第3章 Python语言入门与Web信息解析

3.1 Python语言入门

3.2 Python语言进阶

3.3 HTTP解析与Python实现

3.4 HTML解析与Python实现

3.5 Web信息解析

1.《 Python 爬虫开发与项目实战》范传辉编著 机械工业出版社 2017年11月第一版
2.《Python带我起飞 入门、进阶、商业实战》李金洪编著 中国工信出版集团、电子工业出版社 2018年6月第一版 2019年9月第三次印刷

3.2 Python语言进阶

3.2.1 Python变量

3.2.2 控制流

3.2.3 函数、类

3.2.4 文件操作与异常

3.2.5 实例

1.整数变量
2.字符串变量
3.列表类型
4.元组类型
5.字典类型
6.集合类型

3.2.1 Python变

```
Python 3.9.2 (tags/v3.9.2:1a79785,
Type "help", "copyright", "credits
>>> x=23
>>> id(x)
2212056361968
>>> type(x)
<class 'int'>
>>>
```

- 变量的本质——对象
  - Python 内部使用对象模型，由3 部分组成：
    - 身份：对象的唯一的标识。
      - 调用函数id 可以得到
      - 标识值可以被理解成该对象的内存地址
    - 类型： 表明对象可以存放的类型。
      - 具体的类型限制了该对象保存的内容、可以进行的操作、遵循的规则。
      - 调用函数type查看某个对象的类型
    - 值：对象所存储的具体数值。

python的变量不需要声明，但是需要初始化

3.2.1 Pytho

```
>>> x=123
>>> print("Type(x)",type(x))
Type(x) <class 'int'>
>>>
```

1.整数变量 赋值的机制

Python 中，等号赋值是指直接将对象的内存指针赋值

```
x=23
y =45
print("x,y:",id(x),id(y),x,y)
x= y
print("x,y:",id(x),id(y),x,y)
```

执行代码，输出结果如下：

```
x,y: 1725105328 1725106032 23 45
x,y: 1725106032 1725106032 45 45
```

x 与y 在赋值之前各自有自己的指针

当赋值之后， x 与y 不仅有相同的值，而且还有相同的指针。

3.2.

- is 函数与“==”的区别
  - “==”只是判断两个对象的内容是否相等
  - is 函数不仅比较对内容是否相等，还比较指针是否相等。

```
a=-256
b=-256
print(a==b)          #判断a和b的内容是否相等
print(a is b)        #判断a和b的内存是否相等
print(id(a),id(b))   #id的意思是打印出对象的地址
```

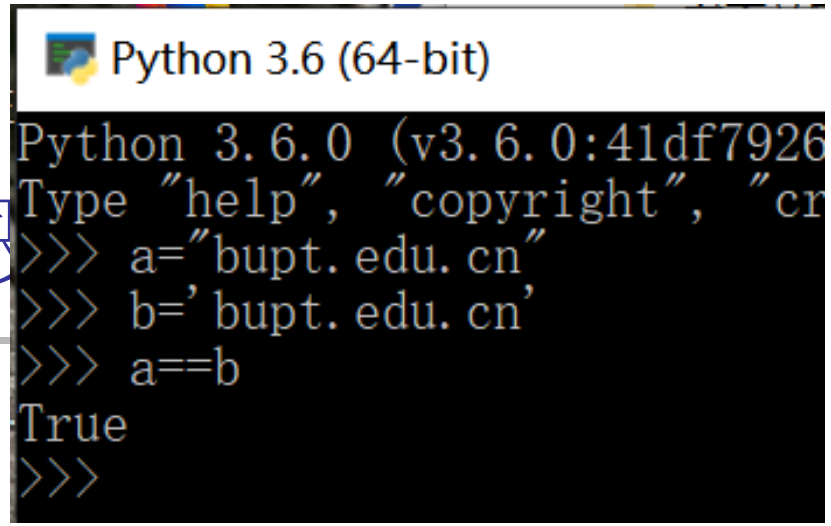
执行代码，输出如下结果：

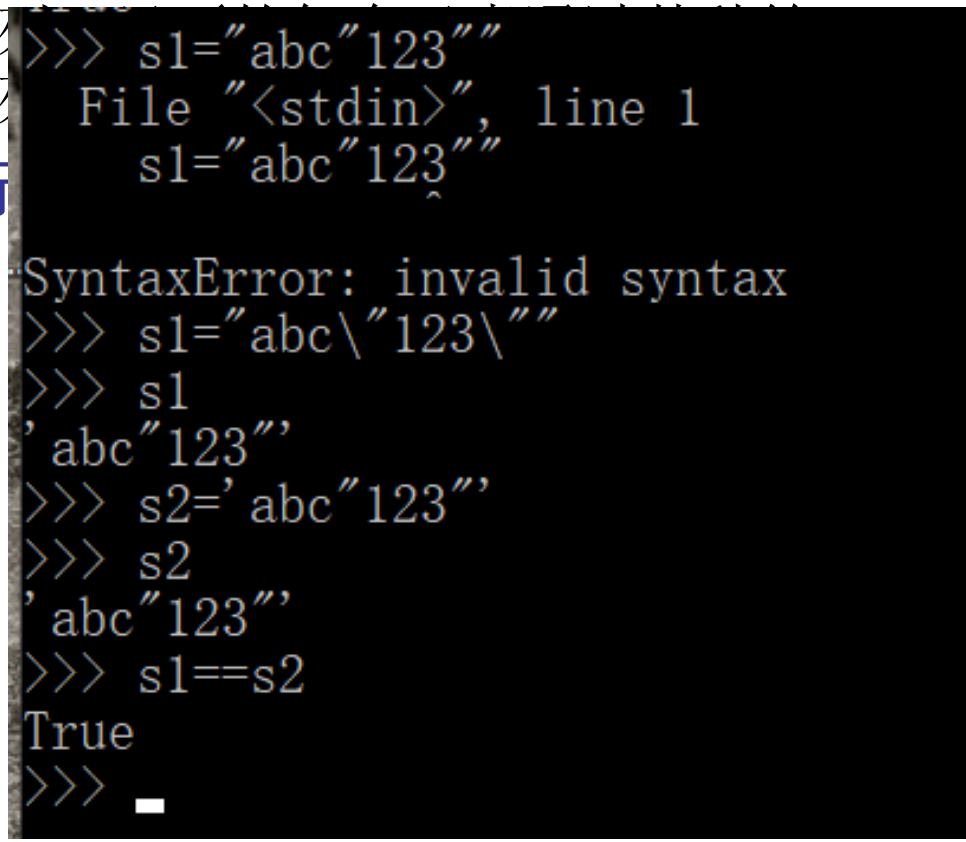
```
True
False
151744048 151743312
```

虽然a 和b 都是－256，但是它们的地址并不相等

3.2.1 Python变

- 2.字符串变量
  - 字符串属于“序列”类型
  - “序列”类型，即有序排列
    号分开，按照顺序排成一列
- 单引号和双引号都可以表示
- 单引号中可以包含双引号
- 双引号中可以包含单引号





3.2.1 Python变量

“序列”类型的基本操作

- 连接：+

```
>>> 'abc'+'def'
'abcdef'
```

- 重复：*

```
>>> 'abcde'*3
'abcdeabcdeabcde'
```

3.2.1 Python变量

“序列”类型的基本操作（续）

- 检索：用序列号检索字符
  - 表示方法([下标])，下标指的是序列中的索引位置
  - 下标为正数：从左往右的索引顺序，从0 开始计算
    如： s ＝”hello”， s [0］ 为“ h”
  - 下标为负数：从右向左的顺序，从1开始计算
    如： s= “hello”，s [- 1 ］ 为 “o”

0	1	2	3	4
h	e	l	l	o
-5	-4	-3	-2	-1

- 反检索：使用index 函数，检索字符的序列号
  - 索引是按照从左向右排列的
  - 如s.index (‘e’)，返回1

```
>>> s='hello'
>>> s.index('e')
1
>>> s.index('l')
2
```

切片应用：回文判断

```
>>> inputstr='abccba'
>>> if inputstr[::] == inputstr[::-1]:
...     print("是回文")
... else:
...     print("不是回文")
...
是回文
```

3.2.1 Python

- “序列”类型的基本操作（续）
- 切片：从字符串中取出相应的元素,重新组成一个新的字符串
  - 语法: 字符串[开始索引 ： 结束索引 ： 步长]
    - 意义：从开始索引取一个数据，跳过步长的长度，再取一个数据，一直到结束索引。
    - 步长为正值: 开始索引默认为0， 结束索引默认为len()+1
    - 步长为负值：开始索引默认为-1， 结束索引默认为开始，不能认为是0,也不能认为是-1
    - 步长默认为1，步长为正值，从开始索引从左往右走，步长为负值，从开始索引从右往左走
  - 例： s ＝ “hello”,
  - s [1 : 4] 结果为 “ell ”。

```
>>> s='hello'
>>> s[1:4]
'ell'
```

0	1	2	3	4
h	e	l	l	o
-5	-4	-3	-2	-1

3.2.1 Python变量

代码 4-6： “序列”类型的基本操作

0	1	2	3	4
h	e	l	l	o
-5	-4	-3	-2	-1

```
s='hello'
s2=' daimayisheng'
#连接
print(s+s2)                    #输出：hello daimayisheng
#重复
print(s*3)                     #输出：hellohellohello
#检索
print(s[0],s[1],s[2],s[4])#当索引为正数，方向从左到右，索引从0开始。输出：h e l o
print(s[-1],s[-2],s[-4])  #当索引为负数，方向从右到左，索引从1开始。输出：o l e
print(s[-5] )                  #因为是从1开始，所有最后一个是5。输出：h
#print(s[5])                   #因为是从0开始，所有最后一个是4。索引5已经超过了字符串的范围，
                               #于是报错输出：IndexError: string index out of range反检索
```

```
print(s.index('e'))          #返回e在s中的索引，该索引是从左向右的顺序。输出：1
print(s.index('l'))          #当有两个l时，返回第一个。输出：2
print(s.index('w'))          #因为s中没有w字符。所以报错：ValueError: substring not found
                             #切片
print(s[1:3])                #从第一个还是到第三个，步长不指定默认为1。输出：el
print(s[:3])                 #开始位置不指定，默认从第一个字符开始截取,取到第3个。输出：hel
print(s[0:])                 #结束位置不指定，默认到最后，即从第一个字符开始截取，一直截取到最后
                             #输出：hello
print(s[:])                  #开始与结束都不指定,即从第一个字符开始截取,一直截取到最后。输出:hello
print(s[::2])                #步长为2，即每取一个之后，光标往后移动2个。输出：hlo
print(s[::-2])               #步长为-2，即反方向读取，每取一个之后，光标往前移动两元素。输出：olh
print(s[::-1])               #实现字符串的逆序。输出：olleh
print(s[-2::-1])             #实现字符串的逆序，然后再切片。输出：lleh
print(s[:0:-1])              #实现字符串的逆序，然后再切片。输出：olle
                             #字符串不能被修改
s[3]='3'                   #报错，输出：TypeError: 'str' object does not support item assignment
```

- 字符串索引

```
>>> s2
'abc"123"'
>>> s2[0]
'a'
>>> s2[3]
'"'
>>> len[(s2)-1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'str' and 'int'
>>> len(s2)
8
>>> s2[-1]
'"'
>>>
```

- 字符串切片

```
>>> s3="123456789"
>>> s3[2:6]
'3456'
>>> s3[:6]
'123456'
>>> s3[2:]
'3456789'
>>> s3[::2]
'13579'
>>> s3[2:6:2]
'35'
>>>
```

- 占位符
  - 占位符在一个字符串中占据着一个位置，在输出时将这个位置替换成具体的内容。
  - 每一个占位符只能替一种特定的类型占位。
    - %d整数占位符，原数字为浮点数会强制转换变成整数
    - %f浮点数占位符，%.xf 则是精确至小数点后x位。
    - %s字符串占位符，字符串

1.%d 整数占位符

```
>>>'我考了%d分' % 20
'我考了20分'
>>>'我考了%d分' % 20.5
;我考了20分'
>>>"我考了%d分，进步了%d分" % (50,10)
"我考了50分，进步了10分"
```

2.%f 浮点数占位符

```
1  >>>"我考了%f" % 66.666
2  "我考了66.666000"
3  >>>'我考了%f' % 66
4  '我考了66.000000'
5  >>>"我考了%f，进步了%.2f" % (66,12.369)
6  "我考了66.000000，进步了12.36"
```

3.%s 字符串占位符

```
>>>'%s' % Ture
'Ture'
>>>'%s,%s' % (123,abc)
'123,abc'
```

3.2.1 Python变量

- 字符串类型格式化format()方法：
  <模板字符串>.format(<逗号分隔的参数>)
- 方便连接不同类型的变量或内容
- 返回一个新的字符串，参数从0 开始编号。

注意！{}间不能有空格！

```
>>> "{}:计算机{}的CPU 占用率为{}%.".format("2016-12-31","PYTHON",10)
'2016-12-31:计算机PYTHON的CPU 占用率为10%.'
```

- Python字符串格式化--format()方法
  https://blog.csdn.net/i_chaoren/article/details/77922939

3.2.1 Python变量

3. list （列表）类型

- Python 中使用最频繁的数据类型
- 是一个有序集合，属于“序列”类型。
- 适合于将值组织到一个结构中，并且**通过编号对其进行引用**
- 列表的数据项不需要具有相同的类型

- 语法：把逗号分隔的不同的数据项使用方括号括起来

```
list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

3.2.1 Python变量

- 列表的基本操作
  - 连接、重复、检索、反检索、切片
  - 列表中的元素是可以改变的
  - 有许多内置方法（函数）

1	list.append(obj) 在列表末尾添加新的对象
2	list.count(obj) 统计某个元素在列表中出现的次数
5	list.insert(index, obj) 将对象插入列表
7	list.remove(obj) 移除列表中某个值的第一个匹配项
8	list.reverse() 反向列表中元素

3.2.1 Python变量

3. list （列表）类型

List内置函数

list.insert(index, obj)

将对象插入列表

代码 4-8：list 实现对列和栈

```
01  queue = []                              #定义一个空 list，当作队列
02  queue.insert(0,1)                       #向队列里存入一个整型元素 1
03  queue.insert(0,2)                       #向队列里存入一个整型元素 2
04  queue.insert(0,"hello")                 #向队列里存入一个字符型元素 hello
05  print("取第一个元素",queue.pop())        #从队列里读取一个元素，根据先进先出原则，输出 1
06  print("取第二个元素",queue.pop())        #从队列里读取一个元素，根据先进先出原则，输出 2
07  print("取第三个元素",queue.pop())        #从队列里读取一个元素，根据先进先出原则，输出 hello
```

列表做队列还是栈，是程序员自己控制的！

```
>>> q=[]
>>> q.insert(0,1)
>>> q.insert(0,2)
>>> q.insert(0,3)
>>> q.append('a')
>>> q.append('b')
>>> q.append('c')
>>> q
[3, 2, 1, 'a', 'b', 'c']
>>> q.pop()
'c'
```

3. list （列表）类型

代码 4-8：list 实现对列和栈（续）

List内置函数

```
08 stack = []                                  #定义一个空 list，当作栈
09 stack.append(1)                             #向栈里存入一个整型元素 1
10 stack.append(2)                             #向栈里存入一个整型元素 2
11 stack.append("hello")                       #向栈里存入一个字符型元素 hello
12 print("取第一个元素",stack.pop())            #从栈里读取一个元素，根据后进先出原则，输出 hello
13 print("取第二个元素",stack.pop())            #从栈里读取一个元素，根据后进先出原则，输出 2
14 print("取第三个元素",stack.pop())            #从栈里读取一个元素，根据后进先出原则，输出 1
```

List几个

- insert、pop操作

```
>>> l2=['a','b','c']
>>> l2.insert(1,'x')
>>> l2
['a', 'x', 'b', 'c']
>>> p1=l2.pop()
>>> p1
'c'
>>>
```

- sort排序操作

```
>>> l=[2,1,5,7]
>>> l.sort(reverse=True)
>>> l
[7, 5, 2, 1]
>>>
```

- 生成1-100之间偶数列表

```
>>> l=[x for x in range(1,101) if x%2==0]
>>> l
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36,
 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70,
 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]
>>>
```

3.2.1 Python变量

4. tuple （元组）类型

- 元组与列表类似，不同之处在于
  - 元组的元素不能修改
  - 元组使用小括号，列表使用方括号
- 元组的定义语法：
  - 使用小括号括起来，元素之间使用逗号来分隔

```
a = (1991, 2014, 'physics', 'math')      #定义一个元组变量a
print(a, type(a), len(a))                #将a的内容、类型、长度打印出来
```

输出结果如下：

```
(1991, 2014, 'physics', 'math') <class 'tuple'> 4
```

(1991, 2014, 'physics', 'math') <class 'tuple'> 4

3.2.1 Python变量

5. dictionary （字典）类型

- Python 唯一内建的的内置映射类型。
- 通过名字来引用值的数据结构，称为映射
- 字典中的值没有特殊的顺序，都存储在一个特定的键（key）下，键可以是数字、字符串甚至元组。
- 定义语法：字典的每个键值 key=>value 对用冒号“：”分割，每个键值对之间用逗号“，”分割，整个字典包括在花括号 {} 中

```
d = {key1 : value1, key2 : value2 }
```

phonebook = { "tom" : '666' , 'cat' : '999' , 'wzw' : '333' }

3.2.1 Python变量

5. dictionary （字典）类型

- 可以用dict函数通过关键字的参数来创建字典：
- d = dict (name='wzw',age= 22)

```
mylist = [('hello',1),('good', 2), ['ok', 3]] #定义一个 list 里面嵌套元组和 list, 每个嵌套的元素都是键值对形式
d = dict(mylist)                              #使用 dict 函数将 mylist 转换成字典
d2 = {'hello': 1, 'good': 2, 'ok': 3}         #使用大括号创建字典
print(d,d2)   #输出{'hello': 1, 'ok': 3, 'good': 2} {'hello': 1, 'ok': 3, 'good': 2}
```

3.2.1 Python变量

基本字典的操作：

len (d)：返回d中键—值对的数量

d[k] ： 返回键k上的值

d[k] = v：将值v关联到键k上

del d[k]：删除键为k的项

k in d：检查d中是否有含有键为k的项

键一般是唯一的，如果重复最后的一个键值对

```
>>>dict = {'a': 1, 'b': 2, 'b': '3'}
>>> dict['b']
'3'
>>> dict
{'a': 1, 'b': '3'}
```

值可以取任何数据类型，但键必须是不可变的，如字符串，数字或元组。

一个简单的字典实例：

```
dict = {'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}
```

Python字典包含了以下内置方法：

序号	函数及描述
1	dict.clear() 删除字典内所有元素
2	dict.copy() 返回一个字典的浅复制
3	dict.fromkeys(seq[, val]) 创建一个新字典，以序列 seq 中元素做字典的键，val 为字典所有键对应的初始值
4	dict.get(key, default=None) 返回指定键的值，如果值不在字典中返回default值
5	dict.has_key(key) 如果键在字典dict里返回true，否则返回false
6	dict.items() 以列表返回可遍历的(键, 值) 元组数组

7	dict.keys() 以列表返回一个字典所有的键
8	dict.setdefault(key, default=None) 和get()类似, 但如果键不存在于字典中，将会添加键并将值设为default
9	dict.update(dict2) 把字典dict2的键/值对更新到dict里
10	dict.values() 以列表返回字典中的所有值
11	pop(key[,default]) 删除字典给定键 key 所对应的值，返回值为被删除的值。key值必须给出。 否则，返回default值。
12	popitem() 返回并删除字典中的最后一对键和值。

3.2.1 Python变量

5. dictionary （字典）类型

- 通过关键字（ key ）来取值。

```
d2 = {'hello': 1, 'good': 2, 'ok': 3}   #使用大括号创建字典
print(d2['hello'])                      #取出字典中 key 为 hello 的值，输出：1
```

```
d2 = {'hello': 1, 'good': 2, 'ok': 3}   #使用大括号创建字典
d2['hello'] = 'e'                       #将字典中 key 为 hello 的值赋值为“e”
print(d2['hello'])                      #取出字典中 key 为 hello 的值，输出：e
```

3.2.1 Python变量

5. dictionary （字典）类型

- 为字典类型添加元素

```
d2 = {'hello': 1, 'good': 2, 'ok': 3}   #使用大括号创建字典
d2['new'] = 100                          #在字典中加入 key 为 new、值为 100 的键值对
print(d2)                    #输出: {'ok': 3, 'hello': 1, 'new': 100, 'good': 2}
```

- 删除字典类型里的元素

```
d2 = {'hello': 1, 'good': 2, 'ok': 3}   #使用大括号创建字典
del d2['hello']                          #在字典中删除 key 为 hello 的键值对
print(d2)                                #输出: {'ok': 3, 'good': 2}
```

3.2.

5. dictionary

- 字典推导：

```
urls_d={i:"www.xyz.com/?page={}".format(i) for i in range(1,11)}
```

- 生成urls_d:

```
{1: 'www.xyz.com/?page=1', 2: 'www.xyz.com/?page=2',
 3: 'www.xyz.com/?page=3', 4: 'www.xyz.com/?page=4',
 5: 'www.xyz.com/?page=5', 6: 'www.xyz.com/?page=6',
 7: 'www.xyz.com/?page=7', 8: 'www.xyz.com/?page=8',
 9: 'www.xyz.com/?page=9', 10: 'www.xyz.com/?page=10'}
```

- item()方法对字典进行循环遍历，每次可以同时获得键和值
- keys()方法对键进行遍历，每次可以利用键获得相应的值

```
>>> for key,val in urls_d.items():
...     print(key,val)
...
1 www.xyz.com/?page=1
2 www.xyz.com/?page=2
3 www.xyz.com/?page=3
4 www.xyz.com/?page=4
5 www.xyz.com/?page=5
6 www.xyz.com/?page=6
7 www.xyz.com/?page=7
8 www.xyz.com/?page=8
9 www.xyz.com/?page=9
10 www.xyz.com/?page=10
>>>
```

```
>>> for key in urls_d.keys():
...     print(key,urls_d[key])
...
1 www.xyz.com/?page=1
2 www.xyz.com/?page=2
3 www.xyz.com/?page=3
4 www.xyz.com/?page=4
5 www.xyz.com/?page=5
6 www.xyz.com/?page=6
7 www.xyz.com/?page=7
8 www.xyz.com/?page=8
9 www.xyz.com/?page=9
10 www.xyz.com/?page=10
>>>
```

- 获取字典类型里的key 与value
  - 类型dict_ keys 和dict_values 都不能直接使用，将其转成list 来使用

```
d2 = {'hello': 1, 'good': 2, 'ok': 3}   #使用大括号创建字典
print(type(d2.values()))   #打印.values()返回值的类型，输出：<class 'dict_values'>
print(d2.values())         #打印.values()返回值，输出：dict_values([3, 1, 2])
print(type(d2.keys()))     #打印.keys()返回值的类型，输出：<class 'dict_keys'>
print(d2.keys())           #打印.keys()返回值，输出：dict_keys(['ok', 'hello', 'good'])
list1 = list(d2.keys())    #将.keys()返回值转成 list 类型
print(list1)               #打印转换后的 list，输出：['ok', 'hello', 'good']
list2 = sorted(d2.keys())  #由于 key 的顺序不固定，常会用将其转成排序后的 list
print(list2)               #打印排序后的 list，输出：['good', 'hello', 'ok']
```

3.2.1 Python变量

6. set （集合）类型

- 是一个无序、不重复元素的集合。
- 主要作用是， 进行成员关系测试和消除重复元素。
- 在数据清洗领域运用得比较广泛。

- **描述方法：**
  - 使用大括号括起来， 元素之间使用逗号进行分隔， 里面的元素可以是任何类型

```
myset = {'hello', 'hello', 'Python', 'tensorflow',2,1,2}  #使用大括号方法定义一个 set（集合）
print(myset)                    #生成的 set 会自动去掉重复的元素
                                #输出: {'hello', 1, 2, 'Python', 'tensorflow'}
```

还可以使用 set 函数将其他类型的变量转成 set（集合）。例如：

```
mylist = ['hello', 'hello', 'Python', 'tensorflow',2,1,2]     #定义一个 list
mytuple = ('hello', 'hello', 'Python', 'tensorflow',2,1,2)    #定义一个 tuple
myset = set(mylist)                                  #使用 set 函数将 list 转成 set（集合）
print(myset)                                         #生成的 set 会自动去掉重复的元素
                                                     #输出：{'hello', 2, 'Python', 'tensorflow', 1}
myset = set(mytuple)                                 #使用 set 函数将 tuple 转成 set（集合）
print(myset)                                         #生成的 set 会自动去掉重复的元素
                                                     #输出：{'hello', 2, 'Python', 'tensorflow', 1}
```

- set 支持差、并、交等操作。
- in :用来测试某个元素是否在某个集合里，返回一个布尔型的结果。

```
>>> h=set('hello')
>>> print(h)
{'h', 'e', 'o', 'l'}
>>> h
{'h', 'e', 'o', 'l'}
```

```
helloset = set('hello')                    #通过 set 函数，将字符串 “hello” 转成一个集合
tensorflowset = set('tensorflow')          #通过 set 函数，将字符串 “tensorflow” 转成一个集合
print('w'in tensorflowset,tensorflowset)   #判断 “w” 是否在集合 tensorflowset 中，并打印 tensorflowset 内容
                                           #输出: True {'t', 's', 'l', 'r', 'e', 'f', 'w', 'o', 'n'}
print('w'in helloset,helloset)             #判断 “w” 是否在集合 helloset 中，并打印 helloset 内容
                                           #输出: False {'e', 'l', 'o', 'h'}
print(helloset -tensorflowset )            #计算集合 helloset 与 tensorflowset 的差集
                                           #helloset 中有而 tensorflowset 中没有。输出: {'h'}
print(helloset |tensorflowset )            #计算集合 helloset 与 tensorflowset 的并集
                                           #既包括 helloset，又包括 tensorflowset
                                           #输出: {'t', 's', 'l', 'r', 'e', 'f', 'w', 'o', 'n', 'h'}
print(helloset &tensorflowset )            #计算集合 helloset 与 tensorflowset 的交集
                                           # helloset 中有，而 tensorflowset 中也有。输出:{'e', 'l', 'o'}
print(helloset ^tensorflowset )            #计算集合 helloset 与 tensorflowset 的对称差集
                                           #该集合中的元素在 helloset 或 tensorflowset 中，但不会同时出现在
                                           # helloset 和 tensorflowset 中。输出: {'t', 's', 'r', 'f', 'w', 'n', 'h'}
```

差

并

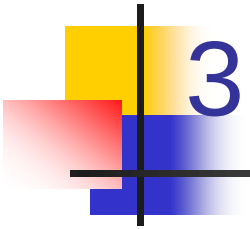
交

对称差

3.2.1 Python变量

- Set有许多内置方法

集合的内置函数	描　述
set.add(x)	往集合里添加一个元素 x
set.update(list)	输入参数是一个列表，将列表里的元素全部添加到集合里
set.remove(x)	删除集合里的元素 x。当 x 不在集合里时，会报错误（输出 KEYERROR）
set.discard(x)	如果结合里有元素 x 时，删除集合里的元素 x



3.2 Python语言进阶

3.2.1 Python变量

3.2.2 控制流

3.2.3 函数、类

3.2.4 文件操作与异常

3.2.5 实例

3.2.2 控制流

- Python 中的流程控制语句包括
  - if 条件语句
  - while 循环语句
  - for 循环语句。还可以细分为
- 内部流程的控制语句
  - break
  - continue
  - pass
  - 等

3.2.2 控制流

- 书5.5 实例 16：演示人机对话中的控制流程

实例描述

通过一个循环来获得用户的输入，并根据不同的输入进行不同的处理：

（1）如果输入“hello”，进入主程序，开启人机对话服务。

（2）如果输入“go away”或是“Bye”，退出程序。

（3）如果输入“pardon”，重新等待用户输入。

实例 16：演示人机对话中的控制流程

代码 5-4：人机对话控制流程

```
getstr = ''                                    #定义一个空字符串，用来接收输入
while("Bye"!=getstr):                          #使用 while 循环
    if ''==getstr:                             #如果输入字符为空，输出欢迎语句
        print("hello! Password!")
    getstr = input("请输入字符，并按 Enter 键结束:")#调用 input 函数，获得输入字符串
    if 'hello'==getstr.strip():                #如果输入字符串为 hello，启动对话服务
        print('How are you today?')

        getstr = "start"                       #将 getstr 设为 start，标志是启动对话服务
    elif 'go away'==getstr.strip():            #如果输入的是 go away，则退出
        print('sorry! bye-bye')
        break                                  #使用 break 语句退出循环
```

加 ‘：’

str.strip()：删掉字符串(str)的头和尾的空格

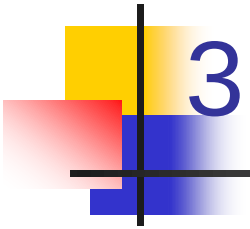
实例 16：演示人机对话中的控制流程

```
elif 'pardon'==getstr.strip():                    #如果输入的是pardon，请重新再输出一次
    getstr = ''
    continue                                      #continue将结束本次执行，开始循环的下一次执行
else:
    pass                                          #什么也不做，保持程序完整性
if 'start'== getstr:                              #如果getstr为start，启动对话服务
    print('…init dialog-serving…')                #伪代码，打印一些语句，代表启动了对话服务
    print('… one thing…')
    print('… two thing…')
    print('……')
```

```
请输入字符，并按Enter键结束：pardon
hello! Password!

请输入字符，并按Enter键结束：
```

```
请输入字符，并按Enter键结束：hello
How are you today?
…init dialog-serving…
… one thing…
… two thing…
……
请输入字符，并按Enter键结束：
```



3.2 Python语言进阶

3.2.1 Python变量

3.2.2 控制流

3.2.3 函数、类

3.2.4 文件操作与异常

3.2.5 实例

3.2.3 函数、类

- Python 函数
  - 函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。
  - 语法

```
def 函数名(参数1，参数2，参数3，...):
    函数体
    return 返回值
```

n:参数，’abc’:缺省值

```
>>> def hi(n='abc'):
...     return 'hi'+n
...
>>> print(hi())
hiabc
>>> print(hi('xxx'))
hixxx
```

函数的基本知识

```
def hi(name="yasoob"):
    return "hi " + name

print(hi())
# output: 'hi yasoob'

# 我们甚至可以将一个函数赋值给一个变量，比如
greet = hi
# 我们这里没有在使用小括号，因为我们并不是在调用hi函数
# 而是在将它放在greet变量里头。我们尝试运行下这个

print(greet())
# output: 'hi yasoob'

# 如果我们删掉旧的hi函数，看看会发生什么！
del hi
print(hi())
#outputs: NameError

print(greet())
#outputs: 'hi yasoob'
```

定义函数:def 函数名(参数)：

调用函数:函数名(参数)：

调用函数greet()与调用hi()一样

原来函数hi()删掉了，greet()还有

在函数中定义函数

```
def hi(name="yasoob"):
    print("now you are inside the hi() function")

    def greet():
        return "now you are in the greet() function"

    def welcome():
        return "now you are in the welcome() function"

    print(greet())
    print(welcome())
    print("now you are back in the hi() function")

hi()
#output:now you are inside the hi() function
#       now you are in the greet() function
#       now you are in the welcome() function
#       now you are back in the hi() function

greet()
#outputs: NameError: name 'greet' is not defined
```

调用hi()：greet()和welcome()将会同时被调用

greet()和welcome()函数在hi()函数之外是不能访问的

3.2.3 函数、类

- enumerate() 枚举函数
  - 用于将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据和数据下标
  - 一般用在 for 循环当中使用
- 语法

  enumerate(sequence, [start=0])
- 参数
  - sequence -- 一个序列、迭代器或其他支持迭代对象。
  - start -- 下标起始位置
- 返回值
  - 返回 enumerate(枚举) 对象

```
>>>seasons = ['Spring', 'Summer', 'Fall', 'Winter']
>>> list(enumerate(seasons))
[(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
>>> list(enumerate(seasons, start=1))       # 下标从 1 开始
[(1, 'Spring'), (2, 'Summer'), (3, 'Fall'), (4, 'Winter')]
```

- Enumerate索引默认从0开始，可以用start重新设置第一个索引的值

```
>>> for ind,val in enumerate([10,20,30,40,50],1):
...     print(ind,val)
...
1 10
2 20
3 30
4 40
5 50
```

```
>>> for ind,val in enumerate([10,20,30,40,50],3):
...     print(ind,val)
...
3 10
4 20
5 30
6 40
7 50
>>>
```

3.2.3 函数、类

- 函数例1:(2.1例first_get.py)定义了函数get_content(url)
  - 函数get_content(url)功能：读取url的网页，打印网页代码
  - first_get.py功能：打印“电子工业出版社”网页代码

函数get_content(url)在程序first_get.py里

```
import requests          #引入requests模块

def get_content(url):                    #定义函数
    resp=requests.get(url)
    return resp.text

url="https://www.phei.com.cn/"           #定义URL值，目标网站地址
content=get_content(url)                 #调用函数值返回值
print("前500个字符为：", content[0:500])
content_len=len(content)
print("内容的长度为：", content_len)
if content_len>=40*1024:                 #判断内容长度是否大于40K
    print("内容长度是否大于等于40KB.")
else:
    print("内容长度是否小于等于40KB.")
```

3.2.3 函数、类

- 编写snd_get.py，引入first_get.py作为一个模块，使用函数get_content(url)，打印“新浪”网页

```
import first_get
url='http://www.sina.com.cn'
sina_content=first_get.get_content(url)
print('------------------------------------')
print('snd_get.py得到内容前200个字符：',sina_content[0:200])
print('snd_get.py得到内容长度：',len(sina_content))
```

- snd_get.py运行结果，上半段是first_get.py运行结果

```
==
前500个字符为:  ï»¿<!DOCTYPE HTML>
<html ng-app="topDivApp"  ng-controller="topDivCtrl">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta http-equiv = "X-UA-Compatible" content = "IE=edge,chrome=1" />
<title>çµå-å·¥ä¸šå°çç¤¾æéå¬å¸</title>
<meta name="keywords" content="" />
<meta name="description" content="" />
<link rel="shortcut icon" href="#" />
<link href="/templates/stylesheets/global.css" rel="stylesheet" />
<link href="/templates/stylesheets/web_base.css" rel="
内容的长度为:  49479
内容长度是否大于等于40KB.
------------------------------------
```

电子工业出版社网页

```
snd_get.py得到内容前200个字符:  <!DOCTYPE html>
<!-- [ published at 2020-03-25 17:03:01 ] -->
<html>
<head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE
snd_get.py得到内容长度:  540360
```

新浪网页

- 为了避免在snd_get.py中执行first_get.py的抓取，修改first_get.py程序：

```
def get_content(url):
    resp=requests.get(url)
    return resp.text

if __name__=='__main__':
        url="https://www.phei.com.cn/"          #定义URL值，目标网站地址
        content=get_content(url)                #调用函数值返回值
```

只有first_get.py运行时，程序名是__main__

- snd_get.py执行，只打印“新浪”网页

```
========== RESTART: C:\Users\fcx\Desktop\2020关注\python代码\snd_get.py ==
==
------------------------------------
snd_get.py得到内容前200个字符： <!DOCTYPE html>
<!-- [ published at 2020-03-25 17:12:01 ] -->
<html>
<head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE
snd_get.py得到内容长度： 540347
>>>
```

3.2.3 函数、类

- 函数例2. 不同模块中的同名函数
  - 3.1中最后的例子，3-1中的getenv模块中有showENV 函数，现在创建代码文件“ 3-2 调用模块”，定义本地showENV 函数

代码 3-1：getenv

```
import platform
import sys
import os
def showENV():                                  #函数
    s = platform.platform()
    print("当前系统：",s)                        #获取系统信息
    p = sys.path
    print("当前安装路径：",p)                    #获取安装路径
    op = os.getcwd()
    print("当前代码路径：",op)                   #获取当前代码路径
    print("Python 版本信息：",sys.version_info)

if __name__ == '__main__':                      #进行模块的单元测试
    showENV()
```

3.2.3 函数、类

代码 3-2：调用模块

```
import getenv                                    #导入自定义模块 getenv

def showENV():                                   #实现同名函数 showENV
    print("this is my env")

showENV()                                        #调用本地函数
getenv.showENV()                                 #调用 getenv 模块里的函数
```

```
this is my env
当前系统:  Windows-7-6.1.7601-SP1
......
```

本地 showENV 的输出

getenv 模块里的 showENV 输出

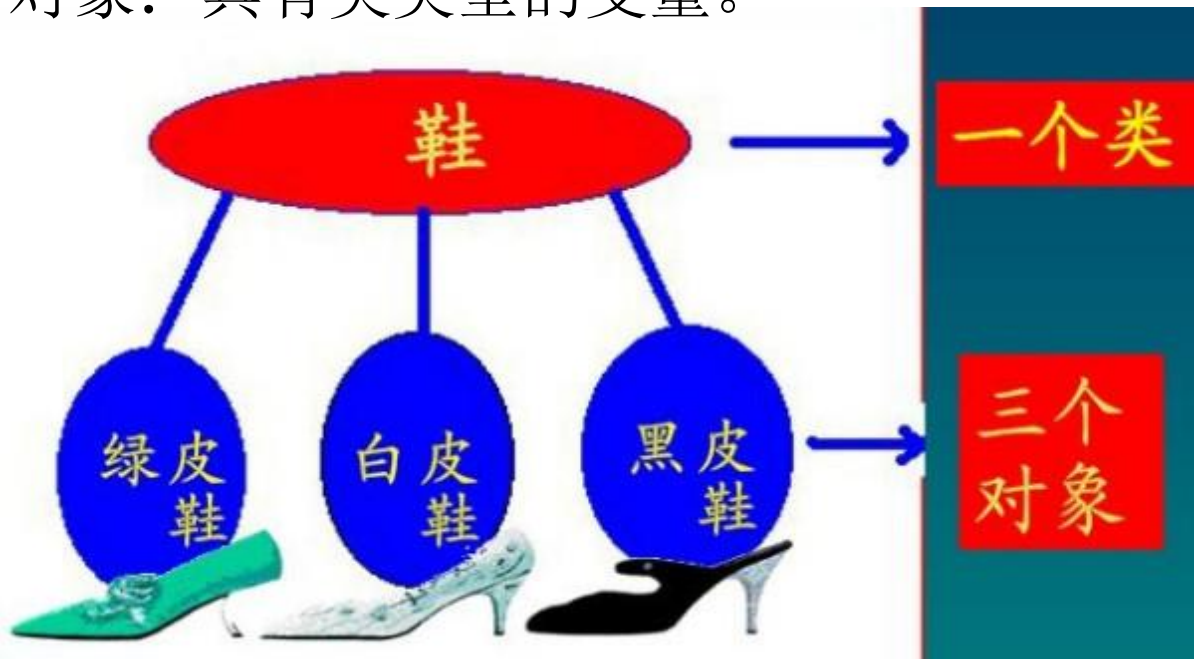
3.2.3 函数、类

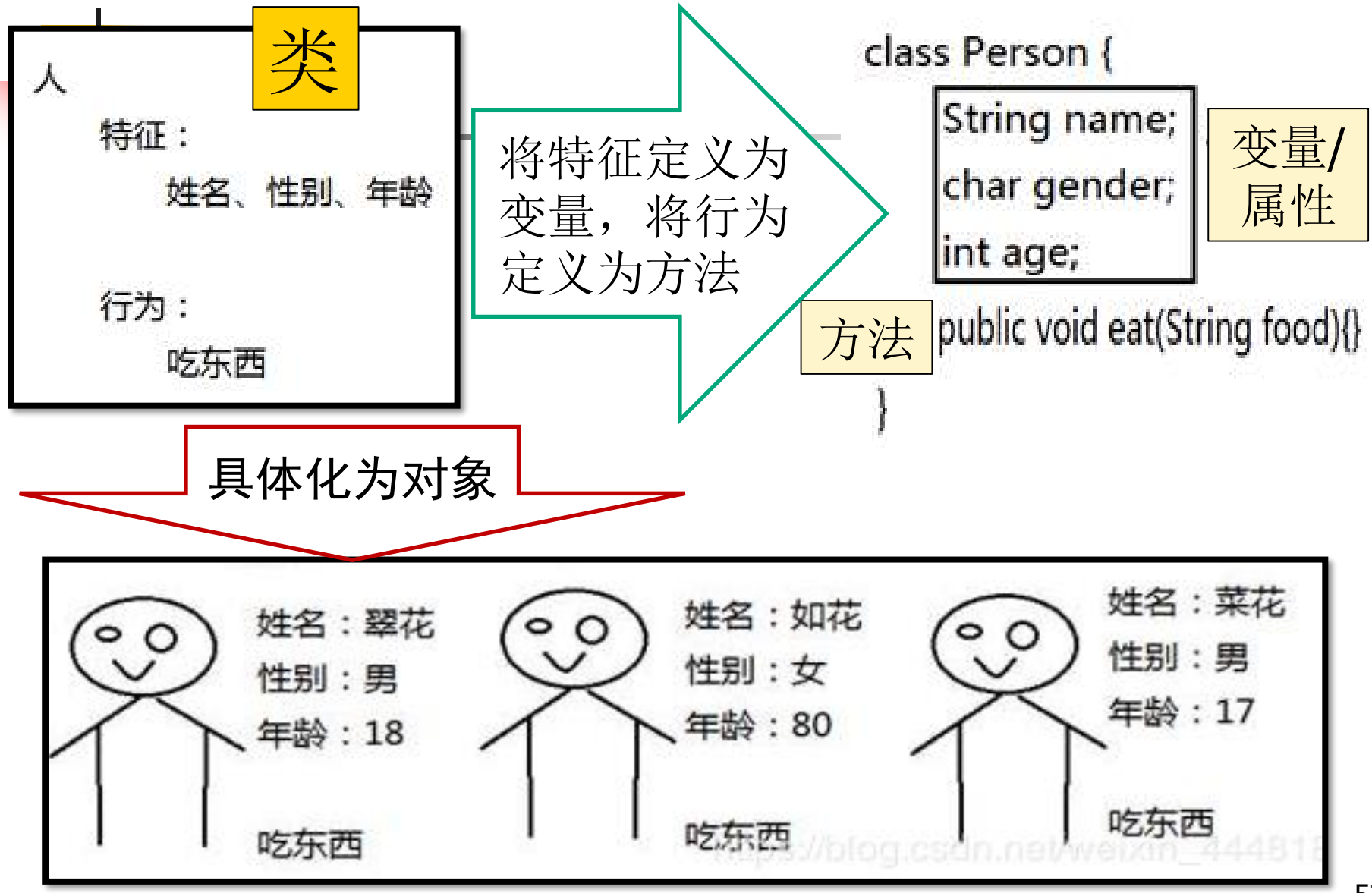
类

- Python 是一种面向对象的脚本语言。
- 面向对象的编程思想符合人们的思考习惯，可以将复杂的事情简单化。
- 面向对象编程思想， 将数据抽象为对象属性，把要实现的功能定位为对象的方法
- Python 中的类，具有面向对象编程的所有基本特征

3.2.3 函数、类

- 类和对象
  - 类：现实世界或思维世界中的实体在计算机中的反映，它将数据以及这些数据上的操作封装在一起。
  - 对象：具有类类型的变量。




类
人
特征：
姓名、性别、年龄
行为：
吃东西
将特征定义为变量，将行为定义为方法
class Person {
String name;
char gender;
int age;
变量/属性
方法
public void eat(String food){}
}
具体化为对象
姓名：翠花
性别：男
年龄：18
吃东西
姓名：如花
性别：女
年龄：80
吃东西
姓名：菜花
性别：男
年龄：17
吃东西

- Python定义类：

```
class MyClass:                          #定义一个类
    """A simple example class"""        #定义该类的说明字符串
    i = 12345                           #定义成员变量
    def f(self):                        #定义方法
        return 'I love Python'
```

- 类的文档字符串；
- 使用__doc__访问：类的内置属性

- 使用类：

```
myc =MyClass ()
print(myc.i)
print(myc.f())
```

```
>>> myc.f()
'I love Python'
>>> MyClass.i
12345
```

- 在定义类方法时，需要有一个默认的形参self，由类的内部机制调用的
- 在调用类方法时，不需要传入对应的实参。

3.2.3 函数、类

- 两个下划线

- 类的内置属性
  - Python 中的类有一些内置属性，用于维护类的基本信息
    - __name__ ：类名称。
    - __doc__ ：类的文档字符串。
    - __module__ ：类定义所在的模块。如果是直接运行当前文件，该值为main
    - __base__： 该类所有的父类＜class 『object′ ＞列表，是一个阳ple 类型的对象。
    - __dict__ ： 该类的所有属性（由类的数据属性组成），是一个diet 类型的对象。

```
print(MyClass.__name__)          #打印类的名字，输出：MyClass
print(MyClass.__doc__)           #打印类的文档字符串，输出：A simple example class
print(MyClass.__module__)        #打印类定义所在的模块，输出：__main__
print(MyClass.__base__)          #打印类的所有父类，输出：<class 'object'>
print(MyClass.__dict__)
```

```
>>> print(MyClass.__name__)
MyClass
>>> print(MyClass.__doc__)
A simple example class
>>> print(MyClass.__module__)
__main__
>>> print(MyClass.__base__)
<class 'object'>
>>> print(MyClass.__dict__)
{'__module__': '__main__', '__doc__': 'A simple example class', 'i': 12345, 'f': <fu
nction MyClass.f at 0x000001D1161D8378>, '__dict__': <attribute '__dict__' of 'MyCla
ss' objects>, '__weakref__': <attribute '__weakref__' of 'MyClass' objects>}
```

```
>>> myc=MyClass()
>>> print(myc.__name__)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'MyClass' object has no attribute '__name__'
```

访问类的内置属性用类名，而不是对象名！

3.2.3 函数、类

```
class MyClass:                                  #定义一个类
    """A record class"""                        #定义该类的说明字符串
    def __init__(self, name, age):              #定义该类的初始化函数
        self.name = name                        #将传入的参数值赋值给成员变量
        self.age = age

    def getrecode(self):                        #定义一个成员函数
        return self.name,self.age               #该成员函数返回该类的成员变量

myc =MyClass ("Anna",42)                        #实例化一个对象，并为其初始化
print(myc.getrecode())                          #调用对象的成员函数，并将返回值打印。
```

```
输出元组类型: ('Anna', 42)
```

3.2.3 函数、类

隐藏调用类的初始化方法

- 每个类在实例化时，都会在内部调用初始化函数__init__
- 对于一个没有初始化函数的类，在实例化时，也会调用内部默认的__init__函数
- 如果在类中实现了函数__init__ ，就会优先调用自定义的函数__init__

```
class MyClass:                    #定义一个类
    """A record class"""          #定义该类的说明字符串
    def __init__(self):           #定义该类的初始化函数
        print("here")             #将传入的参数值赋值给成员变量

myc =MyClass ()                   #实例化一个对象。输出：here
```

3.2 Python语言进阶

3.2.1 Python变量

3.2.2 控制流

3.2.3 函数、类

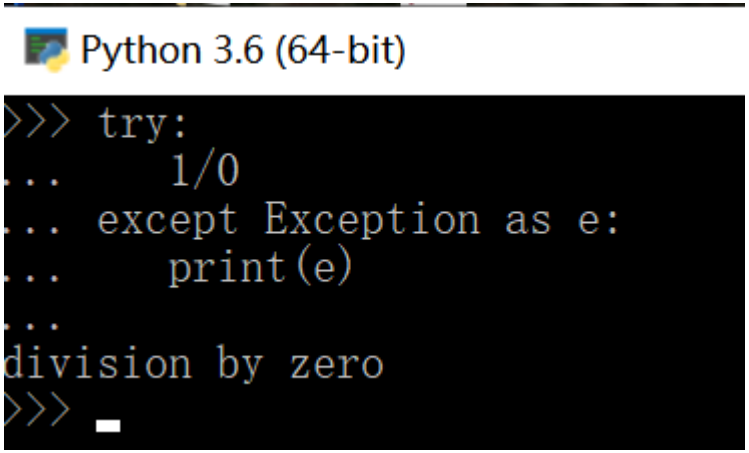
3.2.4 文件操作与异常

3.2.5 实例

3.2.4 文件操作与异常

- 异常：是一个事件，该事件会在程序执行过程中发生，影响了程序的正常执行。
- 一般情况下，在Python无法正常处理程序时就会发生一个异常。

```
try:
    正常的操作
    ......................
except:
    发生异常，执行这块代码
    ......................
else:
    如果没有异常执行这块代码
```



3.2.4 文件操作与异常

- 文件操作——数据持久化的一种方法
- 数据持久化，就是将Python 程序中的对象以文件的方式存储在磁盘上， 便于以后读取。
- Python 中，可以通过调用内置函数的方法进行文件的建立、读、写、删除等操作。
- 编码时，可以导入Python 中的专用模块(os 、sys等)

```
import os
os.remove('a.txt')                    #删除文件 a.txt
```

3.2.4

- r： 只读。文件必须存在。
- w： 只写。如果文件已存在，则将其覆盖。如果该文件不存在，则创建新文件。
- r+ ： 读写。文件必须存在。当写入时，会清空原内容。
- w＋： 读写。如果该文件不存在，则创建新文件。如果文件已存在，则清空原有内容。

- 读写文件
为3 步:
- (1) 打开文件： 使用open 函数，返回的是一个文件对象
- (2)具体读写：使用该文件对象的read 、write 等方法
- (3)关闭文件： 使用该文件对象的close 方法。

```
open（文件名，mode）
```

```
f = open('a.txt','w')
f.write('efgh')
f = open('a.txt','r')
s = f.read()
print(s)
```

```
>>> f=open('aaa.txt','w')
>>> f.write('aaaaaaaaaaa')
11
>>> f=open('aaa.txt','r')
>>> s=f.read()
>>> s
'aaaaaaaaaaa'
>>> print(s)
aaaaaaaaaaa
```

3.2.4 文件操作与异常

- 使用with 语句简化代码
  - 在Python 编程中，有很多代码流程具有事先、事中、事后明显的三个阶段
  - with 语句只需关心事先、事中，不关心事后事情
  - with 语句可以让文件对象在使用后被正常关闭
    - 如文件操作
      - 事先需打开文件(open)
      - 事中需进行文件操作
      - 事后需关闭文件(close)

3.2.4 文件操作与异常

```
with open('a.txt','wb+') as f:          #以二进制形式打开一个文件
    try:
        f.write('I like Python!')       #以文本的方式写入一个用二进制打开的文件，会报错
    except Exception as e:              #将错误异常捕获
        print(e )
        f.write(b'I like Python!')      #以 bytes 对象的形式进行读写

with open('a.txt','r+') as f:           #打开文件
    for line in f:                      #直接使用 for 循环读取文件
        print(line)                     #将内容打印出来
```

3.2.4 文件操作与异常

- 文件操作例：first_get_and_save_file.py

```
import requests          #引入requests模块

def get_content(url):                    #定义函数
    resp=requests.get(url)
    return resp.text

if __name__=='__main__':
    url="https://www.phei.com.cn/"       #定义URL值，目标网站地址
content=get_content(url)                 #调用函数值返回值
print("前200个字符为：", content[0:200])
content_len=len(content)
print("内容的长度为：", content_len)
if content_len>=40*1024:                 #判断内容长度是否大于40K
    print("内容长度是否大于等于40KB.")
else:
     print("内容长度是否小于等于40KB.")
```

3.2.4 文件操作与异常

- 文件操作例first_get_and_save_file.py

```
#方式1
#文件的写入
print('-'*20)
print('方式1:','文件写入')
f1=open('home_page.html', 'w', encoding='utf8')
f1.write(content)
f1.close()

#文件的读取

print('方式1:','文件读取')
f2=open('home_page.html', 'r', encoding='utf8')
content_read=f2.read( )
print("方式1读取的前200个字符为:", content_read[0:200])
f2.close()
```

3.2.4 文件操作与异常

- 文件操作例first_get_and_save_file.py

```
#方式2
#文件的写入
print('-'*20)
print('方式2: ','文件写入')
with open('home_page_2.html' , 'w' , encoding='utf8') as f3:
    f3.write(content)


#文件的读取

print('方式2: ','文件读取')
with open('home_page_2.html' , 'r' , encoding='utf8') as f4:
    content_read2=f4.read( )
    print("方式2读取的前200个字符为: " , content_read[0:200])
```

- first_get_and_save_file.py运行结果

```
前200个字符为:  ï»¿<!DOCTYPE HTML>
<html ng-app="topDivApp"  ng-controller="topDivCtrl">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta http-equiv = "X-UA-Compatible" content = "IE=e
内容的长度为:  49479
内容长度是否大于等于40KB.
--------------------
方式1:  文件写入
方式1:  文件读取
方式1读取的前200个字符为:  ï»¿<!DOCTYPE HTML>
<html ng-app="topDivApp"  ng-controller="topDivCtrl">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta http-equiv = "X-UA-Compatible" content = "IE=e
--------------------
方式2:  文件写入
方式2:  文件读取
方式2读取的前200个字符为:  ï»¿<!DOCTYPE HTML>
<html ng-app="topDivApp"  ng-controller="topDivCtrl">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta http-equiv = "X-UA-Compatible" content = "IE=e
```

3.2 Python语言进阶

3.2.1 Python变量

3.2.2 控制流

3.2.3 函数、类

3.2.4 文件操作与异常

3.2.5 实例

3.2.3 函数、错误与异常

- 例.用字典和列表两种方式描述抓取网页.

```
import requests

urls_dict={
        '电子工业出版社' : 'http://www.phei.com.cn/'
        '在线资源' : 'http://www.phei.com.cn/module/zygl/zxzyindex.jsp',
        'xyz' : 'www.phei.com.cn/',
        '网上书店1' : 'http://www.phei.com.cn/module/goods/wssd_index.jsp',
        '网上书店2' : 'http://www.phei.com.cn/module/goods/wssd_index.jsp'
}


urls_lst=[
('电子工业出版社','http://www.phei.com.cn/'),
('在线资源' , 'http://www.phei.com.cn/module/zygl/zxzyindex.jsp'),
('xyz' , 'www.phei.com.cn/'),
('网上书店1', 'http://www.phei.com.cn/module/goods/wssd_index.jsp'),
('网上书店2' , 'http://www.phei.com.cn/module/goods/wssd_index.jsp')
]
```

抓取目标组织成字典

无效地址

抓取目标组织成列表

```python
#利用字典抓取
crawled_urls_for_dict=set()
for ind,name in enumerate(urls_dict.keys()):
    name_url=urls_dict[name]
    if name_url in crawled_urls_for_dict:
        print(ind,name, "已经抓取过了。")
    else:
        try:
            resp=requests.get(name_url)
        except Exception as e:
            print(ind,name, ':', str(e)[0:50])
            continue
        content=resp.text
        crawled_urls_for_dict.add(name_url)
        with open('bydict_'+name+'.html','w',encoding='utf8') as f:
            f.write(content)
            print("抓取完成:{} {}, 内容长度为{}".format(ind,name,len(content)))

for u in  crawled_urls_for_dict:
    print(u)

print('_'*60)
```

用集合记录已抓取网页地址

Request库抓取

```
抓取完成:0 电子工业出版社, 内容长度为134
抓取完成:1 在线资源, 内容长度为134
2 xyz : Invalid URL 'www.phei.com.cn/': No schema supplied
抓取完成:3 网上书店1, 内容长度为134
4 网上书店2 已经抓取过了。
http://www.phei.com.cn/module/goods/wssd_index.jsp
http://www.phei.com.cn/module/zygl/zxzyindex.jsp
http://www.phei.com.cn/
------------------------------------------------------------
```

```
#利用列表抓取
crawled_urls_for_list=set()
for ind,tup in enumerate(urls_lst):
    name=tup[0]
    name_url=tup[1]
    if name_url in crawled_urls_for_list:
        print(ind,name, "已经抓取过了。")
    else:
        try:
            resp=requests.get(name_url)
        except Exception as e:
            print(ind,name, ':', str(e)[0:50])
            continue
        content=resp.text
        crawled_urls_for_list.add(name_url)
        with open('bylist_'+name+'.html','w',encoding='utf8') as f:
            f.write(content)
            print("抓取完成:{} {}, 内容长度为{}".format(ind,name,len(content)))

for u in  crawled_urls_for_list:
    print(u)
```

```
抓取完成:0 电子工业出版社, 内容长度为134
抓取完成:1 在线资源, 内容长度为134
2 xyz : Invalid URL 'www.phei.com.cn/': No schema supplied
抓取完成:3 网上书店1, 内容长度为134
4 网上书店2 已经抓取过了。
http://www.phei.com.cn/
http://www.phei.com.cn/module/zygl/zxzyindex.jsp
http://www.phei.com.cn/module/goods/wssd_index.jsp
```