Python 程序设计基础

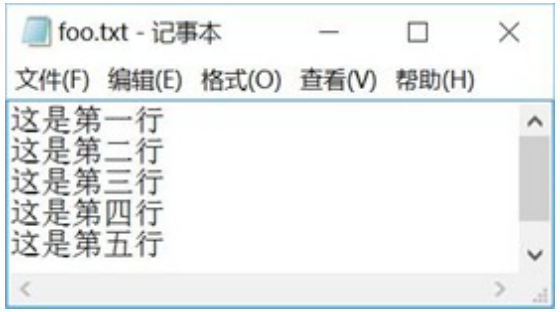
Python Programming

文件读写

- **文本文件读写。**
  - **往文件中写入数据。**

方法	描述
write(str)	写入 str 字符串，不会自动在字符串末尾添加 '\n' 字符
writelines(sequence)	写入 sequence 字符串列表，不会自动在字符串末尾添加 '\n' 字符

```
# 写入数据
fo = open("foo.txt", "w")      # 打开文件
fo.write("这是第一行\n")
fo.write("这是第二行\n")
fo.write("这是第三行\n")
fo.write("这是第四行\n")
fo.write("这是第五行\n")
fo.close()                     # 关闭文件
```



文件读写

- 文件读写时都有可能产生 IOError 异常，一旦出错，后面的 close 方法就不会被调用。为了保证无论是否出错都能正确地关闭文件，可以使用 try ... finally 语句。

```
# 写入数据，使用try-finally语句
try:
    fo = open("foo.txt", 'w')
    fo.write("这是第一行\n")
    fo.write("这是第二行\n")
    fo.write("这是第三行\n")
    fo.write("这是第四行\n")
    fo.write("这是第五行\n")
finally:
    if fo:
        fo.close()
```

文件读写

- 更好的方式是使用上下文管理 with 语句。在 with 语句中，调用 open 函数打开文件，但无需显式调用 close 方法关闭文件，在合适的时候会自动关闭文件。

```
# 写入数据，使用with语句
with open("foo.txt", "w") as fo:
    fo.write("这是第一行\n")
    fo.write("这是第二行\n")
    fo.write("这是第三行\n")
    fo.write("这是第四行\n")
    fo.write("这是第五行\n")
```

文件读写

- 为了防止文件中已存在的数据被意外清除，在打开文件前可以检测该文件是否存在。使用 os.path 模块中的 isfile 方法判断一个文件是否存在，存在返回 True ，否则返回

```
import os.path
import sys
if os.path.isfile("foo.txt"):
    print("文件foo.txt已经存在!")
    sys.exit()
```

sys.exit() 引发一个 SystemExit 异常，若没有捕获这个异常，会直接退出程序。

文件读写

从文件中读取数据。

方法	描述
read([size = -1])	读取 size 指定的字符数。未给定 size 或 size 为负数，则读取所有内容
readline([size = -1])	读取 size 指定的字符数。未给定 size 或 size 为负数，则读取一整行内容，包括 '\n' 换行字符
readlines([hint = -1])	读取 hint 指定的行数。未给定 hint 或 hint 为负数，则读取所有行内容。返回以每行为元素形成的列表

```
# 读取数据
with open("foo.txt", "r") as fo:
    print(fo.read().rstrip())    # 读取文件所有内容

# 读取数据
with open("foo.txt", "r") as fo:
    lines = fo.readlines()    # 按行读取文件所有内容
    for line in lines:
        print(line.rstrip())

# 读取数据
with open("foo.txt", "r") as fo:
    for line in fo:    # 按行读取文件内容
        print(line.rstrip())
```

```
这是第一行
这是第二行
这是第三行
这是第四行
这是第五行
```

文件读写

- **往文件中追加数据。**

```
# 追加数据
with open("foo.txt", "a") as fo:
    list1 = ["我喜欢编程\n", "Python很有趣\n"]
    fo.writelines(list1)

with open("foo.txt", "r") as fo:
    for line in fo:
        print(line.rstrip())
```

```
这是第一行
这是第二行
这是第三行
这是第四行
这是第五行
我喜欢编程
Python很有趣
```

文件读写

- **读写数值数据。**

```
# 读写数值数据
from random import randint
with open("numbers.txt", "w") as fo:
    for i in range(10):
        fo.write(str(randint(0, 9)) + " ")

with open("numbers.txt", "r") as fo:
    s = fo.read()
    numbers = [eval(x) for x in s.split()]
    for number in numbers:
        print(number, end = ' ')
```

```
7 8 5 5 4 6 1 8 8 9
```

为了向文件中写入数字，首先要将它们转换为字符串，然后利用 write 方法将它们写入文件。

为了能从文件中正确读取数字，写入文件时利用空格来分隔数字。因为数字被空格分隔，字符串的 split 方法能够将该字符串分解成列表，从列表中获取数字并显示。

文件读写

- **二进制文件读写。**
  - **使用二进制文件模式 'b' 打开或创建文件。**

```
with open("foo.dat", "wb") as fo:
    fo.write("这是第一行\n".encode("UTF-8"))
    fo.write("这是第二行\n".encode("UTF-8"))
    fo.write("这是第三行\n".encode("UTF-8"))
    fo.write("这是第四行\n".encode("UTF-8"))
    fo.write("这是第五行\n".encode("UTF-8"))
    fo.write("我喜欢编程\n".encode("UTF-8"))
    fo.write("Python很有趣\n".encode("UTF-8"))
with open("foo.dat", "rb") as fo:
    for line in fo:
        print(line.rstrip())
        print(line.rstrip().decode("UTF-8"))
```

```
b'\xe8\xbf\x99\xe6\x98\xaf\xe7\xac\xac\xe4\xb8\x80\xe8\xa1\x8c'
这是第一行
b'\xe8\xbf\x99\xe6\x98\xaf\xe7\xac\xac\xe4\xba\x8c\xe8\xa1\x8c'
这是第二行
b'\xe8\xbf\x99\xe6\x98\xaf\xe7\xac\xac\xe4\xb8\x89\xe8\xa1\x8c'
这是第三行
b'\xe8\xbf\x99\xe6\x98\xaf\xe7\xac\xac\xe5\x9b\x9b\xe8\xa1\x8c'
这是第四行
b'\xe8\xbf\x99\xe6\x98\xaf\xe7\xac\xac\xe4\xba\x94\xe8\xa1\x8c'
这是第五行
b'\xe6\x88\x91\xe5\x96\x9c\xe6\xac\xa2\xe7\xbc\x96\xe7\xa8\x8b'
我喜欢编程
b'Python\xe5\xbe\x88\xe6\x9c\x89\xe8\xb6\xa3'
Python很有趣
>>>
```

  - **encode 方法对字符串进行编码，形成字节码，写入二进制文件。 decode 方法对字节码进行解码。注意观察输出中字节码和原始内容的对比。**

文件读写

- 顺序访问和随机访问。
  - 每个文件都有一个位置指示器（文件指针）。
  - 打开文件时，位置指示器指向文件开头；读写文件时，位置指示器会自动顺序推进，直至文件末尾。这种访问文件的方式称为顺序访问方式。
  - 直接跳到文件的某个位置进行读写操作，而不是从头到尾顺序进行读写操作。这种访问文件的方式称为随机访问方式。

文件读写

- 使用文件对象的 seek 成员方法，移动位置指示器到指定的位置。

  file_object.seek(offset[, whence])

  - 参数 offset 表示以 whence 为基准移动的偏移量（以字节为单位），正偏移量表示从 whence 开始向文件末尾方向移动，负偏移量表示从 whence 开始向文件开头方向移动。
  - 参数 whence 指定从哪个位置开始计算偏移量，有下列几种情况： 0 ，文件开头（默认值）； 1 ，文件当前位置； 2 ，文件末尾。
- 使用文件对象的 tell 成员方法，返回位置指示器当前位置（以字节为单位）。

  file_object.tell()

文件读写

```
with open("foo.dat", "rb+") as fo:
    fo.write("重写第一行\n".encode("UTF-8"))
    fo.seek(0)                  # 定位到文件开头
    line = fo.readline()        # 读取文件第一行
    print(line.rstrip().decode("UTF-8"))
    position = fo.tell()
    print("当前位置:", position)
    fo.seek(96)                 # 跳过文件前六行
    line = fo.readline()        # 读取文件第七行
    print(line.rstrip().decode("UTF-8"))
    position = fo.tell()
    print("当前位置:", position)
```

```
重写第一行
当前位置: 16
Python很有趣
当前位置: 112
>>>
```