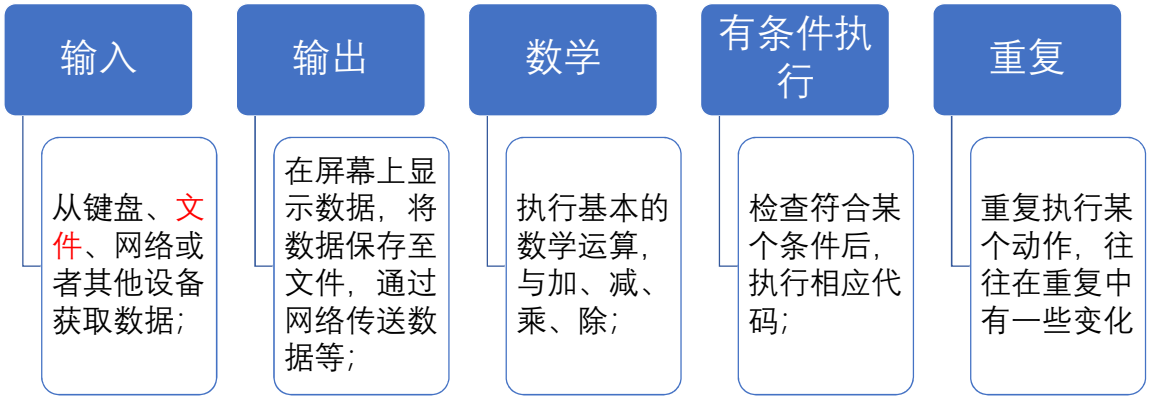
程序设计

算法

- 选择排序
- 冒泡排序
- 顺序查找
- 二分查找

程序的五大基本指令



文件概述

文件概述

文件是一个存储在辅助存储器上的数据序列，可以包含任何数据内容。概念上，文件是数据的集合和抽象（类似地，函数是程序的集合和抽象）。用文件形式组织和表达数据更有效也更为灵活。文件包括两种类型：文本文件和二进制文件。

文件概述

- 文本文件：只包含基本文本字符，不包含字体、大小和颜色信
- 二进制文件：是所有其他文件类型，诸如字处理文档、PDF、图像和可执行程序等。
- 二进制文件和文本文件最主要的区别：是否有统一的字符编码。二进制文件直接由比特0和比特1组成，没有统一字符编码，文件内部数据的组织格式与文件用途有关。
- 无论文件创建为文本文件或者二进制文件，都可以用"文本文件方式"和"二进制文件方式"打开，打开后的操作不同。

文件概述——字符编码

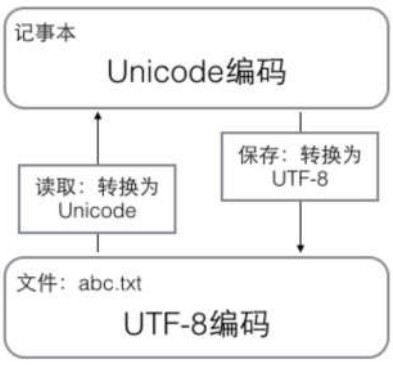
编码：表示符号的过程；

常见编码：ASCII，Unicode，GBK，Shift_JIS等；

为什么要编码：计算机只能处理数字，如果要处理文本，就必须先把文本转换为数字才能处理；

文件概述——字符编码

字符	ASCII	Unicode	UTF-8
A	01000001	00000000 01000001	01000001
中	x	01001110 00101101	11100100 10111000 10101101



文件概述

- 采用文本方式读入文件，文件经过编码形成字符串，打印出有含义的字符；
- 采用二进制方式打开文件，文件被解析为字节（byte）流。由于存在编码，字符串中的一个字符由2个字节表示。

```
>>>
中国是个伟大的国家!
b'\xd6\xd0\xb9\xfa\xca\xc7\xb8\xf6\xce\xb0\xb4
\xf3\xb5\xc4\xb9\xfa\xbc\xd2\xa3\xa1'
```

文件的打开与关闭

文件打开与关闭

Python对文本文件和二进制文件采用统一的操作步骤，即"打开-操作-关闭"

文件打开与关闭

在 Python 中， 读写文件有 3 个步骤：

1. 调用 open()函数， 返回一个 File 对象。
2. 调用 File 对象的 read()或 write()方法。
3. 调用 File 对象的 close()方法，关闭该文件

文件打开

Python通过解释器内置的open()函数打开一个文件，格式如下：

<变量名> = open(<文件名>, <打开模式>)

open()函数有两个参数：文件名和打开模式。文件名可以是文件的实际名字，也可以是包含完整路径的名字。注意在打开一个文件时，要向open()函数传递一个字符串路径，表明希望打开的文件。

文件打开——路径

文件有两个关键属性："文件名"（通常写成一个单词）和"路径"。路径指明了文件在计算机上的位置。

文件打开——路径

制定一个文件的路径有两种方法：

- 绝对路径：总是从根文件夹开始
- 相对路径：它相对于程序的当前工作目录

```
# 读取整个文件
File1 = open('D:\\Hello.txt')
File2 = open('Hello.txt')
```

文件打开——打开模式

打开模式	含义
'r'	只读模式，如果文件不存在，返回异常FileNotFoundError，默认值
'w'	覆盖写模式，文件不存在则创建，存在则完全覆盖源文件
'x'	创建写模式，文件不存在则创建，存在则返回异常FileExistsError
'a'	追加写模式，文件不存在则创建，存在则在原文件最后追加内容
'b'	二进制文件模式
't'	文本文件模式，默认值
'+'	与r/w/x/a一同使用，在原功能基础上增加同时读写功能

文件的读写

文件读入

要使用文本文件中的信息，首先需要将信息读取到内存中。为此，你可以一次性读取文件的全部内容，也可以以每次一行的方式逐步读取：

1. read()方法就返回保存在该文件中的字符串（将整个文件的内容读取为一个字符串）；
2. readlines()方法，从该文件取得一个字符串的列表，每个元素就是文本中的每一行；

文件读入

```
In [4]: File2 = open('Hello.txt')

In [5]: File2.read()
Out[5]: '锄禾日当午，\n汗滴禾下土。\n谁知盘中餐，\n粒粒皆辛苦。'

In [6]: File2.close()

In [7]: File2 = open('Hello.txt')

In [8]: File2.readlines()
Out[8]: ['锄禾日当午，\n', '汗滴禾下土。\n', '谁知盘中餐，\n', '粒粒皆辛苦。']

In [9]:
```

文件读入

如何进行文本逐行打印?

```
# 进行文本逐行打印:
fo = open('hello.txt','r')
for line in fo.readlines():
    print(line)
fo.close()

#
fo = open('hello.txt','r')
for line in fo:
    print(line)
fo.close()
```

```
In [12]: runfile('C:/Users/
锄禾日当午，

汗滴禾下土。

谁知盘中餐，

粒粒皆辛苦。
```

文件写入

Python允许你将内容写入文件，方式与 print()函数将字符串"写" 到屏幕上类似。但是，如果打开文件时用读模式，就不能写入文件。你需要以"写入纯文本模式"或"添加纯文本模式"打开该文件，或简称为"覆盖写模式" 和"添加模式"。

文件写入——覆盖写模式

- 将'w'作为第二个参数传递给 open()，以写模式打开该文件。
- 将覆写原有的文件，从头开始，就像你用一个新值覆写一个变量的值。
- 如果传递给open()的文件名不存在，创建一个新的空文件。

文件写入——覆盖写模式

```
# 文件写入—覆盖写模式
fo = open('bacon.txt','w')
fo.write('ABCDEF\n')
fo.close()
```

文件写入——追加写模式

- 在已有文件的末尾添加文本。
- 将'a'作为第二个参数传递给 open()，以添加模式打开该文件。
- 如果传递给open()的文件名不存在，创建一个新的空文件。

```
# 文件写入—追加写模式
fo = open('bacon.txt','a')
fo.write('ABCDEF\n')
fo.close()
```

文件写入——向文件写入列表

```
# 案例：向文件写入列表
fo = open('new_bacon.txt','w+')
ls = ['唐诗','宋词','元曲']
fo.writelines(ls)
for line in fo:
    print(line)
fo.close()
```

文件写入——向文件写入列表

```
# 案例：向文件写入列表
fo = open('new_bacon.txt','w+')
ls = ['唐诗','宋词','元曲']
fo.writelines(ls)
for line in fo:
    print(line)
fo.close()
```

```
# 案例：向文件写入列表
fo = open('new_bacon.txt','w+')
ls = ['唐诗','宋词','元曲']
fo.writelines(ls)
fo.seek(0)
for line in fo:
    print(line)
fo.close()
```

文件读写——混合打开模式

常见打开模式：r, w, a, r+, w+, a+

如果打开文档不存在，r会报错，w和a则会创建新文档

详见教程： https://www.runoob.com/python3/python3-file-methods.html

Recall：文件打开与关闭

在 Python 中， 读写文件有 3 个步骤：

1. 调用 open()函数， 返回一个 File 对象。
2. 调用 File 对象的 read()或 write()方法。
3. 调用 File 对象的 close()方法，关闭该文件

文件读写——with方法

Python的with语句可自动调用close()方法；

```
# with语句打开文件
with open('new_bacon.txt', 'r') as f:
    print(f.read())
```

凯撒密码

利用凯撒密码对数据进行加密，并写入文件。

凯撒密码：是一种替换加密的技术，明文中的所有字母都在字母表上向后（或向前）按照一个固定数目进行偏移后被替换成密文。例如，当偏移量是 3 的时候，所有的字母 A 将被替换成 D，B 变成 E，以此类推。

例子：

原文：A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

密文：D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

要求：分别完成 4 个函数以及测试用例：加密 enCaesar(s, n)，解密 deCaesar(s, n)，读文件 readFile(filename)，和写文件 writeFile(filename, s)

其中参数：s 为字符串，n 为偏移数，filename 为文件名。

1）随机生成由 26 个英文字母组成的一串字符，字符个数自定。

2）调用 enCaesar 函数对其进行加密，调用 writeFile 函数将密文写入文件中。

3）调用 readFile 函数将密文从文件中读入，调用函数 deCaesar 进行解密并打印输出。

PIL库的使用

PIL库概述

PIL（Python Image Library）库是Python语言的第三方库，支持图像存储、显示和处理，它能够处理几乎所有图片格式，可以完成对图像的缩放、剪裁、叠加以及向图像添加线条、图像和文字等操作。

如何安装?

PIL库概述

PIL库可以完成图像归档和图像处理两方面功能需求：

- 图像归档：对图像进行批处理、生成图像预览、图像格式转换等；
- 图像处理：图像基本处理、像素处理、颜色处理等。

Image子库

在PIL中，任何一个图像文件都可以用Image对象表示Image类的图像读取和创建方法。

方法	描述
Image.open(filename)	根据参数加载图像文件
Image.new(mode, size, color)	根据给定参数创建一个新的图像
Image.open(StringIO.StringIO(buffer))	从字符串中获取图像
Image.frombytes(mode, size, data)	根据像素点data创建图像
Image.verify()	对图像文件完整性进行检查，返回异常

Image子库——打开图像文件

图像的栅格数据不会被直接解码或者加载；

程序只是读取了图像文件头部的元数据信息；

```
from PIL import Image
im = Image.open("D:\\marvel.jpg")
print(im.format, im.size, im.mode)
```

Image子库——打开图像文件

Image类有4个处理图片的常用属性

属性	描述
Image.format	标识图像格式或来源，如果图像不是从文件读取，值是None
Image.mode	图像的色彩模式，"L"灰度图像、"RGB"真彩色图像、"CMYK"出版图像
Image.size	图像宽度和高度，单位是像素（px），返回值是二元元组（tuple）
Image.palette	调色板属性，返回一个ImagePalette类型

Image子库——GIF文件提取

微实例7.1：GIF文件图像提取。

对一个GIF格式动态文件，提取其中各帧图像，并保存为文件。

Image子库——图像转换与保存

方法	描述
Image.save(filename, format)	将图像保存为filename文件名，format是图片格式
Image.convert(mode)	使用不同的参数，转换图像为新的模式
Image.thumbnail(size)	创建图像的缩略图，size是缩略图尺寸的二元元组

Image子库——图像转换与保存

生成"birdnest.jpg"图像的缩略图，（128，128）是缩略图的尺寸。

```
>>>im.thumbnail((128, 128))
>>>im.save("birdnestTN","JPEG")
```

Image子库——图像缩放与旋转

生成"birdnest.jpg"图像的缩略图，（128，128）是缩略图的尺寸。

方法	描述
Image.resize(size)	按size大小调整图像，生成副本
Image.rotate(angle)	按angle角度旋转图像，生成副本

Image子库——颜色交换

Image类能够对每个像素点或者一幅RGB图像的每个通道单独进行操作,split()方法能够将RGB图像各颜色通道提取出来，merge()方法能够将各独立通道再合成一幅新的图像。

方法	描述
Image.point(func)	根据函数func功能对每个元素进行运算，返回图像副本
Image.split()	提取RGB图像的每个颜色通道，返回图像副本
Image.merge(mode，bands)	合并通道，采用mode色彩，bands是新色的色彩通道
Image.blend(im1,im2,alpha)	将两幅图片im1和im2按照如下公式插值后生成新的图像： im1 * (1.0-alpha) + im2 * alpha

Image子库——颜色交换

微实例7.2：图像的颜色交换。

交换图像中的颜色。可以通过分离RGB图片的三个颜色通道实现颜色交换

微实例7.2　　m7.1ChangeRGB.py

```
1 from PIL import Image
2 im = Image.open('birdnest.jpg')
3 r, g, b = im.split()
4 om = Image.merge("RGB", (b, g, r))
5 om.save('birdnestBGR.jpg')
```

Image子库——颜色交换

微实例7.2：图像的颜色交换。

交换图像中的颜色。可以通过分离RGB图片的三个颜色通道实现颜色交换

Image子库——颜色交换

操作图像的每个像素点需要通过函数实现，采用lambda函数和point()方法搭配使用，例子如下

```
>>>im = Image.open('D:\\pycodes\\birdnest.jpg') #打开鸟巢文件
>>>r, g, b = im.split() #获得RGB通道数据
>>>newg = g.point(lambda i: i * 0.9) # 将G通道颜色值变为原来的0.9倍
>>>newb = b.point(lambda i: i < 100) # 选择B通道值低于100的像素点
>>>om = Image.merge(im.mode, (r, newg, newb)) # 将3个通道合形成新图像
>>>om.save('D:\\pycodes\\birdnestMerge.jpg') #输出图片
```

Image子库——颜色交换

操作图像的每个像素点需要通过函数实现，采用lambda函数和point()方法搭配使用，例子如下

图像的过滤和增强

PIL库的ImageFilter类和ImageEnhance类提供了过滤图像和增强图像的方法，共10种

方法表示	描述
ImageFilter.BLUR	图像的模糊效果
ImageFilter.CONTOUR	图像的轮廓效果
ImageFilter.DETAIL	图像的细节效果
ImageFilter.EDGE_ENHANCE	图像的边界加强效果
ImageFilter.EDGE_ENHANCE_MORE	图像的阈值边界加强效果
ImageFilter.EMBOSS	图像的浮雕效果
ImageFilter.FIND_EDGES	图像的边界效果
ImageFilter.SMOOTH	图像的平滑效果
ImageFilter.SMOOTH_MORE	图像的阈值平滑效果
ImageFilter.SHARPEN	图像的锐化效果

图像的过滤和增强

利用Image类的filter()方法可以使用ImageFilter类，如下：

Image.filter(ImageFilter.fuction)

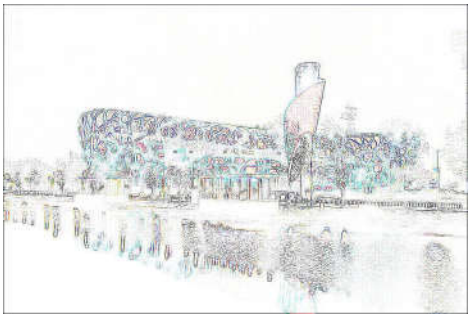
微实例7.3：图像的轮廓获取。

获取图像的轮廓，北京鸟巢变得更加抽象、更具想象空间！

图像的过滤和增强

微实例7.3　　m7.3GetImageContour.py

```
1 from PIL import Image
2 from PIL import ImageFilter
3 im = Image.open('birdnest.jpg')
4 om = im.filter(ImageFilter.CONTOUR)
5 om.save('birdnestContour.jpg')
```



北京鸟巢图片的轮廓效果

图像的过滤和增强

ImageEnhance类提供了更高级的图像增强需求，它提供调整色彩度、亮度、对比度、锐化等功能。

方法	描述
ImageEnhance.enhance(factor)	对选择属性的数值增强factor倍
ImageEnhance.Color(im)	调整图像的颜色平衡
ImageEnhance.Contrast(im)	调整图像的对比度
ImageEnhance.Brightness(im)	调整图像的亮度
ImageEnhance.Sharpness(im)	调整图像的锐度

图像的过滤和增强

微实例7.4：图像的对比度增强。

增强图像的对比度为初始的20倍。

微实例7.4　m7.4EnImageContrast.py

```
1 from PIL import Image
2 from PIL import ImageEnhance
3 im = Image.open('birdnest.jpg')
4 om = ImageEnhance.Contrast(im)
5 om.enhance(20).save('birdnestEnContrast.jpg')
```

图像的过滤和增强

北京鸟巢图片的20倍对比度增强效果

图像字符画绘制

图像字符画绘制

位图图片是由不同颜色像素点组成的规则分布，如果采用字符串代替像素，图像就成为了字符画。

定义一个字符集，将这个字符集替代图像中的像素点。

```
ascii_char =list("$@B%8&WM#*oahkbdpqwmZO0QLCJUYXzcvunxrjf\
           1234568795t/\|()1{}[]?-_+~<>i!;:,\"^`'.")
```

图像字符画绘制

定义彩色向灰度的转换公式如下，其中R、G、B分别是像素点的RGB颜色值：

Gray = R * 0.2126 + G * 0.7152 + B * 0.0722

因此，像素的RGB颜色值与字符集的对应函数如下：

```
def get_char(r, b, g, alpha=256):
  if alpha == 0:
      return ' '
  gray  =  int(0.2126  *  r  +  0.7152  *  g  +  0.0722  *  b)
    unit = 256 / len(ascii_char)

    return ascii_char[gray//unit]
```

实例代码12.1 e12.DrawCharImage.py

```
#e12.1DrawCharImage.py.py
from PIL import Image
ascii_char  = list('"$%_&WM#*oahkbdpqwmZO0QLCJUYXzcvunxr\
jft/\|()1{}[]?-/+@<>i!;:,\^`.')
def get_char(r, b, g, alpha=256):
    if alpha == 0:
        return ' '
    gray = int(0.2126 * r + 0.7152 * g + 0.0722 * b)
    unit = 256 / len(ascii_char)
    return ascii_char[int(gray//unit)]
def main():
    im = Image.open('pic.jpg')
    WIDTH, HEIGHT = 100, 60
    im = im.resize((WIDTH, HEIGHT))
    txt = ""
    for i in range(HEIGHT):
        for j in range(WIDTH):
            txt += get_char(*im.getpixel((j, i)))
        txt += '\n'
    fo = open("pic_char.txt","w")
    fo.write(txt)
    fo.close()
main()
```

图像字符画绘制