1. 请写出一段代码实现单例模式。
2. 答：#-\*- encoding=utf-8 -\*-
3. **print** '----------------------方法1--------------------------'
4. #方法1,实现\_\_new\_\_方法
5. #并在将一个类的实例绑定到类变量\_instance上,
6. #如果cls.\_instance为None说明该类还没有实例化过,实例化该类,并返回
7. #如果cls.\_instance不为None,直接返回cls.\_instance
8. **class** Singleton(object):
9. **def** \_\_new\_\_(cls, \*args, \*\*kw):
10. **if** **not** hasattr(cls, '\_instance'):
11. orig = super(Singleton, cls)
12. cls.\_instance = orig.\_\_new\_\_(cls, \*args, \*\*kw)
13. **return** cls.\_instance
15. **class** MyClass(Singleton):
16. a = 1
18. one = MyClass()
19. two = MyClass()
21. two.a = 3
22. **print** one.a
23. #3
24. #one和two完全相同,可以用id(), ==, is检测
25. **print** id(one)
26. #29097904
27. **print** id(two)
28. #29097904
29. **print** one == two
30. #True
31. **print** one **is** two
32. #True
34. **print** '----------------------方法2--------------------------'
35. #方法2,共享属性;所谓单例就是所有引用(实例、对象)拥有相同的状态(属性)和行为(方法)
36. #同一个类的所有实例天然拥有相同的行为(方法),
37. #只需要保证同一个类的所有实例具有相同的状态(属性)即可
38. #所有实例共享属性的最简单最直接的方法就是\_\_dict\_\_属性指向(引用)同一个字典(dict)
39. #可参看:http://code.activestate.com/recipes/66531/
40. **class** Borg(object):
41. \_state = {}
42. **def** \_\_new\_\_(cls, \*args, \*\*kw):
43. ob = super(Borg, cls).\_\_new\_\_(cls, \*args, \*\*kw)
44. ob.\_\_dict\_\_ = cls.\_state
45. **return** ob
47. **class** MyClass2(Borg):
48. a = 1
50. one = MyClass2()
51. two = MyClass2()
53. #one和two是两个不同的对象,id, ==, is对比结果可看出
54. two.a = 3
55. **print** one.a
56. #3
57. **print** id(one)
58. #28873680
59. **print** id(two)
60. #28873712
61. **print** one == two
62. #False
63. **print** one **is** two
64. #False
65. #但是one和two具有相同的（同一个\_\_dict\_\_属性）,见:
66. **print** id(one.\_\_dict\_\_)
67. #30104000
68. **print** id(two.\_\_dict\_\_)
69. #30104000
71. **print** '----------------------方法3--------------------------'
72. #方法3:本质上是方法1的升级（或者说高级）版
73. #使用\_\_metaclass\_\_（元类）的高级python用法
74. **class** Singleton2(type):
75. **def** \_\_init\_\_(cls, name, bases, dict):
76. super(Singleton2, cls).\_\_init\_\_(name, bases, dict)
77. cls.\_instance = None
78. **def** \_\_call\_\_(cls, \*args, \*\*kw):
79. **if** cls.\_instance **is** None:
80. cls.\_instance = super(Singleton2, cls).\_\_call\_\_(\*args, \*\*kw)
81. **return** cls.\_instance
83. **class** MyClass3(object):
84. \_\_metaclass\_\_ = Singleton2
86. one = MyClass3()
87. two = MyClass3()
89. two.a = 3
90. **print** one.a
91. #3
92. **print** id(one)
93. #31495472
94. **print** id(two)
95. #31495472
96. **print** one == two
97. #True
98. **print** one **is** two
99. #True
101. **print** '----------------------方法4--------------------------'
102. #方法4:也是方法1的升级（高级）版本,
103. #使用装饰器(decorator),
104. #这是一种更pythonic,更elegant的方法,
105. #单例类本身根本不知道自己是单例的,因为他本身(自己的代码)并不是单例的
106. **def** singleton(cls, \*args, \*\*kw):
107. instances = {}
108. **def** \_singleton():
109. **if** cls **not** **in** instances:
110. instances[cls] = cls(\*args, \*\*kw)
111. **return** instances[cls]
112. **return** \_singleton
114. @singleton
115. **class** MyClass4(object):
116. a = 1
117. **def** \_\_init\_\_(self, x=0):
118. self.x = x
120. one = MyClass4()
121. two = MyClass4()
123. two.a = 3
124. **print** one.a
125. #3
126. **print** id(one)
127. #29660784
128. **print** id(two)
129. #29660784
130. **print** one == two
131. #True
132. **print** one **is** two
133. #True
134. one.x = 1
135. **print** one.x
136. #1
137. **print** two.x
138. #1

2、*请用yield写出一个生成器*。

**答：**def generator():

for i in range(5):

yield i

3、*说明os,sys模块不同，并列举常用的模块方法。*

**答：**os模块负责程序与操作系统的交互，提供了访问操作系统底层的接口;

sys模块负责程序与python解释器的交互，提供了一系列的函数和变量，用于操控python的运行时环境。

常用的模块方法：

os常用模块：

os.remove()删除文件

os.rename()重命名文件

os.walk()生成目录树下的所有文件名

os.chdir()改变目录

os.mkdir/makedirs创建目录/多层目录

os.rmdir/removedirs删除目录/多层目录

os.listdir()列出指定目录的文件

os.getcwd()取得当前工作目录

os.chmod()改变目录权限

os.path.basename()去掉目录路径，返回文件名

os.path.dirname()去掉文件名，返回目录路径

os.path.join()将分离的各部分组合成一个路径名

os.path.split()返回（dirname(),basename())元组

os.path.splitext()(返回filename,extension)元组

os.path.getatime\ctime\mtime分别返回最近访问、创建、修改时间

os.path.getsize()返回文件大小

os.path.exists()是否存在

os.path.isabs()是否为绝对路径

os.path.isdir()是否为目录

os.path.isfile()是否为文件

sys常用模块：

sys.argv 命令行参数List，第一个元素是程序本身路径

sys.modules.keys() 返回所有已经导入的模块列表

sys.exc\_info() 获取当前正在处理的异常类,exc\_type、exc\_value、exc\_traceback当前处理的异常详细信息

sys.exit(n) 退出程序，正常退出时exit(0)

sys.hexversion 获取Python解释程序的版本值，16进制格式如：0x020403F0

sys.version 获取Python解释程序的版本信息

sys.maxint 最大的Int值

sys.maxunicode 最大的Unicode值

sys.modules 返回系统导入的模块字段，key是模块名，value是模块

sys.path 返回模块的搜索路径，初始化时使用PYTHONPATH环境变量的值

sys.platform 返回操作系统平台名称

sys.stdout 标准输出

sys.stdin 标准输入

sys.stderr 错误输出

sys.exc\_clear() 用来清除当前线程所出现的当前的或最近的错误信息

sys.exec\_prefix 返回平台独立的python文件安装的位置

sys.byteorder 本地字节规则的指示器，big-endian平台的值是'big',little-endian平台的值是'little'

sys.copyright 记录python版权相关的东西

sys.api\_version 解释器的C的API版本

sys.version\_info

sys.argv 命令行参数List，第一个元素是程序本身路径

sys.modules.keys() 返回所有已经导入的模块列表

sys.exc\_info() 获取当前正在处理的异常类,exc\_type、exc\_value、exc\_traceback当前处理的异常详细信息

sys.exit(n) 退出程序，正常退出时exit(0)

sys.hexversion 获取Python解释程序的版本值，16进制格式如：0x020403F0

sys.version 获取Python解释程序的版本信息

sys.maxint 最大的Int值

sys.maxunicode 最大的Unicode值

sys.modules 返回系统导入的模块字段，key是模块名，value是模块

sys.path 返回模块的搜索路径，初始化时使用PYTHONPATH环境变量的值

sys.platform 返回操作系统平台名称

sys.stdout 标准输出

sys.stdin 标准输入

sys.stderr 错误输出

sys.exc\_clear() 用来清除当前线程所出现的当前的或最近的错误信息

sys.exec\_prefix 返回平台独立的python文件安装的位置

sys.byteorder 本地字节规则的指示器，big-endian平台的值是'big',little-endian平台的值是'little'

sys.copyright 记录python版权相关的东西

sys.api\_version 解释器的C的API版本

sys.version\_info

4、*@classmethod,@staticmethod,@property都是什么意思？*

**答：**@property

这个装饰器可以让你用直接「实例名.属性」来读和改写属性，表面看去是取值或用「=」来赋值，其内核却是调用get或者set方法，其用法是直接修饰在get方法前面，或者在set方法前面用@方法名.setter

@staticmethod

目前为止见过的所有类中的函数都要传入self这个实例。顾名思义，静态方法，@staticmethod可以修饰不需要self的函数，这类函数不加@staticmethod的话报错

@classmethod

顾名思义，类方法，用它来修饰的方法，其第一个参数不是self，而是cls，也就是这个类本身作为参数

5、python里面match() 和 search() 的区别？

答：match（）函数只检测RE是不是在string的开始位置匹配， search()会扫描整个string查找匹配, 也就是说match（）只有在0位置匹配成功的话才有返回，如果不是开始位置匹配成功的话，match()就返回none

例如：

print(re.match(‘super’, ‘superstition’).span())会返回(0, 5)

而print(re.match(‘super’, ‘insuperable’))则返回None

search()会扫描整个字符串并返回第一个成功的匹配

例如：print(re.search(‘super’, ‘superstition’).span())返回(0, 5)

print(re.search(‘super’, ‘insuperable’).span())返回(2, 7)

6、*用Python匹配HTMLtag的时候，<.\*> 和 <>\*?> 有什么区别？*

**答：**<.\*>这种匹配称作贪心匹配 <.\*?>称作非贪心匹配

当重复匹配一个正则表达式时候， 例如<.\*>, 当程序执行匹配的时候，会返回最大的匹配值

例如：

import re

s = ‘<html><head><title>Title</title>’

print(re.match(’<.\*>’, s).group())

会返回一个匹配<html><head><title>Title</title>而不是<html>

而

import re

s = ‘<html><head><title>Title</title>’

print(re.match(’<.\*?>’, s).group())

则会返回<html>

7、*请解释一下Gevent和 threading/multiprocessing的关系。*

**答：**

* Gevent是第三方库，通过greenlet实现协程，其基本思想是：

当一个greenlet遇到IO操作时，比如访问网络，就自动切换到其他的greenlet，等到IO操作完成，再在适当的时候切换回来继续执行。由于IO操作非常耗时，经常使程序处于等待状态，有了gevent为我们自动切换协程，就保证总有greenlet在运行，而不是等待IO。

* threading用于提供线程相关的操作，线程是应用程序中工作的最小单元。python当前版本的多线程库没有实现优先级、线程组，线程也不能被停止、暂停、恢复、中断。

threading模块提供的类：

　　Thread, Lock, Rlock, Condition, [Bounded]Semaphore, Event, Timer, local。

threading 模块提供的常用方法：

　　threading.currentThread(): 返回当前的线程变量。

　　threading.enumerate(): 返回一个包含正在运行的线程的list。正在运行指线程启动后、结束前，不包括启动前和终止后的线程。

threading.activeCount(): 返回正在运行的线程数量，与len(threading.enumerate())有相同的结果。

* multiprocessing是python的多进程管理包，和threading.Thread类似。直接从侧面用subprocesses替换线程使用GIL的方式，由于这一点，multiprocessing模块可以让程序员在给定的机器上充分的利用CPU。

在multiprocessing中，通过创建Process对象生成进程，然后调用它的start()方法

8、*请解释一下cookie 和 session 的关系，以及xsrf的攻击和防范方法。*

**答：**cookie保存在浏览器端，session保存在服务器端，但是为了区分不同的客户端，服务器会在浏览器中发送一个对应的sessionid保存到cookies中，下次浏览器请求服务器的时候会将sessionid一并发送给服务器。所以session机制依赖于cookie机制。

csrf攻击：cross site request forgery 跨站请求伪造

原理：

1.User(C)访问信任网站Web(A)，Web(A)会在User(C)处设置A的cookie；

2.User(C) 在没有登录A的情况下,访问危险网站Web(B),B要求User(C)访问Web(A);

3.User(C)在Web(B)的要求下，带着Web(A)的cookie访问Web(A)，这样Web(B)就达到了模拟用户操作的目的

防御：

1.通过 referer、token 或者 验证码 来检测用户提交。

2.尽量不要在页面的链接中暴露用户隐私信息。

3.对于用户修改删除等操作最好都使用post 操作 。

4.避免全站通用的cookie，严格设置cookie的域。

9、*请设计一个系统消息功能，支持对个人、组、全员发布消息。用户可以看到未读消息数量。*

**提示：**

·使用Django的signals来获取Model的新建/删除操作更新

·使用数据库的select for update来正确处理并发的数据库操作

·使用redis的sorted set来缓存计数器的修改操作

*10、字典 a = {1:1} 是否正确？*

**答：**正确。

*11、请合并列表 a = [1,2,3,4] 和 列表 b =[5,6,7,8]*

**答：**a.extend(b)

*12、列表a，请写出实现正序排列，倒序排列，逆序排列的方法。*

**答：**正序排列 a.sort()

倒序排列 a.sort(reverse=True)

逆序 a.reverse()

*13、字典 d = {“k”:1,”v”:2},请写出d.items()结果。*

**答：**dict\_items([('v', 2), ('K', 1)])

*14、复杂列表[{“k”:1,”v”:2},{“k”:12,”v”:22},{“k”:13,”v”:32}]，请用内置方法写出按K的倒序排列的代码。*

**答：**li = [{“k”:1,”v”:2},{“k”:12,”v”:22},{“k”:13,”v”:32}]

li.sort(key=lambda dict:dict[“k”],reverse=True)

print(li)

*15、集合 s = set([1,2,3,4]), d = set([2,4,9,0,3]),请用内置方法写出他们的并集，交集，对称公差。*

**答：**并集 s.union(d) = {0,1,2,3,4,9}

交集 s.intersection(d) = {2,3,4}

对称公差 s.symmetric\_difference(d) = {0,1,9}

或

s | d #交集

s & d #并集

s ^ d #对称差集

*16、如何把列表 a = [“a”,”b”]里的各项，转为字符串并用逗号”,”连接。*

**答：**”,”.join(a)

*17、请随机输出大于0小于1的数；请随机输出一个100以内的整数。*

**答：**import random

random.random() #随机输出一个大于0小于1的数

random.randint(1,100) #随机输出一个100以内的整数

*18、长度未知的元组 a = (1,3,4,5,6,……..)，请随机输出n个项。*

**答：**import random

a = (1,3,4,5,6,…)

a = list(a)

for i in range(n):

b = a[random.randint(0,len(a))-1]

print(b)

1. remove(b)

*19、代码 a = [{i:j} for i,j in enumerate(xrange(5))],请写出a的最终结果。*

答：a = [{0:0},{1:1},{2:2},{3:3},{4:4}]

#enumerate()函数：枚举、列举，返回一个可迭代对象的索引和索引对应的值

*20、分别说说函数cmp，map，filter 的作用？*

**答：**# cmp:compare比较

cmp(a,b) # a<b返回-1 a>b返回1 a=b返回0

# map: 映射

Map(function,iterable) #将function作用于iterable每个元素，将对应输出结果保存为一个list

# 例：

def plus\_one(x):

return x+1

map(plus\_one,[1,2,3,4]) ===>[2,3,4,5]

# filter:过滤器

filter(function, iterable) # 将 function依次作用于iterable的每个元素，如果返回值为true, 保留元素，否则从iterable里面删除

# 例：

def bigger\_than\_three(x):

return (x>3)

filter(bigger\_than\_three,[1,2,3,4,5]) ===>[4,5]

*21、请阐述\_\_new\_\_ 和 \_\_init\_\_ 的区别？*

**答：**1、\_\_new\_\_是一个静态方法,而\_\_init\_\_是一个实例方法.

2、\_\_new\_\_方法会返回一个创建的实例,而\_\_init\_\_什么都不返回.

3、只有在\_\_new\_\_返回一个cls的实例时后面的\_\_init\_\_才能被调用.

4、当创建一个新实例时调用\_\_new\_\_,初始化一个实例时用\_\_init\_\_.