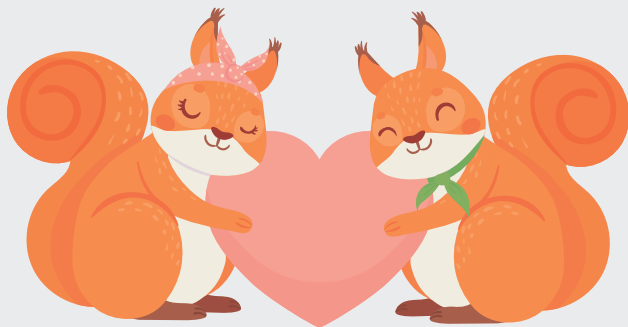




在日常生活中，电路好像是一门离生活很远的学科，一些电路现象常常让我们感到高深莫测，摸不到头脑。为什么按下开关灯就会亮？电热水器又是如何工作的？其实，电路的知识在我们生活的很多地方都有体现，许多神奇的电路现象都向我们展示了电路的奇妙，生活中的许多现象都与电路有着联系。

本章通过观察物理现象，将物理知识与编程知识相结合，利用 Python 实现电路中电流的计算，通过绘制正常情况下和短路情况下电流的图表，认识电路中的短路现象，并引导学生在学的过程中不断体会物理与生活的紧密关联与物理学科的乐趣。





1.1 物理现象观察

欧姆定律是电学的重要定律，是组成电学内容的主干知识。欧姆定律不仅在理论上非常重要，在实际应用中也非常广泛，将欧姆定律运用于人们的工作生活，去分析生活中简单的电学现象，是实现理论联系实际的重要方式。本案例通过带领学生观察真实的电路并设计电路计算短路电流，带领学生多方面地感知与学习电路知识，能够帮助学生更好地理解掌握相关的基础知识。



1.1.1 电流知识介绍

一个电路中必须有电源、开关、用电器，再用导线连接起来。当电路闭合时，电路中才有电流产生，而形成电压。



1.1.2 问题情境

一辆汽车的车灯接在 12V 电源两端，灯丝电阻为 30Ω ，求通过灯丝的电流。

- (1) 如何利用欧姆定律求通过灯丝的电流？
- (2) 如何利用欧姆定律解释电路短路现象？





1.2 案例：电流峰值现象

现实生活中，每家每户无时无刻都在使用电器，那我们是如何理解电器用电的运作原理的呢？当发生短路时又是哪里出了问题了呢？下面用程序编写计算出每个用电器经过的电流，同时可以解释为什么导线直接连在电源两端会导致短路。

首先要弄清，短路指的是电源的短路，而不是某个元件的短接。

如图 1-1 所示，在开关 K 已经闭合的情况下，把电阻 R_1 用导线短接，总电流 I 会因此而增大一些，但不是短路。

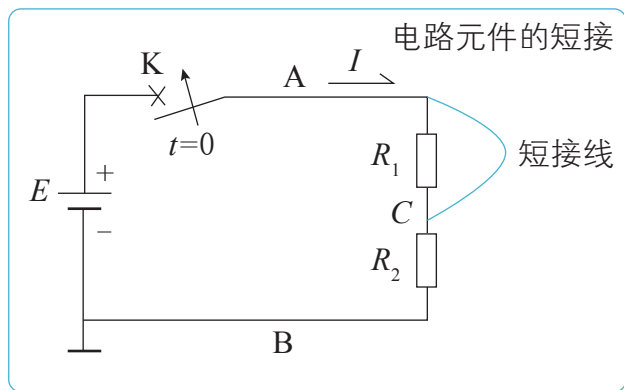


图 1-1 电路的短路现象 1

如图 1-2 所示，在开关 K 已经闭合的情况下，把电阻 R_1 用导线短接，总电流 I 会因此而增大一些，但不是短路。

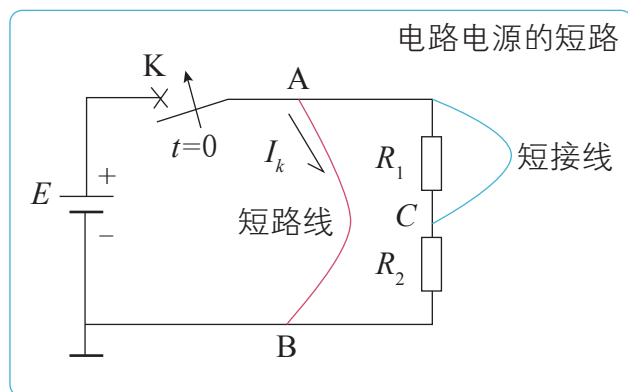


图 1-2 电路的短路现象 2

根据图 1-1 和图 1-2 我们提出一个实例：设电池电动势 $E=3\text{V}$ ，内阻 $r=0.4\Omega$ ，电阻 R_1 和 R_2 均为 10Ω ，短路线为截面 $S=1\text{mm}^2$ 、长度为 $L=1\text{m}$ 的铜线。我们来算一算短路前后的电流是如何变化的？（当然此时 R_1 不能用短接线短路。）

先求外电路总电阻： $R = R_1 + R_2 = 10\Omega + 10\Omega = 20\Omega$ 。

要解决此问题，首先要了解什么是欧姆定律。通过导体的电流，跟导体两端的电压成正比，跟导体的电阻成反比。这个规律叫作欧姆定律，表达式为 $I=U/R$ ，其中， I 表示电流，单位是安培（A）； U 表示电压，单位是伏特（V）； R 表示电阻，单位是欧姆（ Ω ）。根据欧姆定律，可以将上式代入公式。

$$\text{此时正常运行的电流：} I = \frac{E}{r+R} = \frac{3}{0.4+20} = 0.147\text{A}$$

$$\text{此时路端电压：} U = IR = 0.125 \times 20 = 2.5\text{V}。$$

现在把短路线接在 A 点与 B 点之间，此时电源被短接导线的电阻 R_x 给短路。注意， R_x 实质上是与负载电阻 R 并联的，为此先求出短路导线的电阻：

$$R_x = \rho \frac{L}{S} = 1.7 \times 10^{-8} \times \frac{1}{1 \times 10^{-6}} = 0.017\Omega$$



短路线 R_x 与负载电阻 R 并联后的电阻为：

$$R_x // R = \frac{R_x R}{R_x + R} = \frac{0.017 \times 20}{0.017 + 20} \approx 0.01699 \Omega$$

我们看到，短路线与负载电阻并联后，总电阻几乎就等于短接线电阻。

再来求短路电流 I_k ：

$$I_k = \frac{E}{r + R_x // R} = \frac{3}{0.4 + 0.01699} \approx 7.1944 \text{ A}$$

这就是短路电流了。



1.2.1 编程前准备

1. 下载并安装 matplotlib 库文件

我们使用 matplotlib 库里面的 pyplot 包。

Python 中的类库主要指一些写好的 Python 程序片段，有些库是 Python 内置的，有些是其他人创造的，当引入类库之后便可以使用库中已经写好的方法和属性。

(1) matplotlib 是 Python 中最常用的可视化工具之一，可以非常方便地创建海量类型的 2D 图表和一些基本的 3D 图表，可根据数据集 (DataFrame, Series) 自行定义 X,Y 轴，绘制图形 (线形图、柱状图、直方图、密度图、散布图等)，能够解决大部分的需要。

(2) matplotlib 中最基础的模块是 pyplot。每个 pyplot 函数对图形进行一些更改。例如，创建图形、在图形中创建绘图区域、绘制绘图区域中的某些线条、使用标签装饰图形等。



下面将用它进行创建画布以及画出数据线条等。

2. 图像的加载

显示图像前需要划分 x 、 y 轴，这里我们使用 NumPy 包来对 x 轴的值进行取样：

NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库；主要用于数组计算。

这里主要用到 NumPy 中的 `linspace` 函数，此函数的定义如下，接收三个参数，返回一个取样数组：

`numpy.linspace(start, end, num=num_points)`，将在 `start` 和 `end` 之间生成一个统一的序列，共有 `num_points` 个元素。

其中：

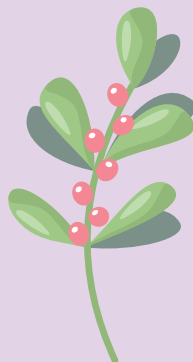
`start`：范围的起点（包括）

`end`：范围的端点（包括）

`num`：序列中的总点数。

```
import numpy as np
import matplotlib.pyplot as plt

y = np.zeros(5)
x1 = np.linspace(0, 10, 5)
x2 = np.linspace(0, 10, 5)
plt.plot(x1, y, 'o')
plt.plot(x2, y + 0.5, 'o')
plt.ylim([-0.5, 1])
plt.show()
```





程序显示结果如图 1-3 所示。

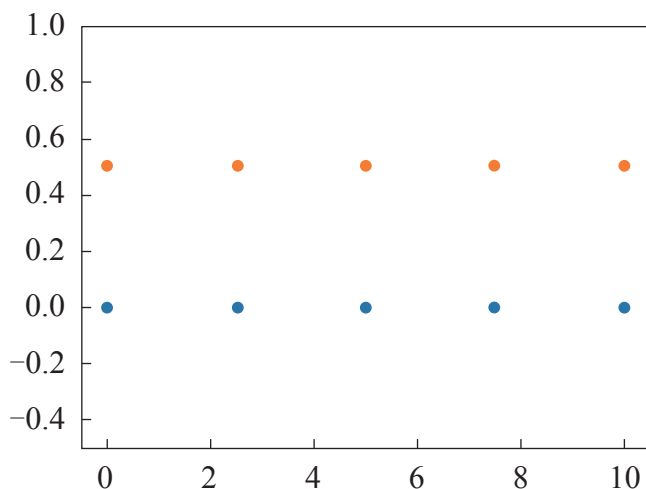


图 1-3 程序显示结果

3. 图像的显示

(1) 使用 `pyplot.plot(x,y,format_string)` 方法将点用线连接起来。

其中, `x,y` 分别为 X 轴数据与 Y 轴数据, 可以为列表或数组; `format_string` 为控制曲线格式的字符串。

(2) `pyplot.show()` 的功能为显示所有打开的图形。



1.2.2 算法设计

程序流程图是用规定的符号描述一个专用程序中所需要的各项操作或判断的图示。这种流程图着重说明程序的逻辑性与处理顺序, 具体描述了微机解题的逻辑及步骤。当程序中有较多循环语句和转移语句时, 程序的结构将比较复杂, 给程序设计与阅读造成困难。程序流程图用图的形式画出程序流向, 是



算法的一种图形化表示方法,具有直观、清晰、更易理解的特点。

程序流程图由处理框、判断框、起止框、连接点、流程线、注释框等构成,并结合相应的算法,构成整个程序流程图。处理框具有处理功能;判断框(菱形框)具有条件判断功能,有一个入口,两个出口;起止框表示程序的开始或结束;连接点可将流程线连接起来;流程线表示流程的路径和方向;注释框是为了对流程图中某些框的操作做必要的补充说明。

(1) 一开始给出杠杆的长度数值,同时给出左右两端物体的质量和力矩长度数据输入程序中。

(2) 通过 `turtle` 库生成图形化界面进行展示。

(3) 程序根据输入的数值和方程计算平衡时左右力矩的长度。

(4) 判断力矩是否超过 $\text{height}/2$, 否则退回上一步, 是则绘制出平衡后的图像。

(5) 结束。

本案例主要包括利用欧姆定律计算电阻和利用编程绘制短路和正常情况下电流的变化, 案例的主要思路如下:

(1) 初始化电路, 包括给出电源电动势与各个电阻的阻值, 及其之间的连接情况。

(2) 对于短路电路:

- ① 计算短路的导线电阻;
- ② 计算短路导线与电阻并联后的总电阻;
- ③ 利用欧姆定律计算此时的短路电流。

(3) 对于非短路电路:

- ① 计算总电阻;
- ② 通过欧姆定律计算正常运行时的电流。

最后, 将计算得到的数据传入绘图函数中, 完成短路电流与正常电流图像的绘制。

程序流程图如图 1-4 所示。

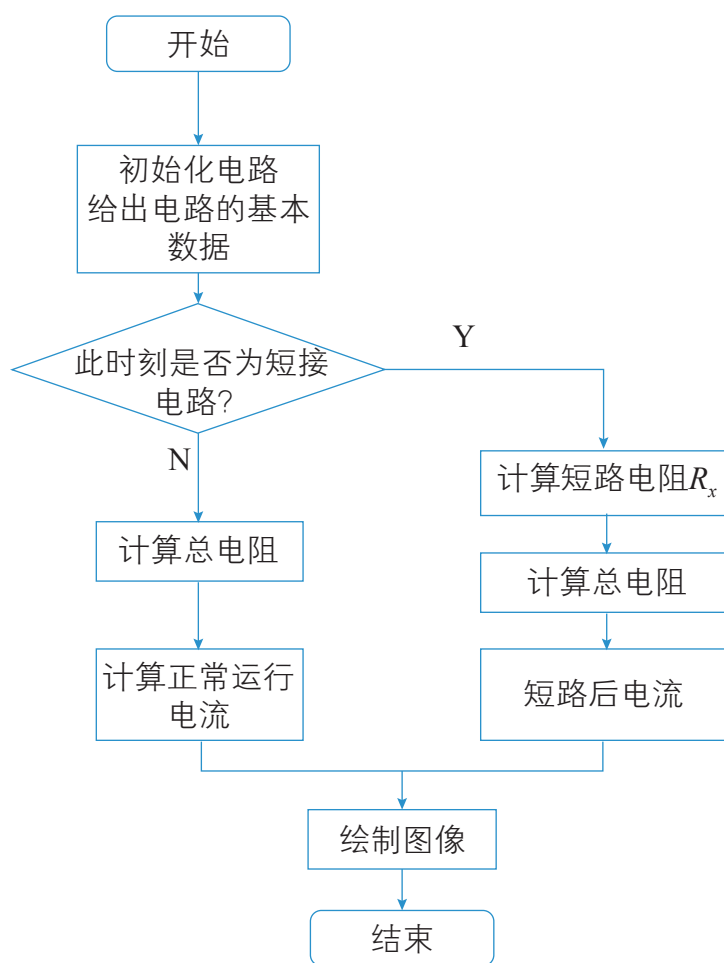


图 1-4 程序流程图



1.3 编写程序及运行

1.3.1 程序代码

```
import matplotlib.pyplot as plt
import numpy as np

def main():
    e, r, r0, s, l, ro = 3, 0.4, 20, 1e-6, 1, 1.7e-8
    i1 = np.array([e / (r + r0)]*10)

    rx = ro * l/s
    i2 = np.array([e / (rx * r0 / (rx + r0) + r)]*10)
    t1 = np.linspace(0, 5, 10)
    t2 = np.linspace(5, 10, 10)
    plt.figure()
    plt.plot(t1, i1, label="normal")
    plt.plot(t2, i2, label="short circuit")
    plt.legend()
    plt.arrow(t1[9], i1[9], 0, i2[9]-i1[9], width=0.01)
    plt.show()

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



1.3.2 运行程序

步骤一：配置环境与新建项目

1. 下载并安装 Python

在 <https://www.python.org/downloads/> 中下载 Python 3.10.1，如图 1-5 所示。

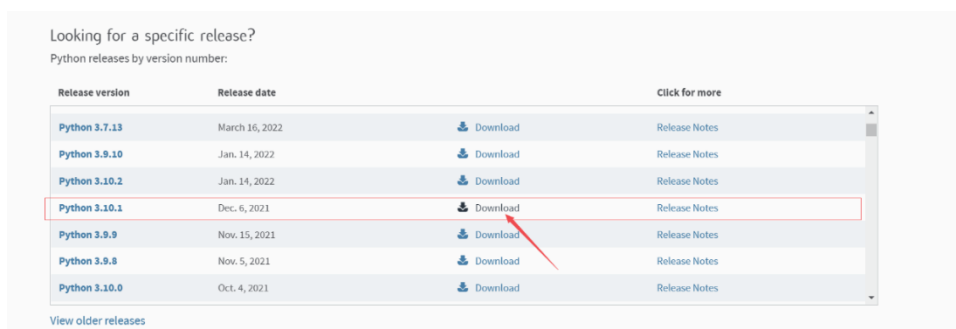


图 1-5 Python 下载界面

如图 1-6 所示，进入页面后，根据系统选择 Windows 64bit 版本下载，下载后依据提示安装即可。

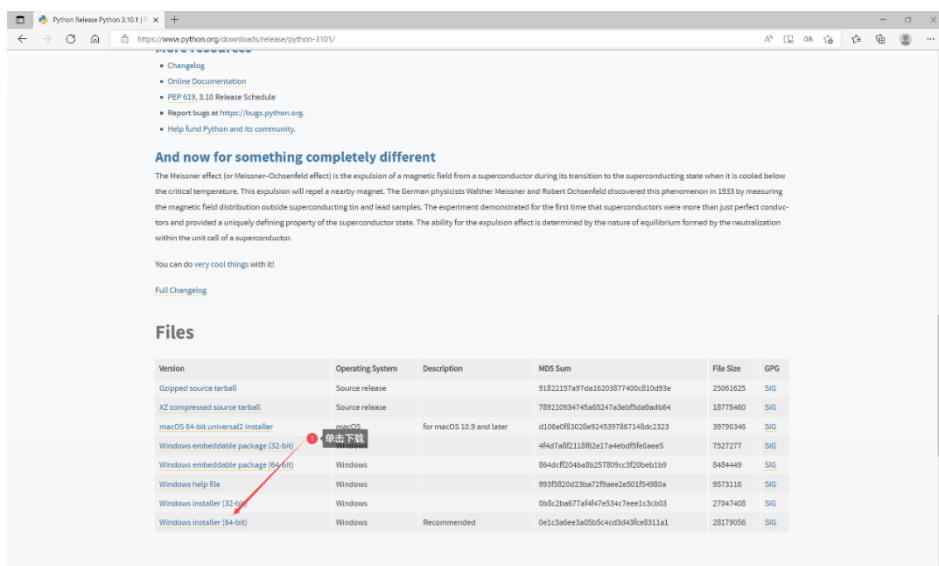


图 1-6 下载 Windows 64bit 版本



2. 下载并安装 PyCharm

打开 <https://www.jetbrains.com/pycharm/download/#section=windows>, 下载 PyCharm Community 的安装程序, 如图 1-7 所示, 下载后, 依据提示安装即可。

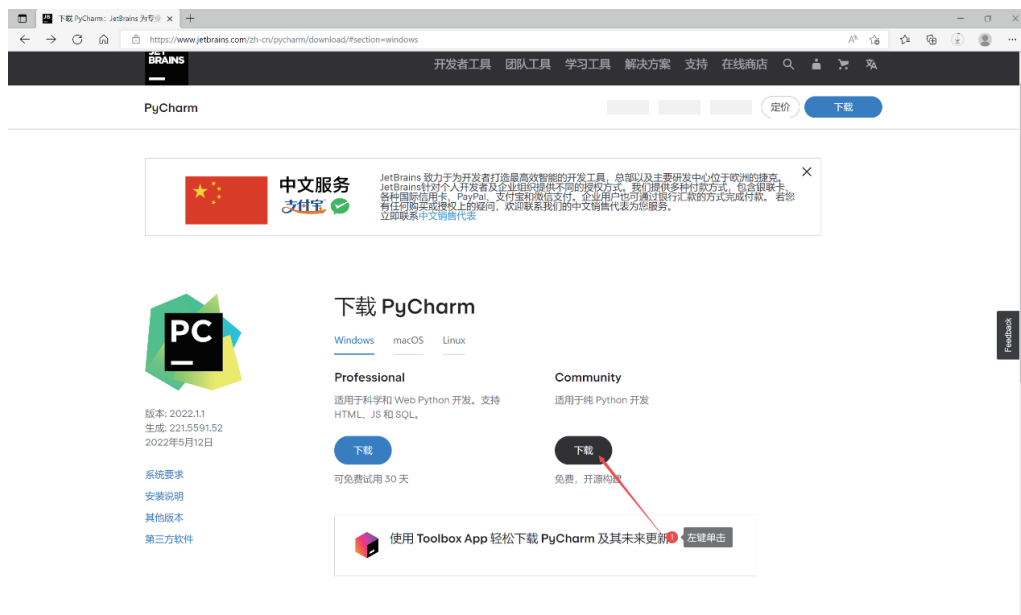


图 1-7 PyCharm 下载界面

3. 新建 Python 项目

运行 PyCharm, 单击 New Project 按钮后, 输入项目名称为 “BookCase” 后, 单击 Create 按钮即可完成新建, 如图 1-8 和图 1-9 所示。



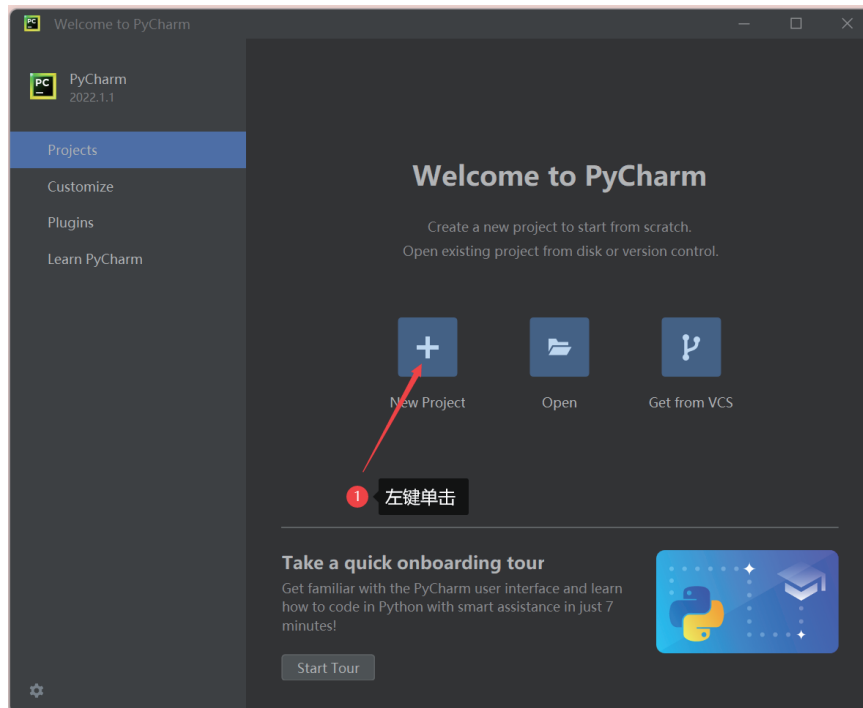


图 1-8 “新建”工程界面

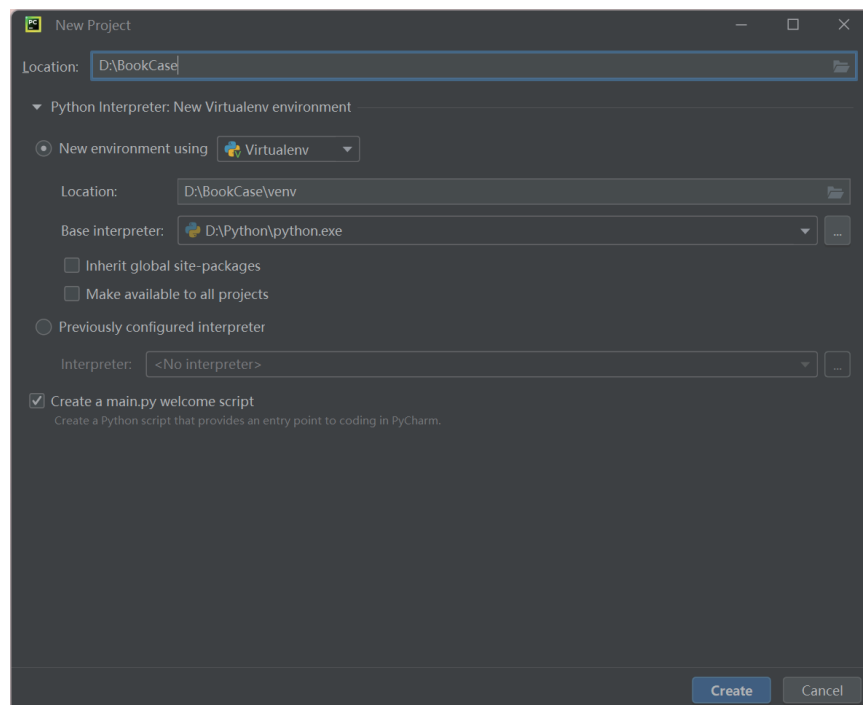


图 1-9 选择“新建工程”的文件位置



4. 安装中文环境、numpy 包与 matplotlib 包

1) 安装中文环境

打开 PyCharm，单击右上角的设置图标后，选择 **Plugins** 命令后，在弹出的对话框中搜索“**Chinese**”，并单击 **Marketplace**，找到第二个，单击 **Install** 按钮。完成安装后重启 PyCharm 即可，如图 1-10 和图 1-11 所示。

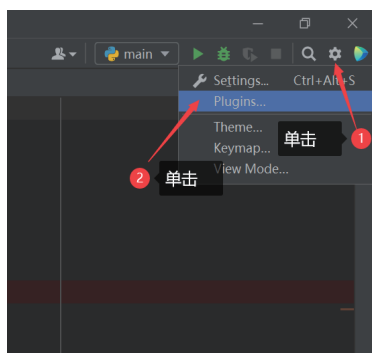


图 1-10 中文环境设置 1

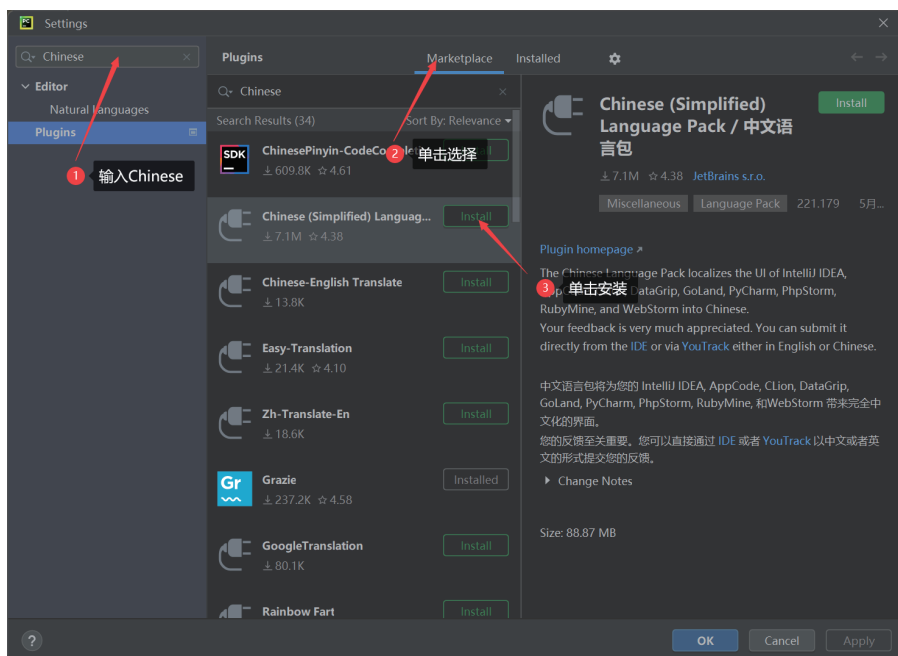


图 1-11 中文环境设置 2

2) 安装 numpy 包与 matplotlib 包

单击下方的终端，分别输入命令：

Pip3 install numpy

Pip3 install matplotlib

等待下载完成即可，如图 1-12 所示。

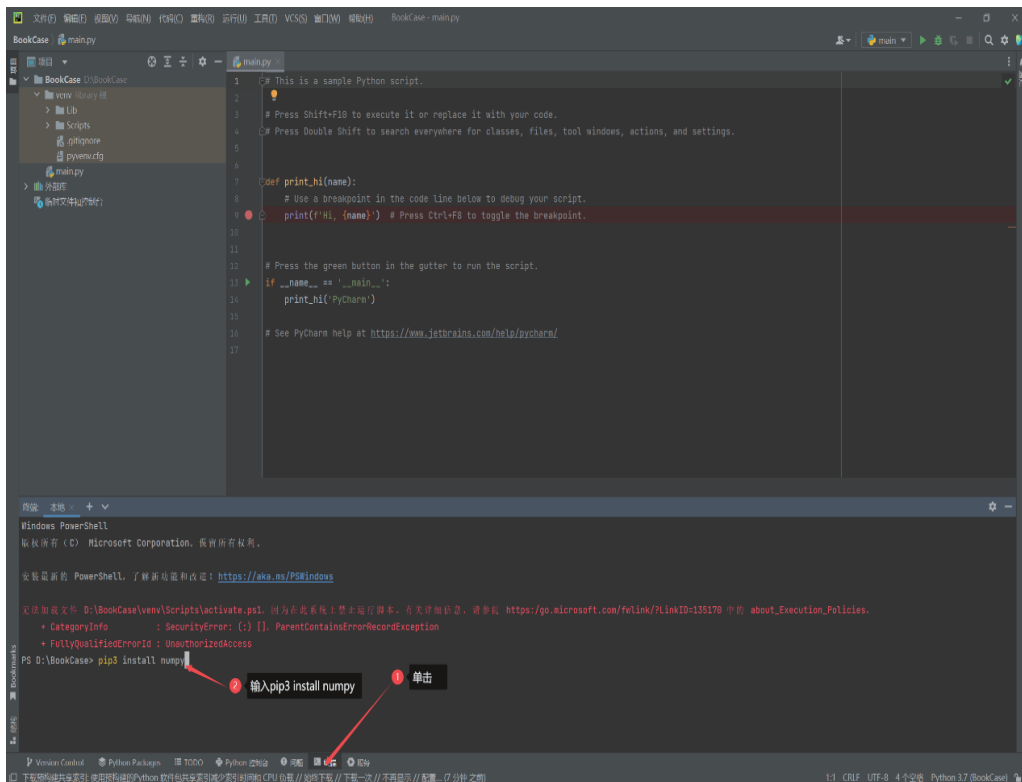


图 1-12 安装 numpy 和 matplotlib 包

步骤二：新建文件

在刚才新建的项目中，右击左侧项目栏，在弹出的选择栏中选择“新建”→“Python 文件”，单击后，在页面中间弹出的命名栏中输入文件名“Case1”，并选择“Python 文件”，回车，即可新建成功，如图 1-13 和图 1-14 所示。

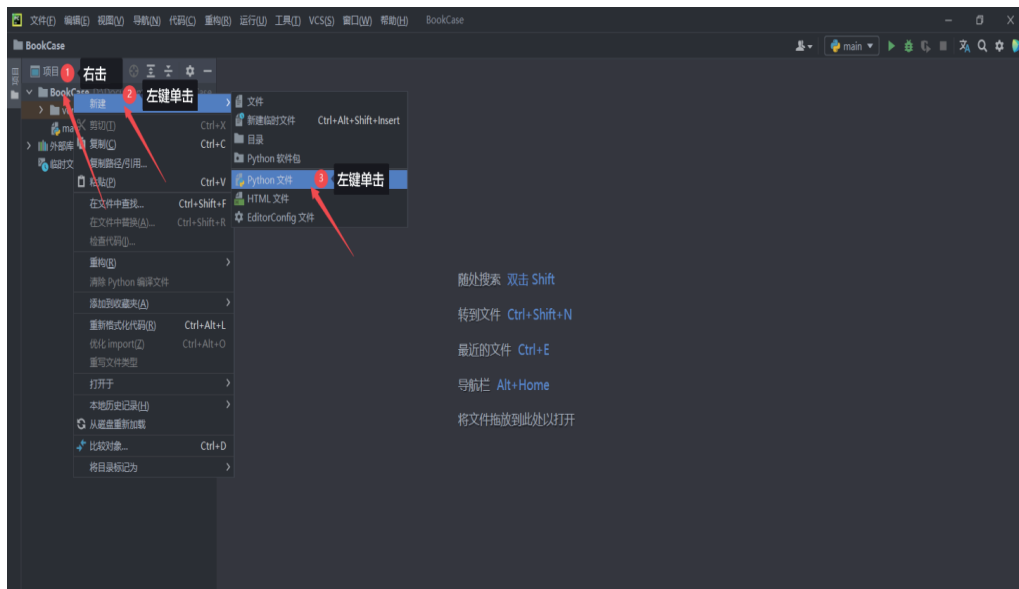


图 1-13 选择“新建工程”的文件位置

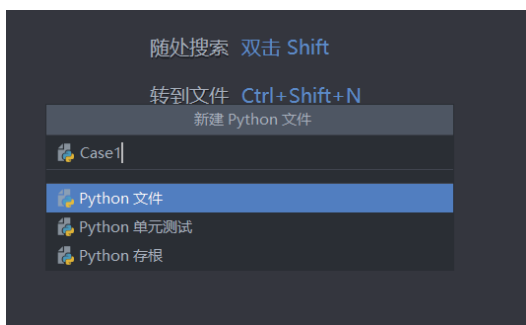


图 1-14 在“新工程”下创建新文件



步骤三：编写代码

将案例的代码粘贴至 `Case1.py` 中，如图 1-15 所示。

```
Case1.py
1 import matplotlib.pyplot as plt
2 import numpy as np
3 def main():
4     e, r, r0, s, l, ro = 3, 0.4, 20, 1e-6, 1, 1.7e-8
5     i1 = np.array([e / (r + r0)]*10)
6
7     rx = ro * l/s
8     i2 = np.array([e / (rx * r0 / (rx + r0) + r)]*10)
9     t1 = np.linspace(0, 5, 10)
10    t2 = np.linspace(5, 10, 10)
11    plt.figure()
12    plt.plot(t1, i1, label="normal")
13    plt.plot(t2, i2, label="short circuit")
14    plt.legend()
15    plt.arrow(t1[9], i1[9], 0, i2[9]-i1[9], width=0.01)
16    plt.show()
17 if __name__ == "__main__":
18     main()
19
```

图 1-15 程序代码的编写

步骤四：运行程序

如图 1-16 所示，在任意空白处右击后，单击右键菜单中的“运行 'Case1'”选项，即可运行。也可使用快捷键 `Ctrl+Shift+F10`。

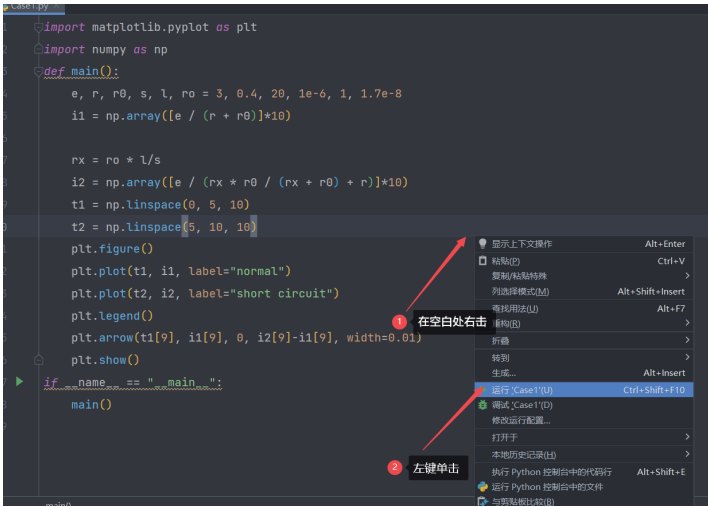


图 1-16 程序的执行



程序运行结束之后，即可看到程序的运行结果，如图 1-17 所示。

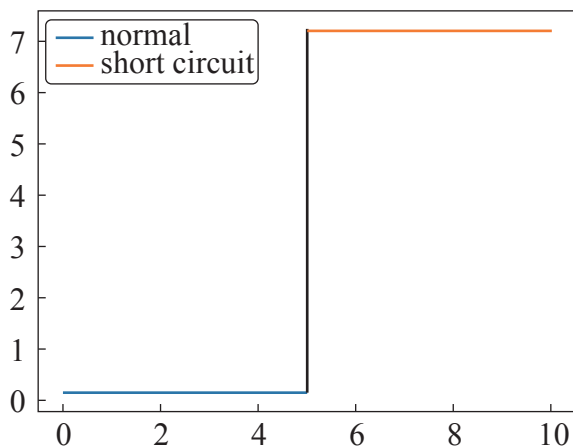


图 1-17 程序运行结果

1.4 拓展训练

根据上述案例，试计算图 1-1 中接 R_1 的短接线和不接 R_1 的短接线的电流变化。

