进击的小鸟

本节重点：

- 常用模块的加载方法
- 游戏编程的一些基本概念
- pygame的下载及使用
- 课上完成“进击的小鸟”游戏
- 课后，使用pyinstaller 打包发布

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，我们把很多函数分组，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）。

模块的作用?

- 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。
- 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

模块的分类?

- 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(pygame)
- 自定义模块

第三方开源软件的安全与使用?

https://pypi.python.org/pypi是python的开源模块库，截止2019年4.30日，已经收录了175, 870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。

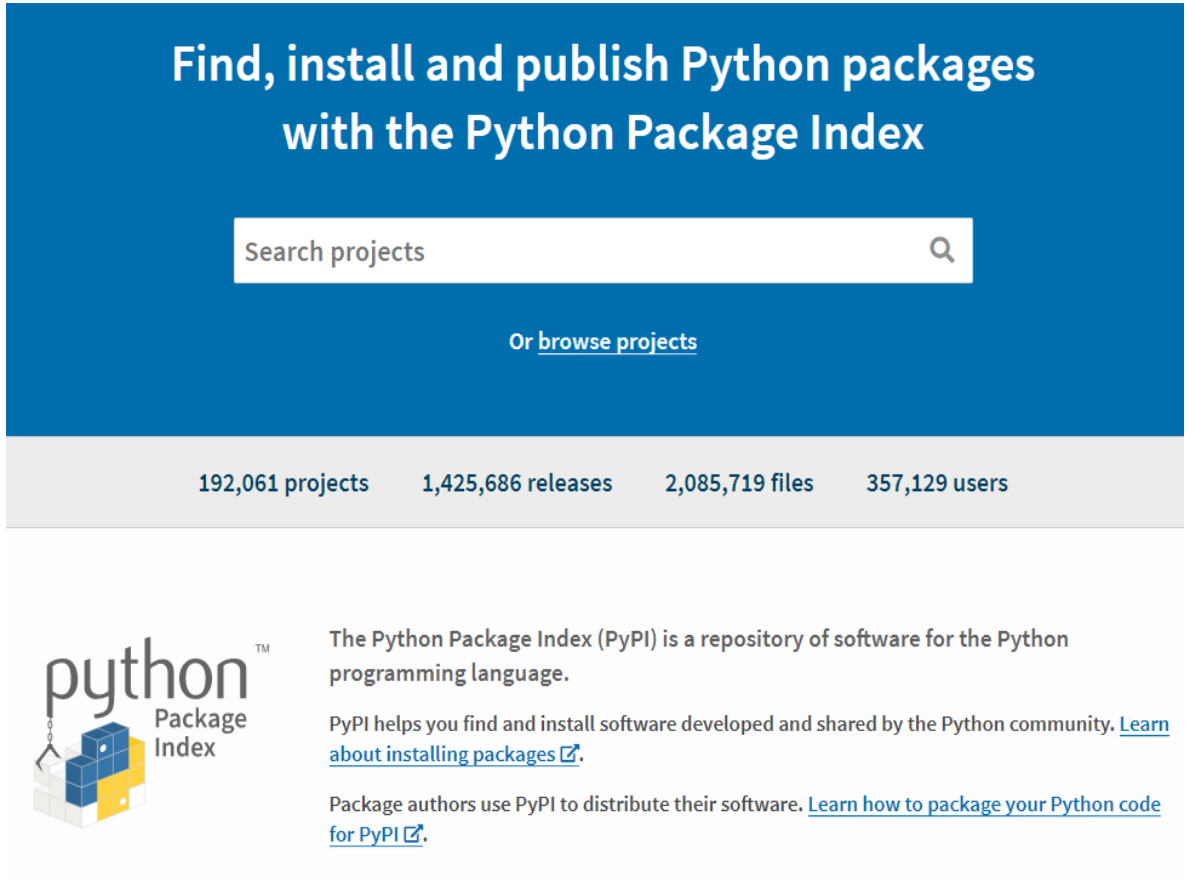
pip命令会自动下载模块包并完成安装。

软件安装目录： /your_python_lib_path/site-packages

默认连接国外python官方服务器，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

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

- -i 后面跟的是豆瓣源地址
- --trusted-host， 网站https安全验证

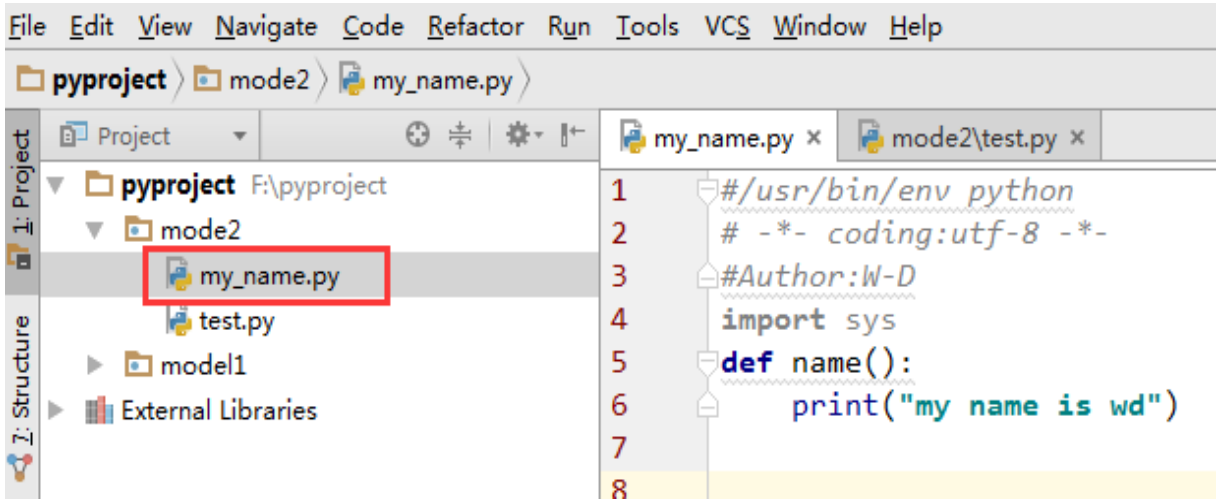


模块导入&调用

```
import module_a  # 导入
from module import xx
from module import xx as yy
from module.xx.xx import xx as rename  # 导入后重命令
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
```

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

自定义模块



如果不想在import时，执行某些程序，怎么办？（单元测试的代码）

```
if __name__ == '__main__':
    while True:
        idx = input("请选择序号>>")
        fun = func_dict.get(idx, None)
        fun()
```

当执行文件本身时候__name__变量等于main，此时判断成立并执行判断语句中的代码，当调用该模块的时候__name__并不等于__main__条件不成立，不执行判断下面的语句，可以认为为了调试模块，因为在模块导入的时候并不执行if下面的语句。

模块导入的查找路径

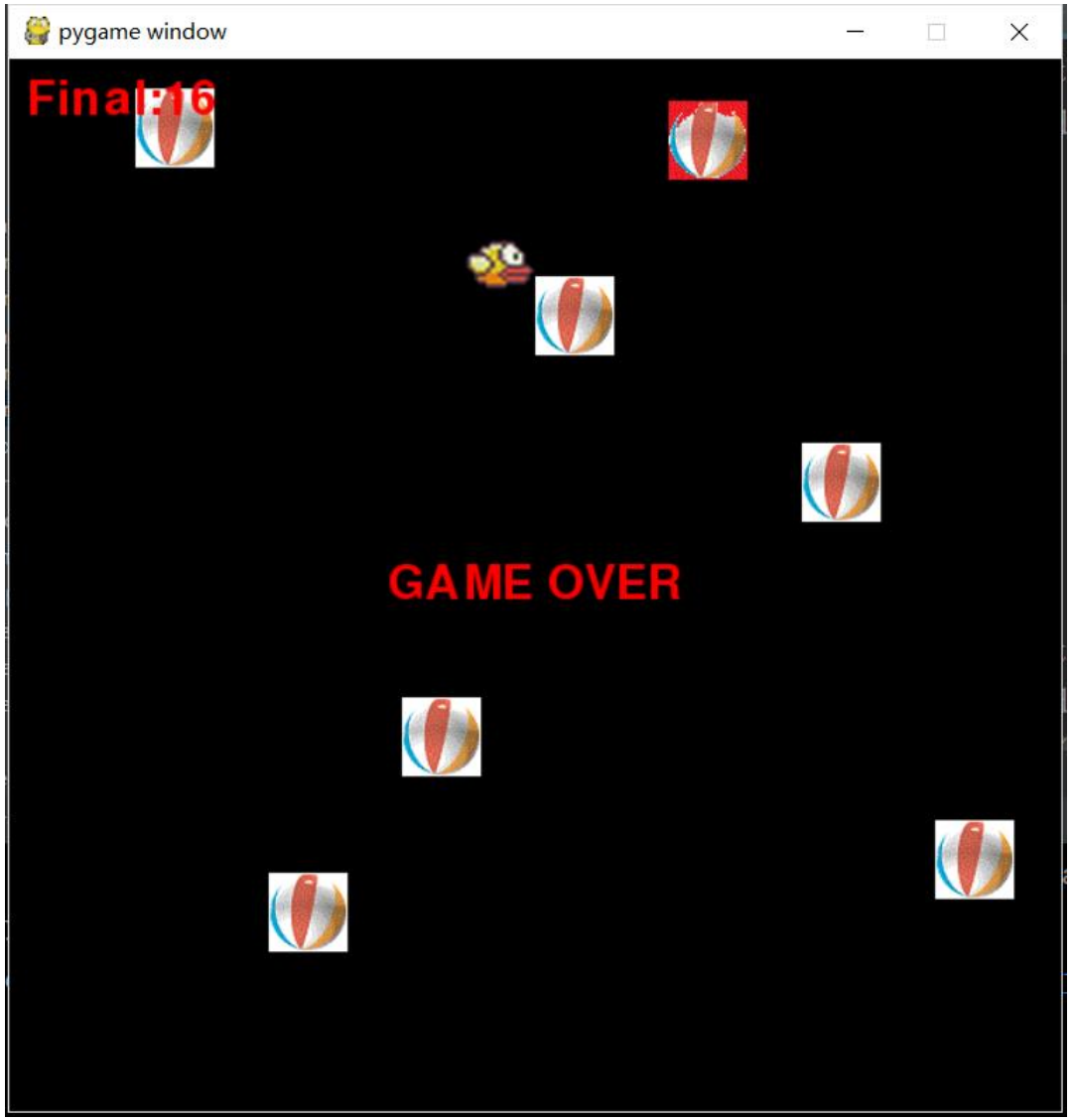
- 自己写的模块只能在当前路径下的程序里才能导入，换一个目录则导入报错，为什么？
- Python如何查找并自动定位第三方路径？
- 如何我定义的模块名称与系统路径冲突了，用哪个？

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

```
'D:\\python\\PycharmProjects\\WebDevelop\\pythontest', 'E:\\tools\\Python\\Python37\\DLLs',
'E:\\tools\\Python\\Python37\\lib', 'E:\\tools\\Python\\Python37', 'E:\\tools\\Python\\Python37\\lib\\site-packages',
'E:\\tools\\JetBrains\\PyCharm 2018.2.3\\helpers\\pycharm_matplotlib_backend'
```

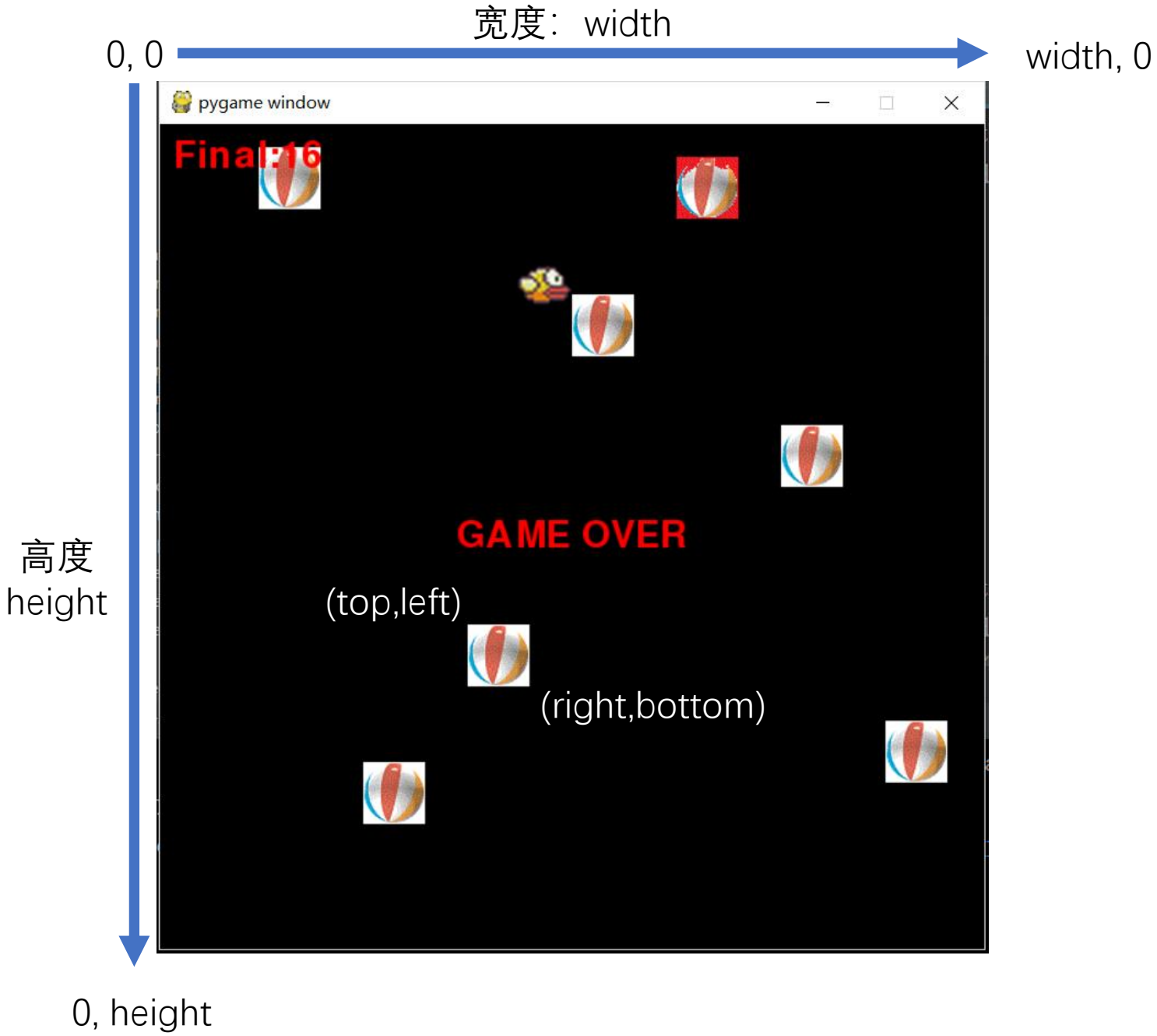
- 你导入一个模块时，Python解释器会按照上面列表顺序去依次到每个目录下去匹配你要导入的模块名，只要在一个目录下匹配到了该模块名，就立刻导入，不再继续往后找。
- 注意列表第一个元素为空，即代表当前目录，所以你自己定义的模块在当前目录会被优先导入。
- 若想在任何地方都能调用自创建模块，那就得确保你的模块文件至少在模块路径的查找列表中。
- 我们一般把自己写的模块放在一个带有“site-packages”字样的目录里，我们从网上下载安装的各种第三方的模块一般都放在这个目录。

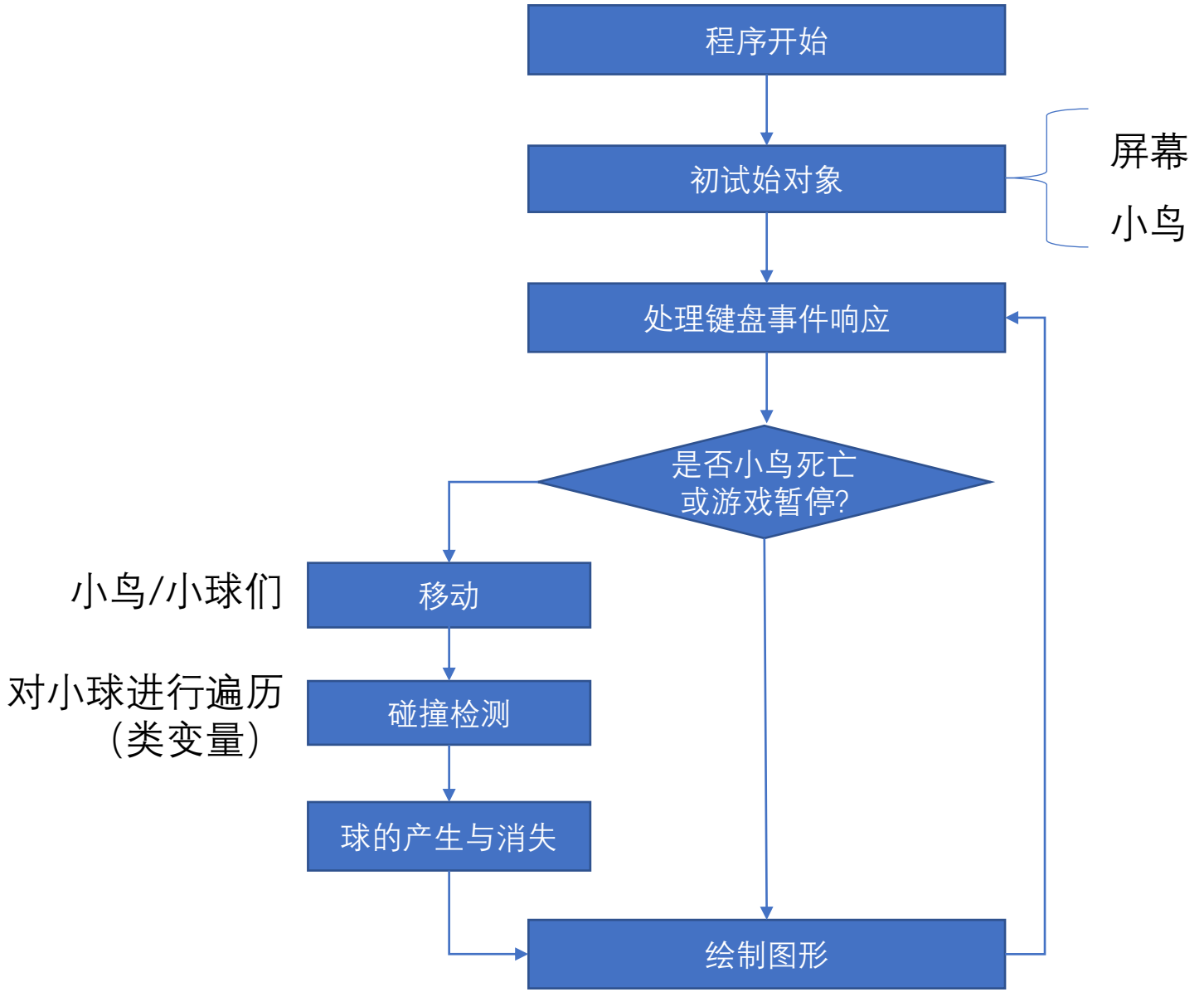
《进击的小鸟》功能需求



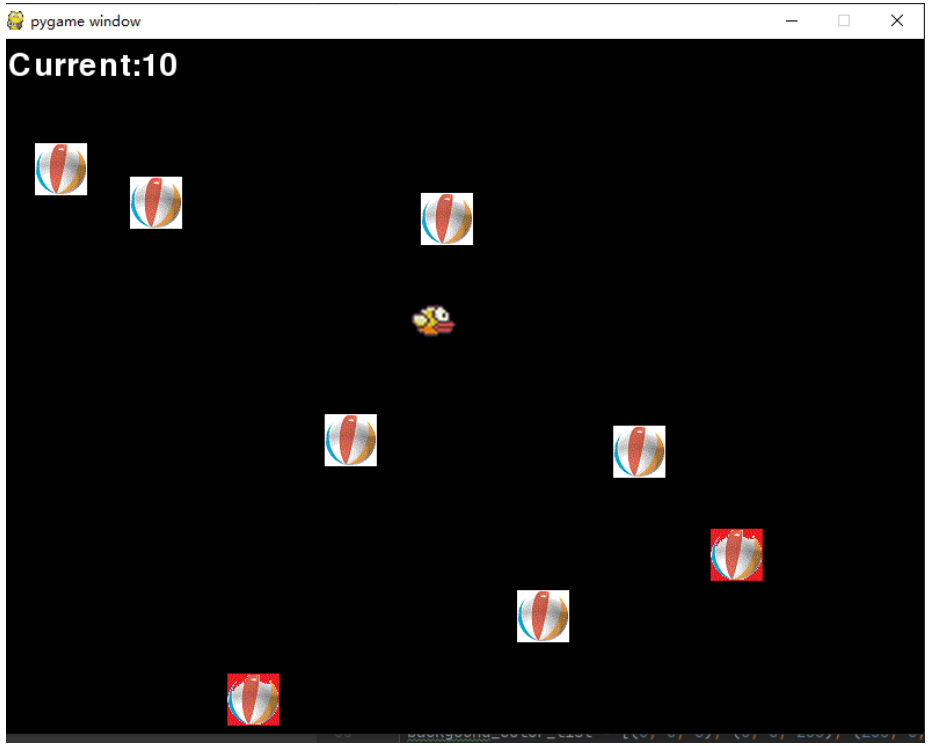
游戏组件	功能
画面屏幕	背景：黑色 尺寸：800 * 600
小鸟	键盘控制小鸟的前后上下。 小鸟跟随鼠标而移动
球	随机定时出现在屏幕上 从左向右移动
游戏得分条件	每躲过一个球，得1分
游戏结束条件	小鸟碰到白球
扩展条件	随机出现红包 小鸟吃红包，得10分

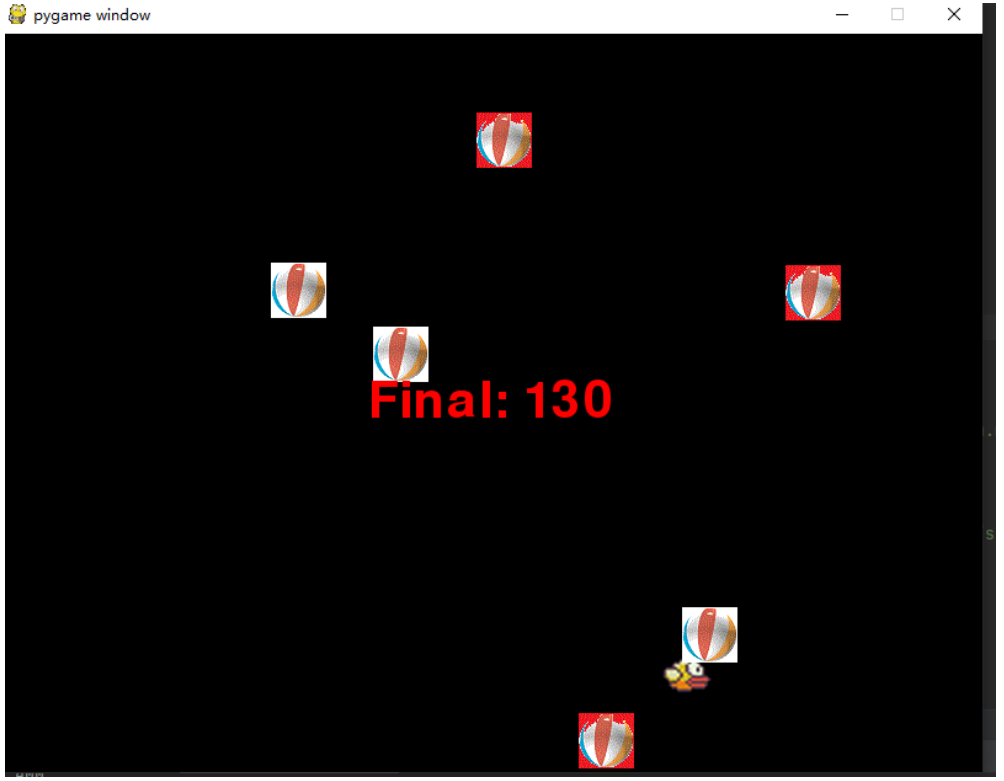
预习知识：


宽度：width
0, 0
width, 0
pygame window
Final:16
GAME OVER
高度
height
(top,left)
(right,bottom)
0, height


程序开始
初试始对象
屏幕
小鸟
处理键盘事件响应
是否小鸟死亡
或游戏暂停?
小鸟/小球们
移动
对小球进行遍历
(类变量)
碰撞检测
球的产生与消失
绘制图形

最终效果





第1步

- pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
- pip install pygame --trusted-host https://pypi.tuna.tsinghua.edu.cn/simple
- pip config set global.index-url https://pypi.douban.com/simple

```
"""
功能点：
1、显示指定大小的黑窗口
"""
```

```
"""
知识点：
1、加载pygame包
    import pygame

2、初始化
    pygame.init()
    width = 800
    height = 600
    screen = pygame.display.set_mode((width, height))

3、启动运行（阻塞式）
    while True:
        pass

4、退出游戏
    pygame.quit()
```

第2步

```
"""
功能点：
1、点击退出按钮，则窗口退出
2、按下空格键或回车键，则窗口退出。。
"""
```

```
"""
知识点：
1、对事件进行遍历。。。
    for event in pygame.event.get():
        pass

2、点击退出按钮事件。。。
   if event.type == pygame.QUIT:
        pass

3、键盘按钮事件。。。
   if event.type == pygame.KEYDOWN:
       if event.key == pygame.K_RETURN or event.key == pygame.K_SPACE:
            b_quit = True
            pass
"""
```

第3步

```
"""
功能点：
1、填充界面颜色
2、按下任意键，界面颜色发生变化。。。
"""
```

```
"""
知识点：
1、添加界面颜色
    screen.fill((0,0,0))
    pygame.display.flip()

2、响应键盘事件，修改页面颜色
    backgound_color_list = [(128, 128, 128), (0, 0, 255), (255, 0, 0), (0, 255, 0), (255, 255, 255)]
    backgound_color_idx = 0
    if event.type == pygame.KEYDOWN:
        backgound_color_idx += 1
        if backgound_color_idx >= len(backgound_color_list):
            backgound_color_idx = 0
    screen.fill(backgound_color_list[backgound_color_idx])
"""
```

第4步

```
"""
功能点：
1、添加小鸟
2、小鸟自动运行
3、小鸟全身在屏幕范围内。
4、按下SPACE键或回车键，则运动暂停。。。
5、暂停状态下，再次按下SPACE键或回车键，则运动回复。。。。
"""
```

```
"""
知识点：
1、添加小鸟
    bird = pygame.image.load('pic/bird.png')
    bird_rect = bird.get_rect()
    bird_rect.right = 100
    bird_rect.top = 100
    screen.blit(bird, bird_rect)

2、修改图片大小。。。
    pygame.transform.scale(image,(80,60))

2、让小鸟运行
    x_step = 1
    y_step = 1
    bird_rect.left += x_step
    bird_rect.top += y_step

3、自动修改小鸟运动轨迹
    if bird_rect.right >= width:
        x_step = -1.0
    if bird_rect.left <= 0:
        x_step = 1.0
    if bird_rect.bottom >= height:
        y_step = -1.0
    if bird_rect.top <= 0:
        y_step = 1.0

4、修改页面刷新频率
    tick = 100
    clock = pygame.time.Clock()
    while True:
        clock.tick(tick)
"""
```

第5步

```
"""
功能点：
1、当鼠标在小鸟图标上且被按下，小鸟随着鼠标的移动而移动。
"""
```

```
"""
知识点：
1、监测鼠标事件。。。
    鼠标按下：  pygame.MOUSEBUTTONDOWN
    鼠标弹起：  pygame.MOUSEBUTTONUP
    鼠标移动：  pygame.MOUSEMOTION

2、获取鼠标位置：
    x = event.pos

3、监测某一个点是否在矩形内：
    if bird_rect.collidepoint(x):
        b_mouse_press = True

"""
```

第6步

```
"""
功能点：
1、小鸟响应上下左右按键操作。。。
2、小鸟全身始终在屏幕范围之内。。。
"""
```

```
"""
知识点：
1、监测按键的事件。。。
    按下：  pygame.KEYUP
    弹起：  pygame.KENDOWN
    上：  pygame.K_UP
    下：  pygame.K_DOWN
    左：  pygame.K_LEFT
    右：  pygame.K_RIGHT

2、善于利用dict，降低代码量
    press_dict = {pygame.K_UP: (0, -1), pygame.K_DOWN: (0, 1), pygame.K_LEFT: (-1, 0), pygame.K_RIGHT: (1, 0)}
    if event.key in press_dict:
        press = press_dict[event.key]
        step[0] += press[0]
        step[1] += press[1]

    bird_rect = bird_rect.move(step)
"""
```

第7步

```
"""
功能点：
1、 以0.5秒为间隔，定期在屏幕最右侧生成小球,纵坐标位置随机
2、小球以-1的匀速，从左向右移动。
3、当小球移动到最左侧时，自动消失；
4、如果小鸟和小球发生了碰撞， 则游戏结束
5、游戏结束时，画面静止在游戏结束前一刻
6、游戏结束后， 如果有任何鼠标点击或按键操作，则退出游戏。
"""
```

```
"""
知识点：

1、利用 dict 来存储每个小球的信息（body，rect）
    ball = {"body": ball_body, "rect": ball_rect}

2、绘制多个小球。。。
    for ball in ball_list:
        screen.blit(ball['body'], ball['rect'])

3、 利用random包，来生成随机数。。
    if last_time < this_time - 0.5:
        last_time = this_time
        ball_body = pygame.image.load('pic/ball.png')
        ball_rect = ball_body.get_rect()
        ball_rect.right = width
        ball_rect.top = random.randint(0, height - ball_rect.height)
        ball = {"body": ball_body, "rect": ball_rect}
        ball_list.append(ball)

4、调用 colliderect，来检测两个矩形是否发生碰撞。。。
    b_collide = False
    for ball in ball_list:
        ball_rect = ball['rect']
        if bird_rect.colliderect(ball_rect):
            b_collide = True
            break
"""
```

第8步

```
"""
功能点：
1、游戏中，在左上角显示分值，字体是 白色 40号
2、游戏结束后，在屏幕正中央显示最终得到， 字体是红色 60号字
3、某小球在最左侧消失后，游戏得分+10
"""
```

```
"""
知识点：

1、文字初始化。。。
    pygame.font.init()  # 初始化字体


2、设置字体和大小：
   font = pygame.font.SysFont("Song", 40)  # 设置字体和大小

3、文字 -> 图片
    str_tip = 'Final: %s' % scores
    color_font = (255, 0, 0)
    bmp_gameover = font.render(str_tip, -1, color_font)

4、打印图片：
   screen.blit(bmp_gameover, (top, left))

"""
```

第9步

```
"""
功能：

1、20%的机率会产生红色。。。
    rand_val = random.randint(0, 10)
    is_red = rand_val <= 3
    if is_red:
        ball_body = pygame.image.load('pic/red_ball.png')
    else:
        ball_body = pygame.image.load('pic/ball.png')
    ball = {"body": ball_body, "rect": ball_rect, "red":is_red}


2、如果吃到了红包，则分值 + 40，如果吃到的不是红色，则game over
   for ball in ball_list:
    ball_rect = ball['rect']
    is_red = ball['red']
    if bird_rect.colliderect(ball_rect):
        if is_red:
            scores += 40
            ball_list.remove(ball)
        else:
            is_game_over = True

3、如果没有吃到红包，则 游戏速度 * 2
    tick *= 2
4、如果分值  < 0，则游戏结束
   if scores < 0:
        is_game_over = True

"""
```

```
"""
知识点：

1、如何生成红包，并区分红包与普通包。。。
    # 20%的几率为红包。。。
    rand_val = random.randint(0, 10)
    is_red = rand_val <= 3

    if is_red:
        ball_body = pygame.image.load('pic/red_ball.png')
    else:
        ball_body = pygame.image.load('pic/ball.png')

    ball_rect = ball_body.get_rect()
    ball_rect.right = width
    ball_rect.top = random.randint(0, height - ball_rect.height)
    ball = {"body": ball_body, "rect": ball_rect, "red":is_red}
"""
```

第10步

```
"""
功能：

1、循环播放背景音乐，直到游戏结束
2、吃到红球，有成功的声音特效。
3、碰到白球，有爆炸的声音特效。

"""
```

```
"""
知识点：

1、初始化声音。。。
    pygame.mixer.init()

2、加载背景声音。。。
    pygame.mixer.music.load("sound/bgm.mp3")
    pygame.mixer.music.play(-1,0)

3、添加特效声音：
    sound = pygame.mixer.Sound("sound/succeed.wav")
    sound.play()
"""
```

Python 打包 方法

https://blog.csdn.net/weixin_51111267/article/details/123599008

1、打包库安装

```
pip install pyinstaller
```

2、初步了解

```
1、打开cmd
2、输入【F: 】进入F盘
3、输入【cd cs01】进入F盘的cs01文件夹（里面存放着要打包的xxx.py文件）
4、pyinstall -F xxx.py （直接打包成exe文件，没图标ico，会有黑色窗口）
5、pyinstaller -F -i test.ico test.py （打包成exe文件，带有test.ico图标，会有黑色窗口）
6、pyinstaller -w -i aa.ico -F aaa.py （打包成exe文件，带aa.ico图标，没有黑色窗口）
7、pyinstall -F xxx.py -p D:\MyApp\Python3.8\Lib\site-packages
# -p后面为搜寻模块的地址（一般为python模块库的地址），想要使用自己的工程文件（照片、视频）要和ex
```

3、我最终使用

```
pyinstaller -w -i crab.ico -F test.py -p D:\MyApp\Python3.8\Lib\site-packages
```

> *调用自己建的py文件要放在和打包py文件同一个文件夹

Python模块

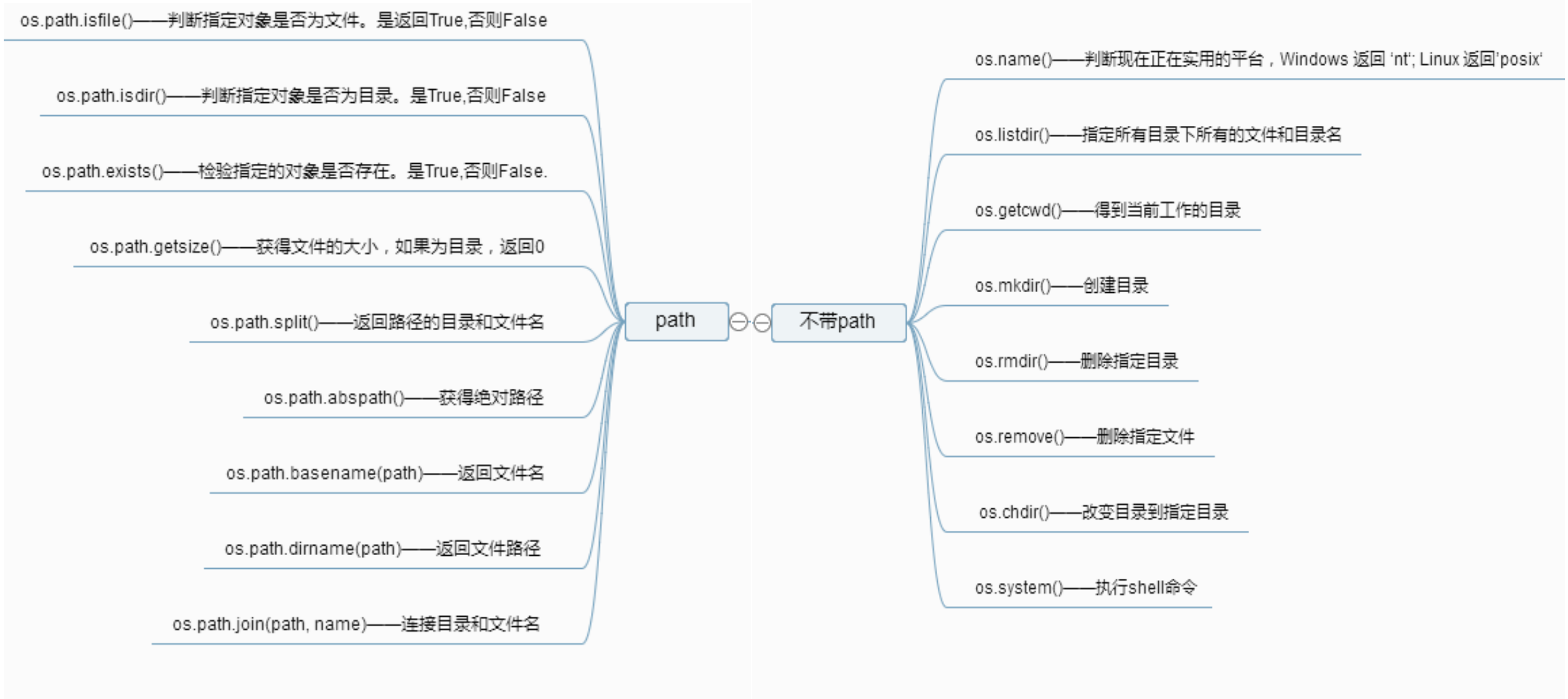
本节重点：

- 常用内置模块
- 基础语法归纳

常用内置模块

模块名称	作用
os	允许你的程序与操作系统直接交互的功能
sys	提供对解释器使用或维护的一些变量的访问，以及与解释器交互的函数
time & datetime	时间/日期
random	生成随机数
json/pickle	序列化
hashlib	hash加密
re	正则表达式
math	数学运算库

os模块


os.path.isfile()——判断指定对象是否为文件。是返回True,否则False
os.path.isdir()——判断指定对象是否为目录。是True,否则False
os.path.exists()——检验指定的对象是否存在。是True,否则False.
os.path.getsize()——获得文件的大小，如果为目录，返回0
os.path.split()——返回路径的目录和文件名
os.path.abspath()——获得绝对路径
os.path.basename(path)——返回文件名
os.path.dirname(path)——返回文件路径
os.path.join(path, name)——连接目录和文件名
path
不带path
os.name()——判断现在正在实用的平台，Windows 返回 'nt'; Linux 返回'posix'
os.listdir()——指定所有目录下所有的文件和目录名
os.getcwd()——得到当前工作的目录
os.mkdir()——创建目录
os.rmdir()——删除指定目录
os.remove()——删除指定文件
os.chdir()——改变目录到指定目录
os.system()——执行shell命令

sys模块

函数	功能
sys.argv	实现从程序外部向程序传递参数
sys.exit([arg])	程序中间的退出，arg=0为正常退出
sys.getdefaultencoding()	获取系统当前编码，一般默认为是utf-8
sys.setdefaultencoding()	设置系统默认编码
sys.getfilesystemencoding()	获取文件系统使用编码方式，Windows下返回’mbcs’，mac下返回’utf-8’
sys.path：	获取指定模块搜索路径的字符串集合，可以将写好的模块放在得到的某个路径下，就可以在程序中import时正确找到。
sys.platform：	获取当前系统平台
sys.stdin sys.stdout sys.stderr：	stdin，stdout和stderr变量包含与标准I/O流对应的流对象。如果需要更好的控制输出，而print不能满足你的需求，他们就是你所需要的，你也可以替换他们，这时候你就可以重定向输出和输入到其他设备，或者以非标准的方式处理他们。

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

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”

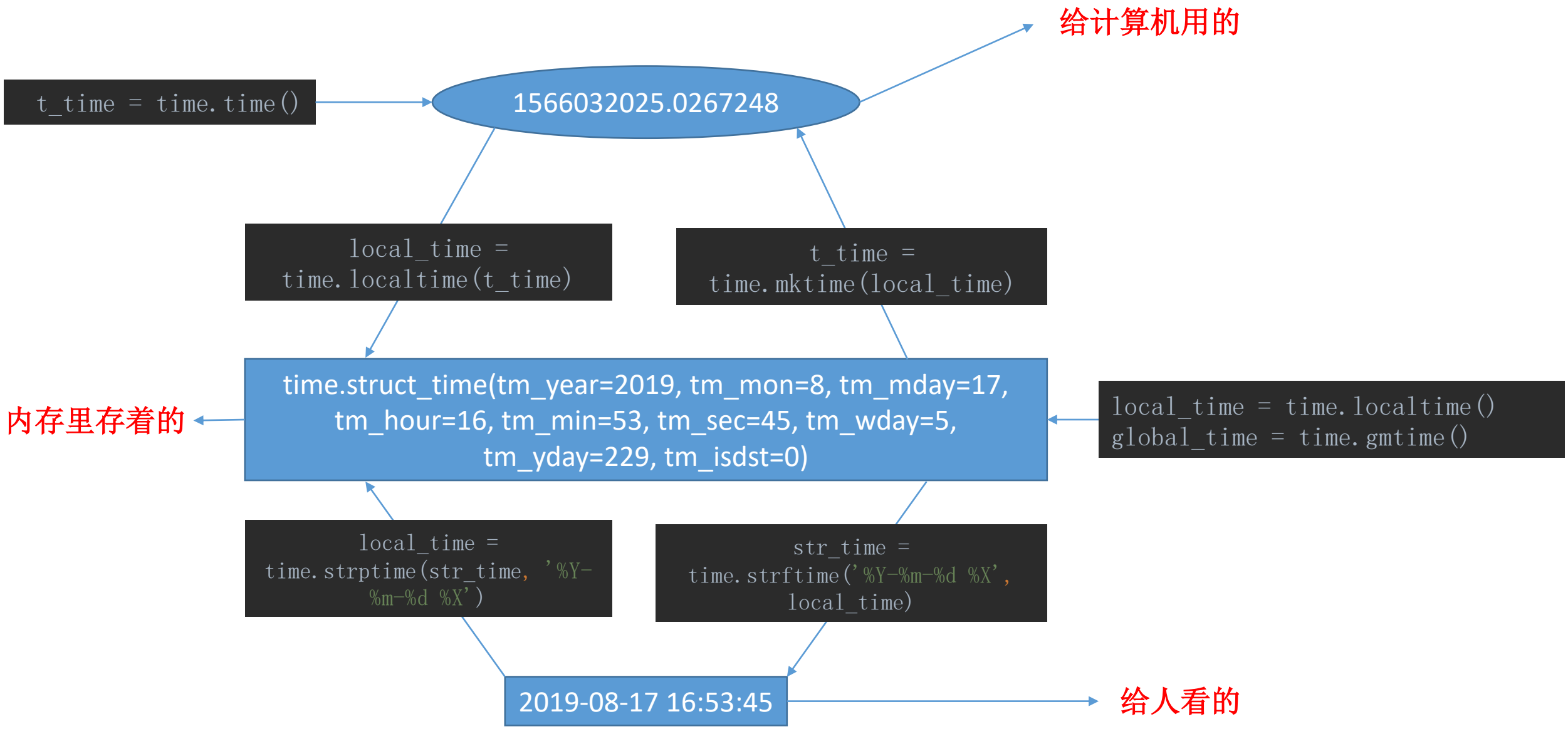
3.元组（struct_time）共九个元素。由于Python的time模块实现主要调用C库。

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

函数	功能
time.localtime(t)	将时间戳转换为当前时区的struct_time。则以当前时间为准。
time.gmtime(t)	和localtime()方法类似，将一个时间戳转换为UTC时区（0时区）
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.asctime()	把一个表示时间的元组或者struct_time表示为默认形式
time.ctime([secs])：	把一个时间戳（按秒计算的浮点数）转化为time.asctime()的形式
time.strftime	把一个代表时间的元组或者struct_time转化为格式化的时间字符串
time.strptime(string[, format])：	把格式化时间字符串转化为struct_time，是strftime()是逆操作
time.perf_counter / process_time	用于以浮点数计算的秒数返回当前的 CPU (处理器)时间

time模块的使用方法



datetime模块

相比于time模块，datetime模块的接口则更直观、更容易调用，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。

```
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


datetime.date.fromtimestamp(322222)
#把一个时间戳转为datetime日期类型
```

```
时间运算
d+ datetime.timedelta(4)  #当前时间 +4天
d + datetime.timedelta(hours=4) #当前时间+4小时


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

日期练习

1. 写一个定时程序，以1秒为周期， 周期输出当前时间，输出格式：“2021-8-21 00:00:00”
2. 计算1年后的当日当时当刻，与此时此刻的 时间差（秒数）， 并按照格式输出：

   1年后的时刻是“2022-9-2 12:22:22”，与现在相比，时间差为212121121.222秒

3. 编写一个函数：

   （1）生成1个给定长度n的自然数列表li = [1,…,n]

   （2） 等待1秒。

   （3）返回1个元组，(函数用时,占用cpu时间,li)

```python
t0 = time.time()
t1 = t0 + 365 * 86400
if time.localtime(t0).tm_mday != time.localtime(t1).tm_mday:
    t1 += 86400
print(f'1年后的时刻是“{time.strftime("%Y-%m-%d %X",time.localtime(t1))}”，与现在相比，时间差为{t1-t0}秒')
```

```python
import time
while True:
    t_time = time.time()
    local_time = time.localtime(t_time)
    str_time = time.strftime('%X',  local_time)
    print(f'【{str_time}】')
    time.sleep(1)
```

```python
def func_test(n):
    t0 = time.time()
    t1 = time.process_time()
    li = [i for i in range(1, n+1)]
    time.sleep(1)
    return (time.time() - t0, time.process_time() - t1, li)

print(func_test(100))
```

random模块

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

```
import random
import string

# 随机整数:
random.randint(1, 50)

# 随机选取0到100间的偶数:
random.randrange(0, 101, 2)

# 随机浮点数:
random.random()
random.uniform(1, 10)

# 随机字符:
random.choice('abcdefghijklmnopqrstuvwxyz!@#$%^&*()')

# 多个字符中生成指定数量的随机字符:
random.sample('zyxwvutsrqponmlkjihgfedcba', 5)
```

```
# 从a-zA-Z0-9生成指定数量的随机字符:
ran_str = ''.join(random.sample(string.ascii_letters + string.digits, 8))
ran_str

# 多个字符中选取指定数量的字符组成新字符串:
''.join(random.sample(
    ['z', 'y', 'x', 'w', 'v', 'u', 't', 's', 'r', 'q', 'p', 'o', 'n', 'm', 'l', 'k', 'j', 'i', 'h', 'g', 'f', 'e', 'd',
     'c', 'b', 'a'], 5))

# 随机选取字符串:
random.choice(['剪刀', '石头', '布'])

# 打乱排序
items = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
random.shuffle(items)
```

json模块

序列化是一个用于将对象状态转换为字节流的过程，可以将其保存到磁盘文件中或通过网络发送到任何其他程序；从字节流创建对象的相反的过程称为**反序列化**。而创建的字节流是与平台无关，与语言无关，在一个平台上序列化的对象可以在不同的平台上反序列化。

在很多应用中，需要对某些对象进行序列化，让它们离开内存空间，入住物理硬盘，以便长期保存。比如最常见的是Web服务器中的Session对象，当有 10万用户并发访问，就有可能出现10万个Session对象，内存可能吃不消，于是Web容器就会把一些seesion先序列化到硬盘中，等要用了，再把保存在硬盘中的对象还原到内存中。

当两个进程在进行远程通信时，彼此可以发送各种类型的数据。无论是何种类型的数据，都会以二进制序列的形式在网络上传送。发送方需要把这个对象转换为字节序列，才能在网络上传送；接收方则需要把字节序列再恢复。

```
import json
data = {'k1':123,'k2':'Hello'}
# 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)
```

json练习

课堂练习：用户登录认证程序：

- 从json文件里读用户和密码（以及其他信息）信息。
  - 文件 -> json -> 内存dict
  - 需要考虑异常判断，如果json文件不存在怎么办？ 自己定义一个users， users-> json -> file， 再去读文件。
  - users=[{“name”:”p1”, “pwd”:”ppp”, “status”:1}, {“name”:”p2”, “pwd”:”ddd”, “status”:1}, {“name”:”p3”, “pwd”:”333”, “status”:1}]
- 输入用户
  - 如果用户已经被锁定，则提示，并直接退出。
  - 如果用户没有被锁定，则继续输入密码。
- 输入密码：
  - 输入成功， 则打印登录成功，并退出。
  - 输入失败， 则重试。
  - 如果重试次数 超过 3次， 则 锁定用户（ user[‘status’] = 0 -> json -> 文件）

hashlib模块

- Hash，一般翻译做“散列”，也有直接音译为”哈希”的，就是把任意长度的输入（又叫做预映射，pre-image），通过散列算法，变换成固定长度的输出，该输出就是散列值。这种转换是一种压缩映射，也就是，散列值的空间通常远小于输入的空间，不同的输入可能会散列成相同的输出，而不可能从散列值来唯一的确定输入值。
- 简单的说就是一种将任意长度的消息压缩到某一固定长度的消息摘要的函数。
- HASH主要用于信息安全领域中加密算法，他把一些不同长度的信息转化成杂乱的128位的编码里,叫做HASH值.也可以说，hash就是找到一种数据内容和数据存放地址之间的映射关系

MD5讯息摘要演算法（英语：MD5 Message-Digest Algorithm），一种被广泛使用的密码杂凑函数，可以产生出一个128位的散列值（hash value），用于确保信息传输完整一致。MD5的前身有MD2、MD3和MD4。

特点

1.压缩性，输出固定。

2.容易计算。

3.抗修改性。

4.强抗碰撞。

用途

1.防止被篡改（电子支票 or 文件下载）：

2.防止直接看到明文(密码)：

3.防止抵赖（数字签名）：

hashlib模块

用于加密相关的操作，3.x里用hashlib代替了md5模块和sha模块，主要提供 SHA1, SHA224, SHA256, SHA384, SHA512 ， MD5 算法

```
import hashlib
# md5
m = hashlib.md5()
m.update( “Hello” .encode ( ‘utf-8’ ))
m.update(b"It's me")
print(m.digest())  # 返回2进制格式的hash值
m.update(b"It's been a long time since last time we ...")
print(m.hexdigest()) # 返回16进制格式的hash值
```

思考：网盘的实现原理？

https://www.cmd5.com/

```
# sha1
s1 = hashlib.sha1()
s1.update("小猿圈".encode("utf-8"))
s1.hexdigest()
# sha256
s256 = hashlib.sha256()
s256.update("小猿圈".encode("utf-8"))
s256.hexdigest()
# sha512
s512 = hashlib.sha256()
s512.update("小猿圈".encode("utf-8"))
s512.hexdigest()
```

hashlib练习

课堂练习2：用户注册/登录的程序

1、用户通过界面选择注册 or 登录 or 退出。

2、如果是注册， 输入 用户名称(str)、密码(str)、确认密码、年龄(int)、收入(float)等信息，并存储到文件，

要求：

（1）用户名称不能重复。

（2）以json格式存储账户信息。

（3）以md5码存储用户密码。

3、如果是登录，类似原来课堂作业的要求：

（1）判断账户状态，如果是锁定，则直接提示“账户已经锁定” 并退出程序。

（1）如果登录成功，则提示“登录成功”并退出。

（2）给3次登录验证机会，如果连续3次登录失败，则锁定账户。

单账户 {} -》 多账户 [{},{},{},{},{}]。

re正则表达式

正则表达式就是字符串的匹配规则，在多数编程语言里都有相应的支持，python里对应的模块是re

4大功能：

```
# 1. match 从头匹配字符串，从头开始找，匹配所有满足要求的最长字符串，
# s='123abc '
# val = re.match("[0-9a-zA-Z]+", s)
```

```
# 2. search 搜索字符串，从已有字符串，搜索出1个满足要求的字符串
# val = re.search("[0-9a-zA-Z]+", s)
# print(val)
```

```
# 3. findall，  搜索字符串，从已有字符串，搜索出所有满足要求的字符串
# val = re.findall("[0-9a-zA-Z]+", s)
# print(val)
```

```
# 4. split，  分割字符串, 分隔的字符串会被去掉
# val = re.split("[+-]", s)
```

```
# 4. split，  分割字符串，保留分隔的字符串
# val = re.split("([+-])", s)
```

re正则表达式

针对字符的模糊匹配

```
# 任意一个字符：  .
            # print(re.match('.', ' +[ba12'))
# 数字(0-9)/非数字: \d \D
            # print(re.match('\d', '12abc’))
            # print(re.match('\D', 'a12bc'))
# 非特殊字符，即a-z、A-Z、0-9、_、汉字  与  特殊字符(+-*/以及其他特殊符号包括.): \w \W
            # print(re.match('\w', '_中国’))
            # print(re.match('\W', '=-='))
# 空白符/非空白符：  \s \S
            # print(re.match('\s', '  ‘))
            # print(re.match('\S', '_+-='))
```

re正则表达式

针对字符/字符串数量的模糊匹配

```
# 0个或多个： *
            # print(re.match('.*', ' +[ba12'))
# # 0个或1个：  ?
            # print(re.match('.?', '+[ba12’))
            # print(re.match('\d', 'abc123+[ba12'))
# # >= 1个：   +
            # print(re.match('\d+', '123+[ba12'))
# # 指定数量： {m}
            # print(re.match('\d{2}', '123+[ba12’))
            # print(re.match('\d{2}', '1+[ba12'))
# # 指定区间: {m, n}
            # print(re.match('\d{1,3}', '+[ba12’))
            # print(re.match('\d{1,3}', '1+[ba12’))
            # print(re.match('\d{1,3}', '1234+[ba12'))
```

re正则表达式

不同字符/字符串的选择功能

```
# 分隔符：|
# 字符串的选择: (abcd|1234)
            # print(re.match('(12|ab)', '1234+[ba12’))
            # print(re.match('(12|ab)', 'ab1234+[ba12’))
            # print(re.match('(12|ab)', '1a2b34+[ba12'))
# 字符的选择:[abcd1234]
            # print(re.match('1|2', '1234+[ba12’))
            # print(re.match('2|3', '1234+[ba12'))
# 多字符的选择:[abcd1234]
            # print(re.match('[\d+]+', '1234+[ba12’))
            # print(re.match('[\d|+]+', '1234+[ba12'))
# 排除式选择： ^
             # print(re.match('[^\d]+', '+1+1234+[ba12'))
```

re正则表达式

支持不同的匹配位置

```
# 开头：^，  从第1个字符开始查找
            # print(re.search('^[a-zA-Z]+', 'abcd1234+[ba12’))
            # 等价于
            # print(re.match('[a-zA-Z]+', 'abcd1234+[ba12'))

# 结尾: $ 从最后一个字符向前查找.
            # print(re.search('[a-zA-Z]+', '1234+[ba12’))
            # print(re.search('[a-zA-Z]+$', '1234+[ba12'))
```

re正则表达式

- 验证手机号是否合法
  - 全部是数字 \d
  - 13位 \d{13}
  - 开头是(13|18)
- 验证邮箱是否合法
  - a@b
  - a: 字母数字或中划线下划线 [a-z|0-9|+-|*_]
  - b: sina.com.cn|.cn|.com|.edu|.edu.cn ([\w]+\.)+(cn|edu|edu.cn|com|)+
- 验证邮箱是否合法
  - 从字符串“<title>这是一个标签</title>”中， 提取到字符串“这是一个标签”

math数学运算库

函数	含义
ceil	取大于等于x的最小的整数值
copysign	把y的正负号加到x前面，可以使用0
cos	求x的余弦，x必须是弧度
degrees	把x从弧度转换成角度
e	表示一个常量
exp	返回math.e,也就是2.71828的x次方
expm1	返回math.e的x(其值为2.71828)次方的值减 1
fabs	返回x的绝对值
factorial	取x的阶乘的值
floor	取小于等于x的最大的整数值，如果x是一个整数，则返回自身
fmod	得到x/y的余数，其值是一个浮点数

math数学运算库

函数	含义
frexp	返回一个元组(m,e),其计算方式为：x分别除0.5和1,得到一个值的范围
fsum	对迭代器里的每个元素进行求和操作
gcd	返回x和y的最大公约数
hypot	如果x是不是无穷大的数字,则返回True,否则返回False
isfinite	如果x是正无穷大或负无穷大，则返回True,否则返回False
isinf	如果x是正无穷大或负无穷大，则返回True,否则返回False
isnan	如果x不是数字True,否则返回False
ldexp	返回x*(2**i)的值
log	返回x的自然对数，默认以e为基数，base参数给定时，将x的对数返回给定的base,计算式为：log(x)/log(base)
log10	返回x的以10为底的对数
log1p	返回x+1的自然对数(基数为e)的值

math数学运算库

函数	含义
log2	返回x的基2对数
modf	返回由x的小数部分和整数部分组成的元组
pi	数字常量，圆周率
pow	返回x的y次方，即x**y
radians	把角度x转换成弧度
sin	求x(x为弧度)的正弦值
sqrt	求x的平方根
tan	返回x(x为弧度)的正切值
trunc	返回x的整数部分

课堂练习

1. 输出未来1年内的所有周日所对应日期字符串的列表，比如：['2021-09-05', '2021-09-12'...]。
2. 用两种方式来判断文件是否相同：(1)直接内存对比，(2)通过hashlib的方式。
3. 用户输入目录名称，以列表形式显示该目录下的所有文件与文件夹信息。
4. 写一个6位随机验证码程序（使用random模块)，要求验证码中至少包含一个数字、一个小写字母、一个大写字母.

基本用法

知识点分类	
基本语法	■ 安装过程、调用方法、IDE环境、注释、对齐
变量	■ 掌握变量命名语法、命名习惯 ■ 变量使用过程 ■ 基本变量及使用
流程控制	■ if else / while / for / range，以及与 c的区别 ■ and / or / not ■ 三元运算赋值
输入输出	■ 格式化输入/输出 ■ 文件IO
复杂变量	■ list/ tuple / str / dict / set ■ 熟练掌握使用方法
列表生成式	■ 基本语法 ■ 熟练掌握

常见问题-流程控制

```
########################################################################
# 问题1：区别与 c的  for 的一种方法， for ... else....
########################################################################
# 需求：  从一个列表中找某一个数，如果找到了做什么，没有找到则做什么。
# 常规做法：
# flag = False
# for i in range(5):
#     if i == 5:
#         flag = True
#         break
# if flag:
#     print("找到了，执行找到后的处理语句")
# else:
#     print("没有找到，执行没有找到的处理语句")

# python 的推荐用法：
# for i in range(10):
#     print(i)
#     if i > 5:
#         print("i > 5, 在此处break,则不会执行 for 相连的 else")
#         break
# else:
#     print("for 循环正常结束, for 相连的else语句会被调用后")
```

常见问题-流程控制

```
#########################################################################
# 问题1：区别与 c的 for 的一种方法， for ... else....
#########################################################################
# 新的需求： 我需要检测输入字符串(choice)是否在某一个列表(li)中。

# 方法1:
# find_flag = False
# for ele in li:
#     if ele == choice:
#         find_flag = True
#         break
# if find_flag:
#     print("找到了")
#     break

# 方法2：
# for ele in li:
#     if ele == choice:
#         break
# else:
#     continue
# print("找到了")

# 方法3
# if choice in li:
#     print("找到了")
```

常见问题-文件操作

```
# 1. open了，没有 close
# f = open('aa.txt', 'w', encoding='utf-8')
# (f.read().)

# ‘cc’从用户态内存 -》 内核态内存。
# f.write("cc")

# 内核态内存 -> 硬盘:
# 1、f.close()
# 2、f.flush()
# 3、写大量内容，超过内核缓存。

# 2. with 帮你做close()， 不需要程序再显示调用close()
#     with open('aa.txt', 'r') as f:
#         f.write("cc")
#
# with open('aa.txt', 'r', encoding='utf-8') as f:
#     data = f.read()
#     # print(data)
```

常见问题-文件操作

```
# 如何取出一行？
# 如何判断结束？
# 为啥会多出一个空行？
# with open('aa.txt', 'r', encoding='utf-8') as f:
#     while True:
#         data = f.readline()
#         # print(len(data))
#         if not data:
#             break
#         print(data, end='')
```

```
# with open('aa.txt', 'r', encoding='utf-8') as f:
#     while True:
#         data = f.read(1)
#         print(len(data))
#         if not data:
#             break
#         print(data)
```

常见问题-文件操作

```
# 如何取出一行？
# 如何判断结束？
# 为啥会多出一个空行？
# with open('aa.txt', 'r', encoding='utf-8') as f:
#     while True:
#         data = f.readline()
#         # print(len(data))
#         if not data:
#             break
#         print(data, end='')
```

```
# readlines：一次性取出所有数据
# # 按照\n来做split，形成列表。
# with open('aa.txt', 'r', encoding='utf-8') as f:
#     data = f.readlines()
#     print(data)
```

常见问题-复杂变量

```
#########################################################################
# 问题2：从 dict中，  取出某key-value对的几种方法：
#########################################################################
# 方法1：  value = dict[key]
# 方法2：
#          value = None
#          if key in dict:
#              value = dict[key]
# 方法3：
#       value = dict.get(key, default_value)
```

常见问题-复杂变量

```
########################################################################
# 问题3：如何把某个字典插入list？
########################################################################
# 方法1：
# li = [{"name":'eric',"age":18},{"name":'amao',"age":11},{"name":'jess',"age":10}]
# 方法2：
# li = []
# eric = {"name":'eric',"age":18}
# li.append(eric)
# amao = {"name":'eric',"age":18}
# li.append(amao)
# li[0]['name'] = 'cc'
# print(li[1]['name'])

# amao = {"name":'amao',"age":11}
# li.append(amao)
# jess = {"name":'jess',"age":10}
# li.append(amao)
```

常见问题-复杂变量

```
########################################################################
# 问题4：对嵌套容器中元素的修改
########################################################################

# 提问1：  以下操作后，li[1]['name'] = ?
# li = [{"name":'eric',"age":18},{"name":'amao',"age":11},{"name":'jess',"age":10}]
# amao = li[1]
# amao['name'] = "new_amao"
#

# 提问2：  以下操作后，li[1]['name'] = ?
# li = [{"name":'eric',"age":18},{"name":'amao',"age":11},{"name":'jess',"age":10}]
# amao = li[1]
# amao = {'name':'new_amao',"age":33}
```

常见问题-复杂变量

```
########################################################################
# 问题4：对嵌套容器中元素的修改
########################################################################

# 提问3：  li[2]['name'] = ?
# li = []
# eric = {"name":'eric',"age":18}
# for i in range(10):
#     li.append(eric)
# li[1]['name'] = 'cc'
# print(li[2]['name'])

# 提问4：  li[2]['name'] = ?
# li = []
# for i in range(10):
#     li.append({"name":'eric',"age":18})
# li[1]['name'] = 'cc'
# print(li[2]['name'])

# a = max(1,5)
# print(a)
```

列表生成式

https://blog.csdn.net/weixin_43936969/article/details/103721875

- 列表生成式：python内置非常简单却强大的可以用来创建list的生成式，列表生成式也可以叫做列表解析。
- 列表生成式的格式：[expression for i in 序列 if …] == 表达式+循环+条件运用列表生成式，可以写出非常简洁的代码。
- 一般情况下循环太繁琐，而列表生成式则可以用一行语句代替多行循环生成列表。

1、成一个列表,列表元素分别为[1 ** 1,2 ** 2,...,9 ** 9]

```
import math
li = []
for i in range(1, 10):
    li.append(i ** i)
print(li)
print([i ** i for i in range(1, 10)])
```

列表生成式

```
print([i ** i for i in range(1, 10)])
print([i_** i for i in range(1, 10) if i % 2 == 0])
```

2、找出1～10之间的所有偶数

```
print([i for i in range(1, 11) if i % 2 == 0])
```

6.列表的字符串的大写改成小写，不是字符串的去掉

```
li = ['hello','World',24,24,41,7,8,False,'Apple']
print([s.lower() for s in li if isinstance(s,str)])
```

作业

作业：购物车
需求分析：
1、启动程序后，分别从文件加载商品(goods.txt)和用户信息(user.txt)
2、输入用户名密码后，检查用户密码是否正确，不正确重复登录，3次失败，则退出程序。
3、用户密码核实后，支持用户选择以下操作代码：（查看商品 (g)、查看余额 (u)、购买商品(b) 、退出购买(q) ）
    3.1、输入q，则退出。
    3.2、输入g，则查看所有商品信息。
    3.3、输入u，则查看用户余额信息。
    3.4、输入b，则进入购买买流程：
        3.4.1、允许用户根据商品编号购买商品。
        3.4.2、用户选择商品后，检测余额是否够，够就直接扣款，不够就提醒。
4、可随时退出，退出后，打印已购买商品和余额，并更新文件信息。

```
goods = [
    {"name": "电脑", "price": 1999},
    {"name": "鼠标", "price": 10},
    {"name": "书籍", "price": 50},
    {"name": "空调", "price": 2000},
]
users = [
    {"name": "张三", "password": 'zhangsan', 'age':11,'balance': 1000.0},
    {"name": "李四", "password": 'lisi', 'age':22,'balance': 2000.0},
    {"name": "王五", "password": 'wangwu', 'age':33,'balance': 2000.0},
]
```