20180626

第 1 章　程序设计基本方法

✧　1.1　计算机的概念

计算机是根据指令操作数据的设备，具备以下两个特性：

1) 功能性

对数据的操作表现为数据计算、输入输出处理和结果存储等

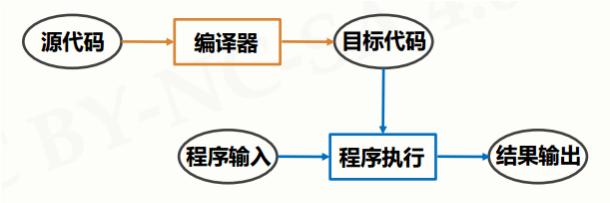
2) 可编程性

根据一系列指令自动地、可预测地、准确地完成操作者的意图

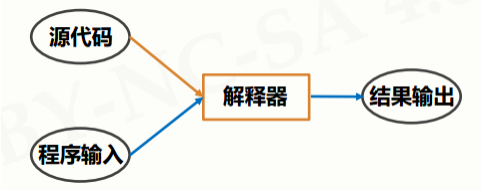
摩尔定律(Moore's Law)是计算机发展历史上最重要的预测法则，由 Intel 公司创始人之一戈登 摩尔在 1965 年提出。其内容是：单位面积集成电路上可容纳晶体管的数量约每两年翻一番；CPU/GPU、内存、硬盘、电子产品价格等都遵循摩尔定律。

✧　1.2　编译和解释

1) 编译：将源代码一次性转换成机器语言；一次性翻译，之后不需要源代码；



2) 解释：将源代码逐条转换成目机器语言同时逐条运行；



根据执行方式不同，编程语言分为静态语言和动态语言:

1) 静态语言：使用编译器执行的编程语言，如 C/C++、Java

编译器一次性生成目标代码，优化更充分，程序运行速度更快；

2) 脚本语言：使用解释器执行的编程语言，如 Python、JavaScript、PHP

执行程序时需要源代码，但维护更灵活、跨平台。

✧　1.3　IPO

程序的基本编写方法是 IPO

1) I：Input，输入，文件输入、网络输入、控制台输入等；

2) P：Process，处理，程序的主要逻辑，对输入数据进行计算得到结果，处理方法统称为算法，是程序最重要部分；

3) O：Output，输出，控制台输出、文件输出、网络输出等。

求解计算问题的精简步骤：

1) 确定 IPO；2) 编写程序；3) 调试程序。

第 2 章　Python 入门程序

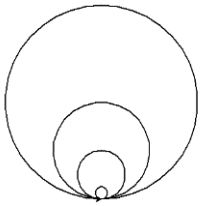
✧　2.1　Python 两种编程方式

交互式：IDLE 中编写，对每个输入语句即时运行结果，适合语法练习；
文件式：批量执行一组语句并运行结果，编程的主要方式。

练习：绘制多个同切圆

```
import turtle
def draw_circles(size, lst):
    # 绘制多个同切圆
    turtle.pensize(size)
    for i in lst:
        turtle.circle(i)
def main():
    lst = [10, 40, 80, 160]
    draw_circles(2, lst)
if __name__ == '__main__':
    main()
```

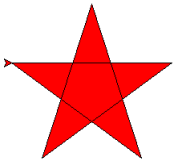
结果：



练习：绘制五角星

```
import turtle
def draw_star():
    # 五角星绘制
    turtle.color('black', 'red')  # 黑画笔,红背景
    turtle.begin_fill()
    for i in range(5):
        turtle.fd(200)  # 长度
        turtle.rt(144)  # 五角星每个角 36°,画笔转 144°
    turtle.end_fill()
    turtle.done()
```

结果：



✧　2.2　实例 1：温度转换问题

中国等世界大多数国家使用摄氏度，以 1atm 下水的熔点为 0 度，沸点为 100 度，将温度进行等分刻画；美国、英国等使用华氏度，以 1atm 下水的熔点为

32 度，沸点为 212 度，将温度进行等分刻画。

温度可以是直接输入的温度值，也可以是网络爬取的温度信息，或者是根据图片和声音等转换后的温度。
此处采用直接将温度值进行转换，温度值后带标志，F 为华氏度，C 为摄氏度，如 82F 表示华氏 82 度，28C 表示摄氏 28 度。
华氏度和摄氏度转换为：C = (F - 32) / 1.8 | F = 1.8×C + 32

```
import re
def temp_convert():
    t = input('请输入带符号的温度值：')
    p = r'^-?\d+[cCfF]$'
    if re.match(p, t):
        if t[-1] in ['F', 'f']:
            c = (int(t[:-1])-32)/1.8
            print("{}转换后为{:.2f}C".format(t.upper(), c))
        else:
            f = 1.8*int(t[:-1])+32
            print("{}转换后为{:.2f}F".format(t.upper(), f))
    else:
        print('格式有误！')
```

第 3 章　Python 基本图形绘制

✧　3.1　计算机技术的演进过程

2017-	人工智能时代	人类的问题
2008-2016	复杂信息系统时代	数据问题
1981-2008	网络和视窗时代	交互问题
1946-1981	计算机系统结构时代	计算能力问题

不同编程语言的初心和适用对象

编程语言	学习内容	语言本质	解决问题	适用对象
C	指针、内存、数据类型	理解计算机系统结构	性能	计算机类专业
Java	对象、跨平台、运行时	理解主客体关系	跨平台	软件类专业
C++	对象、多态、继承	理解主客体关系	大规模程序	计算机类专业
VB	对象、按钮、文本框	理解交互逻辑	桌面应用	不确定
Python	编程逻辑、第三方库	理解问题求解	各类问题	所有专业

各编程语言所处历史时期和使命不同，Python 是**计算时代**演进的选择！

✧　3.2　编程语言的种类

① 机器语言

一种二进制语言，直接使用二进制代码表达指令。计算机硬件(CPU)可以直接执行，与具体 CPU 型号有关。

② 汇编语言

一种将二进制代码直接对应助记符的编程语言。汇编语言与 CPU 型号有关，程序不通用，需要汇编器转换。
完成 2+3 功能的汇编语言：add 2,3,result

③ 高级语言
更接近自然语言，同时更容易描述计算问题。高级语言代码与具体 CPU 型号**无关**，编译器编译后运行。
完成 2+3 功能的高级语言：result = 2 + 3

④ 超级语言
具有庞大计算生态，可以很容易利用已有代码功能。编程思维不再是刀耕火种，而是集成开发。
完成 2+3 功能的超级语言：result = sum(2,3)

✧ 3.3 turtle 库

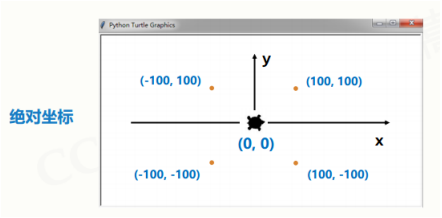
turtle 库是 turtle 绘图体系的 Python 实现。turtle 绘图体系在 1969 年诞生，主要用于程序设计入门级的图形绘制。turtle 库是 Python 语言的标准库之一。

turtle 的原(wan)理(fa)
turtle(海龟)真实存在：有一只海龟，在窗体正中心，在画布上走动，走过的轨迹形成了绘制的图形。海龟由程序控制，可以变换颜色、改变宽度等。

① turtle 的绘图窗体

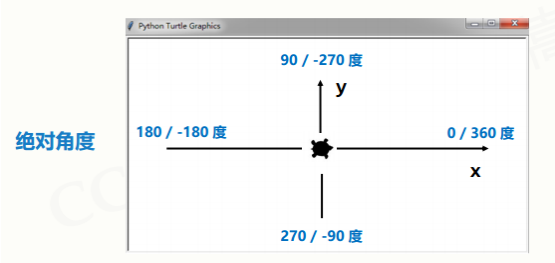
turtle.setup(width, height, startx, starty)：设置主窗体的大小和位置
width、height 表示主窗体宽高；默认 50%宽 75%高；
startx、starty 表示主窗体左上角的坐标；默认水平垂直居中。

② turtle 空间坐标体系



1) turtle.goto(x, y=None)：将海龟移动到指定绝对坐标
2) turtle.fd(distance)：将海龟向前移动指定距离
3) turtle.bk(distance)：向后移
4) turtle.circle(radius, extent=None, steps=None)：画圆
   radius：半径
   extent：表示画指定弧度的圆弧，默认 None 表示画一个圆；
   steps：圆是正多边形的近似。steps 表示走的步数，默认 None，自动计算步数；指定特定值绘制多边形。

③ turtle 角度坐标体系



1) turtle.seth(to_angle)：改变海龟行进方向
   to_angle：绝对度数
2) turtle.rt(angle)：右转 angle 度，相对度数
3) turtle.lt(angle)：左转 angle 度，相对度数

此三个方法只改变海龟的方向，不行进。

④ RGB 色彩模式

RGB 可以用 0~1 之间小数表示或 0~255 整数表示。
默认采用小数值，可以切换为整数值。

turtle.colormode(cmode=None)：切换色彩模式
   cmode：色彩模式，可以传 1.0 或 255 表示小数模式或整数模式。

✧ 3.4 实例 2：Python 蟒蛇绘制

```
import turtle
def draw_python():
    turtle.setup(650, 350, 200, 200)  # 绘图窗体大小和位置
    turtle.penup()
    turtle.fd(-250)  # 画笔向后移不留轨迹
    turtle.pendown()
    turtle.pensize(25)
    turtle.pencolor("purple")
    turtle.seth(-40)  # 海龟面向第四象限的 320°
    for i in range(4):  # 4 个曲折
        turtle.circle(40, 80)
        turtle.circle(-40, 80)
    turtle.circle(40, 80/2)
    turtle.fd(40)
```

```
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
    turtle.done()  # 结束保留绘图窗体，需要手动退出
```

结果：

① turtle 画笔控制函数

1) turtle.penup()：移动时不留下轨迹，海龟在飞行
别名：turtle.pu()、turtle.up()

2) turtle.pendown()：落下画笔，up 和 down 成对出现
别名：turtle.pd()、turtle.down()

3) turtle.pensize(width=None)：设置画笔粗细
别名：turtle.width(width=None)

4) turtle.pencolor(*args)：设置画笔颜色
四种支持的格式：pencolor()、pencolor(colorstring)、pencolor((r, g, b))、pencolor(r, g, b)

② turtle 运动控制函数

1) turtle.forward(distance)：向前直行，distance 负数表示后退
别名：turtle.fd(distance)

2) turtle.circle(radius, extent=None, steps=None)：根据半径 radius 绘制 extent 角度的弧形，默认圆心在海龟左侧 radius 位置。

③ turtle 方向控制函数

1) turtle.setheading(to_angle)：控制海龟面对方向，绝对角度
别名：turtle.seth(to_angle)

2) turtle.right(angle)：右转 angle 度，相对度数
别名：turtle.rt(angle)

3) turtle.left(angle)：左转 angle 度，相对度数
别名：turtle.lt(angle)

第 4 章　基本数据类型

✧ 4.1　数字类型

整数(int)、浮点数(float)、复数(complex)

一些数值运算的内建函数：

1） `divmod(x, y)`：同时求商和余数，返回(x//y, x%y)的元组

2） `pow(x, y, z=None)`：幂运算，x**y (双参数)或 x**y % z (三参数)

```
pow(3, pow(3, 99), 10000)  # 4587, 数字太大了,但是对 10000 取余看后四位数
```

3） round(number, ndigits=0)：四舍五入，保留小数位数默认为 0

✧ 4.2 实例 3：天天向上的力量

一年 365 天，每天进步 1%，累计进步多少? $\mathbf{1.01^{365}}$

一年 365 天，每天退步 1%，累计还剩多少? $\mathbf{0.99^{365}}$

```
def day_day_up():
    print('向上: {:.2f}; 向下: {:.2f}'.format(
        1.01**365, 0.99**365))  # 向上: 37.78; 向下: 0.03
```

工作日的力量：

一年 365 天，每周 5 个工作日每天进步 1%；每周 2 个休息日每天退步 1%。

```
def day_day_up(up_rate, down_rate):
    up = 1
    for i in range(365):
        if i % 7 in [0, 6]:  # 周末
            up *= 1-down_rate
        else:  # 工作日
            up *= 1+up_rate
    return up

if __name__ == '__main__':
    print('工作日的力量: {:.2f}'.format(day_day_up(0.01, 0.01)))
    # 工作日的力量: 4.63
```

工作日的努力

工作日模式要努力到什么水平，才能与每天努力 1%一样?

A 君：一年 365 天，每天进步 1%，不停歇；

B 君：一年 365 天，每周工作 5 天休息 2 天，休息日下降 1%，要多努力?

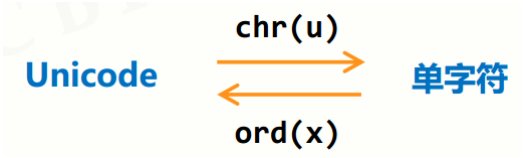
```
def need_up(rate):
    up = (1+rate)**365
    while day_day_up(rate, 0.01) < up:
        rate += 0.001
    return round(rate, 3)

if __name__ == '__main__':
    print('工作日的努力参数: {}'.format(need_up(0.01)))  # 工作日的努力参数: 0.019
```

✧ 4.3 字符串类型

① 一些字符串处理函数

chr(i)：返回 Unicode 编码是 i 的单字符，0 <= i <= 0x10ffff；

ord(c)：返回单字符 c 对应 Unicode 编码。



Unicode 编码是统一字符编码，即覆盖几乎所有字符的编码方式，从 0 到 1114111 (0x10FFFF)空间，每个编码对应一个字符。
Python 字符串中每个字符都是 Unicode 编码字符。

十二星座符号：

```
>>> for i in range(12):
        print(chr(9800+i),end=' ')

♈ ♉ ♊ ♋ ♌ ♍ ♎ ♏ ♐ ♑ ♒ ♓
```

② format()方法的格式控制
{}槽内部格式化配置：{参数序号:格式控制标记}
格式控制标记：

:	填充	对齐	宽度	,	.精度	类型
引导符号		<左对齐(默认) >右对齐 ^居中对齐	槽{}的宽度	数字千位分隔符	浮点数精度	b,c,d,o,x,X,e,E,f,%

示例：

```
>>> '{:*^20}'.format('hikari')
'*******hikari*******'
>>> '{:20}'.format('hikari')
'hikari              '
>>> '{:,.2f}'.format(123456.789)
'123,456.79'
>>> '{0:b}, {0:c}, {0:d}, {0:o}, {0:x}, {0:X}'.format(108)
'1101100, l, 108, 154, 6c, 6C'
```

20180627

✧ 4.4 time 库
① 时间获取
1) time()：获取当前时间戳，浮点数
2) ctime(seconds)：时间戳转为易读的字符串形式，默认当前时间
3) gmtime([seconds])：获取 UTC 时间，时间格式元组
4) localtime([seconds])：获取当前本地时间，时间格式元组

```
>>> import time
>>> time.time()
1530031763.693609
```

```
>>> time.ctime()
'Wed Jun 27 01:49:26 2018'
>>> time.gmtime()
time.struct_time(tm_year=2018, tm_mon=6, tm_mday=26, tm_hour=16, tm_min=49,
tm_sec=31, tm_wday=1, tm_yday=177, tm_isdst=0)
```

② 时间格式化

将时间以合理的方式展示，类似于字符串格式化，需要有展示模板。

1) strftime(format[, tuple]) -> string

根据指定格式将元组转为字符串，tuple 默认使用 localtime()返回值

常用格式：

```
%Y：年份
%m：月份，[01, 12]
%d：日，[01, 31]
%H：小时，[00, 23]
%M：分，[00, 59]
%S：秒，[00, 61]
%a：星期缩写，Sun~Sat
%A：星期完整，Sunday~Saturday
%b：月份名称写，Jan~Dec
%B：月份名称，January~December
```

2) strptime(string, format)：根据时间字符串和和格式模板转为结构化时间元组

```
>>> t = time.gmtime()
>>> time.strftime('%Y-%m-%d %H:%M:%S', t)  # 指定时间
'2018-06-27 02:18:38'
>>> time.strftime('%Y-%m-%d %H:%M:%S')  # 本地时间
'2018-06-27 10:20:10'
>>> time.strptime('20180627 12:12:12', '%Y%m%d %H:%M:%S')
time.struct_time(tm_year=2018, tm_mon=6, tm_mday=27, tm_hour=12,
                 tm_min=12, tm_sec=12, tm_wday=2, tm_yday=178, tm_isdst=-1)
```

③ 程序计时

perf_counter()：返回一个 CPU 级别的精确时间计数值，单位秒

sleep(s)：模拟休眠 s 秒

```
import time

def test():
    lst = [str(x) for x in range(100000)]
    s = ', '.join(lst)
    time.sleep(3)
    s += ', '.join(lst)
    print(len(s))  # 1377776
```

```
if __name__ == '__main__':
    start = time.perf_counter()
    test()
    end = time.perf_counter()
    print('test()用时: {:.2f}s'.format(end-start))  # test()用时: 3.17s
```

✧ 4.5 实例 4：文本进度条(Progress Bar)

在任何运行时间较长的程序增加进度条可以提高用户体验。
需求：采用字符串方式打印动态变化的文本进度条，能在一行中逐渐变化。

① 简单的多行进度条

```
import time
def test():
    scale = 10
    print('------执行开始------')
    for i in range(scale+1):
        a = '*' * i  # 已完成
        b = '.' * (scale - i)  # 未完成
        c = (i/scale)*100
        print('{:^3.0f}%[{}->{}]'.format(c, a, b))
        run()  # 执行任务
    print('------执行结束------')
def run():
    time.sleep(0.1)
if __name__ == '__main__':
    test()
```

结果：

```
------执行开始------
 0 %[->.........]
10 %[*->........]
20 %[**->.......]
30 %[***->......]
40 %[****->.....]
50 %[*****->....]
60 %[******->...]
70 %[*******->..]
80 %[********->.]
90 %[*********->]
100%[**********->]
------执行结束------
```

② 单行动态刷新

刷新的本质是用后打印的字符串覆盖之前的字符串。
所以不能换行，而且需要回退，打印后光标退回之前的位置。
不换行 print()需要指定 end=''，光标回退到行首使用\r

```python
import time
from random import random

def test():
    scale = 50
    print('执行开始'.center(scale//2, '-'))
    start = time.perf_counter()
    for i in range(scale+1):
        a = '*' * i  # 已完成
        b = '.' * (scale - i)  # 未完成
        c = (i/scale)*100
        dur = time.perf_counter()-start
        print("\r{:^3.0f}%[{}->{}]{:.2f}s".format(c, a, b, dur), end='')
        run()  # 执行任务
    print('\n'+'执行结束'.center(scale//2, '-'))

def run():
    time.sleep(random())  # 模拟任务

if __name__ == '__main__':
    test()
```

因为 IDLE 屏蔽了\r 功能，需要命令行执行：

```
$ python "TextProBar.py"
----------执行开始----------
100%[************************************************->]21.46s
----------执行结束----------
```

③ 非线性显示进度条

实际执行进度与进度条显示可以不是线性，前期慢后期快可能会给用户惊喜

```python
def test2():
    scale = 50
    print('执行开始'.center(scale//2, '-'))
    lst = [x/scale for x in range(scale+1)]
    start = time.perf_counter()
    for i in lst:
        # 进度条前期慢后期快
        x = (i+(1-i)/2)**8  # FastPower, Speedsup
        a = '*' * (int(x*scale))  # 已完成
        b = '.' * (int((1-x)*scale))  # 未完成
        c = x*100  # 进度百分比
        dur = time.perf_counter()-start
        print("\r{:^3.0f}%[{}->{}]{:.2f}s".format(c, a, b, dur), end='')
        run()  # 执行任务
    print('\n'+'执行结束'.center(scale//2, '-'))
```

第 5 章　程序控制结构

✧　5.1　实例 5：身体质量指数 BMI

BMI(Body Mass Index)是国际上常用的衡量人体肥胖和健康程度的重要标准，主要用于统计分析。

BMI = 体重(kg) / 身高 ^{2} (m^2)

国际和国内 BMI 标准：

分类	国际 BMI 值	国内 BMI 值
偏瘦	<18.5	<18.5
正常	18.5 ~ 25	18.5 ~ 24
偏胖	25 ~ 30	24 ~ 28
肥胖	≥30	≥28

可以按照国际和国内标准写两个函数，也可以根据条件合并为一个函数：

```
def calBMI(h, w):
    bmi = w / (h/100)**2
    print('BMI 数值为：{:.2f}'.format(bmi))
    world, cn = '', ''
    if bmi < 18.5:
        world, cn = '偏瘦', '偏瘦'
    elif 18.5 <= bmi < 24:
        world, cn = '正常', '正常'
    elif 24 <= bmi < 25:
        world, cn = '正常', '偏胖'
    elif 25 <= bmi < 28:
        world, cn = '偏胖', '偏胖'
    elif 28 <= bmi < 30:
        world, cn = '偏胖', '肥胖'
    else:
        world, cn = '肥胖', '肥胖'
    print('BMI 指标为:[国际]{}; [国内]{}'.format(world, cn))

if __name__ == '__main__':
    height, weight = eval(input('请输入身高(cm)和体重(kg)[逗号隔开]: '))
    calBMI(height, weight)
```

测试：

```
请输入身高(cm)和体重(kg)[逗号隔开]: 174, 74
BMI 数值为：24.44
BMI 指标为:[国际]正常; [国内]偏胖
```

✧　5.2　random 模块

random 模块是 Python 标准库，用于生成随机数。计算机不能生成真正随机数，采用梅森旋转算法生成(伪)随机序列中的元素。

① 基本随机数函数

1) seed(a=None)：初始化随机数种子，默认为当前系统时间

相同随机数种子产生的随机序列相同。

2) random()：生成一个[0, 1)的随机数

```
import random
# 两次随机数种子为 10,产生的随机数序列相同, 第 1 个随机数当然也相同
random.seed(10)
print(random.random())  # 0.5714025946899135
random.seed(10)
print(random.random())  # 0.5714025946899135
```

② 扩展随机数函数

1) randint(a, b)：生成一个[a, b]之间随机整数

2) randrange(start, stop, step=1)：生成一个[start, stop)之间 k 为步长的随机整数

3) getrandbits(k)：生成一个 k 长比特的随机整数

4) uniform(a, b)：生成一个[a, b]之间随机小数

5) choice(seq)：从序列中随机选出一个元素

6) shuffle(seq)：将序列中元素打乱

✧ 5.3 实例 6：圆周率的计算

① 圆周率近似计算公式：

$$\pi = \sum_{k=0}^{\infty}[\frac{1}{16^k}(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6})]$$

```
from time import perf_counter

def cal_pi_v1():
    n = 1000
    pi = 0
    start = perf_counter()
    for k in range(n):
        pi += (1/16**k)*(4/(8*k+1) - 2/(8*k+4) - 1/(8*k+5) - 1/(8*k+6))
    end = perf_counter()
    print('运行时间: {:.2f}s'.format(end-start))
    return pi

if __name__ == '__main__':
    print(cal_pi_v1())
```

结果：

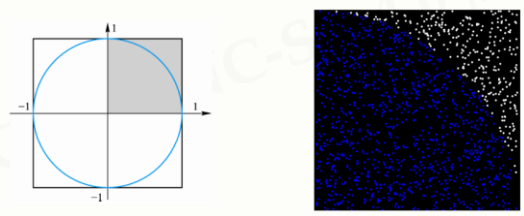
```
运行时间: 0.02s
3.141592653589793
```

② 蒙特卡罗方法

也称统计模拟方法，使用随机数解决很多计算问题。

比如计算圆周率，在边长为 1 的正方形区域随机撒点 n 个，距离圆心≤1 的点个数为 m。$S_{扇形} / S_{正方形} = \pi / 4 = m / n$，所以 $\pi = 4 * m / n$。

撒点个数越多，得到的结果越精确。



```python
from random import random
from time import perf_counter

def cal_pi_v2():
    # 蒙特卡罗方法
    darts = 1000*1000  # 随机撒点个数
    hits = 0  # 记录落在扇形中的点数
    start = perf_counter()
    for i in range(darts):
        # 随机撒点
        x, y = random(), random()
        distance = (x**2+y**2)**0.5  # 该点到圆心距离
        if distance <= 1:
            hits += 1
    pi = hits/darts*4
    end = perf_counter()
    print('运行时间: {:.2f}s'.format(end-start))
    return pi

if __name__ == '__main__':
    print(cal_pi_v2())
```

结果：// 电脑太垃圾了...

```
运行时间: 13.33s
3.14124
```

第 6 章　函数

✧ 6.1　函数参数

① POSITIONAL_OR_KEYWORD：位置或关键字参数，Python 最普通的参数类型，可以通过位置或关键字传参数；

```python
def f(a):
    print(a)

f(1)   # 位置传参调用
```

```
f(a=1)   # 关键字传参调用
```

② VAR_POSITIONAL：可变参数*args，位置参数的元组，不能用关键字传参：

```
def f(*args):
    print(args)

# 可以传入任意个位置参数调用,不传参数也可以,传入关键字参数报错
f()   # ()
f(1, 'a', True)   # (1, 'a', True)
f(a=1)   # TypeError: f() got an unexpected keyword argument 'a'
```

③ KEYWORD_ONLY：关键字参数，在*或*args (VAR_POSITIONAL)后面的参数，只能用关键字传参数，因为位置参数被前面的*args 全部接收了；

```
def f(*, a):   # VAR_POSITIONAL 不需要使用时,可以匿名化
    print(a)

# 只能关键字传参,位置传参报错
f(a=1)   # 1
f(1)   # TypeError: f() takes 0 positional arguments but 1 was given
```

④ VAR_KEYWORD：可变关键字参数**kwargs，字典形式，此类型的参数只允许有一个，只能在函数最后声名：

```
def f(**kwargs):
    print(kwargs)

# 可以传入任意个关键字参数,不传也可以,传入位置参数报错
f()  # {}
f(a=1, b='b', c=False, d=[1, 2])  # {'a': 1, 'b': 'b', 'c': False, 'd': [1, 2]}
f(1)  # TypeError: f() takes 0 positional arguments but 1 was given
```

⑤ POSITIONAL_ONLY：位置参数，不重要，历史遗留产物，高版本 Python 无法使用此类参数，推荐用 VAR_POSITIONAL 来代替。

默认参数(可选参数)
1) VAR 类型不允许设置默认参数；
2) 默认参数靠后放；
3) 默认参数不要设为可变类型(如 list、dict 等)，因为如果在函数内改变了默认参数，下次再调用时就不再是默认值。

示例：默认参数使用可变类型的问题

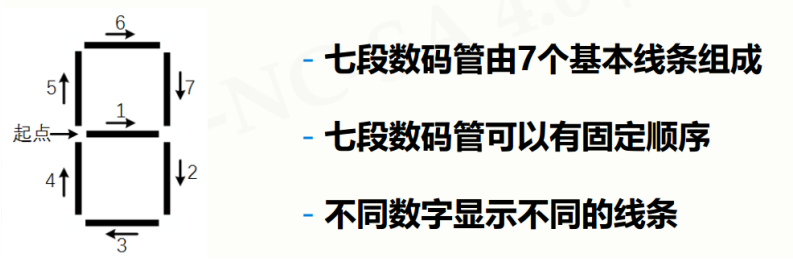
```
def f(a, lst=[]):
    lst.append(a)
    print(lst)
```

```
f(1)  # [1]
f(2)  # [1, 2]
f(3, ['a', 'b'])  # ['a', 'b', 3]
f(4)  # [1, 2, 4]
```

✧ 6.2 实例 7：七段数码管绘制

基本思路：

1) 绘制单个数字对应的数码管；
2) 获得一串数字，绘制对应的数码管；
3) 获得当前系统时间，绘制对应的数码管；

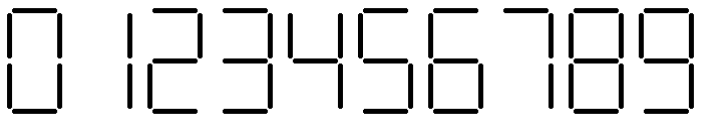


① 绘制一串数字

```
import turtle

def draw_gap():
    # 绘制数码管间隔
    turtle.penup()
    turtle.fd(5)

def draw_line(flag=True):
    # 绘制单段数码管，两端有间隔
    draw_gap()
    turtle.pendown() if flag else turtle.penup()
    turtle.fd(40)
    draw_gap()
    turtle.rt(90)

def draw_digit(digit):
    # 绘制一个数字的七段数码管
    # 按照顺序将七段数码管走一遍，根据数字决定每一段要不要画
```

```
    draw_line() if digit in [2, 3, 4, 5, 6, 8, 9] else draw_line(False)
    draw_line() if digit in [0, 1, 3, 4, 5, 6, 7, 8, 9] else draw_line(False)
    draw_line() if digit in [0, 2, 3, 5, 6, 8, 9] else draw_line(False)
    draw_line() if digit in [0, 2, 6, 8] else draw_line(False)
    turtle.lt(90)
    draw_line() if digit in [0, 4, 5, 6, 8, 9] else draw_line(False)
    draw_line() if digit in [0, 2, 3, 5, 6, 7, 8, 9] else draw_line(False)
    draw_line() if digit in [0, 1, 2, 3, 4, 7, 8, 9] else draw_line(False)
    turtle.lt(180)
    # 为绘制后续数字确定位置
    turtle.penup()
    turtle.fd(20)

def draw_date(date):
    for i in date:
        draw_digit(int(i))

def main():
    turtle.setup(800, 400, 200, 200)
    turtle.penup()
    turtle.fd(-350)
    turtle.pensize(5)
    draw_date('0123456789')
    turtle.hideturtle()  # 隐藏海龟
    turtle.done()

if __name__ == '__main__':
    main()
```

结果：

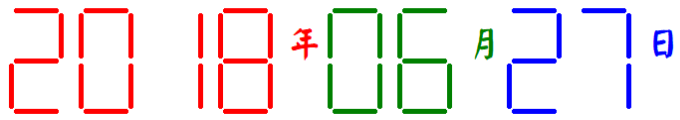


② 获取系统时间绘制七段数码管

```
import time

def draw_date(date):
    turtle.pencolor('red')
    font=('田氏颜体大字库', 24, 'normal')
    for i in date:
        if i == '年':
            turtle.write('年', font=font)
            turtle.pencolor('green')  # '年'后面为月份，字体颜色改为绿色
```

```
            turtle.fd(40)
        elif i == '月':
            turtle.write('月', font=font)
            turtle.pencolor('blue')  # '月'后面为日期，字体颜色改为蓝色
            turtle.fd(40)
        elif i == '日':
            turtle.write('日', font=font)
        else:  # 非'年月日'就是数字，调用绘制数字的函数
            draw_digit(int(i))

def main():
    turtle.setup(800, 400, 200, 200)
    turtle.penup()
    turtle.fd(-350)
    turtle.pensize(5)
    draw_date(time.strftime('%Y 年%m 月%d 日'))
    turtle.hideturtle()  # 隐藏海龟
    turtle.done()
```

结果：



✧ 6.3 PyInstaller 库

第三方库 PyInstaller 可以将.py 源代码转换成无需源代码的可执行文件。

pyinstaller -F "文件名.py"

常用参数

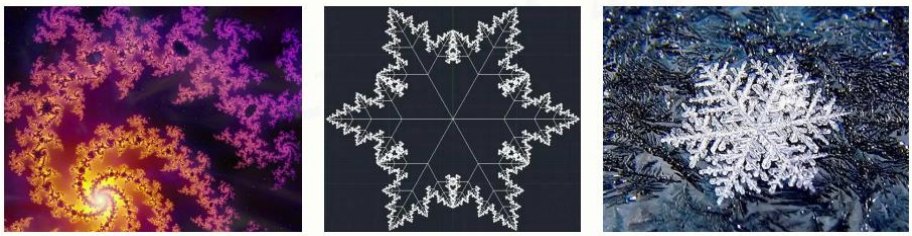
参数	描述
-h	查看帮助
--clean	清理打包过程中的临时文件
-D,--onedir	默认值，生成 dist 文件夹
-F,--onefile	在 dist 文件夹中只生成独立的打包文件
-i "图标文件名.ico"	指定打包程序使用的图标(icon)文件

pyinstaller -i curve.ico -F SevenDigitsDraw.py
生成带图标的可执行文件。

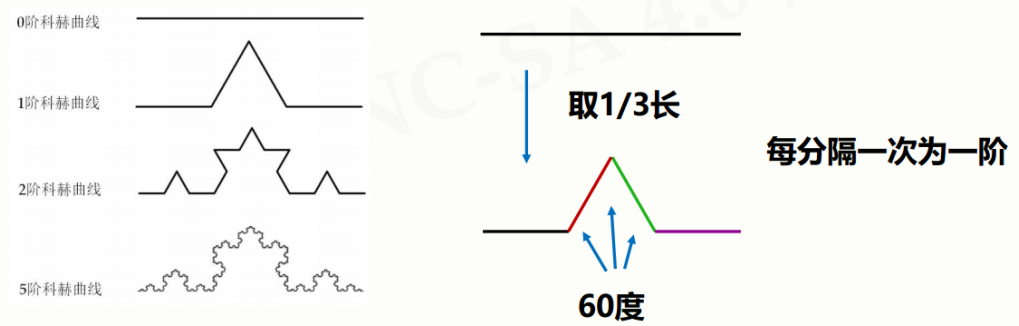
✧ 6.4 实例 8：科赫雪花小包裹

分形几何是一种迭代的几何图形，可理解为整体与局部存在相似性。

分形几何有一个特殊的曲线为科赫曲线，也叫雪花曲线。



科赫曲线，每一阶将所有线段撑开一次

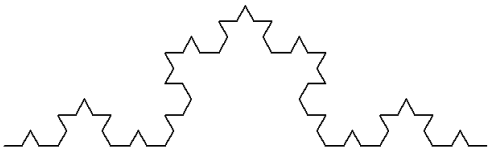


① 科赫曲线的绘制

```
import turtle

def koch(size, n):
    if n == 0:
        turtle.fd(size)
    else:
        # 分成 4 部分递归
        for angle in [0, 60, -120, 60]:
            turtle.left(angle)
            koch(size/3, n-1)

def main():
    turtle.setup(800, 400)
    turtle.penup()
    turtle.goto(-300, -50)
    turtle.pendown()
```

```
    turtle.pensize(2)
    koch(600, 3)  # 3 阶科赫曲线
    turtle.hideturtle()
    turtle.done()

if __name__ == '__main__':
    main()
```

3 阶科赫曲线：

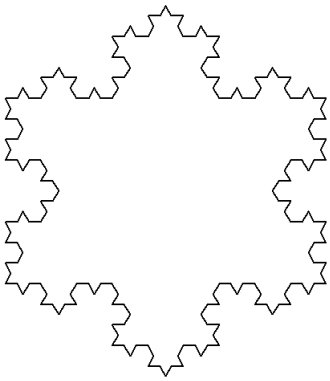


② 科赫雪花的绘制

以科赫曲线为边绘制闭合图形，比如以三角形为轮廓。

```
def draw_koch(level, size, n):
    for i in range(n):
        koch(size, level)
        turtle.right(360/n)

def main():
    turtle.setup(600, 600)
    turtle.penup()
    turtle.goto(-200, 100)
    turtle.pendown()
    turtle.pensize(2)
    level = 3  # 3 阶科赫曲线
    draw_koch(level, 400, 3)
    turtle.hideturtle()
    turtle.done()
```

结果：

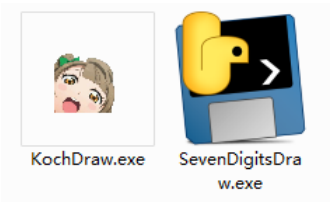


③ 打包

将编写的代码(.py 文件)打包处理

```
pyinstaller -i shortcuticon.ico -F KochDraw.py
```

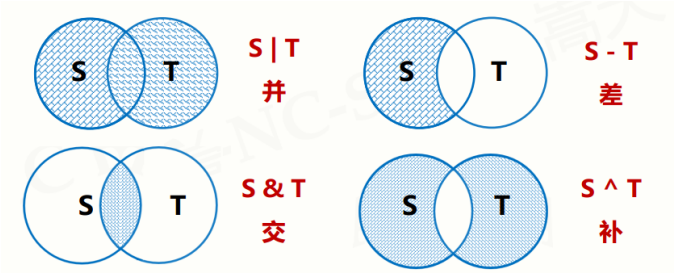
使用自定义图标和默认图标：



第 7 章 组合数据类型

✧ 7.1 集合 (set)

① 集合之间操作



1) S | T：并，返回一个新集合，S 和 T 中所有元素；
2) S - T：差，返回一个新集合，在 S 但不在 T 中的元素；
3) S & T：交，返回一个新集合，同时在 S 和 T 中的元素；
4) S ^ T：补，返回一个新集合，S 和 T 中的非相同元素。

```
>>> s = set('abc')
>>> t = {'a', 's'}
>>> s-t
{'b', 'c'}
>>> t-s
{'s'}
>>> s & t
{'a'}
>>> s | t
{'b', 'a', 's', 'c'}
>>> s ^ t
{'s', 'b', 'c'}
```

② 集合方法

1) add(x)：将 x(如果不存在)添加到集合；
2) discard(x)：集合中移出 x，如果不存在，不报错；
3) remove(x)：集合中移出 x，如果不存在，报错 KeyError；
4) clear()：清空集合元素；
5) pop()：随机弹出集合一个元素，如果集合为空报错 KeyError；

③ 集合应用

主要应用于元素去重：

```
>>> lst = [1, 2, 3, 4, 4, 3, 2, 3, 4, 5, 1, 6, 5, 4]
>>> list(set(lst))
[1, 2, 3, 4, 5, 6]
```

✧ 7.2 序列类型

列表、元组、字符串

✧ 7.3 实例 9：基本统计值计算

给出一组数，对它们求总个数、总和、平均值、方差、中位数等。

```python
from random import randint

def get_nums(n=10):  # 获取指定个数随机数
    n = n if n > 0 else 10
    return [randint(10, 99) for x in range(n)]

def average(nums):  # 计算平均值
    return sum(nums)/len(nums)

def dev(nums, avg):  # 计算方差
    s = 0
    for i in nums:
        s += (i-avg)**2
    return (s/(len(nums)-1))**0.5

def median(nums):  # 计算中位数
    nums_copy = sorted(nums)
    length = len(nums_copy)
    if length % 2 == 0:  # 长度为偶数,取中间两数平均
        med = (nums_copy[length//2-1]+nums_copy[length//2])/2
    else:  # 长度为奇数,取中间数
        med = nums_copy[length//2]
    return med

if __name__ == '__main__':
    nums = get_nums(12)
    avg = average(nums)
    print(nums)  # [13, 55, 56, 94, 41, 16, 55, 25, 16, 27, 58, 57]
    print('平均值:{:.2f}, 方差:{:.2f}, 中位数:{}'.format(
        avg, dev(nums, avg), median(nums)))
    # 平均值:42.75, 方差:24.16, 中位数:48.0
```

✧ 7.4 字典 (dict)

字典主要方法：

1) keys()：返回字典所有键；
2) values()：返回字典所有值；
3) items()：返回字典所有键值对；
4) get(k[, d])：键 k 存在返回对应值；否则返回 d 值；
5) pop(k[, d])：键 k 存在弹出对应值；否则返回 d 值；
6) popitems()：随机取出一个键值对，元组形式；
7) clear()：清空字典；

✧ 7.5 jieba 库

中文文本需要通过分词获得单个的词语，jieba 是优秀的中文分词第三方库。

① jieba 分词的原理：
1) 利用一个中文词库，确定汉字之间的关联概率；
2) 汉字间概率大的组成词组，形成分词结果；
3) 除了分词，用户还可以添加自定义的词组。

② jieba 库提供三种分词模式：
1) **精确模式**：把文本精确地切分开，不存在冗余单词；
2) 全模式：把文本中所有可能的词语都扫描出来，有冗余；
3) 搜索引擎模式：在精确模式基础上，对长词再次切分。

③ jieba 常用函数：
1) jieba.**lcut**(s)：精确模式，返回一个列表分词结果；
2) jieba.lcut(s, cut_all=True)：全模式，返回一个列表分词结果，存在冗余；
3) jieba.lcut_for_search(s)：搜索引擎模式，返回一个列表分词结果，存在冗余；
4) jieba.add_word(word)：向分词词典增加新词 word；

```
>>> import jieba
>>> s = '中华人民共和国'
>>> jieba.lcut(s)
['中华人民共和国']
>>> jieba.lcut(s, cut_all=True)
['中华', '中华人民', '中华人民共和国', '华人', '人民', '人民共和国', '共和', '共和国']
>>> jieba.lcut_for_search(s)
['中华', '华人', '人民', '共和', '共和国', '中华人民共和国']
```

✧ 7.6 实例 10：文本词频统计

需求：一篇文章，出现了哪些词？哪些词出现得最多？

① 英文文本：Hamet 分析词频

```
import requests

headers = {"User-Agent": "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;
Trident/5.0;"}
```

```
def get_text(url):
    txt = requests.get(url, headers=headers).text.lower()
    # 将一些字符转为空格
    for i in '!"#$%&()*+,-./:;<=>?@[\\]^_‘{|}~':
        txt = txt.replace(i, ' ')
    return txt

def cal_hamlet():
    url = 'https://python123.io/resources/pye/hamlet.txt'
    txt = get_text(url)  # 获取哈姆雷特英文文章
    words = txt.split()  # 按空格拆分单词
    dct = {}
    for i in words:
        dct[i] = dct.get(i, 0)+1
    items = list(dct.items())
    items.sort(key=lambda x: x[1], reverse=True)
    for i in range(10):  # 统计前十词频
        word, cnt = items[i]
        print('{:<10}{:>5}'.format(word, cnt))

if __name__ == '__main__':
    cal_hamlet()
```

结果：

```
the        1138
and         965
to          754
of          669
you         550
i           542
a           542
my          514
hamlet      462
in          436
```

② 中文文本：《三国演义》分析人物

1) 词频统计

```
import jieba

def cal_3kingdoms():
    url = 'https://python123.io/resources/pye/threekingdoms.txt'
    txt = requests.get(url, headers=headers).text
    words = jieba.lcut(txt)  # 中文分词
    dct = {}
    for i in words:
        i = i.strip()
        if len(i) < 2:  # 过滤单个字或空字符串
```

```
            continue
        dct[i] = dct.get(i, 0)+1
    items = list(dct.items())
    items.sort(key=lambda x: x[1], reverse=True)
    for i in range(15):  # 统计前 15 中文词语
        word, cnt = items[i]
        print('{:<10}{:>5}'.format(word, cnt))

if __name__ == '__main__':
    cal_3kingdoms()
```

结果：

```
曹操          953
孔明          836
将军          772
却说          656
玄德          585
关公          510
丞相          491
二人          469
不可          440
荆州          425
玄德曰         390
孔明曰         390
不能          384
如此          378
张飞          358
```

2) 人物出场统计

将词频与人物相关联，面向问题

```
def cal_3kingdoms():
    url = 'https://python123.io/resources/pye/threekingdoms.txt'
    txt = requests.get(url, headers=headers).text
    # 出现频率高但不是人物的词语
    excludes = {'将军', '却说', '荆州', '二人', '不可', '不能', '如此', '商议', '如何', '主公', '军士', '左右', '军马', '引兵', '次日', '大喜', '天下', '东吴', '于是', '今日', '不敢', '魏兵', '陛下', '一人', '都督', '人马', '不知', '汉中', '只见', '众将', '蜀兵', '上马', '大叫'}
    words = jieba.lcut(txt)  # 中文分词
    dct = {}
    for i in words:
        i = i.strip()
        if len(i) < 2:  # 过滤单个字或空字符串
            continue
        # 手动合并常见相同人物
        elif i in ['孔明', '孔明曰', '亮曰']:
            j = '诸葛亮'
        elif i in ['关公', '云长', '云长曰']:
            j = '关羽'
```

```
        elif i in ['玄德', '玄德曰', '刘皇叔']:
            j = '刘备'
        elif i in ['孟德', '丞相']:
            j = '曹操'
        elif i == '翼德':
            j = '张飞'
        elif i == '奉先':
            j = '吕布'
        else:
            j = i
        dct[j] = dct.get(j, 0)+1
    # 删除常见非人物词
    for i in excludes:
        del dct[i]
    items = list(dct.items())
    items.sort(key=lambda x: x[1], reverse=True)
    for i in range(10):  # 统计前 10 出现人物
        word, cnt = items[i]
        print('{:<10}{:>5}'.format(word, cnt))
```

结果：

```
曹操        1451
诸葛亮      1383
刘备        1288
关羽         784
张飞         376
吕布         304
赵云         278
孙权         264
司马懿       221
周瑜         217
```

20180628

第 8 章　文件的使用

✧　8.1　文件

文件是数据的抽象和集合，是数据存储的一种形式，是存储在辅助存储器上的数据序列，分为文本文件和二进制文件。

文本文件和二进制文件只是文件的展示方式，本质都是二进制形式存储的。

① 打开模式：

1) **'r'**：只读模式，默认值；如果文件不存在，返回 FileNotFoundError；
2) **'w'**：覆盖写模式，文件不存在则创建，存在则完全覆盖；
3) 'x'：创建写模式，文件不存在则创建，存在则返回 FileExistsError；
4) **'a'**：追加写模式，文件不存在则创建，存在则在文件最后追加内容；
5) **'b'**：二进制模式；
6) 't'：文本模式，默认值；
7) '+'：与 r/w/x/a 一同使用，在原功能基础上增加同时读写功能

② 文件读取：

1) `read(size=-1)`：读取 size 长度内容，默认全部；
2) `readline(size=-1)`：读取一行内容，给定 size 则读取该行前 size 长度；
3) `readlines(hint=-1)`：读取所有行为列表，每个元素对应一行，给定 hint 则读取前 hint 行；

③ 文件写入：

1) `write(s)`：向文件写入一个字符串或字节流；
2) `writelines(lines)`：将字符串列表写入文件；
3) `seek(offset)`：改变当前文件操作指针的位置；
   offset：0 表示文件开头，1 表示当前位置，2 表示文件结尾；

writelines()示例：

```
lst = ['rin', 'maki', 'nozomi']
with open('1.txt', 'w') as f:
    f.writelines(lst)
with open('1.txt') as f:
    print(f.read())  # rinmakinozomi
```

所以并没有在每个字符串之间添加换行，还需要手动追加，那 writelines()也就意义不大了...还不如使用 `f.write('\n'.join(lst))`

✧ 8.2 实例 11：自动轨迹绘制

需求：根据脚本来绘制图形

不是写代码，而是写数据绘制轨迹，数据脚本是自动化最重要的第一步。

1) 定义数据文件格式(接口)

格式自己随意定义，可以使用一行表示一次操作，比如格式为：

<forward_distance>,<turn_right?>,<angle>,<r>,<g>,<b>

如：300,0,144,1,0,0 表示画笔颜色为(1,0,0)，前进 300 左转 144°

2) 编写程序，根据文件接口解析参数绘制图形

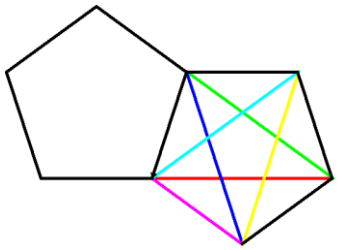
```
import turtle as t

def draw_trace():
    t.title('自动轨迹绘制')
    t.setup(800, 600, 0, 0)
    t.pensize(5)
    lst = []
    with open("data.txt") as f:  # 数据读取
        for line in f:
            lst.append(list(map(eval, line.strip().split(','))))
    for i in lst:  # 根据文件读取内容自动绘制
        t.pencolor(i[3], i[4], i[5])
        t.fd(i[0])
```

```
        t.right(i[2]) if i[1] else t.left(i[2])
    t.done()

if __name__ == '__main__':
    draw_trace()
```

3) 编制数据文件 data.txt：

```
300,0,144,1,0,0
300,0,144,0,1,0
300,0,144,0,0,1
300,0,144,1,1,0
300,0,108,0,1,1
184,0,72,1,0,1
184,0,72,0,0,0
184,0,72,0,0,0
184,0,72,0,0,0
184,1,72,1,0,1
184,1,72,0,0,0
184,1,72,0,0,0
184,1,72,0,0,0
184,1,72,0,0,0
184,1,720,0,0,0
```

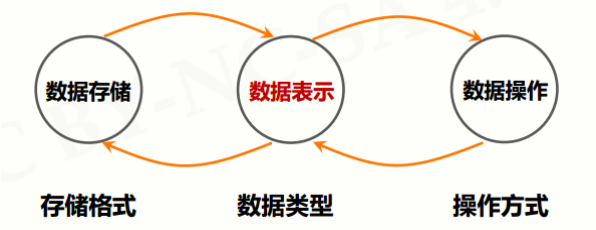
结果：



自动化思维：数据和功能分离，数据驱动的自动运行；
接口化设计：格式化设计接口；
二维数据应用：应用维度组织数据，二维数据(如二维列表)最常用。

✧ 8.3 一维数据格式化和处理

数据的操作周期：存储 <-> 表示 <-> 操作



存储到文件：
使用空格或逗号等符号作为分隔符，缺点是数据中不能有此特殊符号。
一般一维数据就是采用特殊符号作为分隔符存储到文件。

✧ 8.4 二维数据

使用列表嵌套表示二维数据，如[[1, 2], [3, 4], [5, 6]]

CSV(Comma-Separated Values)数据存储格式
国际通用的一二维数据存储格式，扩展名一般为.csv；每行一个一维数据，采用逗号分隔，无空行；Excel 软件可读写.csv。

如左边表格使用 CSV 描述可以转为右边：

城市	环比	同比	定基
北京	101.5	120.7	121.4
上海	101.2	127.3	127.8
广州	101.3	119.4	120.0
深圳	102.0	140.0	145.5
沈阳	100.0	101.4	101.6

```
城市,环比,同比,定基
北京,101.5,120.7,121.4
上海,101.2,127.3,127.8
广州,101.3,119.4,120.0
深圳,102.0,140.0,145.5
沈阳,100.0,101.4,101.6
```

注意：
1) 如果某个元素缺失，逗号仍要保留；
2) 二维数据的表头可以作为数据存储，也可以另行存储；
3) 逗号为英文半角逗号，逗号与数据之间无额外空格。

```
s = '5:rin:55|6:maki:99|7:nozomi:78'
lst = [x.split(':') for x in s.split('|')]
print(lst)  # [['5', 'rin', '55'], ['6', 'maki', '99'], ['7', 'nozomi', '78']]
with open('data.csv', 'w') as f:
    for i in lst:
        f.write(','.join(i)+'\n')
lst1 = []
with open('data.csv') as f:
    for line in f:
        lst1.append(line.strip().split(','))
print(lst1 == lst)  # True
```

data.csv 使用 excel 打开：

	A	B	C
1	5	rin	55
2	6	maki	99
3	7	nozomi	78

20180701

✧ 8.5 wordcloud 库
wordcloud 是优秀的词云展示第三方库，词云以词语为基本单位，更加直观和艺术的展示文本。

win7 使用 pip 安装报错，提示需要安装 C++14...
网上建议到 https://www.lfd.uci.edu/~gohlke/pythonlibs/#wordcloud 下载编译好的.whl 文件，但是 404 Not Found...

没办法，在 Ubuntu 虚拟机基于 Python 3.5 的虚拟环境安装 wordcloud。

wordcloud 库把词云当作一个 WordCloud 对象。
wordcloud.WordCloud()：代表一个文本对应的词云，可以根据文本中词语出现的频率等参数绘制词云，绘制词云的形状、尺寸和颜色都可以设定。
WordCloud 构造方法参数：

1) width：指定词云对象生成图片的宽度，默认 400 像素；
2) height：指定词云对象生成图片的高度，默认 200 像素；
3) min_font_size：指定词云中字体的最小字号，默认 4 号；
4) max_font_size：指定词云中字体的最大字号，根据高度自动调节；
5) font_step：指定词云中字体字号的步进间隔，默认为 1；
6) font_path：指定字体文件的路径，默认 None；
7) max_words：指定词云显示的最大单词数量，默认 200；
8) stop_words：指定词云的排除集合(set)，即不显示的单词；
9) mask：指定词云形状，默认长方形，需要引用 imread()函数；
10) background_color：指定词云图片的背景颜色，默认为黑色；

WordCloud 对象常用方法：
1) generate(txt)：向 WordCloud 对象中加载文本 txt；
2) to_file(filename)：将词云输出为图像文件.png 或.jpg 格式；

示例：英文生成词云图片

```
import wordcloud

def get_hamlet():
    url = 'https://python123.io/resources/pye/hamlet.txt'
    txt = get_text(url)  # 获取哈姆雷特英文文章
    w = wordcloud.WordCloud(background_color='white',
                            max_words=10, width=600, height=300)
    w.generate(txt)
    w.to_file('hamlet.png')

if __name__ == '__main__':
    get_hamlet()
```

运行报错：

```
ImportError: No module named '_tkinter', please install the python3-tk package
```

根据提示安装 python3-tk：

```
sudo apt-get install python3-tk
```

结果：

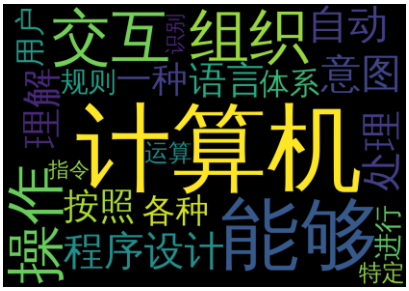
示例：中文生成词云图片

```
import jieba
import wordcloud
txt = "程序设计语言是计算机能够理解和识别用户操作意图的一种交互体系，它按照特定规则组织计算机指令，使计算机能够自动进行各种运算处理。"
w = wordcloud.WordCloud(width=1000, height=700, font_path="wqy-zenhei.ttc")
w.generate(" ".join(jieba.lcut(txt)))
w.to_file("pywcloud.png")
```

安装文泉驿正黑字体：

```
sudo apt-get install ttf-wqy-zenhei
```

结果：



✧ 8.6 实例 12：政府工作报告词云

1) 习近平《决胜全面建成小康社会 夺取新时代中国特色社会主义伟大胜利》，在中国共产党第十九次全国代表大会上的报告(2017 年 10 月 18 日)

2) 中共中央国务院《中共中央 国务院关于实施乡村振兴战略的意见》，2018 一号文件(2018 年 01 月 02 日)

步骤：

1) 读取文件、分词整理；2) 设置并输出词云；3) 观察结果，优化迭代。

```
import requests
import time
import jieba
import wordcloud

headers = {"User-Agent": "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0;"}

def get_text(url):
    while True:
        try:
            txt = requests.get(url, headers=headers).text
        except:
            time.sleep(2)
        else:
            return txt
```

```
def get_word_cloud(url):
    txt = ' '.join(jieba.lcut(get_text(url)))  # 获取文本
    w = wordcloud.WordCloud(font_path='wqy-zenhei.ttc', width=1000,
                            height=700, background_color="white", max_words=15)
    w.generate(txt)  # 词云对象加载文本
    filename = url.split('/')[-1].split('.')[0]
    w.to_file('{}.png'.format(filename))  # 词云输出为图片

if __name__ == '__main__':
    url1 = 'https://python123.io/resources/pye/新时代中国特色社会主义.txt'
    url2 = 'https://python123.io/resources/pye/关于实施乡村振兴战略的意见.txt'
    get_word_cloud(url1)
    get_word_cloud(url2)
```

结果：

词云自定义形状

```
import jieba
import wordcloud
from scipy.misc import imread、

def get_word_cloud(url, img):
    mask = imread(img)  # 指定词云背景
    txt = ' '.join(jieba.lcut(get_text(url)))
    # 使用 mask 后指定宽高貌似没用了，生成图片为 mask 图片宽高
    w = wordcloud.WordCloud(font_path='wqy-zenhei.ttc', width=1000,
                            height=700, background_color="white", mask=mask)
    w.generate(txt)
    filename = url.split('/')[-1].split('.')[0]
    w.to_file('{}.png'.format(filename))

if __name__ == '__main__':
    url1 = 'https://python123.io/resources/pye/新时代中国特色社会主义.txt'
    url2 = 'https://python123.io/resources/pye/关于实施乡村振兴战略的意见.txt'
    img = '1.png'
    get_word_cloud(url1, img)
    get_word_cloud(url2, img)
```

结果：

第 9 章　程序设计方法

✧　9.1　自顶向下和自底向上

①　自顶向下(设计)

将一个总问题表达为若干个小问题组成的形式，使用同样方法进一步分解小问题，直至小问题可以用计算机简单明了的解决。自顶向下是解决复杂问题的有效方法。自顶向下的设计思维：分而治之。

②　自底向上(执行)

分单元测试，逐步组装，按照自顶向下相反的路径操作，直至系统各部分以组装的思路都经过测试和验证。自底向上是逐步组建复杂系统的有效测试方法。自底向上的执行思维：模块化集成。

自顶向下是系统思维的简化。

✧　9.2　实例 13：体育竞技分析

高手过招，胜负只在毫厘之间。毫厘是多少？如何科学分析体育竞技比赛？

输入：球员的水平；输出：可预测的比赛成绩。

体育竞技分析：模拟 n 场比赛

1) 计算思维：抽象 + 自动化；
2) 模拟：抽象比赛过程 + 自动化执行 N 场比赛；
3) 当 n 越大时，比赛结果分析会越科学；

比赛规则：

1) 双人击球比赛：A & B，回合制；
2) 开始时一方先发球，直至判分，接下来胜者发球；
3) 球员只能在发球局得分，15 分胜一局；

步骤：

1) 打印程序的介绍性信息；-> print_info()
2) 获得程序运行参数：能力值 a 和 b, n；-> get_input()
3) 利用球员 A 和 B 的能力值，模拟 n 局比赛；-> sim_n_game()

4) 输出球员 A 和 B 获胜比赛的场次及概率；-> print_result()
其中步骤 3 模拟 n 局比赛可以分解为 n 次模拟 1 局比赛。->sim_one_game()
模拟 1 局比赛可以抽出 game_over()函数判断 1 局比赛是否结束。

```
from random import randint

class Game():
    def __init__(self):
        self.a, self.b = 0, 0
        self.n = 0
        self.wina, self.winb = 0, 0

    def print_info(self):  # 打印介绍信息
        print('这个程序模拟两个选手 A 和 B 的某种竞技比赛')
        print('程序运行需要 A 和 B 的能力值(0~100)')

    def get_input(self):  # 获取输入
        self.a = eval(input('请输入选手 A 的能力值(0~100): '))
        self.b = eval(input('请输入选手 B 的能力值(0~100): '))
        self.n = eval(input("模拟比赛的场次: "))

    def sim_n_game(self):  # 模拟 n 局比赛
        for i in range(self.n):
            scorea, scoreb = self.sim_one_game()
            if scorea > scoreb:
                self.wina += 1
            else:
                self.winb += 1

    def sim_one_game(self):  # 模拟 1 局比赛
        scorea, scoreb = 0, 0
        isA = True  # A 先手
        while not self.game_over(scorea, scoreb):
            # 发球,如果随机值在能力范围就得分,否则就算失败,交出发球权
            if isA:
                if randint(0, 100) <= self.a:
                    scorea += 1
                else:
                    isA = False
            else:
                if randint(0, 100) <= self.b:
                    scoreb += 1
                else:
                    isA = True
```

```
        return scorea, scoreb

    def game_over(self, scorea, scoreb):
        # 有人分数达到 15,一局比赛就结束
        return scorea == 15 or scoreb == 15

    def print_result(self):  # 打印结果
        print("竞技分析开始，共模拟{}场比赛".format(self.n))
        pa = self.wina/self.n
        print("选手 A 获胜{}场比赛，占比{:0.1%}".format(self.wina, pa))
        print("选手 B 获胜{}场比赛，占比{:0.1%}".format(self.winb, 1-pa))

    def __call__(self):  # 流程控制
        self.print_info()
        self.get_input()
        self.sim_n_game()
        self.print_result()

if __name__ == '__main__':
    Game()()
```

结果：

```
这个程序模拟两个选手 A 和 B 的某种竞技比赛
程序运行需要 A 和 B 的能力值(0~100)
请输入选手 A 的能力值(0~100): 80
请输入选手 B 的能力值(0~100): 85
模拟比赛的场次: 1000
竞技分析开始，共模拟 1000 场比赛
选手 A 获胜 412 场比赛，占比 41.2%
选手 B 获胜 588 场比赛，占比 58.8%
```

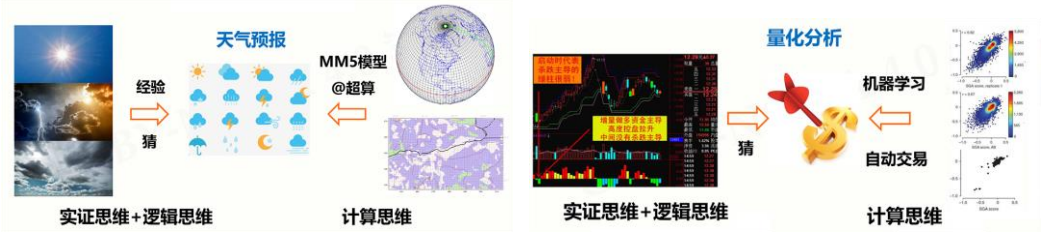
✧ 9.3 Python 程序设计思维

人类思维特征：

1) 逻辑思维：推理和演绎，数学为代表；
2) 实证思维：实验和验证，物理为代表；
3) 计算思维：设计和构造，计算机为代表。

计算思维是人类第 3 中思维特征。

计算思维(Computational Thinking)特点是抽象和自动化，抽象问题的计算过程，利用计算机自动化求解。计算思维是基于计算机的思维方式。



如上面天气预报和买卖股票，之前都是通过实证思维和逻辑思维根据经验或猜想得出，如今借助计算思维，可以使用计算机自动求解。

抽象问题的计算过程，是利用计算机自动化求解的。计算思维基于计算机强大的算力及海量数据；抽象计算过程，关注设计和构造，而非因果；
计算思维以计算机程序设计为实现的主要手段。

编程是将计算思维变成现实的手段：

✧ 9.4 计算生态与 Python 语言

① 开源运动

1983 年，Richard Stallman 启动 GNU 项目，目标是创建一套完全自由的操作系统；1989 年，GNU 通用许可协议诞生，自由软件时代到来。
1991 年，Linus Torvalds 发布了 Linux 内核；1998 年，网景浏览器开源，产生了 Mozilla，开源生态逐步建立。

开源思想深入演化和发展，形成了计算生态。计算生态以开源项目为组织形式，充分利用共识原则和社会利他组织人员，在竞争发展、相互依存和迅速更迭中完成信息技术的更新换代，形成了技术的自我演化路径。

计算生态没有顶层设计，以功能为单位，具备三个特点：
1) 竞争发展；2) 相互依存；3) 迅速更迭；

② 计算生态与 Python 语言

1) Python 提供了超过 13 万个第三方库。
2) 库的建设经过野蛮生长和自然选择，同一个功能，Python 往往有 2 个以上第三方库。
3) 库之间相互关联使用，依存发展。Python 库间广泛联系，逐级封装。
4) 社区庞大，新技术更迭迅速，AlphaGo 深度学习算法采用 Python 语言开源。

使用 Python 编程，不需要刀耕火种，应该站在巨人的肩膀上。
编程的起点不是算法而是系统；编程如同搭积木，利用计算生态为主要模式；
编程的目标是快速解决问题。

✧ 9.5 用户体验与及软件产品

① 用户体验

用户体验是用户对产品建立的主观感受和认识；关心功能实现，更要关心用户体验，才能做出好产品。编程只是手段，不是目的，程序最终为人类服务。
用户体验是程序到产品的关键环节。

② 提高用户体验的方法
1) 进度展示
比如程序可能产生等待，存在大量循环次数，可以增加进度展示。
2) 异常处理
比如检查用户输入的合法性，读取文件如果文件不存在，需要异常处理。
3) 其他类方法
打印输出：打印程序运行的过程信息；
日志文件：对程序异常及用户使用进行定期记录；
帮助信息：给用户多种方式提供帮助信息。

✧ 9.6 基本的程序设计模式
① 自顶向下设计
② 模块化设计
通过函数或对象封装将程序划分为模块及模块间的表达。
采用分而治之、分层抽象、体系化的设计思想。
模块内部紧耦合、模块之间松耦合。
③ 配置化设计
程序引擎+配置文件：程序执行和配置分离，将可选参数配置化。
将程序开发变成配置文件编写，不修改程序，只修改配置文件就可以修改功能。关键在于接口设计，需要清晰明了，灵活可扩展。

✧ 9.7 应用开发的四个步骤
从应用需求到软件产品：
1) 产品定义：对应用需求充分理解和明确定义；
不仅是功能定义，要考虑商业模式。
2) 系统架构：以系统方式思考产品的技术实现；
系统架构，关注数据流、模块化、体系架构。
3) 设计与实现：结合架构完成关键设计及系统实现；
结合可扩展性、灵活性等进行设计优化。
4) 用户体验：从用户角度思考应用效果；
用户至上，体验优先，以用户为中心。

✧ 9.8 Python 第三方库安装
① Python 社区
PyPI：Python Package Index
PyPI 社区是 PSF 维护的、展示全球 Python 计算生态的主站。
利用 PyPI 可以找到合适的第三方库开发程序。

② 安装 Python 第三方库 3 种方法：
1) 使用 pip 命令，主要方法(>99%)，不同操作系统都适用；
2) 集成安装方法，结合特定 Python 开发工具的批量安装；
如 Anaconda，其支持近 800 个第三方库，包含多个主流工具，适合数据计算领域开发。

3) 文件安装方法；
有些第三方库用 pip 可以下载，但无法安装...
因为它们 pip 下载的是源代码，需要编译再安装；但如果操作系统没有编译环境，则安装失败；可以直接下载编译后的.whl 文件用于安装。

UCI 页面：http://www.lfd.uci.edu/~gohlke/pythonlibs/
然而目前基本全是 **404 - File or directory not found.**

③ 常用 pip 命令

```
pip -h：查看 pip 帮助信息；
pip install <package>：安装某个第三方库；
pip install –U <package>：更新已安装的某个第三方库；
pip uninstall <package>：卸载某个第三方库；
pip download <package>：下载但不安装某个第三方库；
pip show <package>：列出某个第三方库的详细信息；
pip search <keyword>：根据关键词在名称和介绍中搜索第三方库；
pip list：列出经安装的第三方库，格式：包名\t+版本号；
pip freeze：列出安装的第三方库，格式：包名==版本号，少一些必须有的包；
```

✧ 9.9 os 库

os 库是 Python 标准库，提供通用的操作系统交互功能。

① 路径操作：os.path 子库，处理文件路径及信息；

```
1) os.path.abspath(path)：返回 path 在当前系统中的绝对路径；
2) os.path.normpath(path)：归一化 path 的表示形式，统一用\\分隔路径；
3) os.path.relpath(path)：返回当前程序与文件之间的相对路径；
4) os.path.dirname(path)：返回 path 中的目录名称；
5) os.path.basename(path)：返回 path 中最后的文件名称；
6) os.path.join(path, *paths)：组合 path 与 paths，返回一个路径字符串；
7) os.path.exists(path)：判断 path 对应文件或目录是否存在；
8) os.path.isfile(path)：判断 path 是否为已存在的文件；
9) os.path.isdir(path)：判断 path 是否为已存在的目录；
10) os.path.getatime(path)：返回 path 对应文件或目录上一次的访问(access)时间；
11) os.path.getmtime(path)：返回 path 对应文件或目录最近一次的修改(modify)时间；
12) os.path.getctime(path)：返回 path 对应文件或目录的创建(create)时间；
13) os.path.getsize(path)：返回 path 对应文件的大小，以字节为单位；
```

② 进程管理：启动系统中其他程序；
os.system(command)：执行程序或命令 command；在 Windows 系统中，返回值为 cmd 的调用返回信息。

使用 Python 调用 ffmpeg 合并视频：

```
import os
name = 'output'
```

```
cmd = 'ffmpeg -f concat -i file.txt -c copy "{}".mp4'.format(name)
os.system(cmd)
```

③ 环境参数：获得系统软硬件信息等环境参数；

1) os.chdir(path)：修改程序操作的路径为 path；

2) os.getcwd()：返回程序的当前路径；

3) os.getlogin()：获得当前系统登录用户名称；

4) os.cpu_count()：获得当前系统的 CPU 数量；

5) os.urandom(n)：获得 n 个字节长度的随机字节字符串，通常用于加解密运算；

```
>>> os.getcwd()
'D:\\software\\python'
>>> os.getlogin()
'hikari 星'
>>> os.cpu_count()
4
>>> os.urandom(6)
b'P\xbe\xaaH\xf1\x95'
```

✧ 9.10 实例 14：第三方库自动安装脚本

需要安装的 20 个第三方库：

库名	用途	pip 安装指令
NumPy	N 维数据表示和运算	pip install numpy
Matplotlib	二维数据可视化	pip install matplotlib
PIL	图像处理	pip install pillow
Scikit-Learn	机器学习和数据挖掘	pip install sklearn
Requests	HTTP 协议访问及网络爬虫	pip install requests
Jieba	中文分词	pip install jieba
Beautiful Soup	HTML 和 XML 解析器	pip install beautifulsoup4
Wheel	Python 第三方库文件打包工具	pip install wheel
PyInstaller	打包 Python 源文件为可执行文件	pip install pyinstaller
Django	Python 最流行的 Web 开发框架	pip install django
Flask	轻量级 Web 开发框架	pip install flask
WeRoBot	微信机器人开发框架	pip install werobot
SymPy	数学符号计算工具	pip install sympy
Pandas	高效数据分析和计算	pip install pandas
Networkx	复杂网络和图结构的建模和分析	pip install networkx
PyQt5	基于 Qt 的专业级 GUI 开发框架	pip install pyqt5
PyOpenGL	多平台 OpenGL 开发接口	pip install pyopengl
PyPDF2	PDF 文件内容提取及处理	pip install pypdf2
docopt	Python 命令行解析	pip install docopt
PyGame	简单小游戏开发框架	pip install pygame

将需要安装的库名写入 req.txt 文件，一行一个。

```
import os


def batch_install(file):
    libs = set()  # set 去重
    with open(file) as f:  # 从文件读取需要安装的库
```

```
        for line in f:
            libs.add(line.strip())
    for mod in libs:
        # 拼接命令，执行命令
        cmd = 'pip install {}'.format(mod)
        try:
            os.system(cmd)
        except:
            print('{}安装失败!'.format(mod))
        else:
            print('{}安装成功!'.format(mod))

def main():
    file = 'req.txt'
    batch_install(file)

if __name__ == '__main__':
    main()
```

一般部署时，从自己电脑 pip freeze > requirements.txt 导出需要的第三方库，然后在服务器的某个虚拟环境安装 pip install -r requirements.txt

第 10 章　Python 计算生态概览

✧　10.1　从数据处理到人工智能

数据表示->数据清洗->数据统计->数据可视化->数据挖掘->人工智能

1) 数据表示：采用合适方式用程序表达数据；
2) 数据清理：数据归一化、数据转换、异常值处理；
3) 数据统计：数据的概要理解，数量、分布、中位数等；
4) 数据可视化：直观展示数据内涵的方式；
5) 数据挖掘：从数据分析获得知识，产生数据外的价值；
6) 人工智能：数据/语言/图像/视觉等方面深度分析与决策。

✧　10.2　Python 库之数据分析

① Numpy：表达 N 维数组的最基础库

提供直接的矩阵运算、广播函数、线性代数等功能，是 Python 数据分析及科学计算的基础库，支撑 Pandas 等其他库。C 语言实现，计算速度优异。

```
import numpy as np

def py_sum(a, b):
    c = []
    for i in range(len(a)):
        c.append(a[i]**2+b[i]**3)
    return c
```

```python
def np_sum(a, b):
    a = np.array(a)
    b = np.array(b)
    return a**2+b**3

def main():
    a = [0, 1, 2, 3, 4]
    b = [9, 8, 7, 6, 5]
    print(py_sum(a, b))  # [729, 513, 347, 225, 141]
    print(np_sum(a, b))  # [729 513 347 225 141]

if __name__ == '__main__':
    main()
```

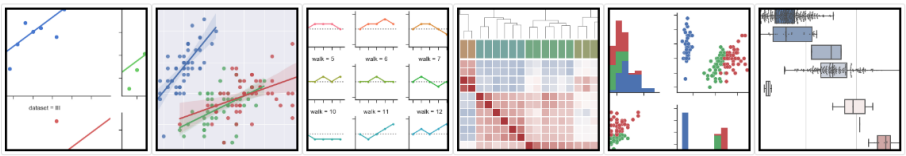
② Pandas：Python 数据分析高层次应用库
Python 最主要的数据分析功能库，基于 Numpy 开发；提供了简单易用的数据结构和数据分析工具；理解数据类型与索引的关系，操作索引即操作数据。
两个数据结构：
Series = 索引 + 一维数据
DataFrame = 行列索引 + 二维数据

③ SciPy：数学、科学和工程计算功能库
Python 最主要的科学计算功能库，基于 Numy 开发；提供了一批数学算法及工程数据运算功能，类似 Matlab，可用于如傅里叶变换、信号处理等应用。

✧ 10.3 Python 库之数据可视化

① Matplotlib：高质量的二维数据可视化功能库
Python 最主要的数据可视化功能库，基于 Numpy 开发；提供了超过 100 种数据可视化展示效果；可以通过 matplotlib.pyplot 子库调用各可视化效果。

② Seaborn：统计类数据可视化功能库
基于 Matplotlib 开发，支持 Numpy 和 Pandas；提供了一批高层次的统计类数据可视化展示效果，主要展示数据间分布、分类和线性关系等内容。



③ Mayavi：三维科学数据可视化功能库

三维可视化最主要的第三方库，目前版本是 Mayavi2；其提供了一批简单易用的 3D 科学计算数据可视化展示效果；支持 Numpy、TVTK、Traits、Envisage 等第三方库。

✧ 10.4 Python 库之文本处理

① PyPDF2：处理 pdf 文件的工具集

完全 Python 语言实现，不需要额外依赖，功能稳定；提供了一批处理 PDF 文件的计算功能；支持获取信息、分隔/整合文件、加密解密等。

```
from PyPDF2 import PdfFileReader, PdfFileMerger

def merge_pdf(file1, file2):
    merger = PdfFileMerger()
    # file1 的 3~5 页(0 开始计数)
    merger.append(file1, pages=(3, 6))
    # file2 的 0~1 页插入到 merger 的第 3 页(0 开始计数)
    merger.merge(3, file2, pages=(0, 2))
    # 结果 5 页页面是:file1 的 p4、p5、p6 和 file2 的 p1、p2(此处页码 p1 开始)
    merger.write('merge.pdf')

def main():
    merge_pdf('10.pdf', '11.pdf')
```

报错：

```
  ...
  File "D:\software\python\lib\site-packages\PyPDF2\utils.py", line 238, in b
    r = s.encode('latin-1')
UnicodeEncodeError: 'latin-1' codec can't encode characters in position 8-9:
```

直接将源码 utils.py 的 r = s.encode('latin-1')的 latin-1 改为 utf-8 居然就成功了...

合并两个 pdf 文件：

```
def merge_pdf(file1, file2):
    merger = PdfFileMerger()
    merger.append(file2)
    # 按照 file1+file2 的顺序拼接两个 pdf 文件
    merger.merge(0, fileobj=file1)
    merger.write('merge.pdf')
```

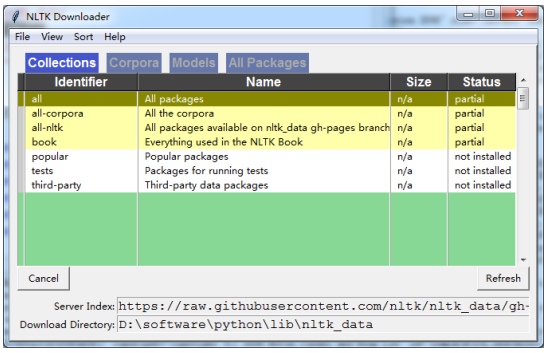
② NLTK：自然语言文本处理第三方库

最优秀的 Python 自然语言处理库，提供了一批简单易用的自然语言文本处理功能，支持语言文本分类、标记、语法句法、语义分析等。

使用 nltk 之前需要使用 NLTK Downloader 下载需要的 package，然后使用：

```
>>> import nltk
>>> nltk.download()
```

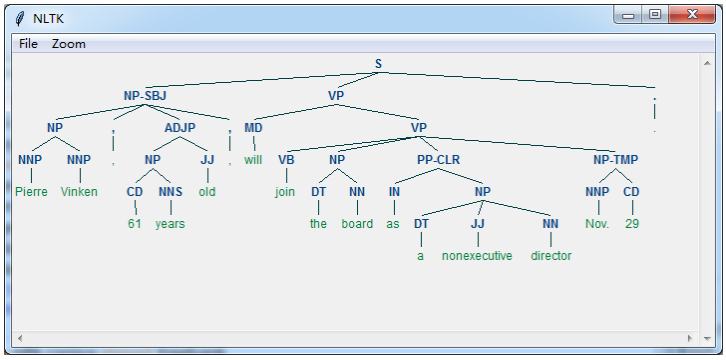
跳出界面：



不知道下载什么，所以下载了 all...

NLTK 官方示例：

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)  # 分词
>>> tagged = nltk.pos_tag(tokens)  # 词性标注
>>> entities = nltk.chunk.ne_chunk(tagged)  # 命名实体识别
>>> entities
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'NN'), ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'), Tree('PERSON', [('Arthur', 'NNP')]), ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'), ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
>>> from nltk.corpus import treebank
>>> t = treebank.parsed_sents('wsj_0001.mrg')[0]
>>> t.draw()
```



// 完全不懂额...

③ Python-docx：创建或更新 Microsoft Word 文件的第三方库
提供创建或更新.doc 和.docx 等文件的功能；可以增加并配置段落、图片、表格、文字等，功能全面。

python-docx 官方示例小改：

```
from docx import Document
from docx.shared import Inches
```

```
class Person(dict):
    def __init__(self, id_, name, age, group, birthday):
        super().__init__(id=id_, name=name, age=age, group=group,
birthday=birthday)

def write_to_docx(lst, attrs):
    document = Document()
    document.add_heading('Document Title', 0)  # 标题
    p = document.add_paragraph('A plain paragraph having some ')  # 段落
    p.add_run('bold').bold = True  # 字体加粗
    p.add_run(' and some ')
    p.add_run('italic.').italic = True  # 字体倾斜
    document.add_heading('Heading, level 1', level=1)
    document.add_paragraph('Intense quote', style='Intense Quote')
    document.add_paragraph( # 点状项目符号
        'first item in unordered list', style='List Bullet'
    )
    document.add_paragraph(  # 数字项目符号
        'first item in ordered list', style='List Number'
    )
    # 添加图片
    pic='E:/LoveLive!查卡器/合并/1599&1615t.png'
    document.add_picture(pic, width=Inches(2))
    # 绘制表格,指定初始行数和列数和表格样式
    table = document.add_table(rows=1, cols=len(attrs),
                               style='Medium Grid 1 Accent 1')
    hdr_cells = table.rows[0].cells  # 表头
    for i in range(len(attrs)):  # 表头文字
        hdr_cells[i].text = attrs[i]
    for p in lst:
        # 一个 Person 对象对应一行数据
        row_cells = table.add_row().cells
        for i in range(len(attrs)):
            row_cells[i].text = str(p[attrs[i]])
    document.add_page_break()  # 添加分页符
    document.save('demo.docx')  # 保存文件

def main():
    lst = [
        Person(5, '星空凛', 15, 'lily white', '11 月 1 日'),
        Person(6, '西木野真姬', 15, 'BiBi', '4 月 19 日'),
        Person(7, '東條希', 17, 'lily white', '6 月 9 日'),
        Person(3, '南ことり', 16, 'Printemps', '9 月 12 日'),
```

```
    ]
    attrs = ['id', 'name', 'age', 'group', 'birthday']
    write_to_docx(lst, attrs)

if __name__ == '__main__':
    main()
```

结果：

Document Title

A plain paragraph having some **bold** and some *italic.*

Heading, level 1

Intense quote

- first item in unordered list

1. first item in ordered list

id	name	age	group	birthday
5	星空凛	15	lily white	11 月 1 日
6	西木野真姬	15	BiBi	4 月 19 日
7	東條希	17	lily white	6 月 9 日
3	南ことり	16	Printemps	9 月 12 日

✧ 10.5 Python 库之机器学习

① Scikit-learn：机器学习方法工具集

机器学习最基本且最优秀的 Python 第三方库；提供一批统一化的机器学习方法功能接口；提供聚类、分类、回归、强化学习等计算功能。

② TensorFlow： AlphaGo 背后的机器学习计算框架

谷歌公司推动的开源机器学习框架；应用机器学习方法的一种方式，支撑谷歌人工智能应用。

将数据流图作为基础，图节点代表运算，边代表张量。

安装失败...原因可能是本机使用 Python 3.6.5，而 TensorFlow 不支持 3.6?

好吧，Ubuntu 虚拟机的 Python 3.5 安装成功...

```
import tensorflow as tf
import numpy as np

# 使用 NumPy 生成假数据(phony data), 总共 100 个点.
x_data = np.float32(np.random.rand(2, 100))  # 随机输入
y_data = np.dot([0.100, 0.200], x_data) + 0.300
```

```
# 构造一个线性模型
b = tf.Variable(tf.zeros([1]))
W = tf.Variable(tf.random_uniform([1, 2], -1.0, 1.0))
y = tf.matmul(W, x_data) + b

# 最小化方差
loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

# 初始化变量
init = tf.global_variables_initializer()

# 启动图(graph)
sess = tf.Session()
sess.run(init)

# 拟合平面
for step in range(0, 201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))
```

结果：

```
0 [[ 0.571042   -0.18406683]] [0.56918967]
20 [[0.14783837 0.08737614]] [0.3352073]
40 [[0.10088515 0.1698456 ]] [0.31565422]
60 [[0.09693602 0.19052537]] [0.3066643]
80 [[0.09811203 0.19663034]] [0.30278713]
100 [[0.09910753 0.19870777]] [0.30115682]
120 [[0.09961111 0.19948465]] [0.30047855]
140 [[0.09983579 0.1997906 ]] [0.3001977]
160 [[0.09993156 0.19991419]] [0.3000816]
180 [[0.09997163 0.1999647 ]] [0.3000337]
200 [[0.09998827 0.19998546]] [0.3000139]
```

得到最佳拟合结果 W: [[0.100 0.200]], b: [0.300]

③ MXNet：基于神经网络的深度学习计算框架

Python 最重要的深度学习计算框架；提供可扩展的神经网络及深度学习计算功能；可用于自动驾驶、机器翻译、语音识别等众多领域。

✧ 10.6 实例 15：霍兰德人格分析雷达图

霍兰德认为：人格兴趣与职业之间应有一种内在的对应关系；

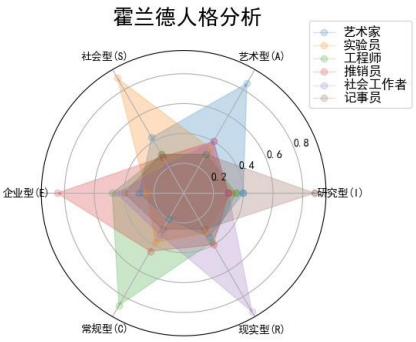
人格分类：研究型、艺术型、社会型、企业型、传统型、现实性；
职业：工程师、实验员、艺术家、推销员、记事员、社会工作者；

需求：雷达图方式验证霍兰德人格分析；雷达图(Radar Chart)是多特性直观展示的重要方式。

输入：各职业人群结合兴趣的调研数据；输出：雷达图
绘制雷达图需要 matplotlib 库；专业的多维数据表示需要 numpy 库。

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family'] = 'SimHei'
radar_labels = np.array(['研究型(I)', '艺术型(A)', '社会型(S)',
                         '企业型(E)', '常规型(C)', '现实型(R)'])
data = np.array([[0.40, 0.32, 0.35, 0.30, 0.30, 0.88],
                 [0.85, 0.35, 0.30, 0.40, 0.40, 0.30],
                 [0.43, 0.89, 0.30, 0.28, 0.22, 0.30],
                 [0.30, 0.25, 0.48, 0.85, 0.45, 0.40],
                 [0.20, 0.38, 0.87, 0.45, 0.32, 0.28],
                 [0.34, 0.31, 0.38, 0.40, 0.92, 0.28]])  # 数据值
data_labels = ('艺术家', '实验员', '工程师', '推销员', '社会工作者', '记事员')
angles = np.linspace(0, 2*np.pi, 6, endpoint=False)
data = np.concatenate((data, [data[0]]))
angles = np.concatenate((angles, [angles[0]]))
fig = plt.figure(facecolor="white")
plt.subplot(111, polar=True)
plt.plot(angles, data, 'o-', linewidth=1, alpha=0.2)
plt.fill(angles, data, alpha=0.25)
plt.thetagrids(angles*180/np.pi, radar_labels, frac=1.2)
plt.figtext(0.52, 0.95, '霍兰德人格分析', ha='center', size=20)
legend = plt.legend(data_labels, loc=(0.94, 0.80), labelspacing=0.1)
plt.setp(legend.get_texts(), fontsize='large')
plt.grid(True)
plt.savefig('holland_radar.jpg')
plt.show()
```

结果：



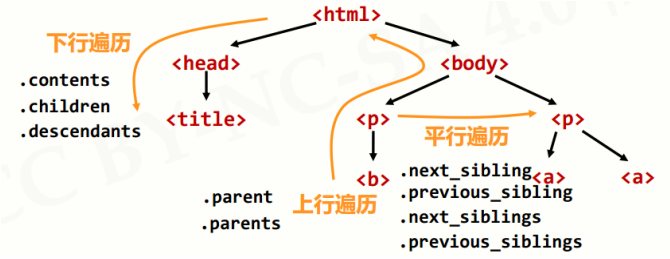
编程不是主要的，编程感觉是重点：
目标 + 沉浸 + 熟练
1) 编程的目标感：寻找感兴趣的目标，寻(wa)觅(jue)之；
2) 编程的沉浸感：寻找可实现的方法，思(zhuo)考(mo)之；
3) 编程的熟练度：练习、练习、再练习，熟练之；

✧ 10.7 Python 库之网络爬虫
① Requests：最友好的网络爬虫功能库
提供了简单易用的类 HTTP 协议网络爬虫功能；是 Python 最主要的页面级网络爬虫功能库；支持连接池、SSL、Cookies、HTTP(S)代理等。

② Scrapy：优秀的网络爬虫框架
Python 最主要且最专业的网络爬虫框架；提供构建网络爬虫系统的框架(功能半成品)功能；支持批量和定时网页爬取、提供数据处理流程等。

③ pyspider：强大的 Web 页面爬取系统
Python 重要的网络爬虫类第三方库；提供了完整的网页爬取系统构建功能；支持数据库后端、消息队列、优先级、分布式架构等。

✧ 10.8 Python 库之 Web 信息提取
① Beautiful Soup：HTML 和 XML 的解析库
提供了解析 HTML 和 XML 等 Web 信息的功能，可以加载多种解析引擎；常与网络爬虫库搭配使用，如 Scrapy、requests 等。

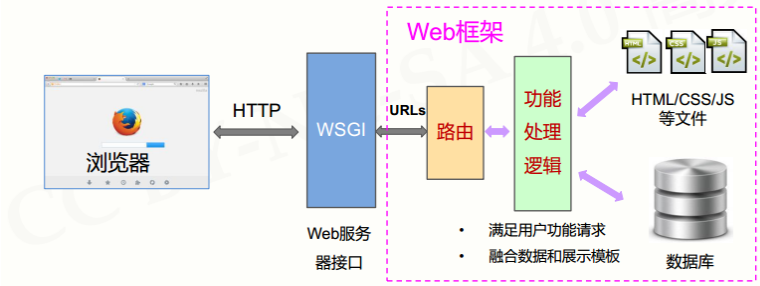


② Re：正则表达式解析和处理功能库
Python 最主要的标准库之一；提供了定义和解析正则表达式的一批通用功能；可用于各类场景，包括定点的 Web 信息提取。
常用方法：
re.search()；re.match()；re.findall()；re.split()；re.finditer()；re.sub()

③ Python-Goose：提取文章类型 Web 页面的功能库
Python 最主要的 Web 信息提取库；提供了对 Web 页面中文章信息/视频等元数据的提取功能；针对特定类型 Web 页面，应用覆盖面较广。

✧ 10.9 Python 库之 Web 网站开发
① Django：最流行的 Web 应用框架

Python 最重要的 Web 应用框架，略微复杂；提供了构建 Web 系统的基本应用框架；基于 MTV 模式：Model、Template、Views。



② Pyramid：规模适中的 Web 应用框架

Python 产品级 Web 应用框架，起步简单可扩展性好；不大不小，规模适中，适合快速构建并适度扩展类应用。

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response

def hello_world(request):
    return Response('<h2 style="color:red">Hello World!</h2>')

if __name__ == '__main__':
    with Configurator() as config:
        # 添加路由,为其注册视图
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('127.0.0.1', 6543, app)
    server.serve_forever()
```

③ Flask：Web 应用开发微框架

提供了最简单构建 Web 系统的应用框架；特点是：简单、规模小、快速。

// 忘得差不多了，见狗书...

✧ 10.10 Python 库之网络应用开发

① WeRoBot：微信公众号开发框架

提供了解析微信服务器消息及反馈消息的功能；

是建立微信机器人的重要技术手段。

② aip：百度 AI 开放平台接口

Python 百度 AI 应用的最主要方式。提供了访问百度 AI 服务的 Python 功能接口，涉及语音、人脸、OCR、NLP、知识图谱、图像搜索等领域。

③ MyQR：二维码生成第三方库

提供了生成二维码的系列功能：基本二维码、艺术二维码和动态二维码。

使用方法：

```
myqr  Words # url 链接
        [-v {1,2,3,...,40}] # 控制编程,范围 1~40
        [-l {L,M,Q,H}] # url 控制水平纠错
        [-n output-filename] # 定义输出名字,默认 qrcode.png
        [-d output-directory] # 定义输出目录,默认当前目录
        [-p picture_file] # 指定与二维码相结合的图片,生成艺术二维码
        [-c] # 加上-c 是彩色,反之为黑白
        [-con contrast] # 调节图片对比度,默认 1.0 表示原始图片
        [-bri brightness] # 调节图片亮度,默认 1.0 表示原始图片
```

示例：

```
$ myqr https://www.github.com -p github.png
$ myqr https://qr.alipay.com/tsx08680cr8mngoa9vzsd60 -p 456t.png -c
```

结果：

✧ 10.11 Python 库之图形用户界面

① PyQt5：Qt 开发框架的 Python 接口

推荐的 Python GUI 开发第三方库；提供了创建 Qt5 程序的 Python API 接口；Qt 是非常成熟的跨平台桌面应用开发系统，完备 GUI。

② wxPython：跨平台 GUI 开发框架

Python 最主要的数据分析功能库，基于 Numpy 开发；提供了专用于 Python 的跨平台 GUI 开发框架；理解数据类型与索引的关系，操作索引即操作数据。

③ PyGObject：使用 GTK+开发 GUI 的功能库

提供了整合 GTK+、WebKitGTK+等库的功能；GTK+是跨平台的一种用户图形界面 GUI 框架；实例：Anaconda 采用该库构建 GUI。

✧ 10.12 Python 库之游戏开发

① PyGame：简单的游戏开发功能库

Python 游戏入门最主要的第三方库。提供了基于 SDL 的简单游戏开发功能及实现引擎；理解游戏对外部输入的响应机制及角色构建和交互机制。

② Panda3D：开源、跨平台的 3D 渲染和游戏开发库

由迪士尼和卡尼基梅隆大学共同开发。是一个 3D 游戏引擎，提供 Python 和 C++两种接口；支持很多先进特性：法线贴图、光泽贴图、卡通渲染等。

③ cocos2d：构建 2D 游戏和图形界面交互式应用的框架

提供了基于 OpenGL 的游戏开发图形渲染功能；支持 GPU 加速，采用树形结构分层管理游戏对象类型；适用于 2D 专业级游戏开发。

✧ 10.13 Python 库之虚拟现实

① VR Zero：在树莓派上开发 VR 应用的 Python 库
非常适合初学者实践 VR 开发及应用；提供大量与 VR 开发相关的功能；针对树莓派的 VR 开发库，支持设备小型化，配置简单化。

② pyovr：Oculus Rift 的 Python 开发接口
针对 Oculus VR 设备的 Python 开发库；基于成熟的 VR 设备，提供全套文档，工业级应用设备；Python+虚拟现实领域探索的一种思路。

③ Vizard：基于 Python 的通用 VR 开发引擎
专业的企业级虚拟现实开发引擎；提供详细的官方文档；支持多种主流的 VR 硬件设备，具有一定通用性。

✧ 10.14 Python 库之图形艺术

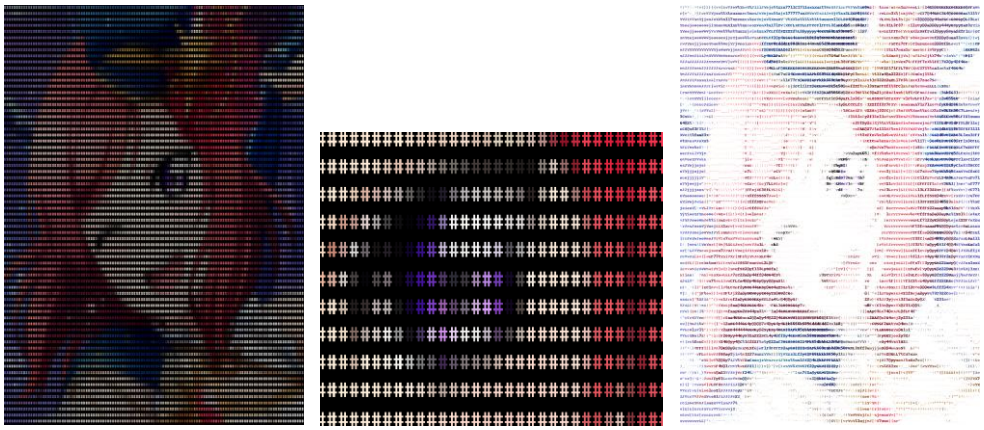
① Quads：迭代的艺术
对图片进行四分迭代，形成像素风；可以生成动图或静图图像；简单易用，具有很高展示度。

② ascii_art: ASCII 艺术库
将普通图片转为 ASCII 艺术风格；输出可以是纯文本或彩色文本；可采用图片格式输出。

```
from ascii_art.ascii import ASCIIArt
from ascii_art.picture import ASCIIPicture

def draw_ascii(name, scale):
    pic = ASCIIArt(name, scale=scale).draw_ascii(curve=1)
    pre = '{}_scale{}'.format(name, scale)
    new_name = '{}_draw_ascii'.format(pre)
    # 1.白底黑字
    ASCIIPicture(pic).save('{}.png'.format(new_name))
    # 2.保存为 txt 文件
    with open('{}.txt'.format(new_name),'w') as f:
        f.write(''.join(pic))
    # 3. 在黑色背景上使用排序的自定义字符集(09215)进行彩色的 ASCII 绘图
    colored_pic = ASCIIArt(name, scale).draw_color_ascii(
        ASCIIArt.sort('09215'))
    ASCIIPicture(colored_pic, 'black').save('{}_color_numbers'.format(pre))
     # 4. 在黑色背景上使用排序的自定义字符集(jontonsoup4)进行彩色的 ASCII 绘图
    colored_pic = ASCIIArt(name, scale).draw_color_ascii(
        ASCIIArt.sort('jontonsoup4'))
    ASCIIPicture(colored_pic, 'black').save('{}_color_name'.format(pre))
```

```
    # 5. 使用'maki'作为黑色背景的字符集将 ASCII 转到 HTML
    html = ASCIIArt(name, scale).draw_html(
        ASCIIArt.sort('maki'), background_color='black')
    with open('{}_html_maki.html'.format(pre), 'w') as f:
        f.write(''.join(html))
    # 6. 仅使用'#'作为黑色背景的字符将 ASCII 转为 HTML
    html = ASCIIArt(name, scale).draw_html(
        ASCIIArt.BLOCK, background_color='black')
    with open('{}_html_block.html'.format(pre), 'w') as f:
        f.write(''.join(html))
    # 7. 黑色背景上只有'#'的彩色 ASCII
    colored_pic = ASCIIArt(name, scale).draw_color_ascii(
        ASCIIArt.BLOCK, curve=1.5)
    ASCIIPicture(colored_pic, 'black').save('{}_block_color.png'.format(pre))
    # 8. 彩色的 ASCII,全灰度
    colored_pic = ASCIIArt(name, scale).draw_color_ascii(
        ASCIIArt.FULL_RANGE, curve=1.5)
    ASCIIPicture(colored_pic).save('{}_full_range_color.png'.format(pre))

def main():
    draw_ascii('maki', 2)

if __name__ == '__main__':
    main()
```

结果：



③ turtle: 海龟绘图体系

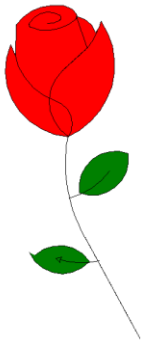
✧ 10.15 实例 16：玫瑰花绘制

需求：用 Python 绘制一朵玫瑰花，献给所思所念

输入：你的想象力！输出：玫瑰花

绘制机理：turtle 基本图形绘制；

绘制思想：因人而异；思想有多大、世界就有多大。



// 代码没什么意义，主要还是思想！

艺术之于编程，设计之于编程
1) 艺术：思想优先，编程是手段；
2) 设计：想法和编程同等重要；
3) 工程：编程优先，思想次之。

编程不重要，思想才重要！
1) 认识自己：明确自己的目标，有自己的思想(想法)；
2) 方式方法：编程只是手段，熟练之，未雨绸缪为思想服务；
3) 为谁编程：将自身发展与祖国发展相结合，创造真正价值。

maki_scale2_draw_ascii.txt 缩小后：