**22个学习Python必备的库，你是否还有没见过的呢**

如今全球各个行业内 Python 的使用状况怎么样呢？

这个问题就是我写这篇文章的初衷。我找出了22个最常用的 Python 包，希望能给你一些启发。

 首先我列出了最近一年内 PyPI 上下载量最高的 Python 包。我们来看看这些包的作用，它们的之间的关系，以及为什么会如此流行。

**很多人学习python，不知道从何学起。  
很多人学习python，掌握了基本语法过后，不知道在哪里寻找案例上手。  
很多已经做案例的人，却不知道如何去学习更加高深的知识。  
那么针对这三类人，我给大家提供一个好的学习平台，免费领取视频教程，电子书籍，以及课程的源代码！  
QQ群：1097524789**

## 1. Urllib3

**8.93亿次下载**

Urllib3 是 Python 的 HTTP 客户端，它提供了许多 Python 标准库没有的功能。

* 线程安全
* 连接池
* 客户端 SSL/TLS 验证
* 使用 multipart 编码进行文件上传
* 用于重传请求并处理 HTTP 重定向的辅助功能
* 支持 gzip 和 deflate 编码
* 支持 HTTP 和 SOCKS 代理

尽管名字叫做 Urllib3，但它并不是 Python 自带的 urllib2 的后继版本。如果你想尽可能使用 Python 的核心功能（比如由于某些限制导致不能安装），那么可以看看 urllib.request。

对于最终用户，我强烈推荐 requests 包（参考列表中的第六项）。Urllib3 之所以排名第一是因为几乎 1200 个软件包都依赖它，许多这些软件包也都在列表中名列前茅。

## 2. Six

**7.32亿次下载**

Six 是 Python 2 和 Python 3 兼容性工具。该项目的目的是让代码能够同时在 Python 2 和  Python 3 上运行。

它提供了许多函数，掩盖了 Python 2 和  Python 3 之间的语法差异。最容易理解的例子就是six.print\_()。在 Python 3 中，输出时需要使用 print() 函数，而 Python 2 中是使用不带括号的 print。因此，使用 six.print\_() 可以同时支持两种语言。

重点：

* 包的名字 six 来自于2 x 3 = 6
* 类似的库还有 future
* 如果想将代码转换成 Python 3（同时不再支持 Python 2），可以看看 2to3

尽管我理解该包如此流行，但还是希望人们尽快抛弃 Python 2，特别是从2020年1月1日起官方已经不再支持 Python 2了。

## 3. botocore, boto3, s3transfer, awscli

这几个项目放在一起说：

* botocore：第3名，6.6亿次下载
* s3transfer：第7名，5.84亿次下载
* awscli：第17名，3.94亿次下载
* boto3：第22名，3.29亿次下载

Botocore 是 AWS 的底层接口。botocore 是 boto3（第22名）库的基础，后者可以让你访问亚马逊的S3、EC2等服务。

Botocore 也是 AWS-CLI 的基础，后者是 AWS 的命令行界面。

s3transfer（第七名）是用于管理S3传输的 Python 库。该库仍在开发中，它的主页依然不建议使用，或者使用时至少要固定版本，因为即使在小版本号之间它的API也可能会发生变化。boto3、AWS-CLI 和许多其他项目都依赖于 s3transfer。

AWS 相关的库的排名如此高，正说明了 AWS 的服务是多么流行。

## 4. Pip

**6.27亿次下载**

我猜许多人都知道并且喜爱 pip（Python的包安装工具）。使用 pip 从 Python Package Index和其他仓库（如本地镜像或包含私有软件的自定义仓库等）安装软件包不费吹灰之力。

关于 pip 的趣事：

* Pip 的名字是个递归定义：Pip Installs Packages
* Pip 非常容易使用。安装一个软件包只需要执行 pip install <软件包名>。删除只需要执行 pip uninstall <软件包名>。
* Pip 最大的好处就是它可以安装一系列包，通常会放在 requirements.txt 文件中。该文件还可以指定每个包的详细版本号。绝大多数 Python 项目都会包含这个文件。
* 与 virtualenv（第57名）结合使用 pip，可以创建可预测的、独立的环境，而不会与系统本身的环境互相影响。

## 5. python-dateutil

**6.17亿次下载**

Python-dateutil 模块为标准的 datetime 模块提供了强大的功能扩展。普通的 Python datetime 无法做到的事情都可以使用 python-dateutil 完成。

用这个库可以完成许多非常酷的功能。我只举一个非常有用的例子：从日志文件中模糊解析日期字符串：

**from dateutil.parser import parse**

**logline = 'INFO 2020-01-01T00:00:01 Happy new year, human.'**

**timestamp = parse(log\_line, fuzzy=True)**

**print(timestamp)**

**# 2020-01-01 00:00:01**

## 6. requests

**6.11亿次下载**

Requests 基于下载量第一的库 urllib3。有了它，发送请求变得极其简单。许多人对 requests 的喜爱超过了 urllib3，因此 requets 的最终用户可能比 urllib3 还要多。后者更底层，通常作为其他项目的依赖出现。

下面的例子演示了 requests 有多么容易使用：

**import requests**

**r = requests.get('https://api.github.com/user', auth=('user', 'pass'))**

**r.status\_code**

**# 200**

**r.headers['content-type']**

**# 'application/json; charset=utf8'**

**r.encoding**

**# 'utf-8'**

**r.text**

**# u'{"type":"User"...'**

**r.json()**

**# {u'disk\_usage': 368627, u'private\_gists': 484, ...}**

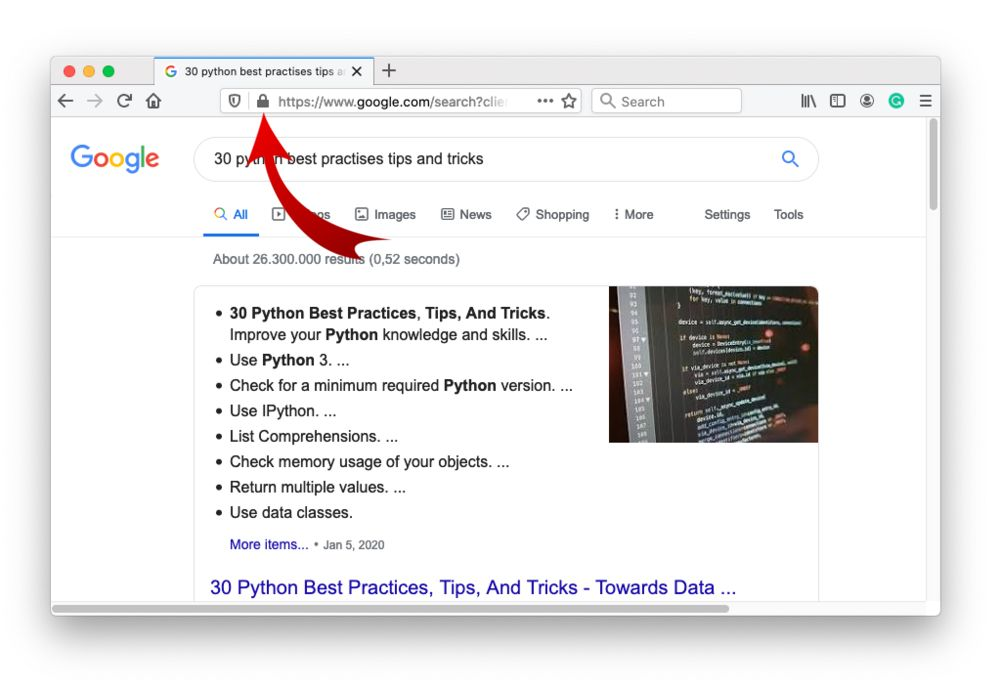
## 7. s3transfer

第3、7、17和22名互相关联，所以请参见第3名的介绍。

## 8. Certifi

**5.52亿次下载**

近年来，几乎所有网站都开始使用SSL，这一点可以从地址栏中的锁图标看出来，该图标的意思是网站是安全的、加密的，可以避免窃听。



加密基于SSL证书，SSL证书由可信的公司或非营利组织负责签发，如 LetsEncrypt。这些组织会对利用它们的证书对签发的证书进行数字签名。

利用这些证书的公开部分，浏览器就可以验证网站的签名，从而证明你访问的是真正的网站，而且别人没有在窃听数据。

Python 也可以做到同样的功能，这就需要用到 certifi。它和 Chrome、Firefox 和 Edge 等Web浏览器中包含的根证书集合没有什么区别。

Certifi 是一个根证书集合，这样 Python 代码就可以验证SSL证书的可信度。

许多项目都信赖并依赖 certifi，可以在这里看到这些项目。这也是为何该项目排名如此高的原因。

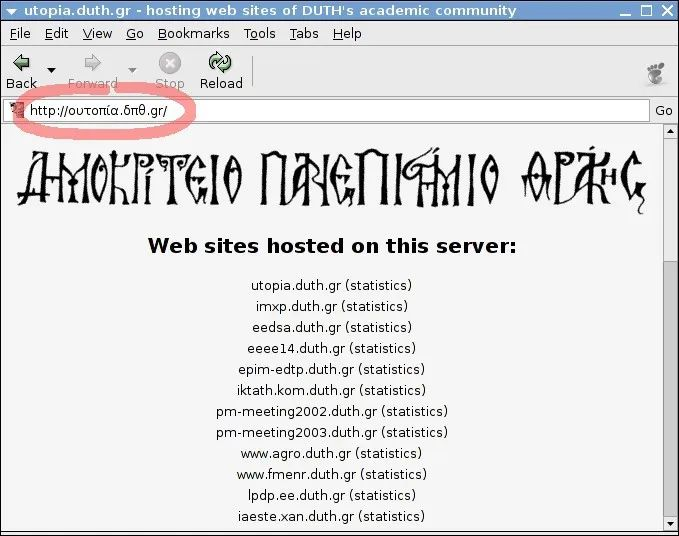
## 9. Idna

**5.27亿次下载**

根据 PyPI 的页面，idna提供“对于RFC5891中定义的IDNA协议（Internationalised Domain Names in Applications）的支持”。

我们来看看 idna 是什么意思：

IDNA 是处理包含非 ASCII 字符的域名的规则。但原始的域名不是已经支持非 ASCII 字符了吗？那么问题何在？



问题是许多应用程序（如Email客户端和Web浏览器等）并不支持非 ASCII 字符。或者更具体地说，Email 和 HTTP 协议并不支持这些字符。

在许多国家这并不是问题，但像中国、俄罗斯、德国、印尼等国家就很不方便。因此，这些国家的一些聪明人联合起来提出了 IDNA，也并非完全偶然。

IDNA 的核心是两个函数：ToASCII 和 ToUnicode。ToASCCI 会将国际化的 Unicode 域名转换成 ASCII 字符串，而 ToUnicode 会做相反的处理。在 IDNA 包中，这两个函数叫做 idna.encode() 和 idna.decode()，参见下面的例子：

**import idna**

**idna.encode('ドメイン.テスト')**

**# b'xn--eckwd4c7c.xn--zckzah'**

**print(idna.decode('xn--eckwd4c7c.xn--zckzah'))**

**# ドメイン.テスト**

该编码的详细内容可以参见 RFC3490。

## 10. PyYAML

**5.25亿次下载**

YAML 是一种数据序列化格式。它的设计目标是同时方便人类和机器阅读——人类很容易读懂，计算机解析也不难。



PyYAML 是 Python 的 YAM 解析器和编码器，也就是说它可以读写 YAML 格式。它可以将任何 Python 对象编码为 YAML：列表，字典，甚至类实例都可以。

Python 提供了自己的配置管理器，但 YAML 提供的功能远胜于 Python 自带的 ConfigParser（只能使用最基本的.ini文件）。

例如，YAML 能存储任何数据类型：boolean，list，float等。ConfigParse 的内部一切都保存为字符串。如果你要用 ConfigParser 来加载证书，就需要指明你需要的是整数：

config.getint(“section”, “my\_int”)

而 pyyaml 能够自动识别类型，因此只需这样就能获得 int：

config[“section”][“my\_int”]

YAML 还允许任意深度的嵌套，尽管并非每个项目都需要，但非常方便。

你可以自行决定使用哪一个，但许多项目都使用 YAML 作为配置文件，因此该项目的流行度非常高。

## 11. pyasn1

**5.12亿次下载**

像 IDNA 一样，这个项目的描述的信息量也非常大：

ASN.1 类型和 DER/BER/CER 编码（X.208）的纯 Python 实现。

幸运的是，我们依然能找到这个几十年之久的标准的许多资料。ASN.1 是 Abstract Syntax Notation One（抽象语法记法一）的缩写，是数据序列化的鼻祖。它来自于通讯行业。也许你知道 protocol buffer 或者 Apache Thrift 吧？ASN.1正是它们的1984年版本。

ASN.1 描述了一种不同系统之间的跨平台的接口，可以通过该接口发送数据结构。

还记得第8名的 certifi 吗？ASN.1 用于定义 HTTPS 协议以及许多其他加密系统中使用的证书的格式。ASN.1 还广泛用于 SNMP、LDAP、Kerberos、UMTS、LTE 和 VOIP 等协议中。

它是个非常复杂的标准，人们已经发现某些实现充满了脆弱性。你可以看看 Reddit 上的这个关于 ASN.1 的讨论（https://www.reddit.com/r/programming/comments/1hf7ds/useful\_old\_technologies\_asn1/）。

除非真正必要，否则我建议不要使用它。但由于许多地方都在使用该协议，因此许多包都依赖于它。

## 12. docutils

**5.08亿次下载**

Docutils 是一个模块化系统，用于将纯文本文档转换成其他格式，如 HTML、XML 和 LaTeX等。docutils 可以读取 reStructuredText 格式（一种类似于 MarkDown 的容易阅读的格式）的纯文本文档。

我猜你一定听说过 PEP 文档，甚至可能阅读过。PEP 文档是什么？

PEP 的意思是 Python Enhanced Proposal（Python增强提案）。PEP 是一篇设计文档，用于给 Pytho n社区提供信息，或者为 Python（或其处理器、环境）描述一个新特性。PEP 应该提供特性的精确的技术标准，并给出该特性的理由。

PEP 文档就是使用固定的 reStructuredText 模板，然后通过 docutils 转换成漂亮的文档。

Sphinx 的核心也使用了 docutils。Sphinx 用于创建文档项目。如果说 docutils 是一台机器，那么 Sphinx 就是一个工厂。它的最初设计目的是构建P ython 本身的文档，但许多其他项目也利用 Sphinx 来创建文档。

你一定度过 readthedocs.org 上的文档吧？那里的文档都是使用 Sphinx 和 docutils 创建的。

## 13. Chardet

**5.01亿下载**

你可以使用 chardet 模块来检查文件或数据流的字符集。在分析大量随机的文本时这个功能非常有用。但也可以用来判断远程下载的数据的字符串。

在安装 chardet 后，就可以使用命令行工具 chardetect，使用方法如下：

**chardetect somefile.txt**

**somefile.txt: ascii with confidence 1.0**

也可以在程序中使用该库，参见文档（https://chardet.readthedocs.io/en/latest/usage.html）。

Requests 和许多其他包都依赖于 chardet。我估计不会有太多人直接使用 chardet，所以它的流行度肯定是来自于这些依赖。

## 14. RSA

**4.92亿次下载**

Rsa是 RSA 的纯 Python 实现。它支持如下功能：

* 加密和解密
* 签名和签名验证
* 根据 PKCS#1 version 1.5生成秘钥

它可以作为 Python 库使用，也可以在命令行上使用。

* RSA 名称中的三个字母来自于三个人的姓：Ron Rivest，Adi Shamir，和Leonard Adleman。他们于1977年发明了该算法。
* RSA 是最早出现的一批公钥加密系统，广泛用于安全数据传输。这种加密系统包括两个秘钥：一个是公钥，一个是私钥。使用公钥加密数据，然后该数据只能用私钥进行解密。
* RSA 算法很慢。通常并不使用 RSA 算法直接加密用户数据，而是用它来加密对称加密系统中使用的共享秘钥，因为对称加密系统速度很快，适合用来加密大量数据。

下面 代码演示了 RSA 的使用方法：

**import rsa**

**# Bob creates a key pair:**

**(bob\_pub, bob\_priv) = rsa.newkeys(512)**

**# Alice ecnrypts a message for Bob**

**# with his public key**

**crypto = rsa.encrypt('hello Bob!', bob\_pub)**

**# When Bob gets the message, he**

**# decrypts it with his private key:**

**message = rsa.decrypt(crypto, bob\_priv)**

**print(message.decode('utf8'))**

**# hello Bob!**

假设 Bob 拥有私钥 private，Alice 就能确信只有 Bob 才能阅读该信息。

但 Bob 并不能确信 Alice 是信息的发送者，因为任何人都可以获得 Bob 的公钥。为了证明发送者的确是 Alice，她可以使用自己的私钥对信息进行签名。Bob 可以使用 Alice 的公钥对签名进行验证，来确保发送者的确是 Alice。

许多其他包都依赖于 rsa，如 google-auth（第37名），oauthlib（第54名），awscli（第17名）。这个包并不会经常被直接使用，因为有许多更快、更原生的方法。

## 15. Jmespath

**4.73亿次下载**

在 Python 中使用 JSON 很容易，因为 JSON 可以完美地映射到 Python 的字典上。我认为这是最好的特性之一。

说实话我从来没听说过 jmepath 这个包，尽管我使用过很多 JSON。我会使用 json.loads() 然后手动从字典中读取数据，或许还得写几个循环。

JMESPath，读作“James path”，能更容易地在 Python 中使用 JSON。你可以用声明的方式定义怎样从 JSON 文档中读取数据。下面是一些最基本的例子：

**import jmespath**

**# Get a specific element**

**d = {"foo": {"bar": "baz"}}**

**print(jmespath.search('foo.bar', d))**

**# baz**

**# Using a wildcard to get all names**

**d = {"foo": {"bar": [{"name": "one"}, {"name": "two"}]}}**

**print(jmespath.search('foo.bar[\*].name', d))**

**# [“one”, “two”]**

这仅仅是它的冰山一角。更多用法参见它的文档和 PyPI 主页。

## 16. Setuptools

4.01亿次下载

Setuptools 是用来创建 Python 包的工具。

这个项目的文档很糟糕。文档并没有描述它的功能，还包含死链接。真正的好文档在这里：https://packaging.python.org/，以及这篇文章中关于怎样创建 Python 包的教程：https://packaging.python.org/tutorials/packaging-projects/。

## 17. awscli

第3、7、17和22名互相关联，所以请参见第3名的介绍。

## 18. pytz

**3.94亿次下载**

类似于第5名的 dateutils，该库可以帮助你操作日期和时间。处理时区很麻烦。幸运的是，这个包可以让时区处理变得很容易。

关于时间，我的经验是：在内部永远使用UTC，只有在需要产生供人阅读的输出时才转换成本地时间。

下面是 pytz 的例子：

**from datetime import datetime**

**from pytz import timezone**

**amsterdam = timezone('Europe/Amsterdam')**

**ams\_time = amsterdam.localize(datetime(2002, 10, 27, 6, 0, 0))**

**print(ams\_time)**

**# 2002-10-27 06:00:00+01:00**

**# It will also know when it's Summer Time**

**# in Amsterdam (similar to Daylight Savings Time):**

**ams\_time = amsterdam.localize(datetime(2002, 6, 27, 6, 0, 0))**

**print(ams\_time)**

**# 2002-06-27 06:00:00+02:00**

更多文档和例子可以参见 PyPI 页面。

## 19. Futures

**3.89亿次下载**

从 Python 3.2 开始，python 开始提供 concurrent.futures 模块，可以帮你执行异步操作。futures 包是该库的反向移植，所以它是用于 Python 2 的。当前的 Python 3 版本不需要该包，因为 Python 3 本身就提供了该功能。

前面我说过，从2020年1月1日起官方已经停止支持 Python 2。我希望明年再做这个列表的时候，不再看到这个包排进前22名。

下面是 futures 包的基本用法：

**from concurrent.futures import ThreadPoolExecutor**

**from time import sleep**

**def return\_after\_5\_secs(message):**

**sleep(5)**

**return message**

**pool = ThreadPoolExecutor(3)**

**future = pool.submit(return\_after\_5\_secs,**

**("Hello world"))**

**print(future.done())**

**# False**

**sleep(5)**

**print(future.done())**

**# True**

**print(future.result())**

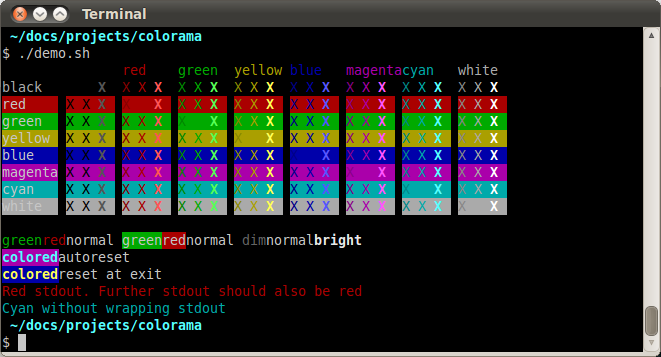
**# Hello World**

可见，我们可以创建一个线程池，然后提交一个函数，让某个线程执行。同时，你的程序会继续在主线程上运行。这是实现并行执行的一种很容易的方式。

## 20. Colorama

**3.70亿次下载**

你可以使用 Colorama 在终端上添加颜色：



下面的示例演示了实现这个功能有多么容易：

**from colorama import Fore, Back, Style**

**print(Fore.RED + 'some red text')**

**print(Back.GREEN + 'and with a green background')**

**print(Style.DIM + 'and in dim text')**

**print(Style.RESET\_ALL)**

**print('back to normal now')**

## 21. Simplejson

**3.41亿次下载**

Python 自带的 json 模块有什么问题导致了这个包有如此高的排名？没有任何问题！实际上， Python 的 json 就是 simplejson。但 simplejson 有一些优点：

* 能在更多 Python 版本上运行
* 更新频率高于 Python
* 一部分代码是用C编写的，运行得非常快

有时候你会看到脚本中这样写：

**try:**

**import simplejson as json**

**except ImportError:**

**import json**

不过，除非确实需要一些标准库中没有的功能，我依然会使用 json。SImplejson 可能比 json快很多，因为它的一部分是用C实现的。但是除非你要处理几千个 JSON 文件，否则这点速度提升并不明显。此外还可以看看 UltraJSON，这是个几乎完全用C编写的包，应该速度更快。

## 22. boto3

第3、7、17和22名互相关联，所以请参见第3名的介绍。

**结束语**

只写22个包很难，因为后面的许多包都是终端用户更倾向使用的包。

写这篇文章给了我一些启示：

* 许多排名靠前的包提供一些核心的功能，如处理时间、配置文件、加密和标准化等。它们通常是其他项目的依赖。
* 最常见的使用场景就是连接。许多包提供的功能就是连接到服务器，或者支持其他包连接服务器。
* 其他包是对 Python 的扩展，比如用于创建 Python 包的工具，创建文档的工具，创建版本兼容性的工具，等等。