常用第三方模块

除了内建的模块外，Python还有大量的第三方模块。

基本上，所有的第三方模块都会在PyPI - the Python Package Index上注册，只要找到对应的模块名字，即可用pip安装。

此外，在安装第三方模块一节中，我们强烈推荐安装Anaconda，安装后，数十个常用的第三方模块就已经就绪，不用pip手动安装。

本章介绍常用的第三方模块。

Pillow

PIL：Python Imaging Library，已经是Python平台事实上的图像处理标准库了。PIL功能非常强大，但API却非常简单易用。

由于PIL仅支持到Python 2.7，加上年久失修，于是一群志愿者在PIL的基础上创建了兼容的版本，名字叫Pillow，支持最新Python 3.x，又加入了许多新特性，因此，我们可以直接安装使用Pillow。

安装Pillow

如果安装了Anaconda，Pillow就已经可用了。否则，需要在命令行下通过pip安装：

```
$ pip install pillow
```

如果遇到`Permission denied`安装失败，请加上`sudo`重试。

操作图像

来看看最常见的图像缩放操作，只需三四行代码：

```
from PIL import Image

# 打开一个jpg图像文件，注意是当前路径:
im = Image.open('test.jpg')
# 获得图像尺寸:
w, h = im.size
print('Original image size: %sx%s' % (w, h))
# 缩放到50%:
im.thumbnail((w//2, h//2))
print('Resize image to: %sx%s' % (w//2, h//2))
# 把缩放后的图像用jpeg格式保存:
im.save('thumbnail.jpg', 'jpeg')
```

其他功能如切片、旋转、滤镜、输出文字、调色板等一应俱全。

比如，模糊效果也只需几行代码：

```
from PIL import Image, ImageFilter

# 打开一个jpg图像文件，注意是当前路径:
im = Image.open('test.jpg')
# 应用模糊滤镜:
im2 = im.filter(ImageFilter.BLUR)
im2.save('blur.jpg', 'jpeg')
```

效果如下:

PIL的`ImageDraw`提供了一系列绘图方法，让我们可以直接绘图。比如要生成字母验证码图片:

```
from PIL import Image, ImageDraw, ImageFont, ImageFilter

import random

# 随机字母:
def rndChar():
    return chr(random.randint(65, 90))

# 随机颜色1:
def rndColor():
    return (random.randint(64, 255), random.randint(64, 255), random.randint(64,
255))

# 随机颜色2:
def rndColor2():
    return (random.randint(32, 127), random.randint(32, 127), random.randint(32,
127))

# 240 x 60:
width = 60 * 4
height = 60
image = Image.new('RGB', (width, height), (255, 255, 255))
# 创建Font对象:
font = ImageFont.truetype('Arial.ttf', 36)
# 创建Draw对象:
draw = ImageDraw.Draw(image)
# 填充每个像素:
```

```
for x in range(width):
    for y in range(height):
        draw.point((x, y), fill=rndColor())
# 输出文字:
for t in range(4):
    draw.text((60 * t + 10, 10), rndChar(), font=font, fill=rndColor2())
# 模糊:
image = image.filter(ImageFilter.BLUR)
image.save('code.jpg', 'jpeg')
```

我们用随机颜色填充背景，再画上文字，最后对图像进行模糊，得到验证码图片如下：

如果运行的时候报错：

```
IOError: cannot open resource
```

这是因为PIL无法定位到字体文件的位置，可以根据操作系统提供绝对路径，比如：

```
'/Library/Fonts/Arial.ttf'
```

要详细了解PIL的强大功能，请请参考Pillow官方文档：

https://pillow.readthedocs.org/

小结

- PIL提供了操作图像的强大功能，可以通过简单的代码完成复杂的图像处理。

requests

我们已经讲解了Python内置的urllib模块，用于访问网络资源。但是，它用起来比较麻烦，而且，缺少很多实用的高级功能。

更好的方案是使用requests。它是一个Python第三方库，处理URL资源特别方便。

安装requests

如果安装了Anaconda，requests就已经可用了。否则，需要在命令行下通过pip安装：

```
$ pip install requests
```

如果遇到Permission denied安装失败，请加上sudo重试。

使用requests

要通过GET访问一个页面，只需要几行代码：

```
>>> import requests
>>> r = requests.get('https://www.douban.com/') # 豆瓣首页
>>> r.status_code
200
>>> r.text
r.text
'<!DOCTYPE HTML>\n<html>\n<head>\n<meta name="description" content="提供图书、电
影、音乐唱片的推荐、评论和...'
```

对于带参数的URL，传入一个dict作为`params`参数：

```
>>> r = requests.get('https://www.douban.com/search', params={'q': 'python',
'cat': '1001'})
>>> r.url # 实际请求的URL
'https://www.douban.com/search?q=python&cat=1001'
```

requests自动检测编码，可以使用`encoding`属性查看：

```
>>> r.encoding
'utf-8'
```

无论响应是文本还是二进制内容，我们都可以用`content`属性获得`bytes`对象：

```
>>> r.content
b'<!DOCTYPE html>\n<html>\n<head>\n<meta http-equiv="Content-Type"
content="text/html; charset=utf-8">\n...'
```

requests的方便之处还在于，对于特定类型的响应，例如JSON，可以直接获取：

```
>>> r = requests.get('https://query.yahooapis.com/v1/public/yql?
q=select%20*%20from%20weather.forecast%20where%20woeid%20%3D%202151330&format=js
on')
>>> r.json()
{'query': {'count': 1, 'created': '2017-11-17T07:14:12Z', ...
```

需要传入HTTP Header时，我们传入一个dict作为`headers`参数：

```
>>> r = requests.get('https://www.douban.com/', headers={'User-Agent':
'Mozilla/5.0 (iPhone; CPU iPhone OS 11_0 like Mac OS X) AppleWebKit'})
>>> r.text
'<!DOCTYPE html>\n<html>\n<head>\n<meta charset="UTF-8">\n <title>豆瓣(手机版)
</title>...'
```

要发送POST请求，只需要把`get()`方法变成`post()`，然后传入`data`参数作为POST请求的数据：

```
>>> r = requests.post('https://accounts.douban.com/login', data={'form_email':
'abc@example.com', 'form_password': '123456'})
```

requests默认使用`application/x-www-form-urlencoded`对POST数据编码。如果要传递JSON数据，可以直接传入json参数：

```
params = {'key': 'value'}
r = requests.post(url, json=params) # 内部自动序列化为JSON
```

类似的，上传文件需要更复杂的编码格式，但是requests把它简化成`files`参数：

```
>>> upload_files = {'file': open('report.xls', 'rb')}
>>> r = requests.post(url, files=upload_files)
```

在读取文件时，注意务必使用`'rb'`即二进制模式读取，这样获取的`bytes`长度才是文件的长度。

把`post()`方法替换为`put()`，`delete()`等，就可以以PUT或DELETE方式请求资源。

除了能轻松获取响应内容外，requests对获取HTTP响应的其他信息也非常简单。例如，获取响应头：

```
>>> r.headers
{Content-Type': 'text/html; charset=utf-8', 'Transfer-Encoding': 'chunked',
'Content-Encoding': 'gzip', ...}
>>> r.headers['Content-Type']
'text/html; charset=utf-8'
```

requests对Cookie做了特殊处理，使得我们不必解析Cookie就可以轻松获取指定的Cookie：

```
>>> r.cookies['ts']
'example_cookie_12345'
```

要在请求中传入Cookie，只需准备一个dict传入`cookies`参数：

```
>>> cs = {'token': '12345', 'status': 'working'}
>>> r = requests.get(url, cookies=cs)
```

最后，要指定超时，传入以秒为单位的timeout参数：

```
>>> r = requests.get(url, timeout=2.5) # 2.5秒后超时
```

小结

用requests获取URL资源，就是这么简单！

chardet

字符串编码一直是令人非常头疼的问题，尤其是我们在处理一些不规范的第三方网页的时候。虽然Python提供了Unicode表示的`str`和`bytes`两种数据类型，并且可以通过`encode()`和`decode()`方法转换，但是，在不知道编码的情况下，对`bytes`做`decode()`不好做。

对于未知编码的`bytes`，要把它转换成`str`，需要先“猜测”编码。猜测的方式是先收集各种编码的特征字符，根据特征字符判断，就能有很大概率“猜对”。

当然，我们肯定不能从头自己写这个检测编码的功能，这样做费时费力。chardet这个第三方库正好就派上了用场。用它来检测编码，简单易用。

安装chardet

如果安装了Anaconda，chardet就已经可用了。否则，需要在命令行下通过pip安装：

```
$ pip install chardet
```

如果遇到Permission denied安装失败，请加上sudo重试。

使用chardet

当我们拿到一个`bytes`时，就可以对其检测编码。用chardet检测编码，只需要一行代码：

```
>>> chardet.detect(b'Hello, world!')
{'encoding': 'ascii', 'confidence': 1.0, 'language': ''}
```

检测出的编码是`ascii`，注意到还有个`confidence`字段，表示检测的概率是1.0（即100%）。

我们来试试检测GBK编码的中文：

```
>>> data = '离离原上草，一岁一枯荣'.encode('gbk')
>>> chardet.detect(data)
{'encoding': 'GB2312', 'confidence': 0.7407407407407407, 'language': 'Chinese'}
```

检测的编码是`GB2312`，注意到GBK是GB2312的超集，两者是同一种编码，检测正确的概率是74%，`language`字段指出的语言是`'Chinese'`。

对UTF-8编码进行检测：

```
>>> data = '离离原上草，一岁一枯荣'.encode('utf-8')
>>> chardet.detect(data)
{'encoding': 'utf-8', 'confidence': 0.99, 'language': ''}
```

我们再试试对日文进行检测：

```
>>> data = '最新の主要ニュース'.encode('euc-jp')
>>> chardet.detect(data)
{'encoding': 'EUC-JP', 'confidence': 0.99, 'language': 'Japanese'}
```

可见，用chardet检测编码，使用简单。获取到编码后，再转换为`str`，就可以方便后续处理。

chardet支持检测的编码列表请参考官方文档Supported encodings。

小结

- 使用chardet检测编码非常容易，chardet支持检测中文、日文、韩文等多种语言。

psutil

用Python来编写脚本简化日常的运维工作是Python的一个重要用途。在Linux下，有许多系统命令可以让我们时刻监控系统运行的状态，如`ps`，`top`，`free`等等。要获取这些系统信息，Python可以通过`subprocess`模块调用并获取结果。但这样做显得很麻烦，尤其是要写很多解析代码。

在Python中获取系统信息的另一个好办法是使用`psutil`这个第三方模块。顾名思义，psutil = process and system utilities，它不仅可以通过一两行代码实现系统监控，还可以跨平台使用，支持Linux / UNIX / OSX / Windows等，是系统管理员和运维小伙伴不可或缺的必备模块。

安装psutil

如果安装了Anaconda，psutil就已经可用了。否则，需要在命令行下通过pip安装：

```
$ pip install psutil
```

如果遇到Permission denied安装失败，请加上sudo重试。

获取CPU信息

我们先来获取CPU的信息：

```
>>> import psutil
>>> psutil.cpu_count() # CPU逻辑数量
4
>>> psutil.cpu_count(logical=False) # CPU物理核心
2
# 2说明是双核超线程，4则是4核非超线程
```

统计CPU的用户 / 系统 / 空闲时间：

```
>>> psutil.cpu_times()
scputimes(user=10963.31, nice=0.0, system=5138.67, idle=356102.45)
```

再实现类似`top`命令的CPU使用率，每秒刷新一次，累计10次：

```
>>> for x in range(10):
...     psutil.cpu_percent(interval=1, percpu=True)
...
[14.0, 4.0, 4.0, 4.0]
[12.0, 3.0, 4.0, 3.0]
[8.0, 4.0, 3.0, 4.0]
[12.0, 3.0, 3.0, 3.0]
[18.8, 5.1, 5.9, 5.0]
[10.9, 5.0, 4.0, 3.0]
[12.0, 5.0, 4.0, 5.0]
[15.0, 5.0, 4.0, 4.0]
[19.0, 5.0, 5.0, 4.0]
[9.0, 3.0, 2.0, 3.0]
```

获取内存信息

使用psutil获取物理内存和交换内存信息，分别使用：

```
>>> psutil.virtual_memory()
svmem(total=8589934592, available=2866520064, percent=66.6, used=7201386496,
free=216178688, active=3342192640, inactive=2650341376, wired=1208852480)
>>> psutil.swap_memory()
sswap(total=1073741824, used=150732800, free=923009024, percent=14.0,
sin=10705981440, sout=40353792)
```

返回的是字节为单位的整数，可以看到，总内存大小是8589934592 = 8 GB，已用7201386496 = 6.7 GB，使用了66.6%。

而交换区大小是1073741824 = 1 GB。

获取磁盘信息

可以通过psutil获取磁盘分区、磁盘使用率和磁盘IO信息：

```
>>> psutil.disk_partitions() # 磁盘分区信息
[sdiskpart(device='/dev/disk1', mountpoint='/', fstype='hfs',
opts='rw,local,rootfs,dovolfs,journaled,multilabel')]
>>> psutil.disk_usage('/') # 磁盘使用情况
sdiskusage(total=998982549504, used=390880133120, free=607840272384,
percent=39.1)
>>> psutil.disk_io_counters() # 磁盘IO
sdiskio(read_count=988513, write_count=274457, read_bytes=14856830464,
write_bytes=17509420032, read_time=2228966, write_time=1618405)
```

可以看到，磁盘`'/'`的总容量是998982549504 = 930 GB，使用了39.1%。文件格式是HFS，`opts`中包含`rw`表示可读写，`journaled`表示支持日志。

获取网络信息

psutil可以获取网络接口和网络连接信息：

```
>>> psutil.net_io_counters() # 获取网络读写字节 / 包的个数
snetio(bytes_sent=3885744870, bytes_recv=10357676702, packets_sent=10613069,
packets_recv=10423357, errin=0, errout=0, dropin=0, dropout=0)
>>> psutil.net_if_addrs() # 获取网络接口信息
{
  'lo0': [snic(family=<AddressFamily.AF_INET: 2>, address='127.0.0.1',
netmask='255.0.0.0'), ...],
  'en1': [snic(family=<AddressFamily.AF_INET: 2>, address='10.0.1.80',
netmask='255.255.255.0'), ...],
  'en0': [...],
  'en2': [...],
  'bridge0': [...]
}
>>> psutil.net_if_stats() # 获取网络接口状态
{
  'lo0': snicstats(isup=True, duplex=<NicDuplex.NIC_DUPLEX_UNKNOWN: 0>, speed=0,
mtu=16384),
  'en0': snicstats(isup=True, duplex=<NicDuplex.NIC_DUPLEX_UNKNOWN: 0>, speed=0,
mtu=1500),
  'en1': snicstats(...),
  'en2': snicstats(...),
  'bridge0': snicstats(...)
}
```

要获取当前网络连接信息，使用`net_connections()`：

```
>>> psutil.net_connections()
Traceback (most recent call last):
  ...
PermissionError: [Errno 1] Operation not permitted

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  ...
psutil.AccessDenied: psutil.AccessDenied (pid=3847)
```

你可能会得到一个`AccessDenied`错误，原因是psutil获取信息也是要走系统接口，而获取网络连接信息需要root权限，这种情况下，可以退出Python交互环境，用`sudo`重新启动：

```
$ sudo python3
Password: ******
Python 3.6.3 ... on darwin
Type "help", ... for more information.
>>> import psutil
>>> psutil.net_connections()
[
    sconn(fd=83, family=<AddressFamily.AF_INET6: 30>, type=1,
laddr=addr(ip='::127.0.0.1', port=62911), raddr=addr(ip='::127.0.0.1',
port=3306), status='ESTABLISHED', pid=3725),
    sconn(fd=84, family=<AddressFamily.AF_INET6: 30>, type=1,
laddr=addr(ip='::127.0.0.1', port=62905), raddr=addr(ip='::127.0.0.1',
port=3306), status='ESTABLISHED', pid=3725),
    sconn(fd=93, family=<AddressFamily.AF_INET6: 30>, type=1,
laddr=addr(ip='::', port=8080), raddr=(), status='LISTEN', pid=3725),
    sconn(fd=103, family=<AddressFamily.AF_INET6: 30>, type=1,
laddr=addr(ip='::127.0.0.1', port=62918), raddr=addr(ip='::127.0.0.1',
port=3306), status='ESTABLISHED', pid=3725),
    sconn(fd=105, family=<AddressFamily.AF_INET6: 30>, type=1, ..., pid=3725),
    sconn(fd=106, family=<AddressFamily.AF_INET6: 30>, type=1, ..., pid=3725),
    sconn(fd=107, family=<AddressFamily.AF_INET6: 30>, type=1, ..., pid=3725),
    ...
    sconn(fd=27, family=<AddressFamily.AF_INET: 2>, type=2, ..., pid=1)
]
```

获取进程信息

通过psutil可以获取到所有进程的详细信息：

```
>>> psutil.pids() # 所有进程ID
[3865, 3864, 3863, 3856, 3855, 3853, 3776, ..., 45, 44, 1, 0]
>>> p = psutil.Process(3776) # 获取指定进程ID=3776，其实就是当前Python交互环境
>>> p.name() # 进程名称
'python3.6'
>>> p.exe() # 进程exe路径
'/Users/michael/anaconda3/bin/python3.6'
>>> p.cwd() # 进程工作目录
'/Users/michael'
>>> p.cmdline() # 进程启动的命令行
['python3']
>>> p.ppid() # 父进程ID
```

```
3765
>>> p.parent() # 父进程
<psutil.Process(pid=3765, name='bash') at 4503144040>
>>> p.children() # 子进程列表
[]
>>> p.status() # 进程状态
'running'
>>> p.username() # 进程用户名
'michael'
>>> p.create_time() # 进程创建时间
1511052731.120333
>>> p.terminal() # 进程终端
'/dev/ttys002'
>>> p.cpu_times() # 进程使用的CPU时间
pcputimes(user=0.081150144, system=0.053269812, children_user=0.0,
children_system=0.0)
>>> p.memory_info() # 进程使用的内存
pmem(rss=8310784, vms=2481725440, pfaults=3207, pageins=18)
>>> p.open_files() # 进程打开的文件
[]
>>> p.connections() # 进程相关网络连接
[]
>>> p.num_threads() # 进程的线程数量
1
>>> p.threads() # 所有线程信息
[pthread(id=1, user_time=0.090318, system_time=0.062736)]
>>> p.environ() # 进程环境变量
{'SHELL': '/bin/bash', 'PATH':
'/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:...', 'PWD': '/Users/michael',
'LANG': 'zh_CN.UTF-8', ...}
>>> p.terminate() # 结束进程
Terminated: 15 <-- 自己把自己结束了
```

和获取网络连接类似，获取一个root用户的进程需要root权限，启动Python交互环境或者`.py`文件时，需要`sudo`权限。

psutil还提供了一个`test()`函数，可以模拟出`ps`命令的效果：

```
$ sudo python3
Password: ******
Python 3.6.3 ... on darwin
Type "help", ... for more information.
>>> import psutil
>>> psutil.test()
USER         PID %MEM     VSZ     RSS TTY           START    TIME  COMMAND
root           0 24.0 74270628 2016380 ?              Nov18   40:51  kernel_task
root           1  0.1 2494140    9484 ?             Nov18   01:39  launchd
root          44  0.4 2519872   36404 ?             Nov18   02:02
UserEventAgent
root          45    ? 2474032    1516 ?             Nov18   00:14  syslogd
root          47  0.1 2504768    8912 ?             Nov18   00:03  kextd
root          48  0.1 2505544    4720 ?             Nov18   00:19  fseventsd
_appleeven    52  0.1 2499748    5024 ?             Nov18   00:00  appleeventsd
root          53  0.1 2500592    6132 ?             Nov18   00:02  configd
...
```

小结

psutil使得Python程序获取系统信息变得易如反掌。

psutil还可以获取用户信息、Windows服务等很多有用的系统信息，具体请参考psutil的官网：https://github.com/giampaolo/psutil