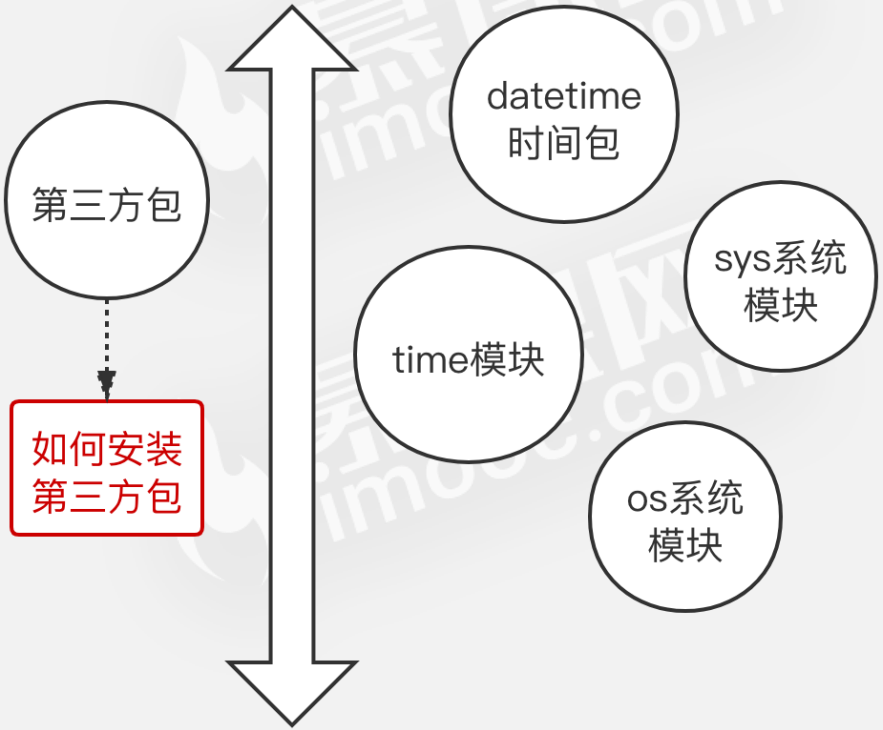
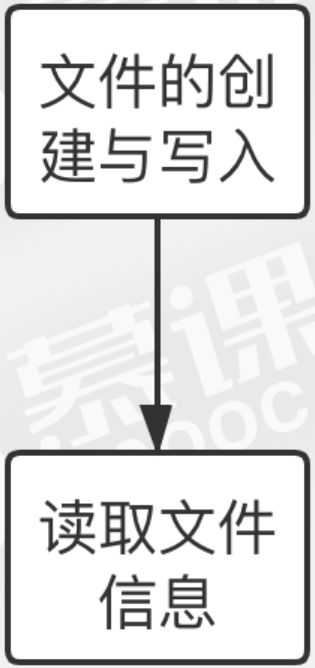
Python模块与文件操作

包与模块


什么是包
什么是模块
如何自定义
包或模块
如何导入包
或模块


第三方包
如何安装
第三方包
datetime
时间包
sys系统
模块
time模块
os系统
模块

文件的读写

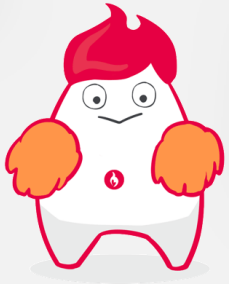


yaml文件模块

json文件与模块

日志模块

Python中的包

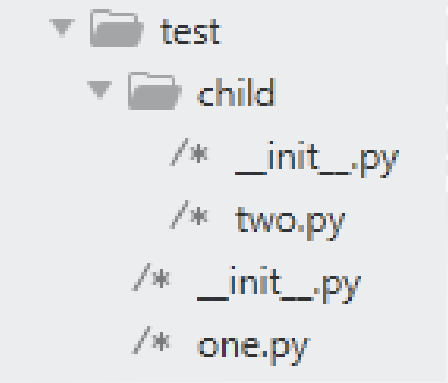


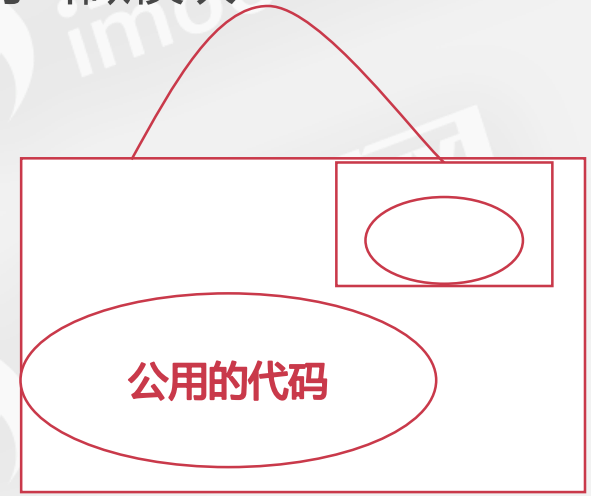
本节课内容

- 什么是python的包与模块
- 包的身份证
- 如何创建包
- 包的倒入
- 模块的导入

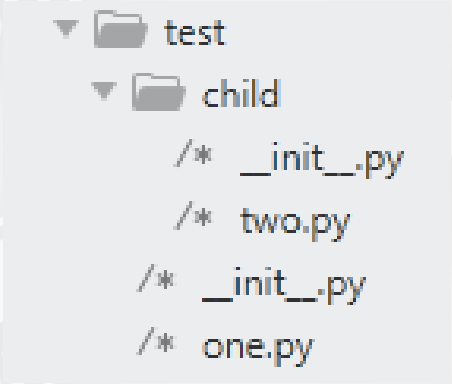
什么是python的包

- 原则上就是一个文件夹，里边放着一个个py文件或者子包
- 在包中可以被调用的一个个py文件，我们叫做模块





包的身份证



- __init__.py是每一个python包里必须存在的文件

如何创建一个包

- 要有一个主题，明确功能，方便使用
- 层次分明，调用清晰

包的导入 import

功能：

将python中的某个包(或模块)，导入到当前的py文件中

用法：

import package

参数：

package：被导入的包的名字

要求：

只会拿到对应包下__init__中的功能或当前模块下的功能

包的导入 import

```
In [1]: import random

In [2]: random.random()
Out[2]: 0.4829682516975815
```

模块的导入 from .. import ..

功能：

通过从某个包中找到对应的模块

用法：

from package import module

参数：

package： 来源的包名

module：包中的目标模块

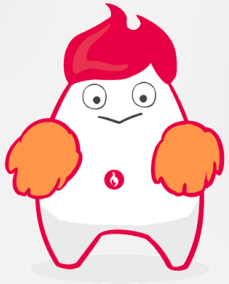
模块的导入 from .. import ..

举例：

```
from animal import dog
dog.shout()
```

- 我们通过from import 直接找到了dog模块，所以只需要使用dog模块直接用.的方式找到里边的方法并执行

Python中的包



本节课内容

- 什么是第三方包
- 利用pip与easy_install 获取第三方包
- 第一个第三方包工具 ipython

什么是第三方包

◆其他程序员写好的功能封装成包（模块）发布到网上

◆提高开发效率

利用pip与easy_install 获取第三方包

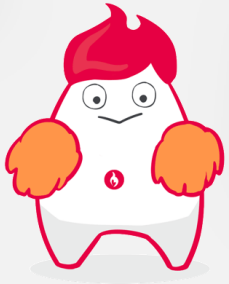
- python的第三方包管理工具，pip的使用率最高
- Python3.4以上版本在安装python的时候已经自带了这两种包管理工具
- 老版python 可通过 https://pip.pypa.io/en/stable/installing/
- pip install 包名
- github.com 搜索 python 第三方包

```
(python_catch) ➜  python_catch pip -V
pip 19.0.3 from /Users/zhangdewei/.virtualenvs/python_catch/lib/python3.8/site-packages/pip-19.0.3-py3.8.egg
/pip (python 3.8)
```

第一个第三方包 --ipython

python是一个python的交互式shell，比默认的python shell好用得多，支持变量自动补全，自动缩进

Python中的时间包1



本节课内容

- datetime包的常用功能
- datetime包中的常用方法
- python的常用时间格式化符号

认识datetime包

- 日期与时间的结合体-date and time
- 获取当前时间
- 获取时间间隔
- 将时间对象转成时间字符串
- 将字符串转成时间类型
- 时间戳转时间类型

获取当前时间

导入包与模块：

```
from datetime import datetime
import datetime
```

使用方法：

```
datetime.now()
datetime.datetime.now()     (today)
```

获取当前时间

```
In [1]: import datetime

In [2]: datetime.datetime.now()
Out[2]: datetime.datetime(2020, 3, 17, 15, 13, 28, 95562)
```

返回当前年月日时分秒毫秒的datetime对象

获取时间间隔

导入包：

```
from datetime import datetime
from datetime import timedelta
```

使用方法：

```
timeobj = timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, week=0)
```

获取时间间隔

```
In [6]: from datetime import datetime

In [7]: from datetime import timedelta

In [8]: before_one_day = timedelta(days=1)

In [9]: yestoday = datetime.now() - before_one_day

In [10]: yestoday
Out[10]: datetime.datetime(2020, 3, 16, 15, 16, 59, 727154)
```

返回时间间隔的datetime对象

时间对象转字符串

获取时间对象：

```
from datetime import datetime
now = datetime.datetime.now()
```

时间转字符串：

```
date_str = now.strftime(format)
```

时间对象转字符串

```
In [12]: from datetime import datetime

In [13]: date = datetime.now()

In [14]: str_date = date.strftime('%Y-%m-%d %H:%M:%S')

In [15]: str_date
Out[15]: '2020-03-17 15:19:27'
```

符合时间格式的字符串

时间字符串转时间类型

获取时间模块：

```
from datetime import datetime
```

时间字符串转时间类型：

```
datetime.strptime(tt, format)
```

参数介绍：

tt： 符合时间格式的字符串

format： tt时间字符串匹配规则

时间字符串转时间类型

```
In [20]: from datetime import datetime

In [21]: str_date = '2021-10-10 13:13:13'

In [22]: date_obj = datetime.strptime(str_date, '%Y-%m-%d %H:%M:%S')

In [23]: date_obj
Out[23]: datetime.datetime(2021, 10, 10, 13, 13, 13)
```

datetime 日期对象

时间格式字符1

字符	介绍
%Y	完成的年份，如 2020
%m	月份，1~12
%d	月中的某一天（1~31）
%H	一天中的第几个小时（24小时，00~23）

时间格式字符1

字符	介绍
%I	一天中的第几个小时（12小时，01~12）
%M	当前的第几分（00~59）
%S	当前分的第几秒（0~61） 闰年多占2秒
%f	当前秒的第多少毫秒

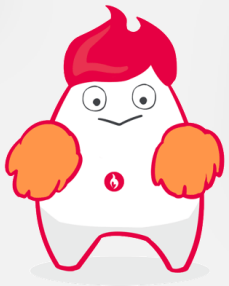
时间格式字符2

字符	介绍
%a	简化的星期， 如星期三 Wed
%A	完整的星期， 如星期三 Wednesday
%b	简化的月份，如二月 Feb
%B	完整的月份，如二月 February

时间格式字符2

字符	介绍
%c	本地日期和时间，如 Wed Feb 5 10:14:49 2020
%p	显示上午还是下午，如 AM代表上午，PM代表下午
%j	一年中的第几天
%U	一年中的星期数

Python中的时间包2



本节课内容

- 认识时间戳
- 认识python的time包与常用方法
- datetime包生成时间戳与时间戳转时间类型的方法

认识时间戳

- 1970年1月1日00时00分00秒至今的总毫秒(秒)数
- timestamp
- float

time包与他的函数们

- ◆时间处理，转换时间格式
- ◆生成时间戳函数 time
- ◆获取本地时间函数 localtime
- ◆localtime对应字段介绍
- ◆暂停函数 sleep
- ◆time中的strftime与strptime

生成时间戳函数 time

导入包：

import time

使用方法：

time.time()

返回值：

秒级别的浮点类型

举例：

1580878485.4009378

获取本地时间函数 localtime

导入包：

```
import time
```

使用方法：

```
time.localtime(timestamp)
```

参数介绍：

timestamp： 时间戳（可不传）

获取本地时间函数 localtime

```
In [1]: import time

In [2]: t = time.localtime()

In [3]: t
Out[3]: time.struct_time(tm_year=2020, tm_mon=3, tm_mday=17, tm_hour=15, tm_min=30, tm_sec=57, tm_wday=1, tm
_yday=77, tm_isdst=0)
```

localtime对应字段介绍

属性名	介绍	取值范围
tm_year	四位数年	2020
tm_mon	月	1~12
tm_mday	日	1~31
tm_hour	小时	0~23
tm_min	分钟	0~59

localtime对应字段介绍

属性名	介绍	取值范围
tm_sec	秒	0~61（依然是闰月问题）
tm_wday	一周的第几天	0~6（0是周一）
tm_yday	一年的第几日	1~366(儒略历)
tm_isdst	夏令时	-1,0, 1 是否是夏时令

暂停函数 sleep

导入包：

```
import time
```

使用方法：

```
time.sleep(second)
```

参数介绍：

second： 希望程序被暂停的秒数

time中的strftime

导入包：

```
import time
```

使用方法：

```
time.strftime(format, t)
```

参数介绍：

format: 格式化规范

t： time.localtime对应的时间类型

time中的strftime

```
In [6]: import time

In [7]: str_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())

In [8]: str_time
Out[8]: '2020-03-17 15:34:59'
```

time中的strptime

导入包：

```
import time
```

使用方法：

```
time.strptime(time_str, format)
```

参数介绍：

time_str: 符合时间格式的字符串

format: 确保与time_str 一致的格式化标准

time中的strptime

```
In [9]: import time

In [10]: time_obj = time.strptime('2020-12-12', '%Y-%m-%d')

In [11]: time_obj
Out[11]: time.struct_time(tm_year=2020, tm_mon=12, tm_mday=12, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=5, tm_
yday=347, tm_isdst=-1)
```

datetime中生成时间戳函数

导入包：

```
import datetime
```

使用方法：

```
now = datetime.datetime.now()
datetime.datetime. timestamp(now)
```

参数介绍：

now: datetime时间对象

datetime中生成时间戳函数

```
In [14]: import datetime

In [15]: now = datetime.datetime.now()

In [16]: now_timestamp = datetime.datetime.timestamp(now)

In [17]: now_timestamp
Out[17]: 1584430694.282982
```

秒级时间戳 浮点类

datetime时间戳转时间对象

导入包：

import datetime

使用方法：

datetime.datetime. fromtimestamp(timestamp)

参数介绍：

timestamp: 时间戳

返回值：

datetime的日期对象

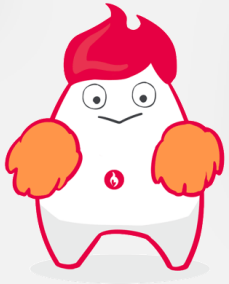
datetime时间戳转时间对象

```
In [18]: now_timestamp
Out[18]: 1584430694.282982

In [19]: datetime_obj = datetime.datetime.fromtimestamp(now_timestamp)

In [20]: datetime_obj
Out[20]: datetime.datetime(2020, 3, 17, 15, 38, 14, 282982)
```

Python的os包



本节课内容

- os的文件与目录函数介绍
- os.path模块常用函数介绍

os的文件与目录函数介绍

import os

函数名	参数	介绍	举例	返回值
getcwd	无	返回当前的路径	os.getcwd()	字符串
listdir	path	返回制定路径下所有的文件或文件夹	os.listdir(‘c://Windows’)	返回一个列表
makedirs	Path mode	创建多级文件夹	os.makedirs(‘d://imooc/py’)	无

os的文件与目录函数介绍

函数名	参数	介绍	举例	返回值
removedirs	path	删除多级文件夹	os.removedirs(‘d://imooc/py’)	无
rename	Oldname newname	给文件或文件夹改名	os.rename(‘d://imooc’ , ‘d://imoc’)	无
rmdir	path	只能删除空文件夹	os.rmdir(‘d://imooc’)	无

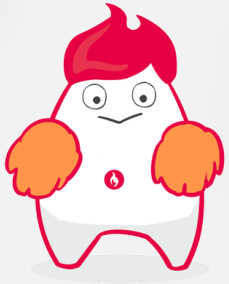
os.path模块常用方法

函数名	参数	介绍	举例	返回值
exists	Path	文件或路径是否存在	os.path.exists(‘d://’)	bool类型
isdir	Path	是否是路径	os.path.isdir(‘d://’)	bool类型
isabs	Path	是否是绝对路径	os.path.isabs(‘test’)	bool类型

os.path模块常用方法

函数名	参数	介绍	举例	返回值
isfile	Path	是否是文件	os.path.isfile(‘d://a.py’)	bool 类型
join	Path, path*	路径字符串合并	os.path.join(‘d://’ , ‘test’)	字符串
split	Path	以最后以层路径为基准切割	os.path.split(‘d://test’)	列表

Python中的sys包



sys中的常用方法

函数名	参数	介绍	举例	返回值
modules	无	Py启动时加载的模块	sys.modules()	列表
path	无	返回当前py的环境路径	sys.path()	列表
exit	arg	退出程序	sys.exit(0)	无
getdefaultencoding	无	获取系统编码	sys.getdefaultencoding()	字符串

sys中的常用方法

函数名	参数	介绍	举例	返回值
platform	无	获取当前系统平台	sys.platform()	字符串
version（属性）	无	获取python版本	sys.version	字符串
argv	*args	程序外部获取参数	sys.argv	列表