

# 一、模块和包

## 1.1 什么是模块

在计算机程序的开发过程中，随着程序代码越写越多，在一个文件里代码就会越来越长，越来越不容易维护。

为了编写可维护的代码，我们把很多代码按功能分组，分别放到不同的文件里，这样，每个文件包含的代码就相对较少，很多编程语言都采用这种组织代码的方式。在Python中，一个.py文件就可以称之为一个模块（Module）。

### 使用模块有什么好处？

1. 最大的好处是大大提高了代码的可维护性。其次，编写代码不必从零开始。当一个模块编写完毕，就可以被其他地方引用。我们在编写程序的时候，也经常引用其他模块，包括Python内置的模块和来自第三方的模块。
2. 使用模块还可以避免函数名和变量名冲突。每个模块有独立的命名空间，因此相同名字的函数和变量完全可以分别存在不同的模块中，所以，我们自己在编写模块时，不必考虑名字会与其他模块冲突

### 模块分类

模块分为三种：

- 内置标准模块（又称标准库）执行`help('modules')`查看所有python自带模块列表
- 第三方开源模块，可通过`pip install 模块名` 联网安装
- 自定义模块

### 模块导入&调用

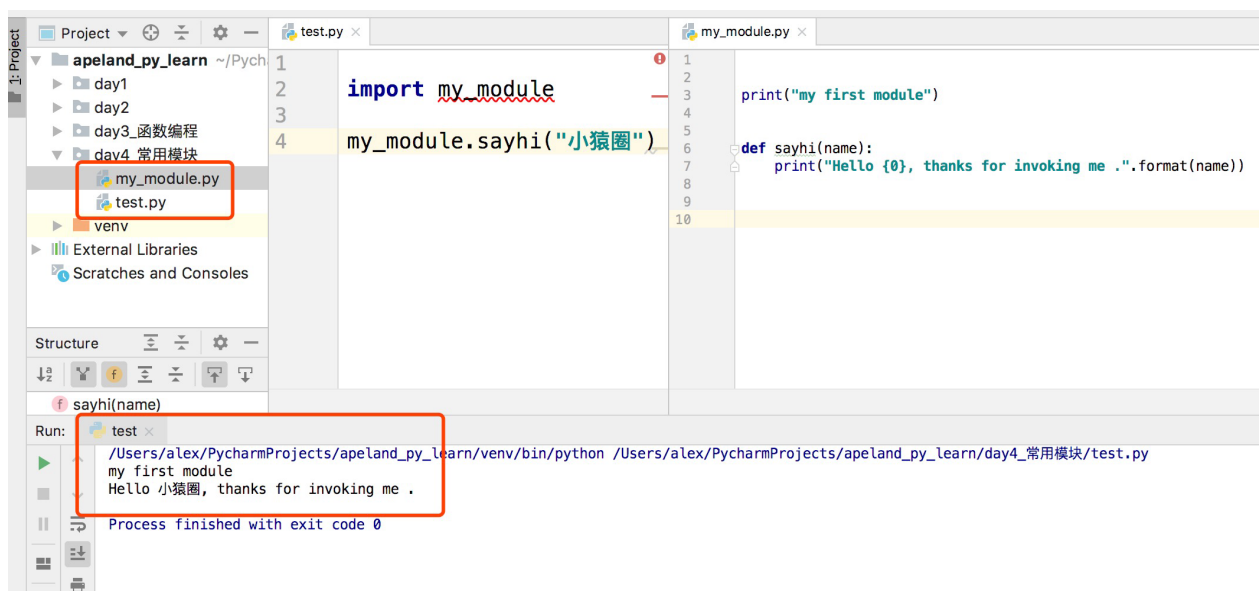
导入模块有以下几种方式：

```
import module_a #导入
from module import xx # 导入某个模块下的某个方法 or 子模块
from module.xx.xx import xx as rename #导入后一个方法后重命名
from module.xx.xx import * #导入一个模块下的所有方法，不建议使用
module_a.xxx #调用
```

注意：模块一旦被调用，即相当于执行了另外一个py文件里的代码

## 1.2 自定义模块

这个最简单，创建一个.py文件，就可以称之为模块，就可以在另外一个程序里导入



## 1.3 模块的查找路径

有没有发现，自己写的模块只能在当前路径下的程序里才能导入，换一个目录再导入自己的模块就报错说找不到了，这是为什么？

这与导入模块的查找路径有关

```
import sys
print(sys.path)
```

输出（注意不同的电脑可能输出的不太一样）

```
['', '/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6.zip',
'/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6',
'/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/lib-dynload',
'/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages']
```

你导入一个模块时，Python解释器会按照上面列表顺序去依次到每个目录下去匹配你要导入的模块名，只要在一个目录下匹配到了该模块名，就立刻导入，不再继续往后找。

注意列表第一个元素为空，即代表当前目录，所以你自己定义的模块在当前目录会被优先导入。

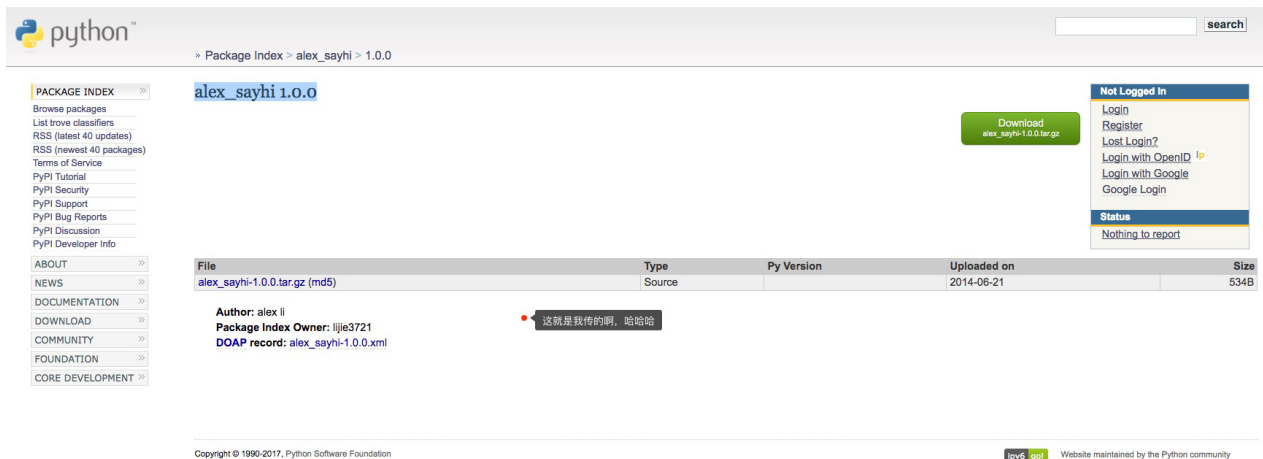
我们自己创建的模块若想在任何地方都能调用，那就得确保你的模块文件至少在模块路径的查找列表中。

我们一般把自己写的模块放在一个带有“site-packages”字样的目录里，我们从网上下载安装的各种第三方的模块一般都放在这个目录。

## 1.3 第3方开源模块安装

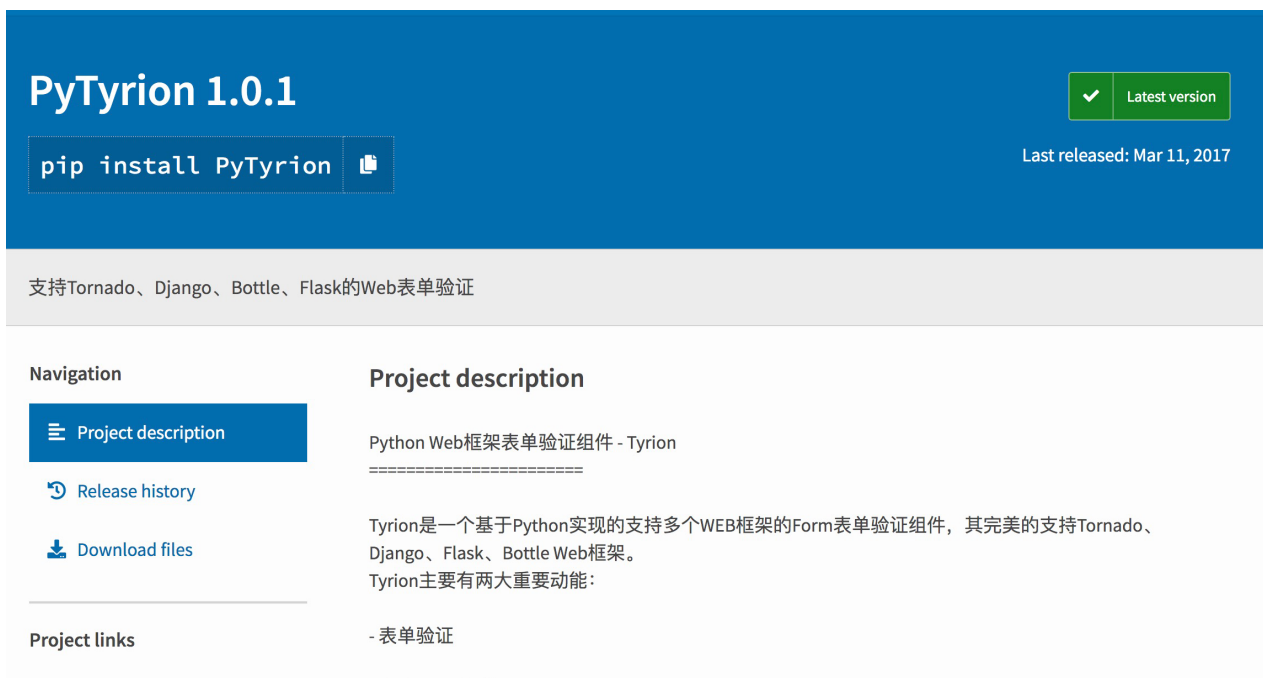
<https://pypi.python.org/pypi> 是python的开源模块库，截止2020年5.26日，已经收录了236,269个来自全世界python开发者贡献的模块，几乎涵盖了你想用python做的任何事情。事实上每个python开发者，只要注册一个账号就可以往这个平台上传你自己的模块，这样全世界的开发者都可以容易的下载并使用你的模块。

这个是我写的



The screenshot shows the PyPI package page for 'alex\_sayhi 1.0.0'. The page layout includes a sidebar on the left with links like 'PACKAGE INDEX', 'Browse packages', and 'PyPI Tutorial'. The main content area displays the package name, version, and a 'Download' button. Below this is a table with columns 'File', 'Type', 'Py Version', 'Uploaded on', and 'Size'. The table lists 'alex\_sayhi-1.0.0.tar.gz (md5)' as a source file uploaded on 2014-06-21. A red comment bubble points to the 'Author' field, stating '这就是我传的啊，哈哈'.

这是我弟弟写的



The screenshot shows the PyTyrion 1.0.1 project page. The header is blue and contains the project name 'PyTyrion 1.0.1', a 'pip install PyTyrion' button, and a 'Latest version' badge. Below the header is a section for 'Project description' and 'Project links'. The 'Project description' section includes a 'Navigation' sidebar with links to 'Project description', 'Release history', and 'Download files'. The 'Project description' text states: 'Python Web框架表单验证组件 - Tyrion' and 'Tyrion是一个基于Python实现的支持多个WEB框架的Form表单验证组件，其完美的支持Tornado、Django、Flask、Bottle Web框架。Tyrion主要有两大重要功能：- 表单验证'.

那如何从这个平台上下载代码呢？

1. 直接在上面这个页面上点download,下载后，解压并进入目录，执行以下命令完成安装

编译源码

```
python setup.py build 安装源码
python setup.py install
```

2. 直接通过pip安装

```
pip3 install paramiko #paramiko 是模块名
```

pip命令会自动下载模块包并完成安装。

软件一般会被自动安装到你python安装目录的这个子目录里

```
/your_python_install_path/3.6/lib/python3.6/site-packages
```

pip命令默认会连接在国外的python官方服务器下载，速度比较慢，你还可以使用国内的豆瓣源，数据会定期同步国外官网，速度快好多

```
pip install -i http://pypi.douban.com/simple/ alex_sayhi --trusted-host  
pypi.douban.com #alex_sayhi是模块名
```

-i 后面跟的是豆瓣源地址

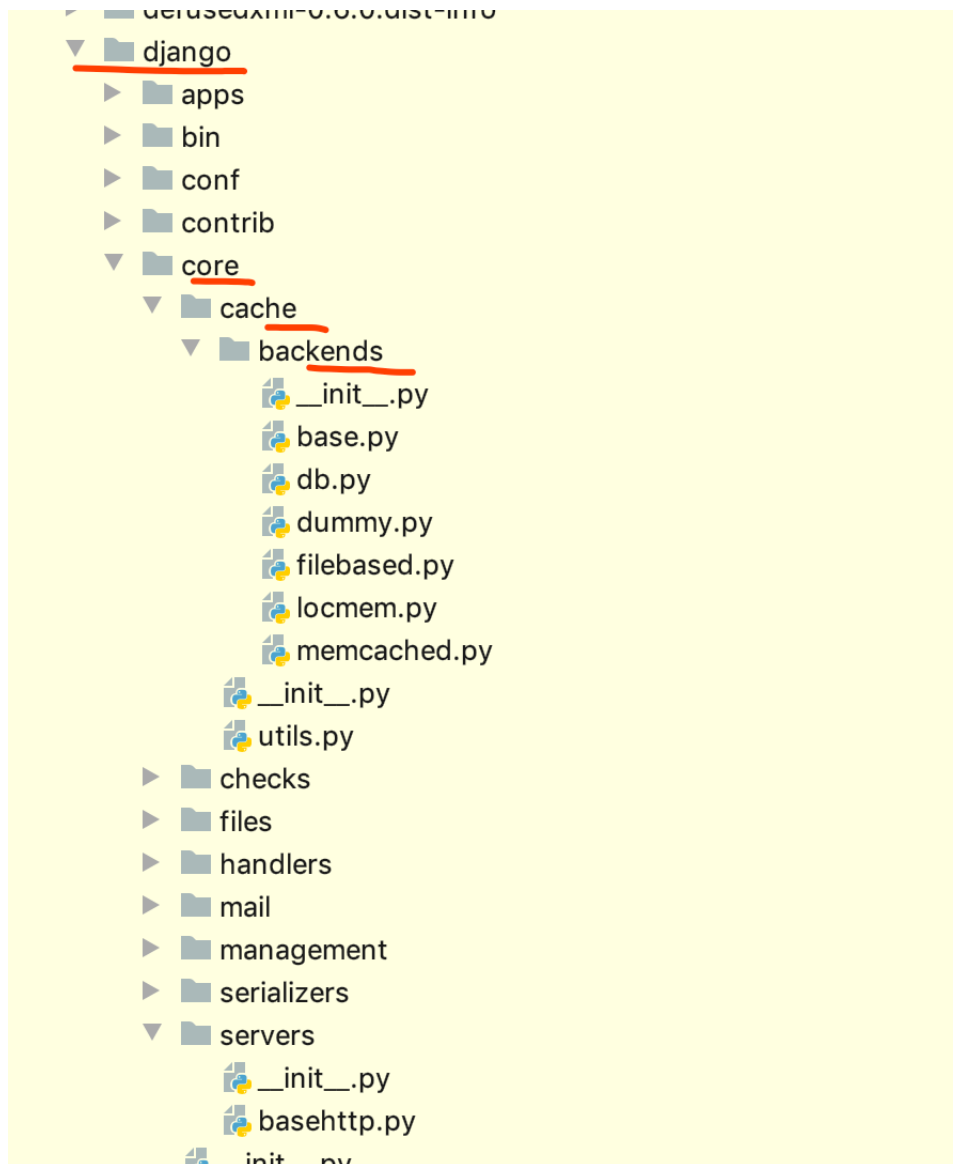
`--trusted-host` 得加上，是通过网站https安全验证用的

## 使用

下载后，直接导入使用就可以，跟自带的模块调用方法无差

## 1.4 什么是包 (package)

若你写的项目较复杂，有很多代码文件的话，为了方便管理，可以用包来管理。一个包其实就是一个文件目录，你可以把属于同一个业务线的代码文件都放在同一个包里。



## 如何创建一个包？

只需要在目录下创建一个空的 `__init__.py` 文件，这个目录就变成了包。这个文件叫包的初始化文件，一般为空，当然也可以写东西，当你调用这个包下及其任意子包的任意模块时，这个 `__init__.py` 文件都会先执行。

以下有a、b 2个包，a2是a的子包，b2是b的子包

```
day6
├── 课件
│   ├── a
│   │   ├── __init__.py
│   │   └── a2
│   │       ├── __init__.py
│   │       ├── a_mod2.py
│   │       └── a_module.py
│   ├── b
│   │   ├── __init__.py
│   │   ├── b2
│   │   │   ├── __init__.py
```

```
| | └─ b2_mod.py
| └─ b_module.py
```

若在**a\_module.py**模块里导入**b2\_mod.py**的话，怎么办？

a\_module.py的文件路径为

```
/Users/alex/Documents/work/PyProjects/py8days_camp/day6/课件/a/a2/a_module.py
```

想导入成功，直接写以下代码就可以

```
from day6.课件.b.b2 import b2_mod
```

为何从day6开始？而不是从 `py8days_camp` 或 `课件` 开始呢？

因为你的sys.path列表里，已经添加了相关的路径

```
['/Users/alex/Documents/work/PyProjects/py8days_camp/day6/课件/a/a2',
'/Users/alex/Documents/work/PyProjects/py8days_camp', # <---就是这个。。
'/Applications/PyCharm.app/Contents/helpers/pycharm_display',
'/Library/Frameworks/Python.framework/Versions/3.6/lib/python36.zip',
'/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6',
'/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/lib-dynload',
'/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages',
'/Applications/PyCharm.app/Contents/helpers/pycharm_matplotlib_backend']
```

## 手动添加sys.path路径

你会说，我没有添加这个 `'/Users/alex/Documents/work/PyProjects/py8days_camp'` 呀，它是怎么进到 `sys.path` 里的？

答案是Pycharm自动帮你添加的,若你脱离pycharm再执行这个 `a_module.py` 就会报错了。

```
Alexs-MacBook-Pro:a2 alex$ python3 a_module.py
Traceback (most recent call last):
  File "a_module.py", line 7, in <module>
    from day6.课件.b.b2 import b2_mod
ModuleNotFoundError: No module named 'day6'
```

不用慌，自己手动添加sys.path路径就可以

```

import os
import sys

base_dir=os.path.dirname(os.path.dirname(os.path.dirname(os.path.dirname(os.path.dirname(os.path.abspath(__file__)))))) # 取到路径/Users/alex/Documents/work/PyProjects/py8days_camp

print(base_dir)
sys.path.append(base_dir) # 添加到sys.path里

from day6.课件.b.b2 import b2_mod

```

## 二、几个常用Python模块

### 2.1 系统调用OS模块

os 模块提供了很多允许你的程序与操作系统直接交互的功能

```
import os
```

得到当前工作目录，即当前Python脚本工作的目录路径：os.getcwd()

返回指定目录下的所有文件和目录名：os.listdir()

函数用来删除一个文件：os.remove()

删除多个目录：os.removedirs(r"c:\python")

检验给出的路径是否是一个文件：os.path.isfile()

检验给出的路径是否是一个目录：os.path.isdir()

判断是否是绝对路径：os.path.isabs()

检验给出的路径是否真地存：os.path.exists()

返回一个路径的目录名和文件名：os.path.split() e.g

os.path.split('/home/swaroop/byte/code/poem.txt') 结果：

(' /home/swaroop/byte/code', 'poem.txt')

分离扩展名：os.path.splitext() e.g os.path.splitext('/usr/local/test.py')

结果：('/usr/local/test', '.py')

获取路径名：os.path.dirname()

获得绝对路径：os.path.abspath()

获取文件名：os.path.basename()

运行shell命令：os.system()

读取操作系统环境变量HOME的值：os.getenv("HOME")

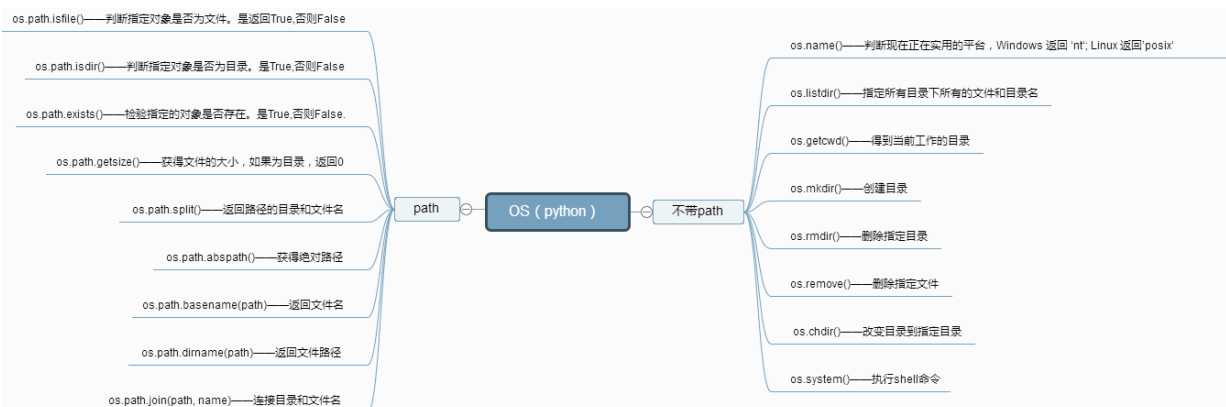
返回操作系统所有的环境变量：os.environ

设置系统环境变量，仅程序运行时有效：os.environ.setdefault('HOME','/home/alex')

给出当前平台使用的行终止符：os.linesep Windows使用'\r\n', Linux and MAC使用'\n'

指示你正在使用的平台：os.name 对于Windows，它是'nt'，而对于Linux/Unix用户，它是'posix'

```
重命名: os.rename (old, new)
创建多级目录: os.makedirs (r"c: \python\test")
创建单个目录: os.mkdir ("test")
获取文件属性: os.stat (file)
修改文件权限与时间戳: os.chmod (file)
获取文件大小: os.path.getsize (filename)
结合目录名与文件名: os.path.join(dir,filename)
改变工作目录到dirname: os.chdir(dirname)
获取当前终端的大小: os.get_terminal_size()
杀死进程: os.kill(10884,signal.SIGKILL)
```



## 2.2 time 模块

在平常的代码中，我们常常需要与时间打交道。在Python中，与时间处理有关的模块就包括：time，datetime,calendar(很少用，不讲)，下面分别来介绍。

我们写程序时对时间的处理可以归为以下3种：

**时间的显示**，在屏幕显示、记录日志等 "2022-03-04"

**时间的转换**，比如把字符串格式的日期转成Python中的日期类型

**时间的运算**，计算两个日期期间的差值等

**在Python中，通常有这几种方式来表示时间：**

1. 时间戳（timestamp），表示的是从1970年1月1日00:00:00开始按秒计算的偏移量。例子：  
1554864776.161901
2. 格式化的时间字符串，比如“2020-10-03 17:54”
3. 元组（struct\_time）共九个元素。由于Python的time模块实现主要调用C库，所以各个平台可能有所不同，mac上：time.struct\_time(tm\_year=2020, tm\_mon=4, tm\_mday=10, tm\_hour=2, tm\_min=53, tm\_sec=15, tm\_wday=2, tm\_yday=100, tm\_isdst=0)



索引 (Index)	属性 (Attribute)	值 (Values)
0	tm_year (年)	比如2011
1	tm_mon (月)	1 - 12
2	tm_mday (日)	1 - 31
3	tm_hour (时)	0 - 23
4	tm_min (分)	0 - 59
5	tm_sec (秒)	0 - 61
6	tm_wday (weekday)	0 - 6 (0表示周一)
7	tm_yday (一年中的第几天)	1 - 366
8	tm_isdst (是否是夏令时)	默认为-1

## UTC时间

UTC (Coordinated Universal Time, 世界协调时) 亦即格林威治天文时间, 世界标准时间。在中国为 UTC+8, 又称东8区。DST (Daylight Saving Time) 即夏令时。



## time模块的常用方法

- `time.localtime([secs])`: 将一个时间戳转换为当前时区的struct\_time。若secs参数未提供, 则以当前时间为准。
- `time.gmtime([secs])`: 和localtime()方法类似, gmtime()方法是将一个时间戳转换为UTC时区 (0时区) 的struct\_time。
- `time.time()`: 返回当前时间的时间戳。
- `time.mktime(t)`: 将一个struct\_time转化为时间戳。
- `time.sleep(secs)`: 线程推迟指定的时间运行,单位为秒。

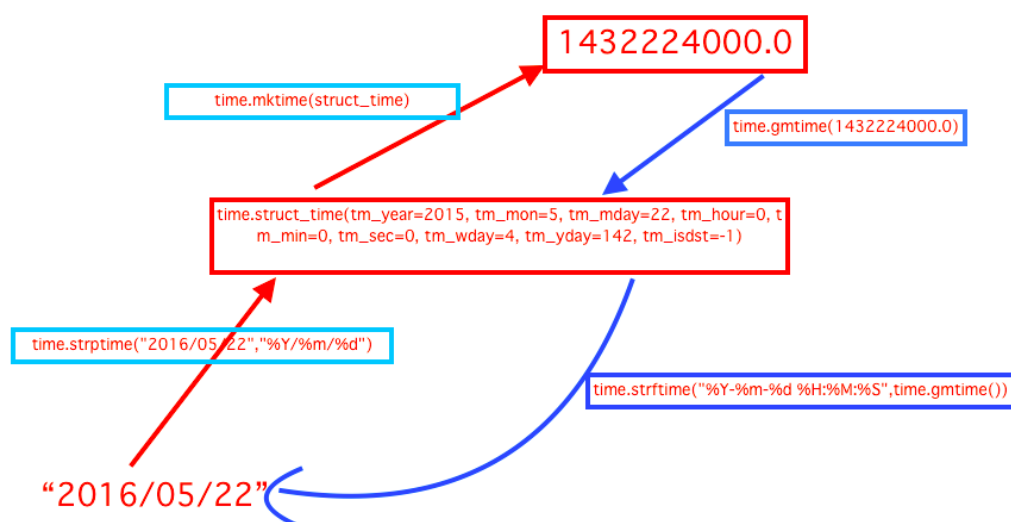
- `time.strftime(format[, t])`: 把一个代表时间的元组或者`struct_time`（如由`time.localtime()`和`time.gmtime()`返回）转化为格式化的时间字符串。如果`t`未指定，将传入`time.localtime()`。
  - 举例: `time.strftime("%Y-%m-%d %X", time.localtime())` #输出'2017-10-01 12:14:23'
- `time.strptime(string[, format])`: 把一个格式化时间字符串转化为`struct_time`。实际上它和`strftime()`是逆操作。
  - 举例: `time.strptime('2017-10-3 17:54', "%Y-%m-%d %H:%M")` #输出  
`time.struct_time(tm_year=2017, tm_mon=10, tm_mday=3, tm_hour=17, tm_min=54, tm_sec=0, tm_wday=1, tm_yday=276, tm_isdst=-1)`
  - 字符串转时间格式对应表
  -

Meaning	Notes	
<code>%a</code>	Locale's abbreviated weekday name.	
<code>%A</code>	Locale's full weekday name.	
<code>%b</code>	Locale's abbreviated month name.	
<code>%B</code>	Locale's full month name.	
<code>%c</code>	Locale's appropriate date and time representation.	
<code>%d</code>	Day of the month as a decimal number [01,31].	
<code>%H</code>	Hour (24-hour clock) as a decimal number [00,23].	
<code>%I</code>	Hour (12-hour clock) as a decimal number [01,12].	
<code>%j</code>	Day of the year as a decimal number [001,366].	
<code>%m</code>	Month as a decimal number [01,12].	
<code>%M</code>	Minute as a decimal number [00,59].	
<code>%p</code>	Locale's equivalent of either AM or PM.	(1)
<code>%S</code>	Second as a decimal number [00,61].	(2)
<code>%U</code>	Week number of the year (Sunday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Sunday are considered to be in week 0.	(3)
<code>%w</code>	Weekday as a decimal number [0(Sunday),6].	
<code>%W</code>	Week number of the year (Monday as the first day of the week) as a decimal number [00,53]. All days in a new year	(3)

	preceding the first Monday are considered to be in week 0.	
<code>%x</code>	Locale's appropriate date representation.	
<code>%X</code>	Locale's appropriate time representation.	
<code>%y</code>	Year without century as a decimal number [00,99].	
<code>%Y</code>	Year with century as a decimal number.	
<code>%z</code>	Time zone offset indicating a positive or negative time difference from UTC/GMT of the form +HHMM or -HHMM, where H represents decimal hour digits and M represents decimal minute digits [-23:59, +23:59].	
<code>%Z</code>	Time zone name (no characters if no time zone exists).	

- 最后为了容易记住转换关系，看下图

•



时间 转换关系 图, by ALEX, OLDBOY PY

## 2.3 datetime模块

相比于time模块，datetime模块的接口则更直观、更容易调用

**datetime**模块定义了下面这几个类：

- datetime.date：表示日期的类。常用的属性有year, month, day；
- datetime.time：表示时间的类。常用的属性有hour, minute, second, microsecond；
- datetime.datetime：表示日期时间。
- datetime.timedelta：表示时间间隔，即两个时间点之间的长度。
- datetime.tzinfo：与时区有关的相关信息。（这里不详细充分讨论该类，感兴趣的童鞋可以参考python手册）

我们需要记住的方法仅以下几个：

1. `d=datetime.datetime.now()` 返回当前的datetime日期类型, `d.timestamp()`,`d.today()`, `d.year`,`d.timetuple()`等方法可以调用
2. `datetime.date.fromtimestamp(322222)` 把一个时间戳转为datetime日期类型
3. 时间运算

```
>>> datetime.datetime.now()
datetime.datetime(2017, 10, 1, 12, 53, 11, 821218)
>>> datetime.datetime.now() + datetime.timedelta(4) #当前时间 +4天
datetime.datetime(2017, 10, 5, 12, 53, 35, 276589)
>>> datetime.datetime.now() + datetime.timedelta(hours=4) #当前时间+4小时
datetime.datetime(2017, 10, 1, 16, 53, 42, 876275)
```

4. 时间替换

```
>>> d.replace(year=2999,month=11,day=30)
datetime.date(2999, 11, 30)
```

## 2.4 random随机数

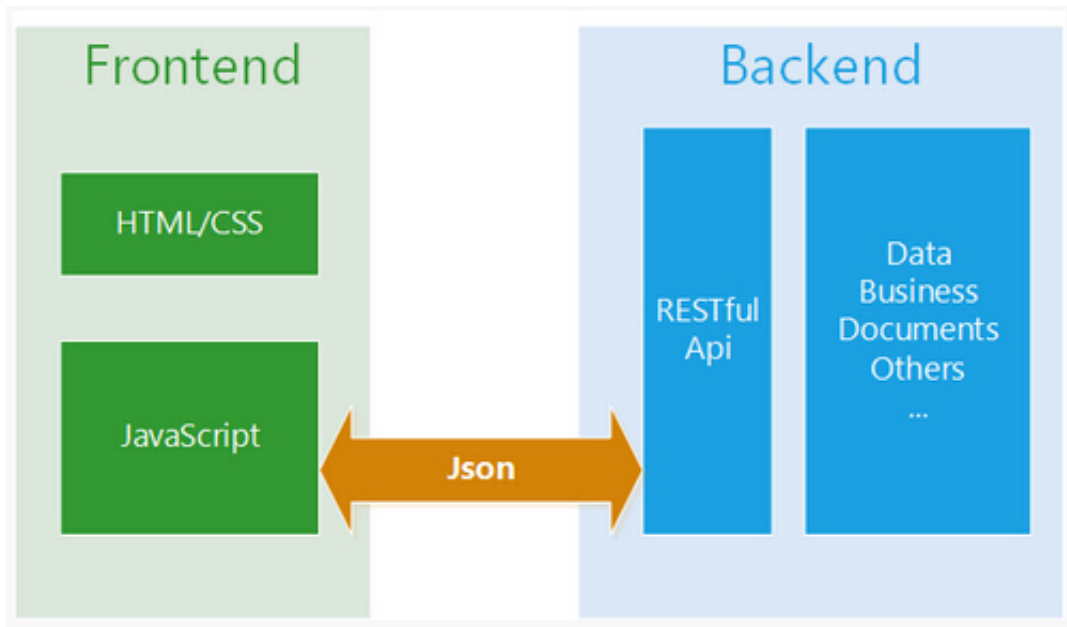
程序中有很多地方需要用到随机字符，比如登录网站的随机验证码，通过random模块可以很容易生成随机字符串

```
>>> random.randrange(1,10) #返回1-10之间的一个随机数，不包括10
>>> random.randint(1,10) #返回1-10之间的一个随机数，包括10
>>> random.randrange(0, 100, 2) #随机选取0到100间的偶数
>>> random.random() #返回一个随机浮点数
>>> random.choice('abce3#$@1') #返回一个给定数据集中的随机字符
'#'
>>> random.sample('abcdefghij',3) #从多个字符中选取特定数量的字符
['a', 'd', 'b']
#生成随机字符串
>>> import string
>>> ''.join(random.sample(string.ascii_lowercase + string.digits, 6))
'4fvdal'
#洗牌
>>> a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> random.shuffle(a)
>>> a
[3, 0, 7, 2, 1, 6, 5, 8, 9, 4]
```

## 2.5 序列化json模块

### 什么是Json?

[JSON](#)(JavaScriptObject Notation, JS 对象简谱) 是一种轻量级的数据交换格式。它采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。



Json的作用是用于不同语言接口间的数据交换，比如你把python的list、dict直接扔给javascript, 它是解析不了的。2个语言互相谁也不认识。Json就像是计算机界的英语，可以帮各个语言之间实现数据类型的相互转换。

```
[>>> names = ["Alex大王","小猪佩奇","黑姑娘"]
[>>> names
['Alex大王 ', '小猪佩奇 ', '黑姑娘 ']
[>>>
[>>> import json
[>>> json.dumps(names)
'["Alex\\u5927\\u738b", "\\u5c0f\\u732a\\u4f69\\u5947", "\\u9ed1\\u59d1\\u5a18"]'
[>>>
[>>>
```

**Python 序列化**

```
> var names = '["Alex\\u5927\\u738b", "\\u5c0f\\u732a\\u4f69\\u5947", "\\u9ed1\\u59d1\\u5a18"]'
< undefined
> JSON.parse(names)
< ▶ (3) ["Alex大王", "小猪佩奇", "黑姑娘"]
>
```

**Json 反序列化**

## JSON支持的数据类型

Python中的字符串、数字、列表、字典、集合、布尔 类型，都可以被序列化成JSON字符串，被其它任何编程语言解析

## 什么是序列化？

序列化是指把内存里的数据类型转变成字符串，以使其能存储到硬盘或通过网络传输到远程，因为硬盘或网络传输时只能接受bytes

## 为什么要序列化？

你打游戏过程中，打累了，停下来，关掉游戏、想过2天再玩，2天之后，游戏又从你上次停止的地方继续运行，你上次游戏的进度肯定保存在硬盘上了，是以何种形式呢？游戏过程中产生的很多临时数据是不规律的，可能在你关掉游戏时正好有10个列表，3个嵌套字典的数据集合在内存里，需要存下来？你如何存？把列表变成文件里的多行多列形式？那嵌套字典呢？根本没法存。所以，若是有一种办法可以直接把内存数据存到硬盘上，下次程序再启动，再从硬盘上读回来，还是原来的格式的话，那是极好的。

用于序列化的两个模块

- json，用于字符串 和 python数据类型间进行转换
- pickle，用于python特有的类型 和 python的数据类型间进行转换

## pickle

模块提供了四个功能：dumps、dump、loads、load

```
import pickle
data = {'k1':123,'k2':'Hello'}
# pickle.dumps 将数据通过特殊的形式转换位只有python语言认识的字符串
p_str = pickle.dumps(data) # 注意dumps会把数据变成bytes格式
print(p_str)
# pickle.dump 将数据通过特殊的形式转换位只有python语言认识的字符串，并写入文件
with open('result.pk',"wb") as fp:
    pickle.dump(data,fp)
# pickle.load 从文件里加载
f = open("result.pk","rb")
d = pickle.load(f)
print(d)
```

## json

Json模块也提供了四个功能：dumps、dump、loads、load，用法跟pickle一致

```
import json
# json.dumps 将数据通过特殊的形式转换位所有程序语言都认识的字符串
j_str = json.dumps(data) # 注意json dumps生成的是字符串, 不是bytes
print(j_str)
#dump入文件
with open('result.json','w') as fp:
    json.dump(data,fp)
#从文件里load
with open("result.json") as f:
    d = json.load(f)
    print(d)
```

## json vs pickle:

### JSON:

优点：跨语言(不同语言间的数据传递可用json交接)、体积小

缺点：只能支持int\str\list\tuple\dict

### Pickle:

优点：专为python设计，支持python所有的数据类型

缺点：只能在python中使用，存储数据占空间大

## 2.6 Excel处理模块

第3方开源模块,安装

```
pip install openpyxl
```

### 2.6.1 打开文件

#### 一、创建

```
from openpyxl import Workbook
# 实例化
wb = Workbook()
# 获取当前active的sheet
ws = wb.active
print(sheet.title) # 打印sheet表名
sheet.title = "salary luffy" # 改sheet 名
```

#### 二、打开已有文件

```
>>> from openpyxl import load_workbook
>>> wb2 = load_workbook('文件名称.xlsx')
```

## 2.6.2 写数据

```
# 方式一：数据可以直接分配到单元格中(可以输入公式)
sheet["C5"] = "Hello 金角大王"
sheet["C7"] = "Hello 金角大王2"

# 方式二：可以附加行，从第一列开始附加(从最下方空白处，最左开始)(可以输入多行)
sheet.append([1, 2, 3])

# 方式三：Python 类型会被自动转换
sheet['A3'] = datetime.datetime.now().strftime("%Y-%m-%d")
```

## 2.6.3 选择表

```
# sheet 名称可以作为 key 进行索引
ws3 = wb["New Title"]
ws4 = wb.get_sheet_by_name("New Title")

print(wb.get_sheet_names()) # 打印所有的sheet
sheet = wb.worksheets[0] # 获得第1个sheet
```

## 2.6.4 保存表

```
wb.save('文件名称.xlsx')
```

## 2.6.5 遍历表数据

按行遍历

```
for row in sheet: # 循环获取表数据
    for cell in row: # 循环获取每个单元格数据
        print(cell.value, end=", ")
    print()
```

按列遍历



```
# A1, A2, A3这样的顺序
for column in sheet.columns:
    for cell in column:
        print(cell.value,end=",")
    print()
```

### 遍历指定行&列

```
# 从第2行开始至第5行，每行打印5列
for row in sheet.iter_rows(min_row=2,max_row=5,max_col=5):
    for cell in row:
        print(cell.value,end=",")
    print()
```

### 遍历指定几列的数据

取得第2-第5列的数据

```
for col in sheet.iter_cols(min_col=2,max_col=5,):
    for i in col:
        print(i.value,end=",")
    print()
```

## 2.6.6 删除工作表

```
# 方式一
wb.remove(sheet)
# 方式二
del wb[sheet]
```

## 2.6.7 设置单元格样式

### 一、需导入的类

```
from openpyxl.styles import Font, colors, Alignment
```

### 二、字体

下面的代码指定了等线24号，加粗斜体，字体颜色红色。直接使用cell的font属性，将Font对象赋值给它。

```
bold_italic_24_font = Font(name='等线', size=24, italic=True, color=colors.RED,
bold=True) # 声明样式

sheet['A1'].font = bold_italic_24_font # 给单元格设置样式
```

### 三、对齐方式

也是直接使用cell的属性alignment，这里指定垂直居中和水平居中。除了center，还可以使用right、left等等参数。

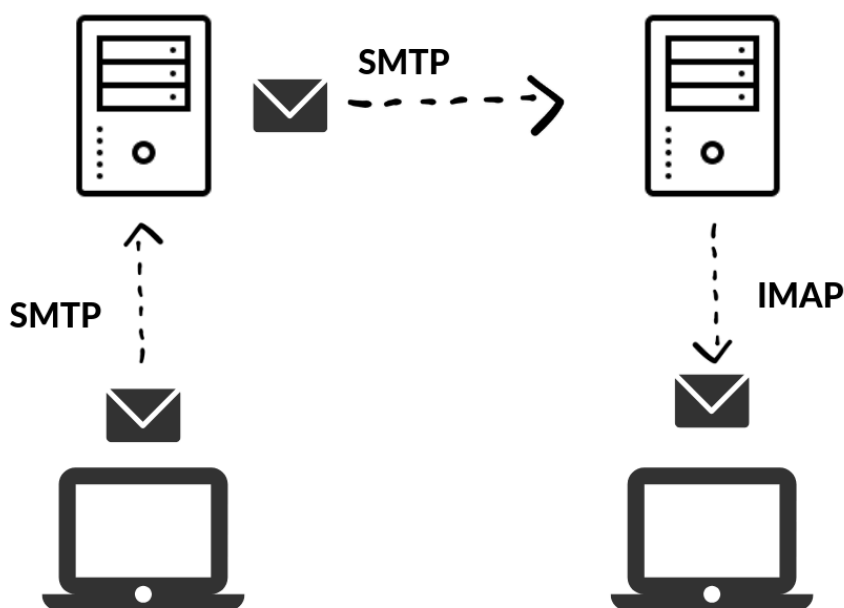
```
# 设置B1中的数据垂直居中和水平居中
sheet['B1'].alignment = Alignment(horizontal='center', vertical='center')
```

### 四、设置行高&列宽

```
# 第2行行高
sheet.row_dimensions[2].height = 40
# C列列宽
sheet.column_dimensions['C'].width = 30
```

## 2.7 邮件发送smtplib

SMTP（Simple Mail Transfer Protocol）即简单邮件传输协议,它是一组用于由源地址到目的地地址传送邮件的规则，由它来控制信件的中转方式。



想实现发送邮件需经过以下几步：

1. 登录 邮件服务器
2. 构造符合邮件协议规则要求的邮件内容（email模块）
3. 发送

Python对SMTP支持有 `smtpplib` 和 `email` 两个模块，`email` 负责构造邮件，`smtpplib` 负责发送邮件,它对smtp协议进行了简单的封装。。

### 2.7.1 发送一封最简单的信语法如下:

```
import smtplib
from email.mime.text import MIMEText # 邮件正文
from email.header import Header     # 邮件头

# 登录邮件服务器
smtp_obj = smtplib.SMTP_SSL("smtp.exmail.qq.com", 465) # 发件人邮箱中的SMTP服务器, 端口是25
smtp_obj.login("nami@luffycity.com", "xxxx-sd#gf") # 括号中对应的是发件人邮箱账号、邮箱密码
#smtp_obj.set_debuglevel(1) # 显示调试信息

# 设置邮件头信息
msg = MIMEText("Hello, 小哥哥, 约么? 800上门, 新到学生妹...", "plain", "utf-8")
msg["From"] = Header("来自娜美的问候", "utf-8") # 发送者
msg["To"] = Header("有缘人", "utf-8") # 接收者
msg["Subject"] = Header("娜美的信", "utf-8") # 主题

# 发送
smtp_obj.sendmail("nami@luffycity.com", ["alex@luffycity.com",
"317822232@qq.com"], msg.as_string())
```

### 2.7.2 发送HTML格式的邮件

只需要改一下 `MIMEText()` 第2个参数为 `html` 就可以

```
# 设置邮件头信息
mail_body = '''
    <h5>hello,小哥哥</h5>
    <p>
        小哥哥, 约么? 800, 新到学生妹.. <a
href="http://wx1.sinaimg.cn/mw1024/5ff6135fgylgdngzh2vbsg205k09ob2d.gif">这是我的照片</a></p>
    </p>
    ...

msg = MIMEText(mail_body, "html", "utf-8")
```

### 2.7.3 在HTML文本中插入图片

有兴趣的可以自己研究

```
# -*- coding:utf-8 -*-
# created by Alex Li - 路飞学城
```

```

import smtplib
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.header import Header

# 登录邮件服务器
smtp_obj = smtplib.SMTP_SSL("smtp.exmail.qq.com", 465) # 发件人邮箱中的SMTP服务器, 端口是25
smtp_obj.login("nami@luffycity.com", "333dsfsf#$") # 括号中对应的是发件人邮箱账号、邮箱密码
smtp_obj.set_debuglevel(1) # 显示调试信息

# 设置邮件头信息
mail_body = '''
    <h5>hello,小哥哥</h5>
    <p>
        小哥哥, 约么? 800, 新到学生妹..

        <p></p>
    </p>
'''

msg_root = MIMEMultipart('related') # 允许添加附件、图片等
msg_root["From"] = Header("来自娜美的问候", "utf-8") # 发送者
msg_root["To"] = Header("有缘人", "utf-8") # 接收者
msg_root["Subject"] = Header("娜美的信", "utf-8") # 主题

# 允许添加图片
msgAlternative = MIMEMultipart('alternative')
msgAlternative.attach(MIMEText(mail_body, 'html', 'utf-8'))

msg_root.attach(msgAlternative) # 把邮件正文内容添加到msg_root里

# 加载图片,
fp = open('girl.jpg', 'rb')
msgImage = MIMEImage(fp.read())
fp.close()

# 定义图片 ID, 在 HTML 文本中引用
msgImage.add_header('Content-ID', '<image1>')
msg_root.attach(msgImage) # 添加图片到msg_root对象里

# 发送
smtp_obj.sendmail("nami@luffycity.com", ["alex@luffycity.com",
"317828332@qq.com"], msg_root.as_string())

```

### 三、实战：给员工自动批量发工资条

处理以下excel表，批量给每个员工发工资条

A	B	C	D	E	F	G	H	I	J	K	L	M
序号	邮箱	姓名	部门	基本工资	绩效工资	提成	电脑补贴	电话补贴	社保	请假	公积金	
1	alex@luffycity.com	李渊	总经办	00.00	2,900.00		100.00		-382.72	-4,000.00	-720.00	
2	alex@luffycity.com	李世民	总经办	00.00	2,000.00		100.00		-382.72		-720.00	
3	alex@luffycity.com	侯君集	销售部	37.93		504.84			-382.72	-1,379.31	-274.00	
4	alex@luffycity.com	李靖	销售部	00.00	5,000.00		100.00		-382.72		-720.00	
5	alex@luffycity.com	魏征	销售部	0.00			100.00		-382.72		-720.00	
6	alex@luffycity.com	房玄龄	研发部	2.00	1,400.00	439.56	100.00		-382.72		-720.00	
7	alex@luffycity.com	杜如晦	研发部	1.00	1,600.00				-382.72	-2,942.53	-720.00	
8	alex@luffycity.com	柴绍	研发部	0.00	1,500.00		100.00		-382.72	-149.43	-720.00	
9	alex@luffycity.com	程知节	研发部	0.00	1,000.00	824.00			-382.72		-720.00	
10	alex@luffycity.com	尉迟恭	运营部	00	3,375.00				-382.72	720.69	-720.00	
11	alex@luffycity.com	长孙无忌	运营部	00	4,000.00	2,359.25	100.00		-382.72		-720.00	
12	alex@luffycity.com	李存恭	运营部	90					-382.72			
13	alex@luffycity.com	封德彝	运营部	00		585.16			-382.72		-274.00	
14	alex@luffycity.com	段志玄	运营部	00		433.98			-382.72		-274.00	
15	alex@luffycity.com	刘弘基	运营部	00		1,282.98			-382.72	-467.24	-274.00	
16	alex@luffycity.com	徐世绩	运营部	18	00		100.00		-382.72		-720.00	
17	alex@luffycity.com	武则天	总经办	1	17				-382.72			
合计				143	83	22,775.00	6,429.77	800.00	0.00	-5,740.80	-8,217.82	-9,016.00

格式如下：

老男孩教育财务部Inbox - Luffycity

2020-4月工资条

To:

你好,

请查收上月工资条，谢谢。

2020-05-24 15:59:14

序号	邮箱	姓名	部门	基本工资	绩效工资	提成	电脑补贴	电话补贴	社保	请假	公积金	本月应扣缴额	实发
2				14000	2000	None	100	None	-382.72	None	-720	579.15	14418.13

### 四、实战：批量从1000号员工word简历中调取技能关键词

这个不讲了，跟发工资的那个难度 差不多，只不过是操作word, 需要的同学，可以找我要相关源码

## 五、作业：生成1000张学员海报



需求：

1. 1000张海报名字不同，改在图片LTing处
2. 为每为同学的海报生成单独的2维码（选做）