第2章 Python编程基础

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 人工智能基础与应用 | | 章名 | | | Python编程基础 | |
| 教学内容 | Python编程基础2 | | | | 课时 | 2 | |
| 项目性质 | □演示性 □验证性 □设计性 √综合性 | | | | | | |
| 授课班级 |  | 授课日期 | |  | | 授课地点 |  |
| 教学目标 | 了解面向对象的概念  掌握Python第三方库的使用 | | | | | | |
| 教学内容 | 1. 面向对象 2. 第三方库的使用 | | | | | | |
| 教学重点 | Python第三方库的使用 | | | | | | |
| 教学难点 | Python第三方库的使用 | | | | | | |
| 教学准备 | 装有Python的计算机  教学课件PPT  教材：《人工智能基础与应用（微课版）》 | | | | | | |
| 作业设计 |  | | | | | | |

教学过程

|  |  |
| --- | --- |
| **教学环节** | **教学内容与过程**  **（教学内容、教学方法、组织形式、教学手段）** |
| **课前组织** | 做好上课前的各项准备工作（打开计算机、打开课件、打开软件、打开授课计划、教案等），吸引学生注意力。 |
| **课程说明** | 【课前说明】  从面向对象的几个重要名词引入本节课内容。  【目的】  使学生从了解本节课的学习目标、学习重点、考评方式等方面明确课程学习的要求和目标。 |
| **课程内容描述** | 2.4 面向对象  对象是人对各种具体物体进行抽象之后的概念，如一本书就是一个对象。对象有很多特性，如高度、宽度、颜色等。对象还有很多功能，如一部手机，可以听歌、看视频、打电话，这些可以理解成一个对象的技能。  在Python中，用变量表示特征，用函数表示技能，具有相同特征和技能的一类事物就是“类”，对象则是这一类事物中具体的一个。  在Python中，所有数据类型都可以视为对象。用户也可以自定义对象，自定义对象的数据类型就是面向对象中类（Class）的概念。  面向对象编程是一种编程方式。此编程方式需要使用“类”“对象”来实现，所以，面向对象编程其实就是对“类”“对象”的使用。每个对象都可以接收其他对象发过来的消息，并处理这些消息，消息在各个对象之间传递，一起协同工作来完成复杂的功能。  下面是关于面向对象的几个名词解释。  （1）类：一个类即是一系列具有相似特征和技能的对象的结合体，如同一个模板。类中定义了这些对象都具备的属性、共同的方法。  （2）属性：人类具有很多特征，把这些特征用程序来描述，就叫作属性，如年龄、身高、性别、姓名等都叫作属性。一个类中可以有多个属性。  （3）方法：人类不仅有身高、年龄、性别等属性，还能做很多事情，如说话、走路、吃饭等，属性是名词，说话、走路、吃饭是动词，这些动词用程序来描述就叫作方法。  （4）实例（对象）：一个类实例化后就是一个对象，一个类必须经过实例化才能在程序中调用执行类中的函数。一个类可以实例化成多个对象，每个对象亦可以有不同的属性，就像人类是指所有人，每个人是指具体的对象，人与人之间有共性，亦有不同。  一个类转变为一个对象的过程称为实例化。  面向过程和面向对象的思想是不同的，面向过程和面向对象的比较如表所示。  面向过程和面向对象的比较   |  |  |  | | --- | --- | --- | |  | 面向过程 | 面向对象 | | 思想 | 核心是“过程”二字，过程指的是解决问题的步骤，如设计一条流水线 | 特征和技能的结合体 | | 优点 | 将复杂的问题流程化，进而简单化 | 扩展性好，维护简单，易阅读 | | 缺点 | 扩展性较差 | 编程复杂度高 |   2.5 第三方库的使用  在介绍第三方库之前，需要首先介绍Python模块。Python模块（Module）是以.py结尾的文件，包含了Python对象定义和Python语句。它的特点是让开发者能够有逻辑地组织Python代码段。把相关的代码分配到一个模块里能让代码更好用，更易懂。模块能定义函数、类和变量，模块也可以包含可执行的代码。库是具有相关功能模块的集合。模块和库其实都是模块，只是具有个体和集合的区别。  Python有很多常用的内置模块，如表所示。  Python常用的内置模块   |  |  | | --- | --- | | 模块名称 | 模块功能 | | os模块 | （文件和目录）用于提供系统级别的操作 | | sys模块 | 用于提供对解释器的相关操作 | | time和datetime模块 | 用于提供与时间相关的操作 | | random模块 | 提供产生随机数的操作 | | logging模块 | 用于便捷地记录日志和线程安全的模块 | | json和pickle模块 | 用于数据传输的数据类型模块 | | shutil模块 | 高级的文件、文件夹、压缩包的处理模块（递归、文件复制等） |   【例2-41】 使用内置模块输出系统当前时间，在交互模式下输入代码。  >>> import datetime  >>> nowTime=datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')  >>> print(nowTime)  2019-06-13 13:06:50  第三方库的存在使得Python更加灵活。第三方库的安装有以下两种方式。  第一种方式是源文件安装，下载源码进行安装。  第二种方式是使用pip工具在线安装。  接下来将介绍3个第三方库，先安装将在2.5.1小节中介绍的NumPy。  在交互模式（联网状态）下输入：  pip install numpy==1.14.5  使用pip工具安装时要注意库的版本，输入的内容中：“pip”表示使用pip工具，“install”表示安装，要卸载需要使用“uninstall”，“numpy”表示要安装的库的名称，“==1.14.5”表示安装的版本号。本书中使用pip工具安装的都是现阶段的稳定版本，在安装时如果不加“==1.14.5”则安装最新版本。最新版本和本书中安装的版本可能会出现API不兼容的情况，建议依照本书安装对应的版本，pip工具在线安装NumPy界面如图所示。    pip工具在线安装NumPy界面  可以从Python包索引网站（PyPI）上将.whl文件下载下来，执行“pip install \*.whl”命令来离线安装第三方库。  安装完成后，在交互模式下输入“import numpy as np”来查看是否报错，其中“np”是NumPy的别名。不报错表示安装成功，NumPy安装成功界面如图所示。    NumPy安装成功界面  2.5.1 NumPy  NumPy（Numerical Python）是Python语言的一个扩展程序库，其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了大量的数学函数库。  1．安装与简介  NumPy官网上将NumPy概括为：NumPy具有强大的*N*维数组对象ndarray，广播功能函数，整合C、C++、Fortran代码的工具。另外，其还包括傅里叶变换、随机数生成等功能。  NumPy的特点之一是其*N*维数组对象ndarray，它是一系列同类型数据的集合，以0为下标开始进行集合中元素的索引。ndarray对象是用于存放同类型元素的多维数组。  2．基础操作  （1）数组的创建  方式一：通过array()函数将Python的列表或者元组转换为数组，数组中的类型是由列表或者元组原有的数据类型推导出的。  【例2-42】 利用array()函数生成数组，在交互模式下输入以下代码。  >>> import numpy as np  >>> a = np.array([2,3,4]) # 列表转换为NumPy  >>> a  array([2, 3, 4])  >>> a.dtype # NumPy的类型  dtype('int32')  >>> b = np.array((1.2, 3.5, 5.1)) # 元组转换为NumPy  >>> b  array([1.2, 3.5, 5.1])  >>> b.dtype  dtype('float64')  >>> c = np.array([(1.5,2,3), (4,5,6)]) # 二维NumPy  >>> c  array([[1.5, 2. , 3. ],  [4. , 5. , 6. ]])  >>> c.dtype  dtype('float64')  方式二：NumPy提供了几个函数来创建具有初始占位符内容的数组。zeros()函数可以创建一个全为0的数组；ones()函数可以创建一个全为1的数组；empty()函数可以创建一个初始内容随机的数组，其内容取决于内存的状态。默认情况下，创建的数组类型是float64。  【例2-43】 生成具有占位符的数组，在交互模式下输入以下代码。  >>> np.zeros((3,4))  array([[0., 0., 0., 0.],  [0., 0., 0., 0.],  [0., 0., 0., 0.]])  >>> np.ones((2,3,4), dtype=np.int16) # 可以指定类型  array([[[1, 1, 1, 1],  [1, 1, 1, 1],  [1, 1, 1, 1]],  [[1, 1, 1, 1],  [1, 1, 1, 1],  [1, 1, 1, 1]]], dtype=int16)  >>> np.empty((2,3))  array([[1.5, 2. , 3. ],  [4. , 5. , 6. ]])  方式三：NumPy通过arange()函数得到数组，arange()函数的原型为arange(start, end, step)，起始值为start，终止值为end，但不含终止值，步长为step。arange()函数可以使用float型数据。  【例2-44】 利用arange()函数生成数组，在交互模式下输入以下代码。  >>> np.arange(1, 5)  array([1, 2, 3, 4])  >>> np.arange(1, 5, 2)  array([1, 3])  （2）打印数组  打印数组时，NumPy以与嵌套列表类似的方式显示。  【例2-45】 将一维数组打印为行，将二维数组打印为矩阵，将三维数组打印为矩阵列表，在交互模式下输入以下代码。  >>> a = np.arange(6)  >>> a  array([0, 1, 2, 3, 4, 5])  >>> a = np.arange(12).reshape(4, 3) # reshape用于改变数组形状  >>> a  array([[ 0, 1, 2],  [ 3, 4, 5],  [ 6, 7, 8],  [ 9, 10, 11]])  >>> a = np.arange(24).reshape(4, 3, 2)  >>> a  array([[[ 0, 1],  [ 2, 3],  [ 4, 5]],  [[ 6, 7],  [ 8, 9],  [10, 11]],  [[12, 13],  [14, 15],  [16, 17]],  [[18, 19],  [20, 21],  [22, 23]]])  （3）数组运算  数组上的算术运算符会应用于元素运算。  【例2-46】 进行数组的加减，在交互模式下输入以下代码。  >>> b = np.arange(4, 8)  >>> a = np.arange(1, 4)  >>> a  array([1, 2, 3])  >>> b = np.arange(4, 7)  >>> b  array([4, 5, 6])  >>> a + b  array([5, 7, 9])  >>> a - b  array([-3, -3, -3])  运算符“\*”在NumPy数组中以元素方式运行。矩阵乘积可以使用“@”运算符（Python 3.5及以上版本中）或dot()函数。  【例2-47】 学习数组元素相乘与数组矩阵相乘，在交互模式下输入以下代码。  >>> a = np.array( [[1, 1],  ... [0, 1]])  >>> a  array([[1, 1],  [0, 1]])  >>> b = np.array( [[2, 1],  ... [3, 4]])  >>> a \* b  array([[2, 1],  [0, 4]])  >>> a @ b  array([[5, 5],  [3, 4]])  >>> a.dot(b)  array([[5, 5],  [3, 4]])  （4）形状改变  NumPy可以使用命令更改矩阵的形状。ravel()、reshape()、T这3个方法会返回已修改的矩阵，但不会更改原始矩阵；resize()方法会更改原始矩阵。  【例2-48】 学习改变NumPy矩阵形状的操作，在交互模式下输入以下代码。  >>> a = np.array([(1,2,3), (4,5,6)]) # 创建NumPy  >>> a  array([[1, 2, 3],  [4, 5, 6]])  >>> a.shape  (2, 3)  >>> a.ravel() # NumPy展平  array([1, 2, 3, 4, 5, 6])  >>> a.ravel().shape  (6,)  >>> a.reshape(3, 2) # NumPy修改形状  array([[1, 2],  [3, 4],  [5, 6]])  >>> a.reshape(3, 2).shape  (3, 2)  >>> a.T # NumPy转置  array([[1, 4],  [2, 5],  [3, 6]])  >>> a.T.shape  (3, 2)  NumPy也可以进行索引、切片、迭代、堆叠、拆分等操作。  2.5.2 Pandas  Pandas 是一个开源的、BSD许可的库，常用于数据分析。它基于NumPy库，为Python编程语言提供高性能、易于使用的数据结构和数据分析工具。  1．安装与简介  在交互模式（联网状态）下完成Pandas的安装，输入以下代码。  pip install pandas==0.23.4  Pandas安装界面如图所示。    Pandas安装界面  在交互模式下输入“import pandas as pd”，pd是Pandas的别名，是通用写法，如图所示。    Pandas安装成功  2．基础操作  （1）创建Series对象  Series对象是Pandas中的一维数据结构，能存储不同类型的数据。Series对象中的每个元素都有一组索引与之对应，可以将其看作特殊的Python字典。  【例2-49】 创建一个Series对象并输出，在交互模式下输入以下代码。  >>> import pandas as pd  >>> a = pd.Series([1, -5, [1,2], "aa", {"aa": 7}])  >>> a  0 1  1 -5  2 [1, 2]  3 aa  4 {'aa': 7}  dtype: object  Series对象可以传入数字、列表、字典等类型，将该对象打印会输出元素和与之对应的索引。如果在传入的时候没有加入索引，会默认将非负整数作为索引。Pandas中的整型为int64，浮点型为float64，字符串、布尔型等其他数据类型都为object。  【例2-50】 创建一个Series对象，为其增加索引并输出，在交互模式下输入以下代码。  >>> import pandas as pd  >>> a = pd.Series(["b", 2, [1 ,2]], index = ["a", "b", "c"])  >>> a  a b  b 2  c [1, 2]  dtype: object  打印输出的不仅有元素，还有索引值，索引与元素之间是一种映射关系。如果填充的是一个字典，那么字典的键就会成为索引，值成为元素。  【例2-51】 创建一个字典，将其填充到Series对象并输出，在交互模式下输入以下代码。  >>> import pandas as pd  >>> dict = {"a":1, "b":2, "c":3}  >>> a = pd.Series(dict)  >>> a  a 1  b 2  c 3  dtype: int64  如果元素都是整数，那么默认的数据类型就为int64。  （2）访问Series元素  Series对象中的元素可以像列表那样被访问。由于它拥有索引和下角标两种属性，所以通过这两种方式都可以访问元素。如果不指定索引，那么索引和下角标的数值就默认是一样的。  【例2-52】 访问Series对象的元素并输出，在交互模式下输入以下代码。  >>> import pandas as pd  >>> a = pd.Series(["b", 2, [1 ,2]], index = ["a", "b", "c"])  >>> a  a b  b 2  c [1, 2]  dtype: object  >>> a[1]  2  >>> a["b"]  2  可以对Series对象的元素和索引进行修改。  【例2-53】 修改Series对象的元素和索引并输出，在交互模式下输入以下代码。  >>> import pandas as pd  >>> a = pd.Series([1, 2, 3], index = ["a", "b", "c"])  >>> a  a 1  b 2  c 3  dtype: int64  >>> a[1] = 6 # 修改元素  >>> a  a 1  b 6  c 3  dtype: int64  >>> a.index = ["d", "e", "f"] # 修改索引  >>> a  d 1  e 6  f 3  dtype: int64  修改元素和修改索引一样，都是在原处进行的，不会在内存中进行复制操作。  （3）创建DataFrame对象  DataFrame是Pandas中的二维数据结构，能存储不同类型的数据，有行索引和列索引，并与元素对应。如果将Series对象比作带灵活索引的一维数组，那么DataFrame对象就可以看作既有行索引又有列索引的二维数组。  【例2-54】 从字典中构造DataFrame对象并输出，在交互模式下输入以下代码。  >>> import pandas as pd  >>> a = {"a":[1, 2], "b": [3, 4]}  >>> a  {'a': [1, 2], 'b': [3, 4]}  >>> df = pd.DataFrame(data=a)  >>> df  a b  0 1 3  1 2 4  可以看出，有行索引0、1，也有列索引a、b。  2.5.3 Matplotlib  Matplotlib是Python的绘图库，它可与NumPy一起使用，提供了一种有效的MATLAB开源替代方案。  1．安装与简介  在交互界面（联网状态）完成Matplotlib的安装，代码如下。  pip install matplotlib==3.0.1  安装完成后，在交互模式下输入“import matplotlib.pyplot as plt”，如图所示。    Matplotlib安装成功  Matplotlib可以和图形工具包一起使用，如PyQt和wxPython。使用Matplotlib时，可以用几行代码进行绘图，包括直方图、功率谱图、条形图、错误图、散点图等。  本小节主要介绍Matplotlib的pyplot模块，该模块提供了类似于MATLAB的绘图系统。  2．基础操作  Matplotlib中最重要的功能是plot，它可以绘制二维数据。  【例2-55】 在basis目录下新建文件，命名为sin.py，绘制一个正弦图像，在PyCharm中输入以下代码。  import numpy as np  import matplotlib.pyplot as plt  # 计算余弦曲线上点的x和y坐标  x = np.arange(0, 3 \* np.pi, 0.1)  y = np.sin(x)  # 使用Matplotlib绘制点  plt.plot(x, y)  # 调用plt.show()才能显示图形  plt.show()  导入numpy和matplotlib.pyplot模块，使用NumPy定义*x*轴和*y*轴，*x*轴的范围为0～3倍的π值，np.pi是圆周率，*x*轴每隔0.1的精度取一个值，*y*轴为sin(x)的值，调用plt.plot(x,y)函数绘制点，调用plt.show()函数将绘制的图像显示出来。正弦图像绘制结果如图所示。    正弦图像绘制结果  此时得到了0～3倍的π区间的正弦图像。  在绘制正弦图像的代码中加入其他的元素，可以绘制多条线，并添加标题、图例和轴标签。  【例2-56】 在basis目录下新建文件，命名为sin\_cos.py，绘制一个正弦和余弦图像，并且添加标题、图例和轴标签，在PyCharm中输入以下代码。  import numpy as np  import matplotlib.pyplot as plt  # 计算正弦曲线和余弦曲线上点的x和y坐标  x = np.arange(0, 3 \* np.pi, 0.1)  y\_sin = np.sin(x)  y\_cos = np.cos(x)  # 使用Matplotlib绘制点  plt.plot(x, y\_sin)  plt.plot(x, y\_cos)  plt.xlabel('x axis label')  plt.ylabel('y axis label')  plt.title('Sine and Cosine')  plt.legend(['Sine', 'Cosine'])  plt.show()  导入numpy和matplotlib.pyplot模块，使用NumPy定义*x*轴和*y*轴，*x*轴的范围为0～3倍的π，*y*轴包括y\_sin=sin(x)和y\_cos=cos(x)，然后利用plt.plot()函数绘制图像，并且利用plt.xlabel()函数和plt.ylabel()函数加入*x*轴标签和*y*轴标签，使用plt.title()函数加入标题，使用plt.legend()函数显示图像示例，使用plt.show()函数将绘制的图像显示，加入更多元素后的正弦和余弦图如图所示。    加入更多元素后的正弦和余弦图  在一幅图像中的不同区域绘制不同的图，将sin()函数和cos()函数在一幅图的不同区域绘制。  【例2-57】 在basis目录下新建文件，命名为sin\_cos\_part.py，绘制一幅图，将正弦、余弦图像绘制在图的不同区域，并且添加标题，在PyCharm中输入以下代码。  import numpy as np  import matplotlib.pyplot as plt  # 计算正弦曲线和余弦曲线上点的x和y坐标  x = np.arange(0, 3 \* np.pi, 0.1)  y\_sin = np.sin(x)  y\_cos = np.cos(x)  # 将第一个子图设置为活动状态，制作第一个区域  plt.subplot(2, 1, 1)  plt.plot(x, y\_sin)  plt.title('Sine')  # 将第二个子图设置为活动状态，制作第二个区域  plt.subplot(2, 1, 2)  plt.plot(x, y\_cos)  plt.title('Cosine')  # 显示图  plt.show()  导入模块并确定*x*和*y*的坐标范围，plt.subplot()函数划分一幅图片中的不同区域，第一个参数代表子图的行数，第二个参数代表该行图像的列数，第三个参数代表每行的第几个图像。参数(2,1,1)是指将一幅图像分为上、下两个区域，操作上面的区域，绘制正弦图像并显示标题；参数(2,1,2)是指将一幅图像分为上、下两个区域，操作下面的区域，绘制余弦图像并显示标题。plt.show()函数用于显示图像，在不同区域绘制不同的图，如图所示。    在不同区域绘制不同的图  Matplotlib的plt.scatter()函数可以绘制散点图，函数原型为：  matplotlib.plt.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, verts=None, edgecolor=None, \*,  data=None, \*\*kwargs)  经常用到的参数如下。  （1）x、y：表示的是数组，描述绘制散点图的数据点。  （2）c：颜色，是一个可选项。默认是蓝色“b”。该参数可以是一个表示颜色的字符，或者是一个长度为*n*的、表示颜色的序列等。但是c不可以是一个单独的RGB数字，也不可以是一个RGBA的序列。  （3）marker：标记的样式，默认为“o”。  （4）vmin、vmax：实数，当norm参数存在的时候忽略，用来进行亮度数据的归一化。  （5）edgecolor：边缘颜色，可以利用它将散点在视觉上缩小。  【例2-58】 在basis目录下新建文件，命名为scatter.py，绘制一幅正弦散点图，在PyCharm中输入以下代码。  import numpy as np  import matplotlib.pyplot as plt  # 计算正弦曲线上点的x和y坐标  x = np.arange(0, 3 \* np.pi, 0.1)  y = np.sin(x)  plt.scatter(x, y)  plt.show()  定义*x*和*y*的范围，用plt.scatter()函数画出散点图，使用plt.show()函数显示图像，散点图如图所示。    散点图  调用保存函数plt.savefig()可保存绘制的图像。  【例2-59】 在basis目录下新建文件，命名为save.py，绘制一幅正弦散点图并保存，在PyCharm中输入以下代码。  import numpy as np  import matplotlib.pyplot as plt  # 计算正弦曲线上点的x和y坐标  x = np.arange(0, 3 \* np.pi, 0.1)  y = np.sin(x)  plt.scatter(x, y)  plt.savefig("image.png")  plt.show()  使用plt.savefig()函数保存结果，参数是保存的路径以及名称。 |
| **总结评价** | 本节课主要讲解面向对象的基本概念和第三方库的使用。 |