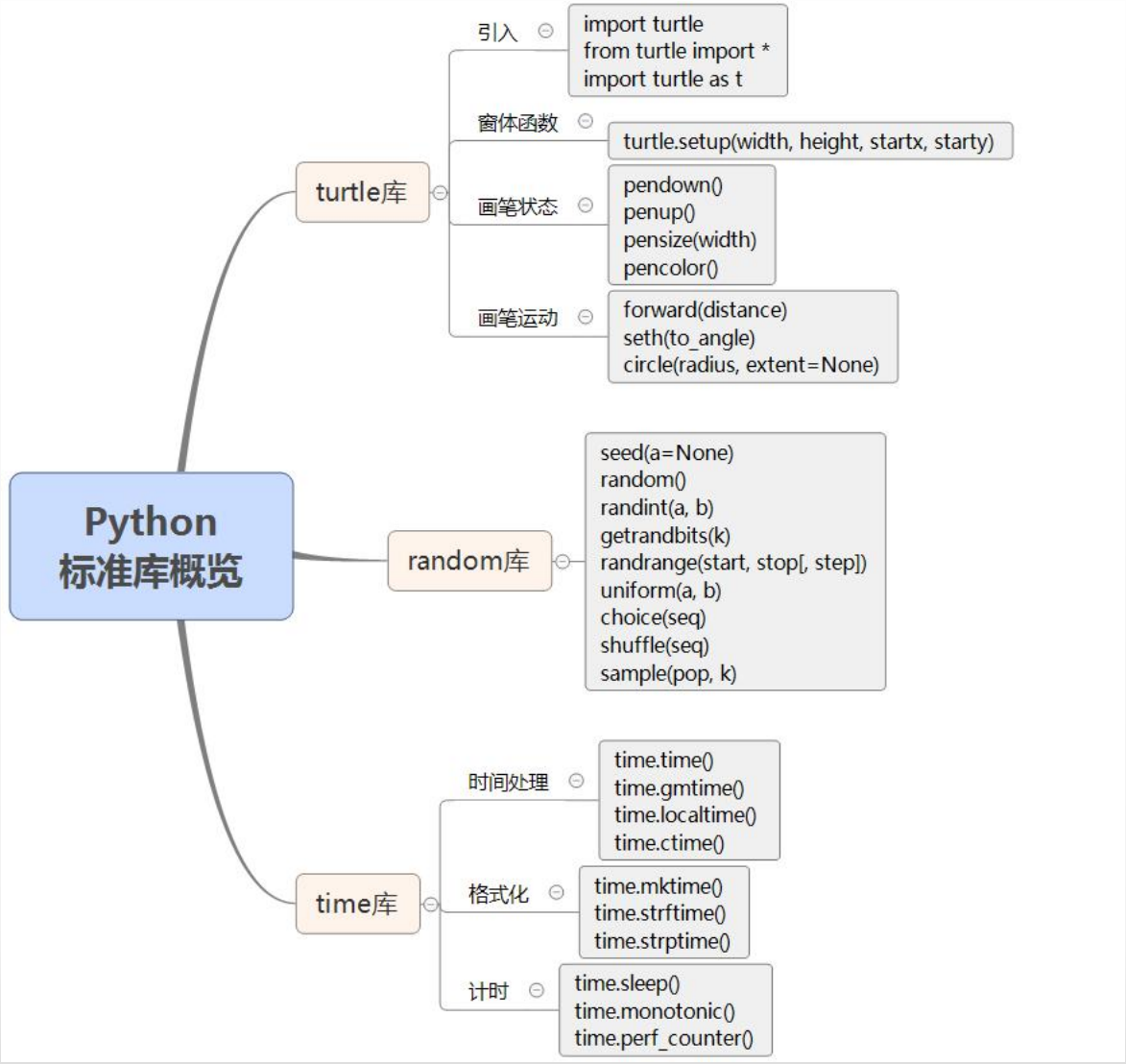
# 全国计算机等级考试二级教程
# Python语言程序设计
## (2018年版)

# 【第9章】
# Python标准库概览

# 考纲考点

- 标准库: turtle库(必选)
- 标准库: random库(必选)、time库(可选）

# 知识导图


Python
标准库概览
turtle库
引入
import turtle
from turtle import *
import turtle as t
窗体函数
turtle.setup(width, height, startx, starty)
画笔状态
pendown()
penup()
pensize(width)
pencolor()
画笔运动
forward(distance)
seth(to_angle)
circle(radius, extent=None)
random库
seed(a=None)
random()
randint(a, b)
getrandbits(k)
randrange(start, stop[, step])
uniform(a, b)
choice(seq)
shuffle(seq)
sample(pop, k)
time库
时间处理
time.time()
time.gmtime()
time.localtime()
time.ctime()
格式化
time.mktime()
time.strftime()
time.strptime()
计时
time.sleep()
time.monotonic()
time.perf_counter()

# turtle库概述

# turtle库概述

- turtle（海龟）是Python重要的标准库之一，它能够进行**基本的图形绘制**。
- turtle库绘制图形有一个基本框架：一个小海龟在坐标系中爬行，其爬行轨迹形成了绘制图形。对于小海龟来说，有“前进”、“后退”、“旋转”等爬行行为，对坐标系的探索也通过**“前进方向”、“后退方向”、“左侧方向”和“右侧方向”**等小海龟自身角度方位来完成。

# turtle库概述

- 使用import保留字对turtle库的引用有如下三种方式
- 第一种，**import turtle**，则对turtle库中函数调用采用turtle.<函数名>()形式。

```
1 import turtle
2 turtle.circle(200)
```

# turtle库概述

- 第二种，**from turtle import ***，则对turtle库中函数调用直接采用<函数名>()形式，不在使用turtle.作为前导。

```
1 from turtle import *
2 circle(200)
```

# turtle库概述

- 第三种，**import turtle as t**，则对turtle库中函数调用采用更简洁的t.<函数名>()形式，保留字as的作用是将turtle库给予别名t。

```
1 import turtle as t
2 t.circle(200)
```

# turtle库与基本绘图

- turtle库包含100多个功能函数，主要包括**窗体函数、画笔状态函数、画笔运动函数**等三类。

# 窗体函数

- turtle.setup(width, height, startx, starty)

**作用**：设置主窗体的大小和位置
**参数**：
width ：窗口宽度，如果值是整数，表示的像素值；如果值是小数，表示窗口宽度与屏幕的比例；
height： 窗口高度，如果值是整数，表示的像素值；如果值是小数，表示窗口高度与屏幕的比例；
startx：窗口左侧与屏幕左侧的像素距离，如果值是None，窗口位于屏幕水平中央；
starty：窗口顶部与屏幕顶部的像素距离，如果值是None，窗口位于屏幕垂直中央；

# 画笔状态函数

| 函数 | 描述 |
| --- | --- |
| pendown() | 放下画笔 |
| penup() | 提起画笔，与pendown()配对使用 |
| pensize(width) | 设置画笔线条的粗细为指定大小 |
| color() | 设置画笔的颜色 |
| begin_fill() | 填充图形前，调用该方法 |
| end_fill() | 填充图形结束 |
| filling() | 返回填充的状态，True为填充，False为未填充 |
| clear() | 清空当前窗口，但不改变当前画笔的位置 |
| reset() | 清空当前窗口，并重置位置等状态为默认值 |
| screensize() | 设置画布的长和宽 |
| hideturtle() | 隐藏画笔的turtle形状 |
| showturtle() | 显示画笔的turtle形状 |
| isvisible() | 如果turtle可见，则返回True |

# 画笔状态函数

- turtle中的画笔（即小海龟）可以通过一组函数来控制，其中turtle.penup()和turtle.pendown()是一组，它们分别表示画笔的和落下，函数定义如下：

**turtle.penup()**　别名　**turtle.pu(), turtle.up()**
**作用**：抬起画笔，之后，移动画笔不绘制形状
**参数**：无

**turtle.pendown()**　别名　**turtle.pd(), turtle.down()**
**作用**：落下画笔，之后，移动画笔将绘制形状
**参数**：无

# 画笔状态函数

- turtle.pensize()函数用来设置画笔尺寸

**turtle.pensize(width)　　别名　　turtle.width()**
**作用**：设置画笔宽度，当无参数输入时返回当前画笔宽度
**参数**：
width ：设置的画笔线条宽度，如果为None或者为空，函数则返回当前画笔宽度。

- turtle.pencolor()函数给画笔设置颜色

**turtle.pencolor(colorstring)** 或者　**turtle.pencolor((r,g,b))**
**作用**：设置画笔颜色，当无参数输入时返回当前画笔颜色
**参数**：
colorstring ：表示颜色的字符串，例如："purple"、"red"、"blue"等
(r,g,b)：颜色对应RGB的01数值，例如：1, 0.65, 0

# 画笔运动函数

| 函数 | 描述 |
| --- | --- |
| forward() | 沿着当前方向前进指定距离 |
| backward() | 沿着当前相反方向后退指定距离 |
| right(angle) | 向右旋转angle角度 |
| left(angle) | 向左旋转angle角度 |
| goto(x,y) | 移动到绝对坐标（x,y）处 |
| setx( ) | 将当前x轴移动到指定位置 |
| sety( ) | 将当前y轴移动到指定位置 |
| setheading(angle) | 设置当前朝向为angle角度 |
| home() | 设置当前画笔位置为原点，朝向东。 |
| circle(radius,e) | 绘制一个指定半径r和角度e的圆或弧形 |
| dot(r,color) | 绘制一个指定半径r和颜色color的圆点 |
| undo() | 撤销画笔最后一步动作 |
| speed() | 设置画笔的绘制速度，参数为0-10之间 |

# 画笔状态函数

- turtle.fd()函数最常用，它控制画笔向当前行进方向前进一个距离

**turtle.fd(distance)**　　别名　**turtle.forward(distance)**
**作用**：向小海龟当前行进方向前进distance距离
**参数**：
distance ：行进距离的像素值，当值为负数时，表示向相反方向前进。

- turtle.seth()函数用来改变画笔绘制方向

**turtle.seth(to_angle)**　　别名　**turtle.setheading(to_angle)**
**作用**：设置小海龟当前行进方向为to_angle，该角度是绝对方向角度值。
**参数**：
to_angle ：角度的整数值。

# 画笔状态函数

- turtle.circle()函数用来绘制一个弧形

**turtle.circle(radius, extent=None)**
**作用**：根据半径radius绘制extent角度的弧形。
**参数**：
radius ：弧形半径，当值为正数时，半径在小海龟左侧，当值为负数时，半径在小海龟右侧；
extent ： 绘制弧形的角度，当不给该参数或参数为None时，绘制整个圆形。

# random库概述

# random库概述

- 使用random库主要目的是**生成随机数**
- 这个库提供了不同类型的随机数函数，其中最基本的函数是random.random()，它生成一个[0.0, 1.0)之间的随机小数，所有其他随机函数都是基于这个函数扩展而来。

```
>>>from random import *
>>>random()
0.5780913011344704
>>>random()
0.2060982321395017 4
```

# random库与随机数运用

- random库的常用函数

| 函数 | 描述 |
| --- | --- |
| seed(a=None) | 初始化随机数种子，默认值为当前系统时间 |
| random() | 生成一个[0.0, 1.0)之间的随机小数 |
| randint(a, b) | 生成一个[a,b]之间的整数 |
| getrandbits(k) | 生成一个k比特长度的随机整数 |
| randrange(start, stop[, step]) | 生成一个[start, stop)之间以step为步数的随机整数 |
| uniform(a, b) | 生成一个[a, b]之间的随机小数 |
| choice(seq) | 从序列类型(例如：列表)中随机返回一个元素 |
| shuffle(seq) | 将序列类型中元素随机排列，返回打乱后的序列 |
| sample(pop, k) | 从pop类型中随机选取k个元素，以列表类型返回 |

# random库与随机数运用

- random库使用random.seed(a)对后续产生的随机数设置种子a。

```
>>>from random import *
>>>seed(10)
>>>random()
0.5714025946899135
>>>random()
0.4288890546751146
>>>seed(10)    #再次设置相同的种子，则后续产生的随机数相同
>>>random()
0.5714025946899135
>>>random()
0.4288890546751146
```

# random库与随机数运用

- 设置随机数种子的好处是可以**准确复现随机数序列**，用于重复程序的运行轨迹。对于仅使用随机数但不需要复现的情形，可以不用设置随机数种子。
- 如果程序没有显式设置随机数种子，则使用随机数生成函数前，将默认以当前系统的运行时间为种子产生随机序列。

# time库概述

# time库概述

- 处理时间是程序最常用的功能之一，time库是Python提供的处理时间标准库。time库提供系统级精确计时器的计时功能，可以用来分析程序性能，也可让程序暂停运行时间。

```
>>>import time
>>>time.localtime()
time.struct_time(tm_year=2017, tm_mon=12,
tm_mday=2, tm_hour=14, tm_min=44, tm_sec=9,
tm_wday=4, tm_yday=26, tm_isdst=0)
```

# time库概述

- time库的功能主要分为3个方面：**时间处理、时间格式化和计时**。
  - 时间处理主要包括4个函数：time.time()、time.gmtime()、time.localtime()、time.ctime()。
  - 时间格式化主要包括3个函数：time.mktime()、time.strftime()、time.strptime()。
  - 计时主要包括3个函数：time.sleep()、time.monotonic()、time.perf_counter()

# time库概述

- 使用time.time()获取当前时间戳

```
>>>import time
>>>time.time()
1516939876.6022282
```

- 使用time.gmtime(secs)获取当前时间戳对应的struct_time对象

```
>>> time.gmtime(now)
time.struct_time(tm_year=2018, tm_mon=1,
tm_mday=26, tm_hour=4, tm_min=11, tm_sec=16,
tm_wday=4, tm_yday=26, tm_isdst=0)
```

# time库概述

- 使用time.localtime(secs)获取当前时间戳对应的本地时间的struct_time对象

```
>>> time.localtime(now)
time.struct_time(tm_year=2018, tm_mon=1, tm_mday=26,
tm_hour=12, tm_min=11, tm_sec=16, tm_wday=4,
tm_yday=26, tm_isdst=0)
```

- 注意结果与gmtime的区别，UTC时间已自动转换为北京时间。

# time库概述

- 使用time.ctime(secs)获取当前时间戳对应的易读字符串表示，内部会调用time.localtime()函数以输出当地时间。

```
>>> time.ctime(now)
'Fri Jan 26 12:11:16 2018'
```

# time库概述

- time库使用time.mktime()、time.strftime()、time.strptime()进行时间格式化。

# time库概述

- 使用time.mktime(t) 将struct_time对象t转换为时间戳，注意t代表当地时间。struct_time对象的元素如下

| 下标 | 属性 | 值 |
| --- | --- | --- |
| 0 | tm_year | 年份，整数 |
| 1 | tm_mon | 月份[1, 12] |
| 2 | tm_mday | 日期[1, 31] |
| 3 | tm_hour | 小时[0, 23] |
| 4 | tm_min | 分钟[0, 59] |
| 5 | tm_sec | 秒[0, 61] |
| 6 | tm_wday | 星期[0, 6]（0表示星期一） |
| 7 | tm_yday | 该年第几天[1, 366] |
| 8 | tm_isdst | 是否夏时令，0否，1是，-1未知 |

# time库概述

- 调用time.mktime(t)函数

```
>>> t = time.localtime(now)
>>> time.mktime(t)
1516939876.0
>>> time.ctime(time.mktime(t))
'Fri Jan 26 12:11:16 2018'
```

- time.strftime()函数是时间格式化最有效的方法，几乎可以以任何通用格式输出时间。该方法利用一个格式字符串，对时间格式进行表达。

```
>>> lctime = time.localtime()
>>> lctime
time.struct_time(tm_year=2018, tm_mon=1, tm_mday=26,
tm_hour=12, tm_min=55, tm_sec=20, tm_wday=4, tm_yday=26,
tm_isdst=0)
>>> time.strftime("%Y-%m-%d %H:%M:%S", lctime)
'2018-01-26 12:55:20'
```

# time库概述

- strftime()方法的格式化控制符

| 格式化字符串 | 日期/时间 | 值范围和实例 |
| --- | --- | --- |
| %Y | 年份 | 0001~9999，例如：1900 |
| %m | 月份 | 01~12，例如：10 |
| %B | 月名 | January~December，例如：April |
| %b | 月名缩写 | Jan~Dec，例如：Apr |
| %d | 日期 | 01～31，例如：25 |
| %A | 星期 | Monday~Sunday，例如：Wednesday |
| %a | 星期缩写 | Mon~Sun，例如：Wed |
| %H | 小时（24h制） | 00～23，例如：12 |
| %I | 小时（12h制） | 01～12，例如：7 |
| %p | 上/下午 | AM, PM，例如：PM |
| %M | 分钟 | 00～59，例如：26 |
| %S | 秒 | 00～59，例如：26 |

# time库概述

- strptime()方法与strftime()方法完全相反，用于提取字符串中时间来生成strut_time对象，可以很灵活的作为time模块的输入接口

```
>>> timeString = '2018-01-26 12:55:20'
>>> time.strptime(timeString, "%Y-%m-%d %H:%M:%S")
time.struct_time(tm_year=2018, tm_mon=1, tm_mday=26,
tm_hour=12, tm_min=55, tm_sec=20, tm_wday=4, tm_yday=26,
tm_isdst=-1)
```

# time库与程序计时

# time库与程序计时

- 程序计时是非常常用的功能，尤其是对于运行时间较长的程序，往往需要先进行小规模（短时间）的实验，并根据实验结果预估最终程序的大致运行时间。
- 程序计时主要要包含三个要素：程序开始/结束时间、程序运行总时间、程序各核心模块运行时间。
- 下面以1千万次循环计时为例介绍程序计时的实现，并进一步理解time模块相关函数的运用。

# time库与程序计时

- 以1千万次循环为主体，模拟实际程序的核心模块，用time.sleep()来模拟实际程序的其他模块。

```
1  import time
2  def coreLoop():
3      limit = 10**8
4      while (limit > 0):
5          limit -= 1
6
7  def otherLoop1():
8      time.sleep(0.2)
9
10 def otherLoop2():
11     time.sleep(0.4)
12
```

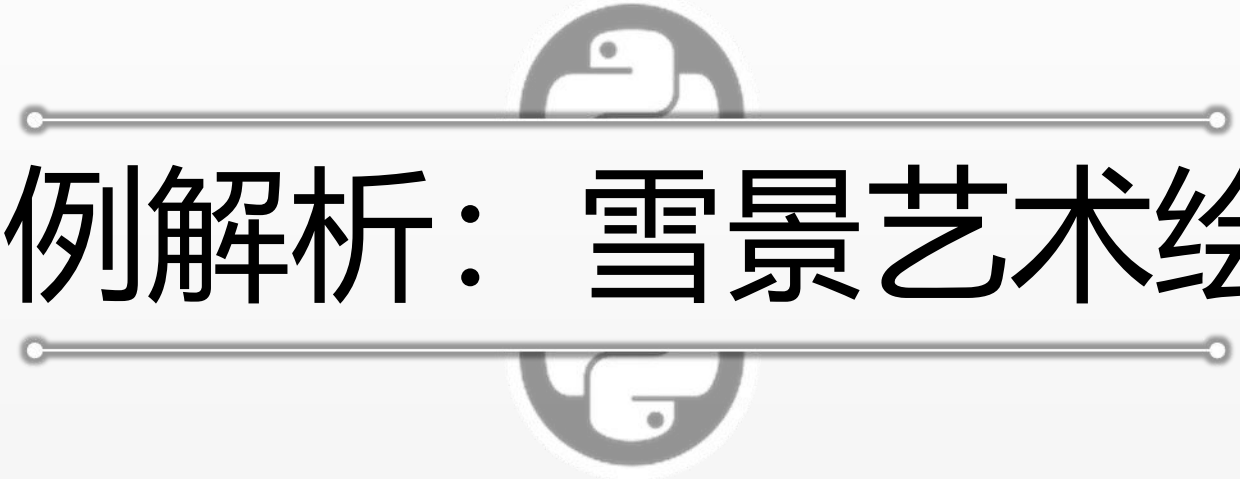
# time库与程序计时

```
13  def main():
14      startTime = time.localtime()
15      print('程序开始时间：', time.strftime('%Y-%m-%d %H:%M:%S', startTime))
16      startPerfCounter = time.perf_counter()
17      otherLoop1()
18      otherLoop1PerfCounter = time.perf_counter()
19      otherLoop1Perf = otherLoop1PerfCounter - startPerfCounter
20      coreLoop()
21      coreLoopPerfCounter = time.perf_counter()
22      coreLoopPerf = coreLoopPerfCounter - otherLoop1PerfCounter
23      otherLoop2()
24      otherLoop2PerfCounter = time.perf_counter()
25      otherLoop2Perf = otherLoop2PerfCounter - coreLoopPerfCounter
26      endPerfCounter = time.perf_counter()
27      totalPerf = endPerfCounter - startPerfCounter
28      endTime = time.localtime()
29      print("模块1运行时间是:{}秒".format(otherLoop1Perf))
30      print("核心模块运行时间是:{}秒".format(coreLoopPerf))
31      print("模块2运行时间是:{}秒".format(otherLoop2Perf))
32      print("程序运行总时间是:{}秒".format(totalPerf))
33     print('程序结束时间：', time.strftime('%Y-%m-%d %H:%M:%S', endTime))
34
35  main()
```

# time库与程序计时

- 程序运行的输出效果如下

```
程序开始时间：  2017-12-26 13:46:39
模块1运行时间是:0.20003105182731706秒
核心模块运行时间是:5.9871016398209 27秒
模块2运行时间是:0.40018931343066555秒
程序运行总时间是:6.587323585324574秒
程序结束时间： 2017-12-26 13:46:45
```



# 实例解析：雪景艺术绘图

# 雪景艺术绘图

- turtle图形艺术，指利用turtle库画笔创造性绘制绚丽多彩艺术图形的过程。
- turtle图形艺术效果中隐含着很多随机元素，如随机颜色、尺寸、位置和数量等。在图形艺术绘制中需要引入随机函数库random。常用randint()函数，生成指定范围内的随机数，

# 雪景艺术绘图

- “雪景”图形艺术背景为黑色，分为上下两个区域，上方是漫天彩色雪花，下方是由远及近的灰色横线渐变。该图运用了随机元素，如雪花位置、颜色、大小、花瓣数目、地面灰色线条长度、线条位置等，需要使用turtle库和random库。

# 雪景艺术绘图

- 绘制分为三个步骤

1. 构建图的背景
2. 绘制雪花效果
3. 绘制雪地效果

# 雪景艺术绘图

- 第一步，构建图的背景

设定窗体大小为800x600像素，窗体颜色为black。然后，定义上方雪花绘制函数drawSnow()和下方雪地绘制函数drawGround()。

# 雪景艺术绘图

- 第二步，绘制雪花效果。

为体现艺术效果，drawSnow()函数首先隐藏turtle画笔、设置画笔大小、绘制速度，然后使用for循环绘制100朵雪花。雪花大小snowsize、雪花花瓣数dens都分别设定为一定数值范围随机数。最后通过for循环绘制出多彩雪花。

# 雪景艺术绘图

- 第三步，绘制雪地效果。

drawGround()函数使用for循环绘制地面400个小横线，画笔大小pensize、位置坐标x、y、线段长度均通过randint()函数作为随机数产生。

# 雪景艺术绘图

实例 9.1

```
# SnowView.py
from turtle import *
from random import *
def drawSnow():
    hideturtle()
    pensize(2)
    for i in range(100):
        r, g, b = random(), random(), random()
        pencolor(r,g,b)
        penup()
        setx(randint(-350,350))
        sety(randint(1,270))
        pendown()
        dens = randint(8,12)
        snowsize = randint(10,14)
        for j in range(dens):
            forward(snowsize)
            backward(snowsize)
            right(360/dens)
```

实例 9.1

```
def drawGround():
    hideturtle()
    for i in range(400):
        pensize(randint(5,10))
        x = randint(-400,350)
        y = randint(-280,-1)
        r, g, b = -y/280, -y/280, -y/280
        pencolor((r,g,b))
        penup()
        goto(x,y)
        pendown()
        forward(randint(40,100))
setup(800,600,200,200)
tracer(False)
bgcolor("black")
drawSnow()
drawGround()
done()
```

# 本章小结

本章主要讲解了3个重要的Python标准库：turtle、random和time，分别用于基本图形绘制、随机数运用和时间处理。再详细讲解各函数库功能基础上，通过雪景随机艺术画的绘制进一步帮助读者掌握这三个有趣且有用的标准库。

能够用Python绘图了，最想绘制的图形是什么？最想送给谁？