Python基础语法精讲

嵩 天

文件的使用

嵩天

文件的使用

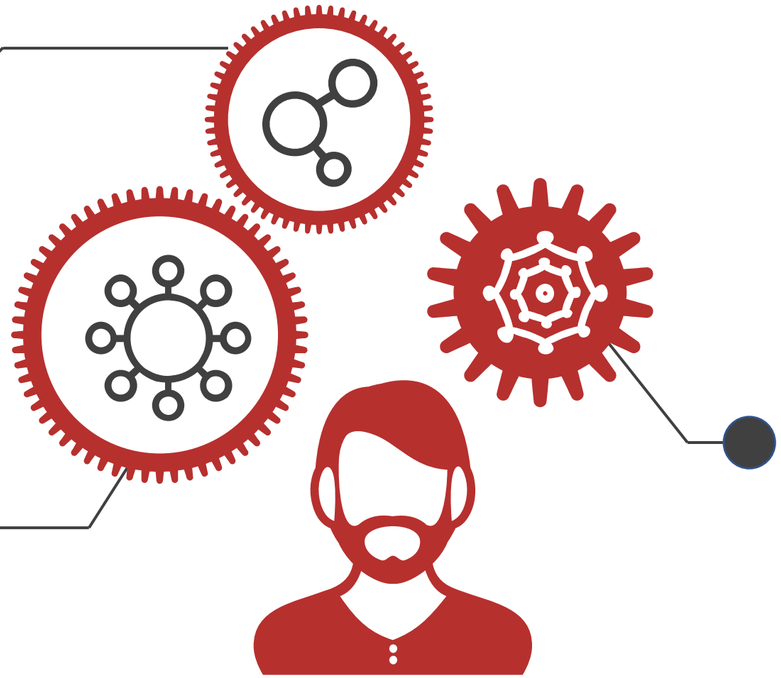
单元开篇

Python 123

单元开篇

(2) 文件内容的读取

(1) 文件的使用方法



(3) 数据的文件写入

文件的使用

文件的使用

文件的使用方法

文件的理解

文件是数据的抽象和集合

- 文本方式和二进制方式是文件的两种不同展示方式
- 文本方式采用统一的编码解释文件，二进制方式采用字节解释文件
- Python语言的文件使用理念与其他语言一致，如C语言

文本方式 vs. 二进制方式

"中国是个伟大的国家!"

- 文本形式 （字符串）

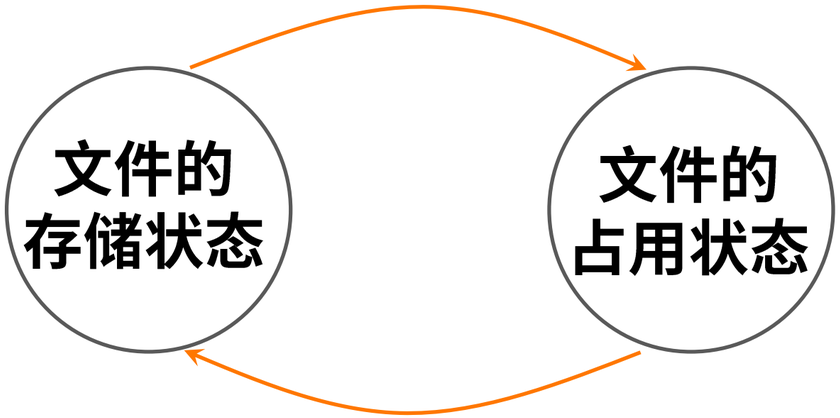
  中国是个伟大的国家!

- 二进制形式 （字节串）

b'\xd6\xd0\xb9\xfa\xca\xc7\xb8\xf6\xce\xb0\xb4\xf3\xb5\xc4\xb9\xfa\xbc\xd2\xa3\xa1'

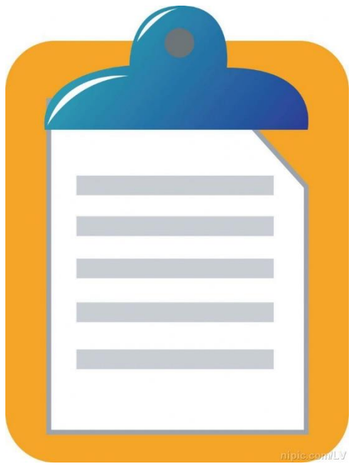
文件的打开关闭

文件处理的步骤: 打开-操作-关闭



a.read(size)
a.readline(size)
a.readlines(hint)

读文件



a.write(s)
a.writelines(lines)
a.seek(offset)

写文件

文件的打开

<变量名> = open(<文件名>, <打开模式>, <编码方式>)

文件句柄

文件路径和名称

源文件同目录可省路径

文本 or 二进制

读 or 写

指定打开文件时采用的编码方式

文件路径

<变量名> = open(<文件名>, <打开模式>, <编码方式>)

D:\PYE\f.txt

文件路径和名称	"D:/PYB/f.txt"	"./PYB/f.txt"
源文件同目录可省路径	"D:\\PYB\\f.txt"	"f.txt"

文件的打开模式

文件的打开模式	描述
'r'	只读模式，默认值，如果文件不存在，返回FileNotFoundError
'w'	覆盖写模式，文件不存在则创建，存在则完全覆盖
'x'	创建写模式，文件不存在则创建，存在则返回FileExistsError
'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
'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

文件的打开模式

```
f = open("f.txt")
f = open("f.txt", "rt")
f = open("f.txt", "w")
f = open("f.txt", "a+")
f = open("f.txt", "x")
f = open("f.txt", "b")
f = open("f.txt", "wb")
```

- 文本形式、只读模式、默认值
- 文本形式、只读模式、同默认值
- 文本形式、覆盖写模式
- 文本形式、追加写模式+ 读文件
- 文本形式、创建写模式
- 二进制形式、只读模式
- 二进制形式、覆盖写模式

文件的编码方式

<变量名> = open(<文件名>, <打开模式>, <编码方式>)

encoding = "utf-8"

可省略，一般指定用utf-8编码打开文件，适用于文本方式

文件的关闭

<变量名>.close()

文件句柄

文件的使用

```
#文本形式打开文件
tf = open("f.txt", "rt", encoding="utf-8")
print(tf.readline())
tf.close()
```

```
#二进制形式打开文件
bf = open("f.txt", "rb")
print(bf.readline())
bf.close()
```

文件的使用

文件内容的读取

文件内容的读取

操作方法	描述
<f>.read(size=-1)	读入全部内容，如果给出参数，读入前size长度 s = f.read(2)
<f>.readline(size=-1)	读入一行内容，如果给出参数，读入该行前size长度 s = f.readline()
<f>.readlines(hint=-1)	读入文件所有行，以每行为元素形成列表，如果给出参数，读入前hint行 s = f.readlines()

文件的全文本操作

方法一：一次读入，统一处理（推荐）

```
fi = open(fname,"r")
txt = fi.read()
#对全文txt进行处理
fi.close()
```

文件的全文本操作

方法二：按数据读入，逐步处理

```
fi = open(fname,"r")
txt = fi.read(20)
while txt != "":
    #对txt进行处理
    txt = fi.read(20)
fi.close()
```

文件的逐行操作

方法一：一次读入，分行处理

```
fi = open(fname,"r")
for line in fi.readlines():
    #对line进行处理
fi.close()
```

文件的逐行操作

方法二：分行读入，逐行处理（推荐）

```
fi = open(fname,"r")
for line in fi:
    #对line进行处理
fi.close()
```

文件的使用

数据的文件写入

数据的文件写入

操作方法	描述
<f>.write(s)	向文件写入一个字符串或字节流 f.write("中国是一个伟大的国家!")
<f>.writelines(lines)	将一个元素全为字符串的列表写入文件 f.writelines(["中国", "法国", "美国"]) 写入结果是：中国法国美国
<f>.seek(offset)	改变当前文件操作指针的位置，offset含义如下： 0 – 文件开头； 1 – 当前位置； 2 – 文件结尾 f.seek(0) #回到文件开头

实例介绍

```
fo = open("output.txt","w+")
ls = ["中国", "法国", "美国"]
fo.writelines(ls)
fo.seek(0)
for line in fo:
    print(line)
fo.close()
```

写入方式打开
文件及关闭

实例介绍

```
fo = open("output.txt","w+")
ls = ["中国", "法国", "美国"]
fo.writelines(ls)
fo.seek(0)
for line in fo:
    print(line)
fo.close()
```

写入内容
此时文件指针位置在
文件结尾处

实例介绍

```
fo = open("output.txt","w+")
ls = ["中国", "法国", "美国"]
fo.writelines(ls)
fo.seek(0)
for line in fo:
    print(line)
fo.close()
```

回到文件开始

再逐行读入并打印内容

文件的使用

单元小结

单元小结

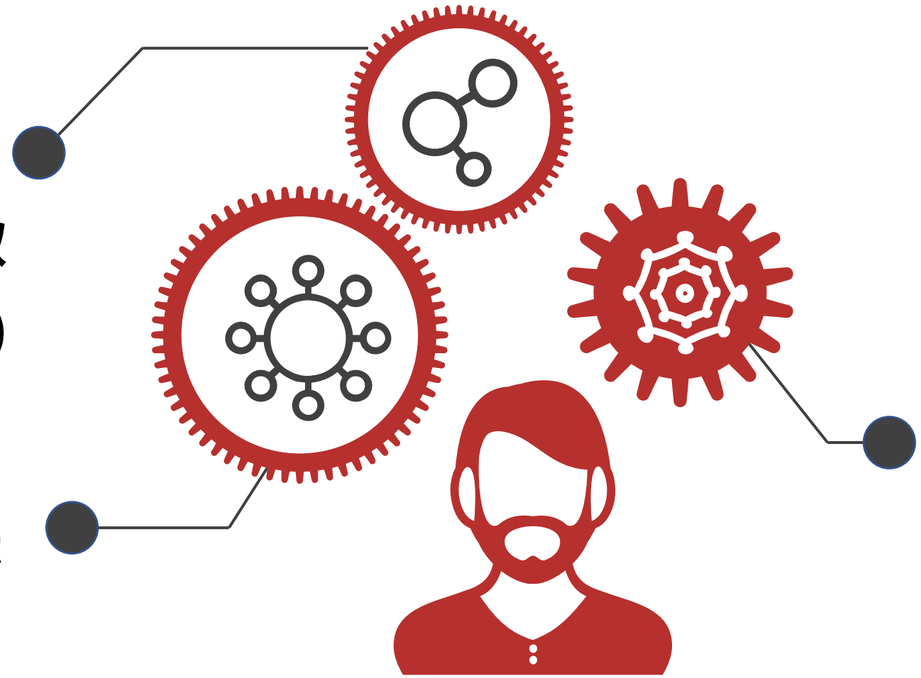
(2) 文件内容的读取

.read()、.readlines()、.readline()

(1) 文件的使用方法

open()、.close()

打开模式、编码方式



(3) 数据的文件写入

.write()、.writelines()、.seek()

文件的使用


Python 123
Thank you