cn.jason.calc.TestCalc2 import TestCalc2 *# 创建测试集* suit = unittest.TestSuite() *# 把测试类放在测试集里* suit.addTest(TestCalc(**"testAdd"**)) suit.addTest(TestCalc2(**"testIncre"**))  runner = unittest.TextTestRunner()  runner.run(suit) |

#### 生成界面版的测试报告

|  |
| --- |
| import unittest from cn.jason.calc.TestCalc import TestCalc from cn.jason.calc.TestCalc2 import TestCalc2 from HTMLTestRunner import HTMLTestRunner *# 创建测试集* suit = unittest.TestSuite() *# 把测试类放在测试集里* suit.addTest(TestCalc(**"testAdd"**)) suit.addTest(TestCalc2(**"testIncre"**)) *# runner = unittest.TextTestRunner() # runner.run(suit) # 生成html文件版的测试报告 > .html* f = open(**"a.html"**,**"w"**,encoding=**"utf-8"**) test = HTMLTestRunner.HTMLTestRunner(  stream = f,  verbosity = 2,  title = **"计算器的测试报告"**,  description = **"测试报告"**, ) test.run(suit) |

结果：



### 参数化测试：（ddt：数据驱动测试）

|  |
| --- |
| import unittest from testdemo.calc import Calc from ddt import ddt from ddt import data from ddt import unpack data1 = [  [1,2,3],  [9,-1,8],  [-9,8,-1],  [0,4,4],  [1000,1000,2000] ] @ddt class TestCalcAdd(unittest.TestCase):  @data(\*data1)  @unpack  def testAdd(self,s,t,y):  a = s  b = t  p = y  calc = Calc()   sum = calc.add(a,b)   self.assertEqual(p,sum) |

## PDBC（Python DataBase Connective）

1. 什么是PDBC？就是python的数据库连接技术，说白了就是用python语言来连接数据库。原来我们操作数据库是在控制台使用sql语句来操作数据库。PDBC使用python语言向数据库发送sql语句。

### 数据库的基本操作

* Create database 【数据库名】 character set utf8;
* Use [数据名];
* Create table [表名] (

字段1 类型,

字段2 类型,

。。。

字段n 类型

);

* 插入数据：Insert into [表名] values(值1，值2，值3…..，值n)
* 更新数据：update [表名] set 字段 = 值 where [条件]
* 删除数据：delete from [表名] where 【条件】;

### 如何操作数据库？

1. 安装pymysql模块
2. 获取数据游标
3. 书写sql语句
4. 使用游标执行sql语句
5. 提交执行结果
6. 关闭游标
7. 关闭数据库连接

|  |
| --- |
| 1. import pymysql **'''  1.获取数据库连接(host地址，user用户，password密码，database数据库，charset编码集)  2.获取游标(所有sql语句等数据库的操作必须经过游标)  3.书写sql 语句  4.使用游标执行sql语句  4.1 若做的是增删改，没有返回结果  4.2 若做的是查询操作，有返回结果  5.处理执行后的结果  6.关闭游标  7.关闭连接 '''** *# 1. 获取数据库 连接* con = pymysql.connect(host=**"localhost"**,user=**"root"**,password=**"root"**,database=**"lttest"**,charset=**"utf8"**)  *# 2.通过连接来获取游标* cursor = con.cursor()  *# 3.书写sql* sql = **"insert into person values('刘日成',45,500.63,'男')"** *# 4.通过游标执行sql语句* s = cursor.execute(sql) print(s)  *# 4.1 提交刚才执行操作，提交给数据* con.commit()  *# 5.关闭资源* cursor.close() con.close() |