#### 1.输入与输出

通过前面的学习，我们已经掌握了基础代码的编写，函数的使用。但是都有一种电脑在跟电脑自己玩，怎么才能让电脑看到我的输入，或者怎么让电脑读取文件，或者帮助我们处理外部数据呢？我们将学习输入与输出，包括键盘控制台的输入与输出和文件的输入与输出。

**重点**：**掌握如何从键盘控制台输入数据，输出数据到控制台；如何读取文件内容，如何将内容写入文件**。

#### 2.控制台的输入与输出

键盘控制台的输入和输出是在命令行或终端窗口中与用户进行交互的方式。Python提供了input()和print()函数来实现键盘控制台的输入和输出操作。

1. 键盘控制台的输入：
   * input()函数用于从键盘获取用户输入。它会阻塞程序的执行，直到用户输入一些内容并按下回车键。input()函数将用户输入的内容作为字符串返回，并可以存储到变量中供后续使用。

* 例如：
* name = input("请输入您的姓名：")  
  print("您的姓名是：" + name)
* 在这个示例中，input()函数会等待用户输入姓名，并将输入的内容存储到变量name中。然后，通过print()函数将输入的姓名打印出来。
* **注意**：python的input()函数的接受内容是字符串，比如说下面代码
* x=input("请输入你的年龄:")  
  print(type(x),x)  
  age=int(x)  
  print(f"十年后你的年龄是{age+10}")

1. 键盘控制台的输出：
   * print()函数用于将数据输出到控制台。它可以接受一个或多个参数，并将它们打印到控制台上。多个参数之间可以使用逗号分隔，它们将按照给定的顺序打印出来。

* 例如：
* name = "John"  
  age = 25  
  print("姓名:", name, "年龄:", age)
* 在这个示例中，print()函数会将姓名和年龄打印到控制台上。各个参数之间使用逗号分隔，在输出时会自动添加空格。
  + print()函数还支持格式化输出。可以使用占位符 % 来指定要插入的变量的位置和格式。
* 例如：
* name = "Alice"  
  age = 30  
  print("姓名：%s，年龄：%d" % (name, age))
* 在这个示例中，print()函数使用了占位符 %s 和 %d，分别表示字符串和整数的格式。name 和 age 变量的值将按照指定的格式插入到字符串中，并打印到控制台上。

#### 3.文件的读取与写入

文件的读取和写入是在Python中处理文件操作的重要部分。通过读取文件，我们可以从外部文件中获取数据；而通过写入文件，我们可以将数据保存到文件中。

##### 3.1文件的读取：

###### 3.1.1.打开文件

可以使用open()函数打开一个文件，并创建一个文件对象。open()函数主要接受三个参数：文件路径，打开模式，编码格式

file = open("example.txt", "r",encoding="utf-8")

在这个示例中，open()函数打开名为example.txt的文件，并创建了一个文件对象file，以供后续读取操作使用。第二个参数"r"表示以只读模式打开文件。读取文件内容：可以使用文件对象的read()、readline()或readlines()方法来读取文件中的内容。

###### 3.1.2.read()

read()方法用于一次性读取整个文件的内容，并将其作为一个字符串返回。

with open("example.txt", "r",encoding="utf-8") as file:  
 content = file.read()  
 print(content)

在这个示例中，我们使用with语句打开名为example.txt的文件，并使用read()方法将整个文件的内容读取到变量content中，最后通过print()函数打印文件内容到控制台。

###### 3.1.3.readline()

readline()方法用于逐行读取文件中的内容。每次调用`readline()`方法，它会返回文件的下一行作为一个字符串。

with open(r"D:\project\tutorial\python基础\example.txt", "r", encoding="utf-8") as file:  
line = file.readline()  
while line:  
 print(line)  
 line = file.readline()

遍历文件的方法：

with open("example.txt", "r",encoding="utf-8") as file:  
 for line in file:  
 print(line)

在这个示例中，我们使用with语句打开名为example.txt的文件，并使用for循环遍历文件对象file。每次循环迭代时，line变量将保存文件的一行内容，并通过print()函数打印到控制台。

###### 3.1.4.readlines()

readlines()方法用于将文件的所有行读取到一个列表中。每一行作为列表的一个元素，以字符串的形式存储。

with open("example.txt", "r") as file:  
 lines = file.readlines()  
 for line in lines[:5]: # 读取前5行  
 print(line)

在这个示例中，我们使用文件对象的readlines()方法将文件的所有行读取到列表lines中。

在这个示例中，我们使用with语句打开名为example.txt的文件，并使用readlines()方法将文件的所有行读取到列表lines中。然后，我们使用切片操作lines[:5]来获取前5行的内容，并通过for循环遍历并打印出来。

###### 3.1.5.关闭文件

在完成文件读取后，应该使用文件对象的close()方法关闭文件。

file.close()

使用with可以方便管理文件的打开与关闭，这个时候就不需要使用file.close()关闭文件。

##### 3.2.文件的写入：

首先还是需要打开文件，然后进行写入。

###### 3.2.1打开文件：

同样使用open()函数打开文件，并创建一个文件对象。在打开文件时，需要指定文件的路径和打开模式以及编码格式。

file = open("example.txt", "w", encoding="utf-8")

在这个示例中，open()函数打开名为example.txt的文件，并创建了一个文件对象file，以供后续写入操作使用。第二个参数"w"表示以写入模式打开文件。

###### 3.2.2.写入文件内容

可以使用文件对象的write()方法将数据写入文件。

file.write("Hello, World!")

在这个示例中，我们使用文件对象的write()方法将字符串"Hello, World!"写入文件中。

###### 3.2.3关闭文件

写入完成后，应该使用文件对象的close()方法关闭文件。

例如：

file.close()

在这个示例中，我们使用文件对象的close()方法关闭文件对象。

需要注意的是，在进行文件操作时，应该始终使用open()函数打开文件，并在操作完成后使用close()方法关闭文件。这样可以确保文件资源得到正确释放。

写入文件的例子。

1. 写入单行内容：  
with open("example.txt", "w") as file:  
 file.write("Hello, World!")  
  
使用`with`语句打开名为`example.txt`的文件，并使用`write()`方法向文件写入内容。在这里，我们将字符串`"Hello, World!"`写入文件中。  
  
2. 写入多行内容：  
lines = ["Line 1", "Line 2", "Line 3"]  
with open("example.txt", "w") as file:  
 file.writelines("\n".join(lines))  
使用`with`语句打开名为`example.txt`的文件，并使用`writelines()`方法向文件写入多行内容。我们使用列表`lines`存储多行内容，然后使用`"\n".join(lines)`将列表中的字符串连接起来，并使用换行符`\n`分隔每行内容，最后将连接后的字符串写入文件中。  
  
3. 追加内容到文件末尾：  
with open("example.txt", "a") as file:  
 file.write("This is an appended line.")  
在这个示例中，我们使用`with`语句打开名为`example.txt`的文件，并使用`"a"`模式来以追加模式打开文件。然后，我们使用`write()`方法将字符串`"This is an appended line."`追加到文件的末尾。

本文介绍常见的文件输入与输出，可以满足绝大多数工作场景，更多资料参考：[菜鸟教程](https://www.runoob.com/python3/python3-file-methods.html)