

第 5 章 现代智能信息处理开发

5.1 开发语言基础

5.1.1 Python 语言基础

Python 的第一个编译器由 Guido van Rossum 于 1989 年发明,并在 1991 年公开发行。Python 是一种易写易读、功能强大、可以跨平台的解释型编程语言。由于其内置的标准库和第三方库的支持,因此可以利用该编程语言进行各种复杂的科学计算。经过多年的发展,Python 在数据处理、人工智能、网页开发等领域已有广泛应用。本章仅介绍 Python 的一些基础知识和几种处理数据扩展库的使用,如果你对 these 内容非常了解。

1) Python 的安装

Python 的安装很简单,进入 Python 官方网站 <https://www.python.org>,单击 Downloads,然后选择想要的版本号进行下载。安装完成后,可以在终端窗口输入“python”命令查看计算机是否已经安装成功,以及安装的 Python 版本。Linux 系统一般都自带 Python 支持,不需要重复安装,只要升级版本就行。

2) 扩展库的安装

进行数值分析时,Python 需要利用 pip 导入相关的扩展库,如果有搭建 Anaconda 的话,则不需要自行导入,其一般都自带常用的工具库(NumPy、SciPy、Matplotlib)。可以访问网址 <https://www.anaconda.com> 下载相关资源和文档。pip 提供了对 Python 包的查找、下载、安装、卸载等功能,下面列出 pip 的常用命令。

- 安装模块: `pip install <包名>`。
- 卸载模块: `pip uninstall <包名>`。
- 查找模块: `pip freeze` 或 `pip list`。
- 升级模块: `pip install --upgrade <包名>`。

1. Python 基础知识

Python 中不需要声明变量,通过赋值操作便可创建不同类型的数据类型。Python 中有 5 种标准的数据类型:字符串(Strings)、列表(List)、元组(Tuple)、字典(Dictionary)、集合(Set)。

1) 字符串(Strings)

字符串通过引号进行创建,在实际编程中通常会接触大量的字符串操作,下面简要介绍一些字符串的基本操作。

表 5-1 令 a 为字符串"intelligent",b 为字符串"communication"。



表 5-1 字符串输入与输出

输 入	输 出
<code>a+b</code> # 连接字符串	<code>intelligentcommunication</code>
<code>a*2</code> # 连续输出字符串	<code>intelligentintelligent</code>
<code>a[2]</code> # 输出指定位置字符	<code>t</code>
<code>a[2:4]</code> # 截取字符串	<code>te</code>
<code>'o' in a</code> # 判断 a 中是否包含字符'o'	<code>False</code>
<code>'b' not in a</code> # 判断 a 中是否不包含字符'b'	<code>True</code>

Python 中也存在大量针对字符串的内建函数,使用这些函数可以极大地简化代码,而且运行时更加稳定、快速。下面介绍一些常用内建函数的使用(见表 5-2),详细的函数介绍可以参考 Python 官方文档。

表 5-2 常用的内建函数

函 数	描 述
<code>max(str),min(str)</code>	返回字符串中最大的字母、最小的字母
<code>str.capitalize()</code>	把字符串的第一个字母大写
<code>str.center(L)</code>	<code>str</code> 居中,用空格填充至长度 <code>L</code>
<code>str.ljust(L)</code>	<code>str</code> 左对齐,用空格填充至长度 <code>L</code>
<code>str.rjust(L)</code>	<code>str</code> 右对齐,用空格填充至长度 <code>L</code>
<code>str.isdigit()</code>	若 <code>str</code> 中存在数字,则返回 <code>True</code> ,否则返回 <code>False</code>
<code>str.join(li)</code>	用 <code>str</code> 连接 <code>li</code> 字符串中的每个元素
<code>str.lower()</code>	<code>str</code> 中所有大写字符转小写
<code>str.upper()</code>	<code>str</code> 中所有小写字符转大写
<code>str.lstrip()</code>	去掉 <code>str</code> 中最左边的空格
<code>str.rstrip()</code>	去掉 <code>str</code> 中最右边的空格

2) 列表(List)

Python 中没有数组,但是有更加强大的列表,同样是把所有元素放在一对方括号“[]”中,各元素用“,”隔开。不同的是,列表内可以存放数、字符串、列表、元组、集合等 Python 可以支持的任何数据类型,并且同一列表可以存放不同的元素类型。对于列表,主要介绍关于列表添加、删除、修改、查找,以及切片的内容。

• 添加元素

有三种方法可用来添加元素,见表 5-3。

表 5-3 添加元素的方法

方 法	描 述
<code>lis.append(obj)</code>	把 <code>obj</code> 整体添加到列表 <code>lis</code> 末尾
<code>lis.extend(obj)</code>	把 <code>obj</code> 中的元素依次添加到列表 <code>lis</code> 末尾
<code>lis.insert(index,obj)</code>	在列表 <code>lis</code> 对应索引位置添加整体的 <code>obj</code>



- 删除元素

有四种方法可用来删除元素,见表 5-4。

表 5-4 删除元素的方法

方 法	描 述
<code>del listname[index]</code>	删除指定位置元素
<code>listname.pop(index)</code>	删除指定位置元素
<code>listname.remove(num)</code>	删除列表中第一个 num,若没有,则报错
<code>listname.clear()</code>	清空列表

- 修改元素

可以修改单个元素和一组元素,见表 5-5。

表 5-5 修改元素的方法

方 法	描 述
<code>listname[index]=num</code>	直接修改位置元素为 num
<code>listname[index1:index2]=[...]</code>	切片修改一组元素

- 查找元素

有两种方法可进行查找,见表 5-6。

表 5-6 查找元素的方法

方 法	描 述
<code>listname.index(num,index1,index2)</code>	查找元素出现的位置
<code>listname.count(num)</code>	统计元素出现的次数

- 切片

切片是处理列表的一种十分重要的操作,在字符串、元组等对象中有广泛的应用。表 5-7 举例说明切片操作,有列表 `lis=['a','b','c','d','e','f']`。

表 5-7 切片操作

操 作	输 出	操 作	输 出
<code>lis[:]</code>	<code>['a','b','c','d','e','f']</code>	<code>lis[::-2]</code>	<code>['f','d','b']</code>
<code>lis[1:3]</code>	<code>['b','c']</code>	<code>lis[::2]</code>	<code>['a','c','e']</code>

3) 元组(Tuple)

元组是 Python 中另一个重要的序列结构,和列表类似,元组也由一系列按特定顺序排序的元素组成。元组和列表的不同之处在于:

(1) 列表的元素是可以更改的,包括修改元素值、删除和插入元素,所以列表是可变序列;

(2) 而元组一旦被创建,它的元素就不可更改了,所以元组是不可变序列。



元组也可以看作不可变的列表,通常,元组用于保存无须修改的内容。因为元组与列表十分相似,所以就不过多列举。元组的主要操作如表 5-8 所示。

表 5-8 元组的主要操作

操 作	描 述
<code>tuple=(num1,num2,...,numn)</code>	创建元组
<code>tuple[index]</code>	访问元素
<code>tuple=(num1,num2,...,numn)</code> <code>tuple=(str1,str2,...,strn)</code>	元组不能修改,只能用新的元组覆盖旧的元组
<code>del tuple</code>	删除元组

4) 字典(Dictionary)

Python 字典是一种无序的、可变的序列,它的元素以“键值对(key-value)”的形式存储。字典的每个元素的“键”和“值”用冒号“:”分割,每个对之间用逗号“,”隔开,所有元素都在一对花括号“{ }”里。

字典中的“键”必须是唯一的,可以是实数、字符串、元组等,但是值不一定,值可以重复。下面介绍一些字典常用的操作和内置函数的使用,见表 5-9。

表 5-9 字典常用的操作和内置函数的使用

方 法	描述/结果
<code>dict={'a':1,'b':2}</code> 或 <code>x=dict(a=1,b=2)</code>	创建字典
<code>dict={'a':1,'b':2}</code> <code>print(dict['a'])</code>	访问字典里的值,输出 1
<code>dict={'a':1,'b':2}</code> <code>dict['b']=3</code> <code>print(dict)</code>	修改元素的值 输出 {'a':1,'b':3}
<code>dict={'a':1,'b':2}</code> <code>del dict['a']</code> <code>print(dict)</code>	删除元素的值 输出 {'b':2}
<code>dict={'a':1,'b':2}</code> <code>dict.update({'c':3,'d':4})</code> <code>print(dict)</code>	添加元素 输出 {'a': 1,'b': 2,'c': 3,'d': 4}
<code>dict={'a':1,'b':2}</code> <code>print(dict.keys())</code>	返回字典中所有的键 输出 dict_keys(['a','b'])
<code>dict={'a':1,'b':2}</code> <code>print(dict.values())</code>	返回字典中所有的值 输出 dict_values([1,2])
<code>dict={'a':1,'b':2}</code> <code>print(dict.items())</code>	返回字典中所有的键值对 输出 dict_items([('a',1),('b',2)])
<code>dict={'a':1,'b':2}</code> <code>print(list(dict.items()))</code>	将返回的数据转换成列表 输出 [('a',1),('b',2)]



5) 集合(Set)

Python 集合会将所有元素放在一对花括号“{}”中,相邻元素之间用“,”分隔。集合中的元素不能重复,每个元素都是唯一的。表 5-10 显示了集合的基本操作。

表 5-10 集合的基本操作

方 法	结 果
set={element1,element2} 或 setname=set(列表/元组/range 对象)	创建集合
set={'a'} set.add('b')	向集合添加元素 输出{'a','b'}
set={'a','b'} set.remove('a')或 set.pop()	从集合中删除元素 都输出{'b'}
set1={'a','b'} set2={'a','c'} print(set1&set2) print(set1 set2) print(set1-set2)	集合的交集、并集、差集 分别输出 {'a'} {'a','c','b'} {'b'}

6) 函数

Python 里用 def 关键词定义函数,后接函数标识符名称和圆括号“()”。简单举例如下。

```
def obj():  
    set={'a','b'}  
    for i in set:    # 输出集合中的每个元素  
        print(i)  
obj()
```

2. Python 扩展模块及可视化

接下来介绍 Python 中用于科学计算的 4 种常用扩展包: NumPy、Pandas、Matplotlib、SciPy。相关程序运行结果通过 Jupyter Notebook 展示,这一工具能直接在网页上编写和执行 Python 代码,相关运行结果也能在代码块下显现。有关 Jupyter 的安装和操作可以参考 Jupyter 的官网 <http://jupyter.org/index.html>。

1) NumPy

NumPy 是使用 Python 进行科学计算的基础包,可提供数组支持以及相应的高效处理函数。相比于 Python 的内置序列,NumPy 数组的执行效率更好,代码量更少。NumPy 也是 Scipy、Pandas 等数据处理的科学计算库最基本的函数功能库,Matplotlib 同样也需要 NumPy 的支持。

通常使用 Python 包管理器 pip 安装 numpy: pip install numpy。

numpy 包的导入: import numpy as np。

● 创建数组

可以通过转换 Python 中的列表和元组创建数组,也可以使用 NumPy 数组创建函数以及其他特殊的库函数来创建数组。

导入 numpy 包: import numpy as np。



表 5-11 创建数组的方法

操 作	描述/输出
<pre>a=np.array((1,2,3)) 或 a=np.array([1,2,3]) print(a)</pre>	创建一维数组 <pre>[1 2 3]</pre>
<pre>a=np.array([[1,2],[3,4]]) print(a)</pre>	创建二维数组 <pre>[[1 2] [3 4]]</pre>
<pre>a=np.arange(2,10,2) print(a)</pre>	用内置函数,规定开始、结束和步长 <pre>[2 4 6 8]</pre>
<pre>a=np.linspace(2,10,2, dtype=int) print(a)</pre>	用内置函数,规定开始、结束和数量 <pre>[2 10]</pre>

- 数组索引

可以使用方括号“[]”索引数组值,通过下标索引单一元素值,通过切片指向元素连续的多个元素值。

表 5-12 数组索引的方法

操 作	描述/输出
<pre>a=np.array([1,2,3,4,5]) print(a[2])</pre>	单元素索引 <pre>3</pre>
<pre>a=np.array([1,2,3,4,5]) print(a[1:3])</pre>	切片操作,多元素索引 <pre>[2 3]</pre>
<pre>a=np.array([[1,2,3],[4,5,6]]) b=(a>3) print(b)</pre>	布尔索引 <pre>[[False False False] [True True True]]</pre>
<pre>a=np.array([1,2,3,4,5]) a[1:3]=1 print(a)</pre>	索引赋值 <pre>[1 1 1 4 5]</pre>

- 数组的相关运算(见表 5-13)

表 5-13 数组的相关运算

操 作	描述/输出
<pre>a=np.array([[1,2],[3,4]]) print(a.T)</pre>	数组转置 <pre>[[1 3] [2 4]]</pre>
<pre>a=np.array([1,2]) b=np.array([3,4]) print(np.dot(a,b))</pre>	向量点积 <pre>11</pre>



续表

操 作	描述/输出
<pre>a=np.array([1,2,3]) print(a*2) print(a/2) print(a**2)</pre>	数组的乘、除、平方 [2 4 6] [0.5 1. 1.5] [1 4 9]
<pre>a=np.array([1,2]) b=np.array([[1,2],[3,4]]) print(a*b)</pre>	数组间的乘法运算 [[1 4] [3 8]]
<pre>a=np.array([[2,1,4,3],[5,0,2,4]]) print(np.sort(a))</pre>	数组按行排序 [[1 2 3 4] [0 2 4 5]]
<pre>a=np.array([[2,1,4,3],[5,0,2,4]]) print(np.sort(a,axis=0))</pre>	数组按列排序 [[2 0 2 3] [5 1 4 4]]

2) Matplotlib 绘图

Matplotlib 是基于 NumPy 的一套 Python 包,这个包提供了一系列的数据绘图工具,可以将很多数据以线条图、饼图、柱状图及其他专业图形呈现出来。通常使用 Python 包管理器 pip 安装 matplotlib: pip install matplotlib。

Matplotlib 包的导入: import matplotlib.pyplot as plt
import matplotlib。

(1) 散点图(见图 5.1)。

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0,20,100)
plt.plot(x,np.sin(x),'*',color="black");
plt.show()
```

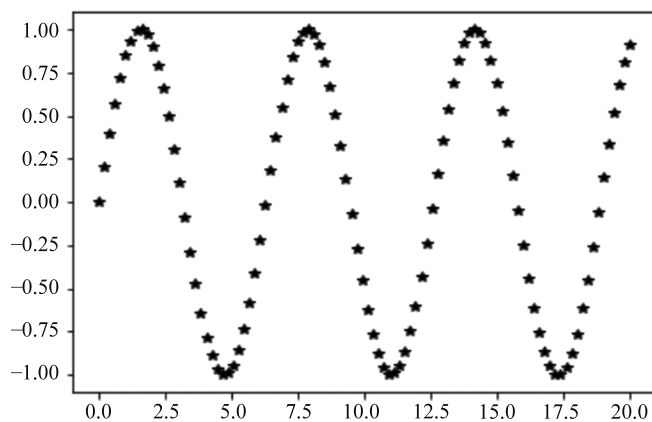


图 5.1 散点图



(2) 柱形图(见图 5.2)。

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(1, 11, 1)
y = np.random.rand(10)*10
plt.bar(x, y);
plt.show()
```

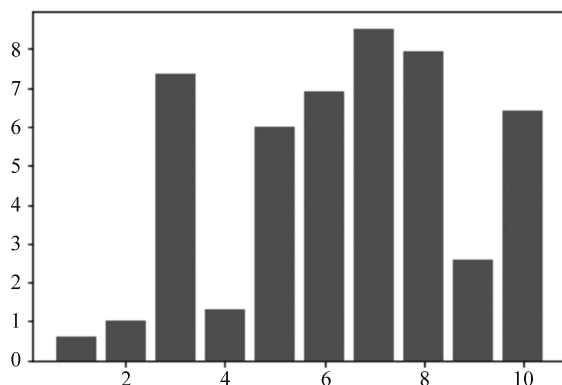


图 5.2 柱形图

(3) 饼图(见图 5.3)。

```
import matplotlib.pyplot as plt
import numpy as np
parts = np.array([10, 45, 5, 25, 15])
plt.pie(parts, labels=['A', 'B', 'C', 'D', 'E'],
        colors=["#00FF00", "#FFD700", "#FF6A6A", "#9400D3", "#9F79EE"],
        explode=(0.1, 0, 0.1, 0, 0),
        autopct='%1f%%',
        )
plt.show()
```

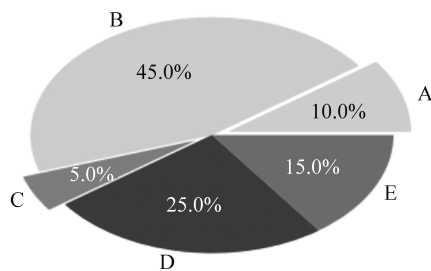


图 5.3 饼图

(4) 多线图(见图 5.4)。

(5) 三维曲线(见图 5.5)。

3. SciPy

SciPy 在 NumPy 的基础上添加了更多用于科学计算的模块,包括统计、优化、整合、线性代数模块、傅里叶变换、信号和图像处理、常微分方程求解器等。SciPy 依赖于 NumPy,并



```
import matplotlib.pyplot as plt
import numpy as np
fig, ax = plt.subplots()
x = np.linspace(0, 20, 100)
plt.plot(x, np.sin(x), color="black", label="sin");
plt.plot(x, np.cos(x), color="red", label="cos");
ax.legend();
plt.show()
```

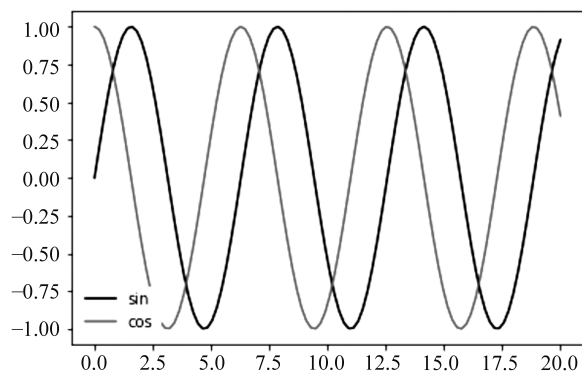


图 5.4 多线图

```
In [2]: from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes(projection='3d')

z = np.linspace(0, 10*np.pi, 1000)
r = np.linspace(0, 1, 1000)
x = r*np.sin(z)
y = r*np.cos(z)
ax.plot3D(x, y, z, color='black')
plt.show()
```

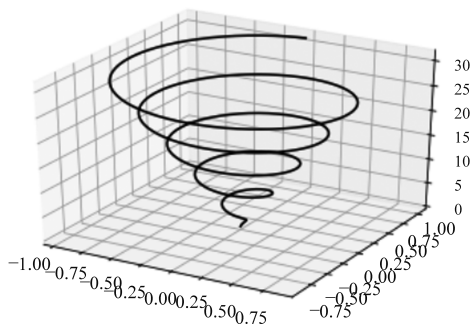


图 5.5 三维曲线

提供许多如数值积分和优化这样的数值例程。

下面列举一个 SciPy 关于信号处理的应用(见图 5.6), `scipy.signal` 模块专门用于信号处。



`scipy.signal.resample()`: 采样(Sampling)。

`scipy.signal.detrend()`: 去除线性趋势(Eliminate linear trend)。

```
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)
y = np.cos(t)
y_resampled = signal.resample(y, 50)
plt.subplot(1, 2, 1)
plt.title("Sampling")
plt.plot(x, y)
plt.plot(x[1::2], y_resampled, "o")

y2 = np.random.rand(100)+t
y2_detrended = signal.detrend(y2)
plt.subplot(1, 2, 2)
plt.title("Eliminate linear trend")
plt.plot(x, y2)
plt.plot(x, y2_detrended)
plt.show()
```

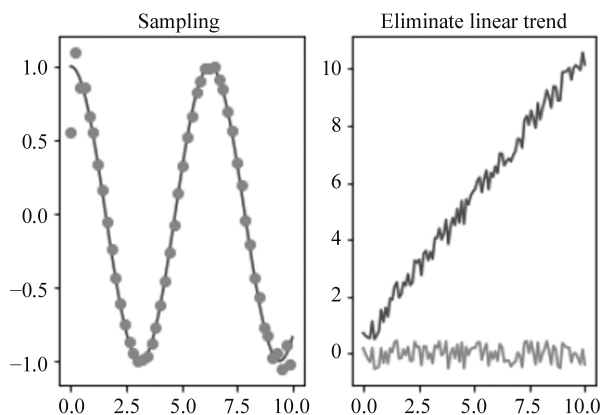


图 5.6 SciPy 关于信号处理的应用

4. Scikit-learn

Scikit-learn 是一个功能强大的 Python 机器学习库,为实现各种监督和半监督机器学习算法提供了简单、高效的工具。Scikit-learn 是开源的,每个人都能访问。它基于 NumPy、SciPy 和 Matplotlib 构建,极大地提高了它的可靠性和健壮性。Scikit-learn 内置丰富的数据集,如鸢尾花等,是一个简单高效的数据挖掘和数据分析工具。

Scikit 主要分为 6 个模块:分类、回归、聚类、降维、模型选择和预处理。机器学习算法主要有近邻分类算法、线性回归分类算法、随机森林分类算法、决策树分类算法、支持向量机、神经网络等。使用 Scikit-learn 训练模型的步骤如下。

- 导入模块
- 读入数据