**打包工具:**

**distutils：**

**介绍：**

**distutils 是 python 标准库的一部分，2000年发布，使用它能够进行 python 模块的安装和发布。**

**要求：**

**setup.py:**

**from distutils.core import setup**

**setup(**

**name="zxf-module",**

**version="1.0",**

**author="zxf",**

**py\_modules=['zxfmo'],**

**)**

**打包：<name>-<version>.tar.gz**

**pyhon setup.py sdist**

**pyhon setup.py bdist\_rpm**

**pyhon setup.py bdist\_wininst**

**从源码安装:**

**sudo pyhon setup.py install**

**从本地包安装：**

**sudo ./dist/easy\_install zxfmodule-1.0.tar.gz**

**setuptools：**

**介绍：**

**setuptools 是一个为了增强 distutils 而开发的集合，2004年发布。它包含了 easy\_install 这个工具。**

**ez\_setup.py 是 setuptools 的安装工具。**

**要求：**

**setup.cfg**

**setup.py:**

**from setuptools import setup**

**setup(**

**name="zxf-module",**

**version="1.0",**

**author="zxf",**

**py\_modules=['zxfmo'],**

**)**

**打包：<name>-<version>.tar.gz**

**pyhon setup.py sdist**

**pyhon setup.py bdist\_rpm**

**pyhon setup.py bdist\_wininst**

**pyhon setup.py bdist\_wheel**

**pyhon setup.py bdist\_egg**

**从源码安装：**

**by manual from local:**

**sudo python setup.py install**

**从安装包安装：**

**by easy\_install from pypi:**

**easy\_install <package>**

**by pip from pypi:**

**pip install <package>**

**打包格式：**

**tar.gz：标准压缩文件格式。**

**.egg格式(supported by setuptools)：Egg包，本质上也是压缩文件，元数据在.egg-info目录。**

**<package>**

**<package>.egg-info**

**setup.cfg**

**setup.py**

**requirements.txt**

**.whl格式：Wheel包，本质上也是压缩文件，元数据在.dist-info目录。**

**包安装工具：**

**Easy\_install－python 2.7**

**说明：**

**easy\_install是setuptools包里带的一个命令，它用来安装egg包；egg包是目前最流行的Python打包部署方式。**

**Install easy\_install(**

**setuptools**

**):**

**方式一：from self setup package**

**python setup.py install**

**方式二：**

**apt-get install python-setuptools**

**yum install python-setuptools**

**Usage:**

**from default pypi：**

**easy\_install sshuttle**

**easy\_install pip**

**from specifed pypi：**

**easy\_install --index-url**[**http://rnd.mirrors.huawei.com/pypi/simple**](http://rnd.mirrors.huawei.com/pypi/simple)**pip**

**easy\_install -i**[**http://rnd.mirrors.huawei.com/pypi/simple**](http://rnd.mirrors.huawei.com/pypi/simple)**pip**

**from tgz file：**

**easy\_install**<http://example.com/path/to/MyPackage-1.2.3.tar.gz>

**from egg file**

**easy\_install /my\_downloads/OtherPackage-3.2.1-py2.3.egg**

**Pip3 － requiments.txt**

**Install pip：**

**方式一：**

**easy\_install pip**

**方式二：**

**apt-get install python-pip**

**yum install python-pip**

**Pip source:**

**Path:**

**~/.pip/pip.conf**

**%User%/pip/pip.ini**

**Content:**

**[global]**

**trusted-host=rnd.mirrors.huawei.com**

**index-url=**[**http://rnd.mirrors.huawei.com/pypi/simple**](http://rnd.mirrors.huawei.com/pypi/simple)

**Usage:**

**#version**

**pip -V**

**#install package**

**pip install xxx**

**pip install xxx==1.0.4**

**pip install xxx>-1.0.4**

**#from specfied index\_url**

**pip install xxx**

**--index-url**

**http://pypi.douban.com/simple/ --trusted-host pypi.douban.com**

**pip install xxx -i  http://pypi.douban.com/simple/ --trusted-host pypi.douban.com**

**#install packages from requirements.txt**

**pip install -r requirements.txt**

**#from specfied index\_url**

**pip install -i  http://pypi.douban.com/simple/ -r requirements.txt**

**#from local folder**

**pip install --no-index -f <DIR> -r requirement.txt**

**#from local archive**

**pip install ./downloads/SomeProject-1.0.4.tar.gz**

**#from local source**

**pip install <path>**

**#uninstall packege**

**pip uninstall xxx**

**#list packages**

**pip list**

**#generate  requirements.txt**

**pip freeze**

**#show detail**

**pip show <pkg-name>**

**#search package**

**pip search xxx**

**#build wheel archives for your requirements and dependencies.**

**pip wheel -i http://pypi.wg-t.com:8000/wg/dev/ --use-wheel -w ~/.pip/wheelhouse -r requirements.txt**

**#config file**

**~/.pip/pip.conf**

**#local install**

**pip install --download <DIR> -r requirement.txt**

**Virtualenv**

**#insall:**

**pip3 install virtualenv**

**#make a python enviroment with specific version.**

**virtualenv -p `which python3` <path-for-env>**

**#active the python environment created previos in current shell on everyt ime you want use it.**

**source <path-for-env>/bin/activate**

**venv:**

**#insall:**

**?????**

#make a python enviroment with specific version.

**?????**

**将在当前目录创建一个venv目录**

**#active the python environment created previos in current shell on everyt ime you want use it.**

**source ./venv/bin/activate**

**Python:**

**#run library module as a script**

**python -m <module> <args-of-module>**

**#Examples**

**#start http server with python2 in current folder**

**python -m SimpleHTTPServer <port:8000>**

**#start http server with python3**

**in current folder**

**python3 -m http.server**

**<port:8000> [—bind <ip>]**

**常犯的错误：**

**使用了未定义的变量**

**next(<list>)**

**Table.field.in\_(<empty\_list>)**

**善用dict.get(“key”, <default>)**

**virtualenv在有些用戶下有问题，导致default encoding变为assiic**

**软件包安装出错，提示找不到pyconfig.h，需要安装python-devel包**

**Python语言特性：**

**交互性**

**解释型**

**可移植**

**脚本**

**面向对象**

**动态类型**

**自动内存管理**

**胶水语言(C，C++，Java，.NET)**

**Python语法特点：**

**.py**

**大小写敏感**

**代码缩进决定代码的逻辑关系**

**注释：＃, ‘’’, “””**

**冒号：**

**Python的编码注释:**

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

**如果要在python2的py文件里面写中文，则必须要添加一行声明文件编码的注释，否则python2会默认使用ASCII编码。**

**Python中的语句：**

**声明型语句－－－》面向对象，函数，声明式**

**执行型语句－－－》脚本型，流水式**

**Python的可移植：**

**大多数Python外围接口都是平台相关的扩展，但是其核心语言和库在任何平台都是一样可以运行的。**

**Python的可混合：**

**Python的C语言API可以帮助Python程序灵活地调用C程序。**

**Python的动态性：**

**Python中所有的事情都是在程序运行时发生的，这甚至包括创建函数和类的操作以及导入模块的操作；这些事情对于静态语言往往是在编译阶段规划好的，而对于Python则是通过代码执行出来的；比如静态语言的类，其内存布局是在编译阶段规划好的，因此其同一个类的每个对象的布局是一致的，而动态语言的类的内存布局是通过代码动态的创建出来的，因此其同一个类的每个对象可以有不同的动态布局（不是提前布局好的，而是动态执行出来的）。**

**Python的对象特性：**

**在Python中，一切皆是对象，模块是对象（类型为module），类是对象（类型为type），函数是对象（类型为function），类的实例也是对象（类型为该类）；用户自定义类型（类，函数）以及类型实例都有一个\_\_dict\_\_属性，其中记录了该对象上的所有属性－－属性表；内置类型被设计为不能够随意地为它增加属性或方法，因此内置类型的实例没有\_\_dict\_\_。**

**在Python中一切皆对象，类也是对象，可以在运行的时候动态创建；有多种方法可以创建出一个类（type的实例）：**

**使用class关键字：**

**使用关键字class的时候，Python解释器在执行的时候就会创建一个对象（类型为type）。**

**使用内建元类type：**

**type实际上是Python的一个内建元类，用来直接指导类的生成。**

**使用自定义元类：**

**用户可以通过继承type来创建自定义元类。**

**自定义元类的使用有两种方式：**

**使用自定义元类的构造函数（name，bases，dict）。**

**在需要被生成的类中设置\_\_metaclass\_\_属性。**

**元类：**

**元类用来指导类的生成；元方法可以从元类或者类中调用，不能从类的实例中调用；类方法既可以从类中调用也可以从类的实例中调用。**

**元方法：**

**定义在元类中的方法为元方法；元类中所定义的元方法为其所创建的类的类方法，并不属于该类的对象。**

**Python模块与包：**

**模块：**

**每个文件即为一个模块（其没有子模块）; 每个目录（包含 \_\_init\_\_.py）也是一个模块（可以包含子模块，如文件或目录）。**

**模块是对象，其中有一些内置属性，如\_\_author\_\_，\_\_doc\_\_， \_\_name\_\_。**

**一个模块的 \_\_name\_\_ 的值取决于您如何应用模块。如果 import 一个模块，那么模块\_\_name\_\_ 的值通常为模块文件名，不带路径或者文件扩展名。但是您也可以像一个标准的程序样直接运行模块，在这 种情况下, \_\_name\_\_ 的值将是一个特别缺省"\_\_main\_\_”。**

**sys.modules中保存有当前导入的所有模块的信息；因此多次导入只是指向了相同的模块实例，也只会执行一次。**

**让你的脚本既可以执行又可以被导入：**

**def main():**

**print("we are in" + \_\_name\_\_)**

**print(sys.argv)**

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

**main()**

**包：**

**package是一组module的集合。package是一种组织module的方法，提供了一个namespace，防止放生名字冲突。package中还可以有package，所以这种方式可以很好的组织一个树状结构，用来管理modules。每一个包目录下必须包含一个\_\_init\_\_.py文件以区别于普通目录，即使该文件内容为空，也可以在里面定义变量或者导出包内的模块。包的本质还是模块，\_\_init\_\_.py就是包对应的py文件。**

**Python模块搜索：**

**模块搜索路径存储在sys模块的sys.path变量中，运行时变量里包含当前目录，Shell变量PYTHONPATH指定的目录和由安装过程决定的默认目录。**

**Python模块管理：**

**外部模块导入：**

**#import root-module[.submodule]**

**import os**

**import os.path**

**#from root-module[.submodule] import submodule or element or \***

**from os import path**

**from os.path import exists**

**from keyword import kwlist**

**from keyword import \***

**工程内模块导入：**

**#Import module or class in current or sub package:**

**from . import <class>**

**from . import <submodule>**

**from .<submodule> import <class>**

**注意：代码在执行过程中可以从\_\_package\_\_中获取当前代码所在的包，并以此为依据做关联.引用。作为脚本执行的python文件中\_\_package\_\_为None，所以只能从Root Pacakge开始引用需要的包。**

**#Import module or class in parent package:**

**from <root\_module> import <class>**

**from <root\_module> import <submodule>**

**from <root\_module>.<submodule> import <class>**

**注意：系统从sys.path变量中查找root包。**

**Python程序架构：**

**脚本文件（顶层模块，启动后可运行程序）**

**模块文件（下层模块，程序库，可被其他模块导入）**

**Python类型系统：**

**简单类型：**

**Int：**

**int(\*\*\*)**

**123，0xff00**

**Long：**

**123L，0xff00L，**

**Float：**

**float(\*\*\*)**

**1.23，1.23e9**

**Complex：**

**3.0 + 1.2j**

**Bool：**

**bool(<value>)**

**True，False**

**容器类型：**

**String：**

**str(\*\*\*)**

**‘abc’     “abc”   ‘’’abc’''  “””abc”””    ‘\u4e42\u6587’    ‘,’.join(‘1’, ’2’, ’3')**

**b’abc’.decode(‘ascii’)**

**‘abcdef’[0:3]**

**‘abcdef’[2:]**

**‘abcdef’[:-2]**

**‘abcdef’[-5:-2]**

**‘abcdef’[::-2]**

**‘   abc    ‘.strip()**

**Bytes：**

**bytes(\*\*\*)**

**b’abc’    b’\xe4\xb8\xad\xe6’**

**‘abc’.encode(‘utf-8’)**

**b’\x31\x32\33’.decode(‘utf-8’)**

**Tuple ：()**

**tpl=tuple([1,2,3])**

**tpl=(‘one’,2,3)   tpl[0]     tpl[-1]**

**List ：[]**

**lst=list([1,2,3])**

**lst=[‘one',2,3]    lst[0]     lst[-1]=’three'     lst.append(4)    lst.insert(1,0)    lst.pop()     lst.pop(0)  lst.extend([4, 5, 6])**

**new\_lst = lst + [4, 5, 6]**

**Set ： {}**

**set1 = set([1,2,3])**

**set1={1,2,3}   set1.add(4)    set1.remove(1)     set1 & set2     set1 | set2**

**Dictionary ：{:}**

**dct=dict(key1=123,key2=345)**

**dct={1:”one”,2:”two”,3:”three”}      dct[4]=‘four’     5 in dct     dct.get(5)     dct.get(5,-1)     dct.pop(1)**

**注：Key必须是不可变对象或者说必须是可以通过hash(\*)求值的类型的实例。**

**Class：**

**class ClassName(base1,base2,base3):**

**self，super()**

**@staticmethod**

**def staticMethod()**

**@classmethod**

**def classMethod(cls)**

**@property**

**def property\_get(self)**

**@score.setter**

**def property\_set(self, value)**

**def instanceMethod(self)**

**注：Class在Python中是一个可执行的语句，因此语法class ClassName(ClassA if <condition> else ClassB)是成立的。**

**实例变量(定义于Class或者Module)命名约定：**

**\_\_xxx\_\_：特殊变量，可以直接访问。**

**\_\_xxx：私有变量，不可以直接访问。**

**\_xxx：约定私有变量，可以但不建议直接访问。**

**基础属性：**

**\_\_metaclass\_\_**

**\_\_class\_\_**

**\_\_base\_\_**

**\_\_bases\_\_（用来存放所有的基类，类型为元组）**

***\_\_*dict\_\_**

**基础重载方法：**

**\_\_init\_\_ ( self [,args...] )**

**构造函数**

**简单的调用方法: *obj = className(args)***

**\_\_del\_\_( self )**

**析构方法, 删除一个对象**

**简单的调用方法 : *dell obj***

**\_\_repr\_\_( self )**

**转化为供解释器读取的形式**

**简单的调用方法 : *repr(obj)***

**\_\_str\_\_( self )**

**用于将值转化为适于人阅读的形式**

**简单的调用方法 : *str(obj)***

**\_\_cmp\_\_ ( self, x )**

**对象比较**

**简单的调用方法 : *cmp(obj, x)***

**高级重载方法：**

**@staticmethod**

**\_\_new\_\_ ( cls, \*args, \*\*kwargs)**

**\_\_dir\_\_(self)**

**\_\_hash\_\_(self, \*args, \*\*kwargs)**

***\_\_getattribute\_\_(self, \*args, \*\*kwargs);***

***\_\_getattr\_\_(self, \*args, \*\*kwargs);***

***\_\_setattr\_\_(self, \*args, \*\*kwargs)***

**运算符重载方法：**

**\_\_add\_\_ ( self , other )**

**\_\_sub\_\_( self, other )**

**\_\_eq\_\_(self, \*args, \*\*kwargs):**

**\_\_ne\_\_(self, \*args, \*\*kwargs)**

**\_\_ge\_\_, \_\_gt\_\_, \_\_le\_\_, \_\_lt\_\_**

**Enum：**

**# Create Enum by code**

**from enum import Enum**

**Animals = Enum('Animals', ‘Cat, Dog’)**

**# Create Enum by declare**

**from enum import Enum**

**class Animals(Enum):**

**Cat = 1; Dog = 2**

**# Create instance of Enum**

**Animals.Dog**

**try:**

**Animals(1)**

**Animals(3)#will throw exception**

**expect ValueError:**

**#value is not valid**

**try:**

**Animals[‘Dog’]**

**Animals[‘Duck']#will throw exception**

**expect KeyError:**

**#key is not valid**

**作用域：**

**定义或修改本地变量：**

**abc = 123**

**定义或修改全局变量：**

**global abc = 123**

**Python预定义函数：**

**help(\*)，dir(\*)，globals()，locals()，vars()**

**id(\*)**

**setattr，getattr，hasattr，delattr，vars(\*)**

**issubclass(\*)，isinstance(\*)，**

**callable(\*)**

**staticmethod，classmethod，super，property**

**eval(\*)，exec(\*)，compile(\*)**

**input()**

**divmod()，pow()，round()，abs()**

**format，repr**

**exit()，quit()**

**oct(\*)，hex(\*)，bin(\*)**

**ord("A”)，chr(52)**

**open(\*)**

**len(\*)**

**hash(\*)**

**iter，all，any，sum，sorted，next，min，max**

**Python预定义类型：**

**object**

**type**

**set，list，dict，tuple，frozenset**

**int，float，bool，complex，**

**str**

**enumerate，slice，reversed，map，filter，zip，range**

**bytearray，bytes**

**Python预定义异常类:**

**BaseException**

**GeneratorExit**

**SystemExit**

**KeyboardInterrupt**

**Exception**

**ArithmeticError**

**FloatingPointError**

**OverflowError**

**ZeroDivisionError**

**AssertionError**

**AttributeError**

**BufferError**

**OSError(IOError,EnvironmentError)**

**BlockingIOError**

**ConnectionError**

**BrokenPipeError**

**ConnectionAbortedError**

**ConnectionRefusedError**

**ConnectionResetError**

**ChildProcessError**

**FileExistsError**

**FileNotFoundError**

**InterruptedError**

**IsADirectoryError**

**NotADirectoryError**

**PermissionError**

**ProcessLookupError**

**TimeoutError**

**Warning**

**BytesWarning**

**DeprecationWarning**

**FutureWarning**

**ImportWarning**

**PendingDeprecationWarning**

**ResourceWarning**

**RuntimeWarning**

**SyntaxWarning**

**UnicodeWarning**

**UserWarning**

**EOFError**

**ImportError**

**SyntaxError**

**IndentationError**

**TabError**

**LookupError**

**IndexError**

**KeyError**

**MemoryError**

**NameError**

**UnboundLocalError**

**RuntimeError**

**NotImplementedError**

**ReferenceError**

**StopIteration**

**SystemError**

**TypeError**

**ValueError**

**UnicodeError**

**UnicodeDecodeError**

**UnicodeEncodeError**

**UnicodeTranslateError**

**Python中的对象协议：是指Python对象为了具有某种能力而必须实现的一个或一组方法。**

**类型转换协议：**

**\_\_str\_\_(), \_\_repr\_\_(), \_\_int\_\_(), \_\_long\_\_(), \_\_float\_\_()**

**比较大小协议：**

**\_\_cmp\_\_()**

**数值运算符协议：反运算**

**\_\_add\_\_(), \_\_sub\_\_(), \_\_mul\_\_(), \_\_div\_\_(), \_\_floordiv\_\_(), \_\_truediv\_\_(), \_\_pow\_\_(), \_\_mod\_\_(), \_\_moddiv\_\_()**

**\_\_radd\_\_(), \_\_rsub\_\_(), \_\_rmul\_\_(), \_\_rdiv\_\_()**

**逻辑运算符协议：**

**\_\_eq\_\_(), \_\_ne\_\_(), \_\_lt\_\_(), \_\_ge\_\_**

**容器类型协议：**

**\_\_len\_\_(), \_\_getitem\_\_(), \_\_setitem\_\_(), \_\_delitem\_\_(), \_\_iter\_\_(), \_\_reversed\_\_(), \_\_contains\_\_()**

**可调用对象协议：**

**\_\_call\_\_()**

**可哈希对象协议：hash**

**\_\_hash\_\_()**

**属性交互协议：**

**\_\_getattr\_\_(), \_\_setattr\_\_(), \_\_delattr\_\_()**

**上下文管理协议：with**

**\_\_enter\_\_(), \_\_exit\_\_()**

**管道运算符协议：**

**\_\_ror\_\_()**

**file-like:**

**mode**

**encoding**

**readable, read, readline, readlines**

**writeable, write, writelines**

**flush**

**close**

**生成器，迭代器与可迭代对象：**

**Iterable：包含有\_\_getitem\_\_()或者\_\_iter\_\_()方法的对象。**

**Iterator：包含有\_\_next\_\_()方法和\_\_iter\_\_()方法的对象。**

**Generator：包含有yield语句的函数，是一种特殊的迭代器（Iterator）也可以称为可迭代对象（Iterable）。**

**Python中的str和repr：**

**str：打印，**

**repr：回显，展现细节类似代码**

**Python集合操作：**

**iter/items/enumerate/zip**

**filter/map/sorted/reversed/sum/min/max/all/any/next**

**Python逻辑系统**

**基本运算符：**

**+，-，\*，/，%，\*\*，//**

**and，or，not**

**in，not in**

**==，!=（相等性测试）**

**is，is not（同一性测试）**

**>，<，>=，<=**

**True，False，None**

**条件－假：**

**在python中，任何对象都可以判断其真假值：True，False。**

**在if或while条件判断中，下面的情况值为False：**

**1. None**

**2. Flase**

**3. 数值为0的情况，如：0, 0.0, 0j**

**4. 所有空序列，如：'', (), []**

**5. 所有空mapping，如：{}**

**6. Instances of user-defined classes, if the class defines a \_\_bool\_\_() or \_\_len\_\_() method, when that method returns the integer zero or bool value False.**

**条件－假与\_\_nonzero\_\_, \_\_bool\_\_, \_\_len\_\_：**

**if a: 的执行结果首先取决于a对象是否为None，0，False；否则则取决于a对象所属的类有没有定义\_\_nonzero\_\_(在Python3中被\_\_bool\_\_替换）及其返回值是否为0/1或True/False；然后才是取决于a对象所属的类有没有定义\_\_len\_\_及其返回值是否为0/1或True/False；最后则认为是True。**

**逻辑结构：**

**单值赋值/多值赋值/高级赋值(解构等**

**or取值（a = b or c）**

**;     分隔单行中的多条语句**

**if:/elif:/else:**

**<var> = <value-for-true> if <condition> else <value-for-false>**

**<method-for-true> if <condition> else <method-for-false>**

**for/in: continue/pass/break—>else:**

**while: continue/pass/break—>else:**

**try:/except as:/else:/finally: —>raise，assert**

**with/as:**

**yield**

**del**

**pass**

**return/return if**

**逻辑单元：**

**函数：**

**def funcName():**

**return 1**

**返回多值－返回多个值会被封装成Tuple：**

**return 1, 2, 3**

**匿名函数：**

**lambda [arg1 [,arg2,.....argn]]:expression**

**Class：**

**Python中Class类似于模版定义了其所有实例都会有的属性和方法，但每个实例还可以自行扩展自己即为自己添加属性或方法(为实例添加实例方法有些复杂)；这也体现了Python不同于Java等静态语言的动态语言特性。**

**用法：**

**class Student(object):**

**pass**

**zan ＝ Student();**

**zan.Name = “Davis Zan”**

**zan.Age = 35**

**类变量：**

**定义：**

**class Student(object):**

**name = “Student”**

**使用：可以通过类名或实例访问。**

**Student.name**

**zan = Student()**

**zan.name**

**注意：通过实例访问时，实例的本地同名属性会覆盖类上的同名属性**

**zan.name = “davis”**

**zan.name**

**#删除实例中的同名属性后才可以再次通过实例访问类上的属性**

**del zan.name**

**zan.name**

**实例变量：**

**定义：**

**class Student(object):**

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

**self.name = name**

**使用：只能通过实例来访问。**

**zan ＝ Student(“davis”)**

**zan.name**

**为实例动态添加实例方法：**

**class Student(object):**

**pass**

**#step 1**

**zan ＝ Student()**

**zan.nam = “Davis”**

**#step 2**

**def set\_age(self, age):**

**self.age = age**

**#step 3**

**from types import MethodType**

**zan.set\_age = MethodType(set\_age)**

**#step 4**

**zan.set\_age(35)**

**zan.age**

**为类动态添加实例方法：**

**class Student(object):**

**pass**

**#step 1**

**def set\_score(self, score):**

**self.score = score**

**#step 2**

**from types import MethodType**

**Student.set\_score = MethodType(set\_score, Student)**

**@property：**

**class Student(object):**

**@property**

**def score(self):**

**return self.\_score**

**@score.setter**

**def score(self, value):**

**if not isinstance(value, int):**

**raise ValueError('score must be an integer!’)**

**if value < 0 or value > 100:**

**raise ValueError('score must between 0 ~ 100!’)**

**self.\_score = value**

**使用\_\_xx\_\_定制类：**

**\_\_slots\_\_：规定类实例所能定义的变量**

**\_\_str\_\_：定制Print的显示效果**

**\_\_repr\_\_：定制交互模式下回显时的显示效果**

**\_\_len\_\_：被len方法调用**

**\_\_iter\_\_：返回迭代器对象**

**\_\_next\_\_：迭代器**

**\_\_getitem\_\_：使实例可以按下标取值**

**\_\_setitem\_\_：使实例可以按下标赋值**

**\_\_getattr\_\_：访问的实例属性不存在时该方法被调用**

**\_\_call\_\_：使该类的实例为Callable，可以直接调用。**

**Python元编程：**

**运行时创建Class：**

**＃创建**

**def fn(self, name='world'):**

**print('Hello, %s.' % name)**

**Hello = type('Hello', (object,), dict(hello=fn))**

**＃使用**

**hi = Hello()**

**hi.hello('Davis’)**

**metaclass：**

**＃定义**

**class ListMetaclass(type):**

**def \_\_new\_\_(cls, name, bases, attrs):**

**print('name: {}, bases: {}, attrs: {}'.format(name, bases, attrs))**

**attrs['add\_the'] = lambda self, value: self.append(value)**

**return type.\_\_new\_\_(cls, name, bases, attrs)**

**class MyList(list, metaclass=ListMetaclass):**

**def my\_func():**

**pass**

**＃使用**

**lst = MyList()**

**lst.add\_the(1)**

**高级特性：**

**切片：[start:stop:step]**

**lst[0:3]    lst[:3]    lst[-2:]  lst[:10:2]  lst[::5]**

**命名切片：**

**slice\_reverse = slice(None, None, -1)**

**'abcdefghi'[slice\_reverse]**

**列表切片赋值：**

**a = [1, 2, 3, 4, 5]**

**a[1:-1] = []**

**解析：**

**a, b, c = 1, 2, 3**

**a, b, c = [1, 2, 3]**

**a, (b, c), d = [1, (2, 3), 4]**

**a, b, c = (2 \* i + 1 for i in range(3))**

**a, \*b, c = [1, 2, 3, 4, 5]**

**迭代：**

**for key in dct:**

**for value in dct.values:**

**for key, value in dct.items:**

**for i, value in enumerate(lst):**

**负索引：**

**‘123456789’[-1]**

**‘123456789’[-3:]**

**生成器：**

**g = [x\*x for x in range(10)]**

**带yield的generator function。**

**迭代器：**

**from collections import iterable, iterator**

**装饰器：**

**定义：**

**例子一：**

**from functools import wraps**

**def log(func):**

**@wraps(func)**

**def wrapper(\*args, \*\*kwargs):**

**print(“Called before {}…”.format(func.\_\_name\_\_))**

**return func(\*args,\*\*kwargs)**

**return wrapper**

**例子二：**

**class EndpointFeatureToggle:**

**def \_\_init\_\_(self, switch):**

**self.\_switch = switch**

**def \_\_call\_\_(self, func):**

**@wraps(func)**

**def \_\_inner\_func(\*args, \*\*kwargs):**

**switch\_on = FeatureToggle.is\_open(self.\_switch)**

**if switch\_on:**

**return func(\*args, \*\*kwargs)**

**else:**

**abort(404)**

**return \_\_inner\_func**

**例子三：**

**使用：**

**@log**

**def now():**

**print(“now is 16:40”)**

**常用：**

**@abstractmethod**

**@classmethod**

**@staticmethod**

**@property**

**@unique**

**偏函数：**

**Example #1:**

**import functools**

**assert 255 == int("ff", base=16)**

**int16 = functools.partial(int, base = 16)**

**assert 255 == int16("ff”)**

**Example #2**

**import functools**

**def mul(a, b):**

**reurn a \* b**

**abc = functools.partial(mul, 12, 12)**

**abc()**

**列表，元组，集合，字典解析：**

**[expr for iter\_item in iterable if cond\_expr]**

**(expr for iter\_item in iterable if cond\_expr)**

**{expr for iter\_item in iterable if cond\_expr}**

**{expr1: expr2 for iter\_item in iterable if cond\_expr}**

**变长参数：**

**使用\*args来实现可变参数列表（args类型为tuple）**

**定义：**

**def func(\*args):**

**for i, value in enumerate(args):**

**print(‘{0} - {1}’.format(i, value))**

**使用：**

**func(2, 4)**

**func(2,3,4,5)**

**使用－数组转可变参数列表：**

**params = [2,3,4,5]**

**func(\*params)**

**使用\*\*kwargs接受字典形式的关键字参数列表（kwargs类型为dict）**

**定义：**

**def func(\*\*kwargs):**

**for name, value in kwargs.items():**

**print(‘{0} - {1}’.format(name, value))**

**使用：**

**func(apple=‘fruit’, corrot=‘vegetable’)**

**func(BMW=‘car’)**

**使用－字典转关键字参数列表：**

**params = {'apple’: ‘fruit’, ‘corrot’: ‘vegetable’}**

**func(\*\*params)**

**Descriptors：**

**Descriptors are objects with any of \_\_get\_\_, \_\_set\_\_, or \_\_delete\_\_. These descriptor objects can be used as attributes on other object class definitions.**

**Descriptor objects can be used to programmatically manage the results of a dotted lookup (e.g. foo.descriptor) in a normal expression, an assignment, and even a deletion. Bound methods, property, classmethod, and staticmethod all use these special methods to manage how they are accessed via the dotted lookup.**

**For an owner-object, obj\_instance, with a descriptor object:**

**descriptor.\_\_get\_\_(self, obj\_instance, owner\_class) (returning a value)**

**is invoked by**

**obj\_instance.descriptor**

**descriptor.\_\_set\_\_(self, obj\_instance, value) (returning None)**

**is invoked by**

**obj\_instance.descriptor = value**

**descriptor.\_\_delete\_\_(self, obj\_instance) (returning None)**

**is invoked by**

**del obj\_instance.descriptor**

**A Data Descriptor has a \_\_set\_\_ and/or \_\_delete\_\_.**

**A Non-Data-Descriptor has neither \_\_set\_\_ nor \_\_delete\_\_.**

**Coroutines:**

**Coroutines used with asyncio may be implemented using the async def statement, or by using generators.**

**The async def type of coroutine was added in Python 3.5, and is recommended if there is no need to support older Python versions.**

**Generator-based coroutines should be decorated with @asyncio.coroutine, although this is not strictly enforced. The decorator enables compatibility with async def coroutines, and also serves as documentation. Generator-based coroutines use the yield from syntax introduced in PEP 380, instead of the original yield syntax.**

**The word “coroutine”, like the word “generator”, is used for two different (though related) concepts:**

**The function that defines a coroutine (a function definition using async def or decorated with @asyncio.coroutine). If disambiguation is needed we will call this a coroutine function (iscoroutinefunction() returns True).**

**The object obtained by calling a coroutine function. This object represents a computation or an I/O operation (usually a combination) that will complete eventually. If disambiguation is needed we will call it a coroutine object (iscoroutine() returns True).**

**Things a coroutine can do:**

**result = await future or result = yield from future – suspends the coroutine until the future is done, then returns the future’s result, or raises an exception, which will be propagated. (If the future is cancelled, it will raise a CancelledError exception.) Note that tasks are futures, and everything said about futures also applies to tasks.**

**result = await coroutine or result = yield from coroutine – wait for another coroutine to produce a result (or raise an exception, which will be propagated). The coroutine expression must be a call to another coroutine.**

**return expression – produce a result to the coroutine that is waiting for this one using await or yield from.**

**raise exception – raise an exception in the coroutine that is waiting for this one using await or yield from.**

**yield && yield from：**

**Example 1:**

**def my\_generator():**

**for i in range(2000):**

**yield i**

**for element in my\_generator():**

**print(element)**

**Example 2:**

**def my\_generator():**

**for i in range(10):**

**yield i**

**for j in range(20,30):**

**yield j**

**Example 3:**

**def generator1():**

**for i in range(10):**

**yield i**

**def generator2():**

**for j in range(10, 20):**

**yield j**

**def my\_generator():**

**for i in generator1():**

**yield i**

**for j in generator2():**

**yield j**

**def generator():**

**yield from generator1()**

**yield from generator2()**

**Example 4:**

**def generator():**

**for i in range(5):**

**input\_value = yield i**

**print("input: ", input\_value)**

**gen = generator()**

**type(gen)**

**print(next(gen))**

**print(gen.send(5))**

**print(gen.send(4))**

**print(next(gen))**

**print(next(gen))**

**Python调试：**

**python3 -m pdb \*.py**

**Python跟踪：**

**python3 -m profile \*.py**

**Python编译：**

**python3 -m compileall <path-of-py>**

**Python变量作用域：**

**global**

**nonlocal**

**作用域解析：**

**Python的作用域解析是基于叫做**[**LEGB**](https://blog.mozilla.org/webdev/2011/01/31/python-scoping-understanding-legb/)**（Local（本地），Enclosing（封闭），Global（全局），Built-in（内置））的规则进行操作的。**

**方法解析顺序（Method Resolution Order (MRO)：**

**MRO是python用来解析方法调用顺序的；MRO对于多重继承中方法调用异常重要。python中有一个内建函数和MRO密切相关——super。**

**Finally 陷阱：**

**在finally中产生新的异常或者执行了return或break语句，try中产生的原始异常会被屏蔽。不要在finally中使用return语句。**

**字符串处理：**

**当进行大量字符串拼接时优先使用join而不是＋以避免产生大量中间数据**

**格式化字符串时尽量使用.format而不是％**

**’%s=%s’ % (‘one’, ‘1’)**

**‘{0}={1}’.format(’one', ‘1')**

**‘{name}={code}’.format(name='one', code=’1’)**

**可变对象／不可变对象：**

**数字，字符串，元组属于不可变对象**

**字典，列表，字节数组属于可变对象**

**Python函数参数是传值还是传引用：**

**正确的叫法是传对象，函数参数在传递的过程中将整个对象传入，对可变对象的修改在函数内部及外部都可见，对不可变对象会在修改时生产新对象然后赋值。**

**默认参数的问题：**

**Def在Python中是一个可执行的语句，当解释器执行def时，默认参数会被计算并存储在函数的.fun\_defaults属性中。**

**Python常用模块：**

**builtins，enum，abc，keyword，operator，sys，os，types，imp，importlib，datetime，dateutil，time，uuid，random，math，string，pipe，pickle，marshal，json，re，zipfile，collections，numbers，logging，pdb，compileall，traceback，profile，inspect，unittest，doctest，io，inspect，ast，copy，shutil，tarfile，abc，threading，multiprocessing，redis，Queue，message，state，base64，struct，hmac，hashlib，itertools，blinker，urllib3，urllib3-mock，PIL，socket，select，email，smtplib，poplib，subprocess，commands，sqllite3，invoke，flask，asyncio，aiohttp，pykka，eventlet，aiomysql，argparse，docopt，requests，Pandas，ConfigParser，ElementTree，PyYAML，pypinyin，PyMongo，scrapy，openpyxl**

**常用模块：**

**sys：**

**import sys**

**#parameters**

**sys.argv[]**

**#module && path**

**sys.modules**

**sys.path**

**#sys info**

**sys.platform**

**sys.version**

**#shell**

**sys.stderr**

**sys.stdin**

**sys.stdout**

**#process**

**sys.exc\_info()**

**sys.exit(0)**

**#encoding**

**sys.getdefaultencoding()**

**sys.getdefaultencoding()**

**re：**

**import re**

**re.sub('[\d].', '', '5. who are you?')**

**matcher = re.match('[\d].', '5. who are you?')**

**if matcher:**

**matcher.group()**

**types：**

**import types**

**imp：**

**import imp**

**file, pathname, desc = imp.find\_module('os')**

**myos = imp.load\_module('myos', file, pathname, desc)**

**imp.reload(sys)**

**importlib：**

**import importlib**

**module = importlib.import\_module(’name-of-module')**

**uuid：**

**str(uuid.uuid1())**

**str(uuid.uuid4())**

**operator：**

**……..**

**pickle：**

**import pickle**

**bts = pickle.dumps(obj); obj = pickle.loads(bts);**

**pickle.dump(obj, f); obj = pickle.load(f);**

**json：**

**import json**

**strObj = json.dumps(dictObj); dictObj = json.loads(strObj)**

**json.dump(dictObj, f); dictObj = json.load(f)**

**ast：表达式求值**

**import ast**

**mylist = ast.literal\_eval('[1,2,3]’)**

**pdb：**

**手动调试：**

**python3 -m pdb <python-script.py> [args]**

**#启动后程序会进入调试控制台，等待用户发送调试命令，help可以查看所有调试命令列表以及使用方式。**

**b可以设置断点，c可以执行程序，Ctol+c可以暂停程序的执行并等待用户指令，此时where可以打印出当前程序停留处的堆栈信息。**

**代码接入：**

**import pdb**

**pdb.set\_trace()#运行到这行程序会暂停**

**pyinstrument:**

**pip install pyinstrument**

**#显示方法调用时间统计**

**pyinstrument --show-all -t <python-script.py> [args]**

**#测试脚本**

**import time**

**def func\_a():**

**print("in function a")**

**def func\_b():**

**print("in function b")**

**def func\_c():**

**print("in function c")**

**def func\_d():**

**print("in function d")**

**counter = 0**

**while True:**

**func\_e(counter)**

**counter += 1**

**def func\_e(counter):**

**print("in function e " + str(counter))**

**time.sleep(.2)**

**def main():**

**func\_a()**

**func\_b()**

**func\_c()**

**func\_d()**

**func\_e**

**main()**

**compileall：**

**import compileall**

**compileall.compile\_dir('examples')**

**traceback：**

**import traceback**

**try:**

**………...**

**except IndexError as ex:**

**traceback.print\_exc()**

**profile：**

**import profile**

**profile.run(’statement’)**

**profile.runctx(’statement’, <globals>, <locals>)**

**subprocess：**

**描述:**

**This module allows you to spawn processes, connect to their input/output/error pipes, and obtain their return codes.**

**例子：**

**import subprocess**

**subprocess.call([‘ls’, ‘-al’, ‘/’ ])**

**subprocess.getoutput(‘ls -al /')**

**commands:**

**描述：**

**Execute shell commands via os.popen() and return status, output.**

**例子：**

**import commands;**

**outtext = commands.getoutput("ls .|grep .gz")**

**(exitstatus, outtext) = commands.getstatusoutput("ls .|grep .gz")**

**inspect：**

**import inspect**

**inspect.stack()**

**inspect.trace()**

**inspect.isgenerator(res)**

**inspect.isgeneratorfunction(func)**

**if inspect.ismethod(func):**

**job.\_instance = func.\_\_self\_\_**

**job.\_func\_name = func.\_\_name\_\_**

**if inspect.isfunction(func) or inspect.isbuiltin(func):**

**job.\_func\_name = '%s.%s' % (func.\_\_module\_\_, func.\_\_name\_\_)**

**eventlet：**

**import eventlet**

**from eventlet.green import urllib**

**def fetch(url):**

**return urllib.urlopen(url).read()**

**urls = [**

**"http://www.google.com/intl/en\_ALL/images/logo.gif",**

**"https://wiki.secondlife.com/w/images/secondlife.jpg",**

**"http://us.i1.yimg.com/us.yimg.com/i/ww/beta/y3.gif",**

**]**

**pool = eventlet.GreenPool()**

**for body in pool.imap(fetch, urls):**

**print("got body", len(body))**

**socket && select:**

**#example 1**

**import socket**

**s = socket.socket()**

**s.connect(('www.google.com', 80))**

**#example 2**

**import socket**

**sockets = {}**

**for i in range(100):**

**s = socket.socket()**

**sockets[s.fileno()] = s**

**s.setblocking(0)**

**try:**

**s.connect(('www.google.com', 80))**

**except:**

**pass**

**import select**

**while sockets:**

**fds = select.select([], list(sockets.keys()), [])[1]**

**for fd in fds:**

**s = sockets.pop(fd)**

**print("%d connected to %s:%d" % ((fd,) + s.getpeername()))**

**异步切换：**

**显式异步切换：Twisted**

**@defer.inlineCallbacks**

**def main(endpoint, username="alice", password=“secret”):**

**endpoint = endpoints.clientFromString(reactor, strport)**

**factory = protocol.Factory()**

**factory.protocol = imap4.IMAP4Client**

**try:**

**client = yield endpoint.connect(factory)**

**yield client.login(username, password)**

**yield client.select('INBOX')**

**info = yield client.fetchEnvelope(imap4.MessageSet(1))**

**print 'First message subject:', info[1]['ENVELOPE'][1]**

**except:**

**print "IMAP4 client interaction failed"**

**failure.Failure().printTraceback()**

**task.react(main, sys.argv[1:])**

**隐式异步切换：**

**import socket**

**from gevent import monkey**

**monkey.patch\_all()**

**s = socket.socket()**

**s.connect(('www.google.com', 80))**

**print("We are connected to %s:%d" % s.getpeername())**

**Gevent 是隐式的异步切换的代表。通过所谓的 monkey patch，Gevent 把系统库里的 socket 等模块，替换成了 Gevent 自己提供的相应的非阻塞模块。这样，上面的代码就变成（底层）异步的了。考虑到 monkey patch 的侵入性，您也可以考虑直接使用 Gevent 提供的模块，比如这样：from gevent import socket**

**urllib:**

**import urllib**

**#url encode**

**urllib.parse.quote\_plus(<value>)**

**#url decode**

**urllib.parse.unquote\_plus(<value>)**

**#url join**

**urllib.parse.urljoin(url\_root, url)**

**#url query split**

**base\_url, query\_args = urllib.parse.splitquery('http://zanxiaofeng.com/index?lang=zh-cn')**

**assert base\_url == 'http://zanxiaofeng.com/index'**

**assert query\_args == 'lang=zh-cn'**

**#url split**

**url\_parts = urllib.parse.urlsplit('http://zanxiaofeng.com/index?lang=zh-cn')**

**assert url\_parts.scheme == 'http'**

**assert url\_parts.netloc == 'zanxiaofeng.com'**

**assert url\_parts.path ==  '/index'**

**assert url\_parts.query == 'lang=zh-cn’**

**#url encode**

**assert urllib.parse.urlencode({'lang': 'zh cn'}) == 'lang=zh+cn'**

**pipe：**

**from pipe import \***

**[1, 2, 3, 4, 5] | where(lambda x: x % 2) | tail(2) | select(lambda x: x \* x) | add**

**redis：**

**from redis import StrictRedis**

**from datetime import timedelta**

**redis\_cli = StrictRedis(“127.0.0.1”, “6379”)**

**redis\_cli.set(“key:test”, “test”, ex=timedelta(minutes=5))**

**redis\_cli.set(“key:test”, “test”)**

**redis\_cli.get(“key:test”)**

**redis\_cli.delete(“key:test”)**

**logging：**

**import logging**

**logging.basicConfig()**

**logging.basicConfig(level=logging.DEBUG)**

**logging.basicConfig(………)**

**logging.info(……)**

**logging.debug(……)**

**logging.error(……)**

**logging.critical(……)**

**blinker：**

**send\_data = signal('send-data’)**

**@send\_data.connect**

**def receive\_data(sender, \*\*kw):**

**print("Caught signal from %r, data %r" % (sender, kw))**

**return 'received!’**

**result = send\_data.send('anonymous', abc=123)**

**getopt:**

**import getopt**

**try:**

**options,args = getopt.getopt(sys.argv[1:],"hp:i:",["help","ip=","port=“])**

**except getopt.GetoptError:**

**sys.exit()**

**for name,value in options:**

**if name in ("-h","--help”):**

**usage()**

**if name in ("-i","--ip”):**

**print 'ip is----‘,value**

**if name in ("-p","--port")**

**print 'port is----‘,value**

**openpyxl：**

**arguparse:**

**import argparse**

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

**parser = argparse.ArgumentParser()**

**parser.add\_argument('-o', '--output')**

**parser.add\_argument('-v', dest='verbose', action='store\_true')**

**args = parser.parse\_args()**

**# ... do something with args.output ...**

**# ... do something with args.verbose ...**

**copy：**

**import copy**

**y = copy.copy(x)**

**y = copy.deepcopy(x)**

**datetime：**

**from datetime import datetime, timedelta**

**t = datetime.now() + timedelta(minutes=1)**

**t = datatime(2016, 1, 1, 9, 0 ,0)**

**t = datetime.utcnow()**

**DATETIME\_DISPLAY\_FORMAT = '%Y-%m-%d %H:%M'**

**DATE\_DISPLAY\_FORMAT = '%Y-%m-%d'**

**DATETIME\_DISPLAY\_FORMAT\_NEW = '%Y.%m.%d %H:%M'**

**DATETIME\_FORMAT = '%Y-%m-%dT%H:%M:%SZ'**

**t.strftime(DATETIME\_FORMAT)**

**dateutil：**

**from dateutil import tz**

**from datetime import datetime**

**datetime.now().replace(tzinfo=tz.tzlocal()).astimezone(tz.tzutc())**

**datetime.now().replace(hour=0, minute=0, second=0, microsecond=0, tzinfo=tz.tzlocal()).astimezone(tz.tzutc())**

**time：**

**import time**

**//secs**

**time.sleep(60)**

**threading：**

**Create Thread:**

**import threading**

**th1 = threading.Thread(target=myFunc, args=(1, 2, 3))**

**th1.setDaemon(True)**

**th1.start()**

**th1.join()**

**Current Thread:**

**threading.currentThread().name**

**threading.currentThread().ident**

**Thread Local:**

**local = threading.local()**

**local.num = 12**

**Thread Lock：**

**#创建**

**mutex = threading.Lock()**

**#锁定**

**mutex.acquire([timeout])**

**#释放**

**mutex.release()**

**Queue：**

**rq\_scheduler：**

**import sys**

**import logging**

**from redis import Redis**

**from rq\_scheduler import Scheduler**

**from datetime import datetime, timedelta**

**logging.basicConfig(level=logging.DEBUG)**

**#usage: python scheduler.py**

**redis = Redis(host='localhost', port='6379')**

**scheduler = Scheduler(queue\_name='my-jobs', connection=redis, interval=10)**

**for job in scheduler.get\_jobs():**

**scheduler.cancel(job)**

**t1 = datetime.utcnow() + timedelta(seconds=120)**

**scheduler.enqueue\_at(t1, print, "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")**

**td = timedelta(seconds=120)**

**scheduler.enqueue\_in(td, print, "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")**

**scheduler.schedule(**

**datetime.utcnow(),**

**print,**

**interval=30,**

**args=["\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"]**

**)**

**scheduler.run()**

**rq\_worker:**

**import os**

**import redis**

**import logging**

**from rq import use\_connection**

**from rq.scripts import rqworker**

**logging.basicConfig(level=logging.DEBUG)**

**#usage: python worker.py my-jobs**

**def setup\_redis(args):**

**redis\_conn = redis.StrictRedis(host='localhost', port='6379')**

**use\_connection(redis\_conn)**

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

**rqworker.setup\_redis = setup\_redis**

**rqworker.main()**

**message：**

**import message**

**message.sub(…)**

**message.unsub(…)**

**message.pub(…)**

**message.declare(…)**

**message.retract(...)**

**@observable**

**state：**

**from state import cure, switch, stateful, State, behavior**

**@stateful**

**class People(object):**

**class Workday(State):**

**default = True**

**@behavior**

**def day(self):**

**print(‘work hard’)**

**class Weekend(State):**

**@behavior**

**def day(self):**

**print(‘play harder!’)**

**zan = People()**

**for i in xrange(1, 8)**

**if i == 6:**

**switch(zan, People.Weekend)**

**if i == 1:**

**switch(zan, People.Workday)**

**zan.day()**

**collections：**

**#频次统计器**

**from collections import Counter**

**#命名元组**

**from collections import namedtuple**

**Point = namedtuple('Point', ['x', 'y'])**

**p1 = Point(1, 3)**

**p2 = Point(x=2, y=4)**

**assert p1.x == 1**

**assert p2.y == 4**

**argparse：docopt**

**import argparse**

**parser ＝ argparse.ArgumentParser()**

**parser.add\_argument(‘-o’, ‘—output’)**

**parser.add\_argument(‘-v’, dest=‘verbose’, action=’store\_true’)**

**args = parser.parse\_args()**

**Pandas：Python Data Analysis Library**

**pd.read\_csv()**

**pd.to\_csv()**

**ConfigParser：**

**from ConfigParser import ConfigParser**

**conf = ConfigParser()**

**conf.read(‘a.conf’)**

**conf.get(’section’, ‘key’)**

**ElementTree：**

**XML Processing**

**PyYAML：**

**import yaml**

**config = yaml.load(open(‘path/to/xxx.yml’))**

**yaml.dump(…...)**

**yaml.load\_all(……)**

**yaml.dump\_all(……)**

**pypinyin：**

**from pypinyin import pinyin, lazy\_pinyin**

**import pypinyin**

**pinyin(u'中心')**

**lazy\_pinyin(u'中心')  # 不考虑多音字的情况**

**pinyin(u'中心', heteronym=True)  # 启用多音字模式**

**pinyin(u'中心', style=pypinyin.FIRST\_LETTER) # 设置拼音风格**

**unittest：**

**#test cases**

**import unittest**

**class TestXXX(unittest.TestCase):**

**def setUp(self):**

**print('setUp…')**

**def tearDown(self):**

**print('tearDown…')**

**def test\_XXX(self):**

**…..**

**self.assertEqual(xxx, xx)**

**#mock object**

**#patch methed of object**

**from unittest.mock import patch**

**with patch.object(<obj\_to\_mock>, “<mothed\_to\_mock>", return\_value=<value>) as mocked\_obj:**

**………….**

**assert mocked\_obj.called == True**

**assert mocked\_obj.call\_count == 1**

**mocked\_obj.call\_args**

**mocked\_obj.call\_args\_list**

**with patch.object(<obj\_to\_mock>, “<mothed\_to\_mock>", side\_effect=<function\_instead\_of>) as mocked\_obj:**

**………….**

**assert mocked\_obj.called == True**

**assert mocked\_obj.call\_count == 1**

**mocked\_obj.call\_args**

**mocked\_obj.call\_args\_list**

**#mock object**

**import json**

**from unittest.mock import Mock**

**mocked\_response = Mock()**

**mocked\_response.status\_code = 200**

**mocked\_response.text = json.dumps({'rtnCode': '0000000', 'bizData': 'result' })**

**base64：**

**import base64**

**str = base64.encodebytes(b'username’)**

**hmac：**

**import hmac**

**hashed = hmac.new(<key>, <msg-bytes>, <digestmod>)**

**hashed.hexdigest()**

**hashlib：**

**Usage：**

**Hash object constructors in hashlib：**

**md5, sha1, sha224, sha256, sha384, and sha512**

**Methods in hash objects：**

**update, copy, digest, and hexdigest**

**Example：**

**Info：**

**import hashlib**

**hashlib.algorithms\_available**

**hashlib.algorithms\_guaranteed**

**Message digest：**

**import hashlib**

**hashlib.md5(‘\*\*\*’/bytes).hexdigest()**

**hashlib.sha224(‘\*\*\*’/bytes).hexdigest()**

**Key derivation function：**

**import hashlib, binascii**

**dk = hashlib.pbkdf2\_hmac('sha256', b'password', b'salt', 100000)**

**binascii.hexlify(dk)**

**requests：**

**import requsts**

**requests.get(url, \*\*kwargs)**

**requests.put(url, data=None, \*\*kwargs)**

**requests.post(url, data=None, \*\*kwargs)**

**requests.delete(url, \*\*kwargs)**

**pykka：**

**Three actor types：**

**ThreadingActor**

**GeventActor**

**EventletActor**

**Usage：**

**from pykka import ThreadingActor**

**class Greeter(ThreadingActor):**

**def \_\_init\_\_(self, greeting='Hi '):**

**super(Greeter, self).\_\_init\_\_()**

**self.greeting = greeting**

**def on\_receive(self, message):**

**if message != 'stop’:**

**greeting\_message = self.greeting + message['msg’]**

**print(greeting\_message)**

**return greeting\_message**

**self.stop()**

**actor\_ref = Greeter.start(greeting='Hi, ')**

**actor\_ref.tell({'msg': 'Hi!’})**

**answer = actor\_ref.ask({'msg': 'Hi?'}, timeout=3)**

**future = actor\_ref.ask({'msg': 'Hi?'}, block=False)**

**answer = future.get(timeout=0.1)**

**actor\_ref.tell('stop’)**

**actor\_ref.stop()**

**PyMongo：**

**import pymongo**

**from pymongo import MongoClient**

**＃Making a Connection**

**client = MongoClient('localhost', 27017)**

**client = MongoClient('mongodb://localhost:27017/')**

**＃Getting all Database**

**client.database\_names()**

**＃Getting a Database**

**db = client.test\_database**

**db = client['test-database']**

**＃Get all Collections**

**db.collection\_names(include\_system\_collections=False)**

**＃Getting a Collection**

**posts = db.test\_collection**

**posts = db['test-collection']**

**＃Inserting a Document¶**

**import datetime**

**post = {"author": "Mike",**

**"text": "My first blog post!",**

**"tags": ["mongodb", "python", "pymongo"],**

**"date": datetime.datetime.utcnow()}**

**post\_id = posts.insert\_one(post).inserted\_id**

**post\_id**

**＃Bulk Inserts**

**new\_posts = [{"author": "Mike",**

**"text": "Another post!",**

**"tags": ["bulk", "insert"],**

**"date": datetime.datetime(2009, 11, 12, 11, 14)},**

**{"author": "Eliot",**

**"title": "MongoDB is fun",**

**"text": "and pretty easy too!",**

**"date": datetime.datetime(2009, 11, 10, 10, 45)}]**

**result = posts.insert\_many(new\_posts)**

**result.inserted\_ids**

**＃Getting a Single Document**

**posts.find\_one({"\_id": post\_id})**

**posts.find\_one({"author": "Mike"})**

**＃Querying for More Than One Document¶**

**for post in posts.find():**

**post**

**for post in posts.find({"author": "Mike"}):**

**post**

**＃Counting**

**posts.count()**

**posts.find({"author": "Mike"}).count()**

**＃Index**

**result = posts.create\_index([('author', pymongo.ASCENDING)], unique=True)**

**list(posts.index\_information())**

**OS：**

**系统信息：os**

**os.name;**

**环境变量：os.environ**

**os.environ.get(‘PATH’)**

**文件操作：os, os.path**

**os.path.isdir(\*)**

**os.path.isfile(\*)**

**os.path.abspath(‘.’)**

**os.path.dirname(‘/Users/davis/test’)**

**os.path.join(‘/Users/davis’, ’test’)**

**os.mkdir(‘xxxxxx’)**

**os.rmdir(‘xxxxxx’)**

**os.path.split(‘xxx/xxxx/xxx.xxx’)**

**os.path.splitext(‘xxx/xxx/xxxx.xxx’)**

**os.rename(‘xxx’, ‘xxx’)**

**os.remove(‘xxx’)**

**os.listdir("")**

os.walk(top, topdown=True, onerror=None, followlinks=False)

**进程信息：**

**os.getpid()**

**运行程序：**

**os.system("ls .|grep .gz")**

**os.popen("ls .|grep .gz")**

**StringIO && BytesIO**

**from io import StringIO**

**fw = StringIO(); fw.write('Str’); fw.getvalue();**

**fr = StringIO(’Str’); fr.readline();**

**from io import BytesIO**

**fw = BytesIO(); fw.write(’str’.encode(‘utf-8’)); fw.getvalue();**

**fr = BytesIO(b’\xe4\xb8’); fr.read();**

**Python文件操作：**

**f=open(“hello.txt”，“r”); f.read() / f.read(size) / f.readline() / f.readlines(); close(f)**

**open-flag：**

**r—>read as string, rb—>read as bytes**

**w—>write with string, wb—>write with bytes**

**with open(“hello.txt”, “r”, encoding='gbk') as f:**

**f.read() /f.read(size) / f.readline() / f.readlines();**

**with open(“file\_in.txt”，“r”) as f\_in, open(“file\_out.txt”，“w”) as f\_out:**

**f\_in.read() / f\_out.write(“str”);**

**协程与异步IO：**

**协程**

**由generator实现－－yield不仅可以返回一个值，还可以接受调用者发出的参数。**

**生产者－消费者并发模型**

**异步IO：**

**asyncio与消息循环**

**异步操作需要在coroutine中通过yield from完成；**

**多个coroutine可以封装成一组Task然后并发执行。**

**aiohttp**

**多进程与多线程**

**多进程－－os.fork()，multiprocessing(.Process，.Pool，.Lock，.Pipe，.Queue)**

**By fork**

**import os**

**child\_pid = os.fork()**

**if child\_pid == 0:**

**＃on sub process**

**else:**

**# on main process**

**try:**

**os.waitpid(child\_pid, 0)**

**break**

**except OSError as e:**

**if e.errno != errno.EINTR:**

**raise**

**By process：**

**from multiprocessing import Process**

**processes = []**

**for i in range(5):**

**process = Process(worker\_func, args)**

**processes.append(process)**

**for process in processes:**

**process.join()**

**By pool：**

**from multiprocessing import Pool**

**pool = Pool(processes = 2)**

**for i in xrange(4):**

**pool.apply\_async(worker\_func, args)**

**pool.close()**

**pool.join()**

**分布式进程－－multiprocessing.managers**

**多线程**

**Thread—>threading.Thread(\*, \*)**

**Lock—>threading.Lock()**

**ThreadLocal—>threading.local()**

**WSGI－－Web Server Gateway Interface**

**# hello.py**

**def application(environ, start\_response):**

**start\_response('200 OK', [('Content-Type', 'text/html')])**

**return [b'<h1>Hello, web!</h1>']**

**# server.py**

**# 从wsgiref模块导入:**

**from wsgiref.simple\_server import make\_server**

**# 导入我们自己编写的application函数:**

**from hello import application**

**# 创建一个服务器，IP地址为空，端口是8000，处理函数是application:**

**httpd = make\_server('', 8000, application)**

**print('Serving HTTP on port 8000...')**

**# 开始监听HTTP请求:**

**httpd.serve\_forever()**

**readFille as base64:**

**f = open("./path", 'rb')**

**bContent = f.read()**

**base64Content=base64.encodebytes(bContent)**

**download from js:**

**a = document.createElement('a');**

**a.setAttribute("href", "**data:image/png;base64,**" + base64Content);**

**a.setAttribute("download", "my.png");**

**a.click();**

os.walk:

import os

for dir, dirs, files in os.walk("/usr"):

print("Dir: " + dir);

for d in dirs:

print("\tSub Dir " + d + " of " + dir);

for f in files:

print("\tSub File " + f + " of " + dir);

**Mapping Operators to Functions**

|  |
| --- |
|  |