Python编程及人工智能应用

第二章 Python语言基础

内容提要

- Python基本语法
  - 变量和数据类型
  - 运算符和表达式
  - 字符串
  - 流程控制
- Python组合数据类型
- Python函数
- 异常处理和文件操作
- Python面向对象基础
- 数值计算库NumPy

Python语法基本特点

- 使用**缩进**来表示程序的层次结构。不同于C语言，**Python变量可不声明直接使用**
  - 同一层次代码块，缩进所包含空格数量必须一致

```
#代码2.1 接收用户从键盘上输入的整数，并判别其奇偶性
num = int(input("请您输入一个整数："))
if num%2==0:
    print(num,"是一个偶数")
    print("这里执行的是程序中的第1个分支")
else:
    print(num,"是一个奇数")
    print("这里执行的是程序中的第2个分支")
```

  - 输入函数input()，从键盘输入一个字符串
    - int()函数用于把字符串转换成整数
  - 输出函数print()，输出后自动换行

Python对象及类型

- Python 所有数据都由**对象**或**对象间关系**来表示
- 对象 约等于 C语言常量
- Python常见的内置标准类型
  - 整型数字（int）、浮点型数字（float）、复数型数字（complex）、逻辑值（bool）、字符串（str）、列表（list）、元组（tuple）、可变集合（set）、不可变集合（frozenset）、字典（dict）

Python对象及类型

```
#代码2.2 在屏幕上输出各种对象的类型
print("100 is", type(100))
print("3.14 is", type(3.14))
print("5+2j is", type(5+2j))
print("True and False are", type(False))
print("\'I love Python.\' is", type("I love Python."))
print("[1,2,3] is", type([1,2,3]))
print("(1,2,3) is", type((1,2,3)))
print("{1,2,3} is", type({1,2,3}))
print("frozenset({1,2,3}) is",
        type(frozenset({1,2,3})))
print("{'name':'Tom','age':18} is",
        type({'name':'Tom','age':18}))
```

代码2.2运行后输出如下：

```
100 is <class 'int'>
3.14 is <class 'float'>
5+2j is <class 'complex'>
True and False are <class 'bool'>
'I love Python.' is <class 'str'>
[1,2,3] is <class 'list'>
(1,2,3) is <class 'tuple'>
{1,2,3} is <class 'set'>
frozenset({1,2,3}) is <class 'frozenset'>
{'name':'Tom','age':18} is <class 'dict'>
```

- type()是Python的内置函数，返回对象所属类型

Python变量和关键字

- 标识符的命名规则
  - 由大写和小写字母、下划线 _ 以及数字组成
  - 不可以数字打头，可以包含Unicode字符（中文）
  - 不可以和Python中的关键字重复
- Python关键字

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Python变量赋值

南京邮电大学 Nanjing University of Posts and Telecommunications

- 赋值语句可以将**变量**和**对象**进行**关联**

```
#代码2.3 使用赋值语句建立变量和对象之间的关联
#这是使用科学记数法表示浮点数0.0178的方法
var = 1.78E-2
print(var, type(var))
#复数的表示中，带有后缀j或者J的部分为虚部
var = 3+7j
print(var, type(var))
#字符串数据既可以用单引号，也可以用双引号进行定义
var = "人生苦短，我用Python"  print(var, type(var))
```

运行代码2.3输出如下

```
0.0178 <class 'float'>
(3+7j) <class 'complex'>
人生苦短，我用Python <class 'str'>
```

```
#代码2.4 赋值语句的连续赋值和多重赋值
#用于将不同的变量关联至同一个对象
x = y = z = 100
print(x, y, z)
#用于将不同的变量关联至不同的对象
a, b, c = 7, 8, 9
print(a, b, c)
```

运行代码2.4后输出如下

```
交换之前x与y的值为： 3 5
交换之后x与y的值为： 5 3
```

Python运算符和表达式

- 表达式可以由多个**运算对象**和**运算符**构成
- Python按照运算符的**优先级**求解表达式的值
  - Python常见运算符（从最低优先级到最高优先级）

运算符	运算符描述
if -- else	条件运算符
or	逻辑或运算
and	逻辑与运算
not x	逻辑非运算
in、not in、is、is not、<、<=、>、>=、!=、==	比较运算
|	按位或运算
^	按位异或运算
&	按位与运算
<<、>>	移位运算
+、-	算术运算符：加、减
*、/、//、%	算术运算符：乘、除、整除、取模（求余数）
+x、-x、~x	单操作数运算符：正、负、按位非运算
**	乘方运算符

Python运算符和表达式

南京邮电大学 Nanjing University of Posts and Telecommunications

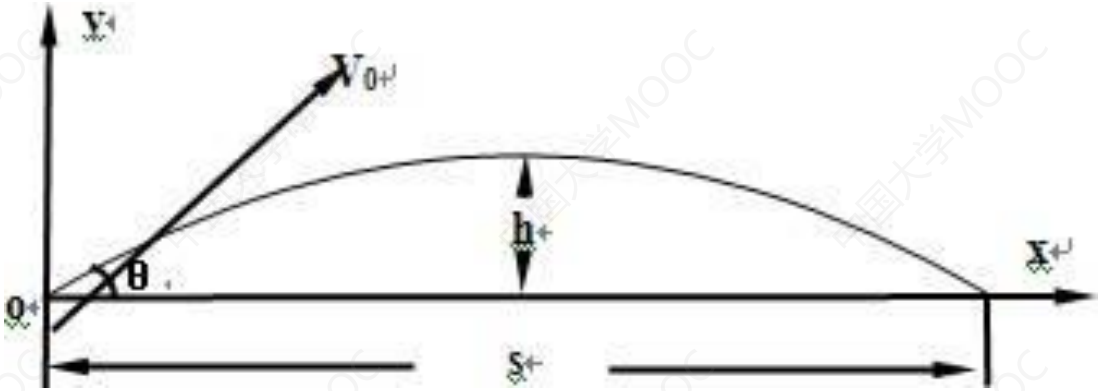
```
#代码2.6 表达式举例
print(1+2*3-4)      #由于乘号的优先级高于加号和减号，所以2*3会优先计算
print(2*3**3)       #乘方运算的优先级比乘法运算还要高
a, b = 10, 20
print(a if a>b else b)          #输出变量a和b中数值较大的那一个
print(True+1)       #运算过程中，True和False可以被自动转换成1和0使用
print(3>2>1)        #连续的比较运算，其结果与表达式3>2 and 2>1等价
print((3>2)>1)      #这条表达式计算的是子表达式3>2的值是否大于1，即True>1
```

运行代码2.6后输出如下：

```
3
54
20
2
True
False
```

数学计算库math的使用

- 背景知识
  - 将物体以一定的初始速率V0和角度θ向空中抛出，仅在重力作用下物体所做的运动叫做抛物运动，如下图所示：



$$R = \frac{V_0^2 \bullet \sin(2\theta)}{g}$$

- 思考题：若将sin函数中的参数“2*theta/180.0*pi”改成“2*theta/180*pi”，对上例的运行结果有无影响？

Python字符串

- 字符串类型的对象使用一对**单引号**、一对**双引号**或者一对**三引号**进行定义
  - 使用三引号定义字符串的时候可以包含换行符

```
s = '''Hello
      World'''
```

- Python字符串转义字符

转义字符	含义	转义字符	含义
\'	单引号	\t	水平制表符
\"	双引号	\v	垂直制表符
\\	字符“\”本身	\r	回车符
\a	响铃	\f	换页符
\b	退格符	\ooo	以最多3位的八进制数作为编码值对应的字符
\n	换行符	\xhh	以必须为2位的十六进制数作为编码值对应的字符

Python字符串举例


```
#代码2.7 定义字符串的几种方法
str1 = '单引号可以用来定义字符串'
str2 = "双引号也可以用来定义字符串"
print(str1, str2)
str3 = '''三引号定义的字符串可以直接换行
这是字符串的第二行
这是字符串的第三行'''
print(str3)
print("转义字符\n表示换行符")
print(r"转义字符\n表示换行符")
#格式标记.2f表示保留小数点后2位小数
print(f"半径为5的圆的面积是{3.14*5**2:.2f}")
```

运行代码2.7后输出如下：

```
单引号可以用来定义字符串 双引号也可以用来定义字符串
三引号定义的字符串可以直接换行
这是字符串的第二行
这是字符串的第三行
转义字符
表示换行符
转义字符₩n表示换行符
半径为5的圆的面积是78.50
```

Python字符串运算

- f-string，亦称为格式化字符串常量，从Python3.6新引入，目的是使格式化字符串操作更加简便
- f-string在形式上是以f或F修饰符引领的字符串
  - 格式：f"{表达式}"

```
#代码2.8 与字符串有关的运算
#字符串之间用加法运算连接，结果为连接后的字符串
print("ABCD"+"1234")
#字符串和整数进行乘法运算，结果为字符串复制多遍并连接
print("Hi"*3)
str1 = "I love Python!"
#获取字符串str1中的首个字符
print(f"{str1[0] = }")
#获取字符串str1的最后一个字符
print(f"{str1[-1] = }")
#获取字符串中索引号从2开始到6之前结束的子字符串，即"love"
print(f"{str1[2:6] = }")
```

运行代码2.8后输出如下：

```
ABCD1234
HiHiHi
str1[0] = 'I'
str1[-1] = '!'
str1[2:6] = 'love
```

Python字符串的常用函数

南京邮电大学 Nanjing University of Posts and Telecommunications

- 字符串对象有一系列已定义好的函数可直接使用

```
#代码2.9 字符串对象的常用方法
str1 = "i have an apple."
print(f"{str1.capitalize() = }") #用于将字符串的首字母大写
print(f"{str1.title() = }") #用于将字符串中每个单词的首字母大写
print(f"{str1.count('a') = }") #用于统计指定的字符在字符串中出现的次数
print(f"{str1.startswith('i am') = }") #用于判定字符串是否以指定的参数开始
print(f"{str1.endswith('apple.') = }") #用于判定字符串是否以指定的参数结尾
print(f"{str1.find('apple') = }") #用于查找指定的子字符串并返回其起始位置
print(f"{str1.find('pear') = }") #若无法找到指定的子字符串，find()方法会返回-1
print(f"{str1.split() = }") #对字符串进行切割，默认以空格作为分隔符
print(f"{','.join(['a','b','c']) = }") #用指定的字符串将若干字符串类型的对象进行连接
print(f"{'ABcd'.upper() = }") #将字符串中的字母转换成大写
print(f"{'ABcd'.lower() = }") #将字符串中的字母转换成小写
print(f"{' ABcd '.strip() = }") #删除字符串前后的特殊字符，比如空格和换行符
```

运行代码2.9后的输出结果：

```
str1.capitalize() = 'I have an apple.'
str1.title() = 'I Have An Apple.'
str1.count('a') = 3
str1.startswith('i am') = False
str1.endswith('apple.') = True
str1.find('apple') = 10
str1.find('pear') = -1
str1.split() = ['i', 'have', 'an', 'apple.']
','.join(['a','b','c']) = 'a,b,c'
'ABcd'.upper() = 'ABCD'
'ABcd'.lower() = 'abcd'
' ABcd '.strip() = 'ABcd'
```

Python选择结构

- 用if...elif...else来构造选择结构的程序代码
- python没有switch语句

```
#代码2.10 将百分制成绩转换成五级制成绩
score = float(input("请输入一个百分制成绩： "))
if score>=90:
    print(f"成绩{score}对应的五级制成绩是优秀")
elif score>=80:
    print(f"成绩{score}对应的五级制成绩是良好")
elif score>=70:
    print(f"成绩{score}对应的五级制成绩是中等")
elif score>=60:
    print(f"成绩{score}对应的五级制成绩是及格")
else:
    print(f"成绩{score}对应的五级制成绩是不及格")
```

运行代码2.10后的输出结果：

```
请输入一个百分制成绩：95
成绩95.0对应的五级制成绩是优秀
```

Python选择结构

- 通过组合多个if...elif...else结构进行嵌套使用

```
#代码2.11 判断从键盘上输入的三个整数构成的三角形形状
a,b,c = input("请输入组成三条边，用空格分隔：").split()
a,b,c = int(a), int(b), int(c)
a,c,b = min(a,b,c), max(a,b,c), a+b+c-min(a,b,c)-max(a,c,b)
if a+b<=c:
    print("输入的整数无法构成三角形")
else:
    if a==b==c:
        print("输入的整数构成一个等边三角形")
    elif a==b:
        print("输入的整数构成一个等腰三角形")
    elif a**2+b**2==c**2:
        print("输入的整数构成一个直角三角形")
    else:
        print("输入的整数构成一个普通三角形")
```

运行代码2.11后的输出结果：

```
请输入组成三角形三条边的整数，用空格分隔：1 1 2
输入的整数无法构成三角形
请输入组成三角形三条边的整数，用空格分隔：3 3 3
输入的整数构成一个等边三角形
请输入组成三角形三条边的整数，用空格分隔：3 3 5
输入的整数构成一个等腰三角形
请输入组成三角形三条边的整数，用空格分隔：3 4 5
输入的整数构成一个直角三角形
请输入组成三角形三条边的整数，用空格分隔：5 6 7
```

Python循环结构

- 用关键字**while**或者**for**可以用来构造循环结构
  - while语句实现累加

```
#代码2.12 使用while循环完成1至100所有整数求和
n = 1
s = 0
while n<=100:
    s += n          #与s = s + n等价
    n += 1          #与n = n + 1等价
print("1到100所有整数的和是：",s)
```

  - for语句实现累加

```
#代码2.13 使用for循环完成1至100所有整数求和
s = 0
for n in range(1,101):
    s += n
print("1到100所有整数的和是：",s)
```

  - range(1,101)所表示的从1到101以内（不含101）的所有整数

Python循环结构

- 如果range()函数只有一个参数，则该参数表示返回结果的终止值（不含）
- 如果range()函数有两个参数，则第1个参数表示返回结果的起始值，第2个参数表示返回结果的终止值（不含）
- 如果range()函数有三个参数，则第1个参数表示返回结果的起始值，第2个参数表示返回结果的终止值（不含），第3个参数表示每次步进的增量值

```
#代码2.14 求100以内所有奇数的和
#sum()函数的功能为对其参数进行求和
s = sum(range(1,100,2))
print("100以内所有奇数的和是：",s)
```

Python循环结构

- **break**语句会终结最近的外层循环，如果循环有可选的else子句，也会跳过该子句
- **continue**语句会终止当前循环中剩下的语句，并继续执行最近的外层循环的下一个轮次

```
#代码2.15 输出从小到大排列的第100个素数
count = 0
num = 2
while True:
    #下方的for循环用于判断num是否为素数，如果是则计数器count加1
    for i in range(2,num):
        if num%i==0:
            break
    else:
        count += 1
```

```
#如果计数器小于100，num加1，且终止当前循环中剩下的语句，并继续执行循环的下一个轮次
    if count<100:
        num += 1
        continue
    #剩下的代码只有在上方if不成立的时候才会被执行到，即计数器count为100的情况下
    print(num,"是从小到大排列的第100个素数")
    break                    #输出完毕后，使用break语句终结外层的while循环
```

控制语句应用举例1

- 数列求和。
  - 已知一个数列如下，求该数列前1000项的和。

$$s=\left\{1,-\frac{1}{3},\frac{1}{5},\ldots,\frac{(-1)^{i+1}}{2i-1}\ldots\right\}$$

- i = 1, 2, ...

控制语句应用举例2

- 加法表打印。
  - 要求打印如下所示的九九加法表。

```
1+1= 2
2+1= 3  2+2= 4
3+1= 4  3+2= 5  3+3= 6
4+1= 5  4+2= 6  4+3= 7  4+4= 8
5+1= 6  5+2= 7  5+3= 8  5+4= 9  5+5=10
6+1= 7  6+2= 8  6+3= 9  6+4=10 6+5=11 6+6=12
7+1= 8  7+2= 9  7+3=10 7+4=11 7+5=12 7+6=13 7+7=14
8+1= 9  8+2=10 8+3=11 8+4=12 8+5=13 8+6=14 8+7=15 8+8=16
9+1=10 9+2=11 9+3=12 9+4=13 9+5=14 9+6=15 9+7=16 9+8=17 9+9=18
```

控制语句应用举例3

- 梯形打印。
  - 要求打印如下所示的等腰梯形。

```
  ***
 *****
*******
*********
```

控制语句应用举例4

- 质数判断。
  - 从键盘输入一个正整数n，判断n是否为质数。
- 分析：质数判断的基本思路是，对于正整数n（n>1），用2~$\sqrt{n}$去除它，如果存在可以整除的情况，则n不是质数，否则必为质数。

控制语句应用举例5

- 百钱百鸡问题。
  - 公鸡5钱一只，母鸡3钱一只，小鸡1钱三只。一百钱买一百只鸡，问公鸡、母鸡、小鸡各几只？
- 分析：设公鸡、母鸡、小鸡的数量分别为a、b、c，则有

$$\begin{cases} a + b + c = 100 \\ 5a + 3b + c / 3 = 100 \end{cases}$$

组合数据类型——列表

- 列表（list）是一种序列数据类型，是一种可变数据类型，内部元素可以任意增删和修改
- 列表对象的定义和基本运算如下：

```
##代码2.16 列表对象的定义和基本运算
lst1 = []  #定义一个空列表
#len()函数可以返回参数对象中包含元素的数量
print(f"{len(lst1) = }")
#list()函数可以将参数对象转换成列表
lst1 = list("Python!")
print(f"{lst1 = }")
#列表也是序列类型对象，所以支持索引和切片
print(f"{lst1[2] = }, {lst1[-2] = }")
print(f"{lst1[2:-2] = }")
```

```
lst1 = [1,2,3]
lst2 = [1,2,3]
#通过id()函数我们可以取得对象的身份标识
print(f"{id(lst1) = }, {id(lst2) = }")
print("lst1和lst2是否相等：", lst1==lst2)
print("lst1和lst2是否相同：", lst1 is lst2)
#不同的变量与同一个列表对象关联
lst1 = lst2 = [1,2,3]
#无论操作变量lst1或者lst2，其实都是在操作同一个列表
lst1[0] = 100
print(f"{lst1 = }, {lst2 = }")
```

列表的常用方法

- 列表对象具有一系列定义好的方法可以直接使用

```
#代码2.17 列表对象的常用方法
lst1 = list("Python")
print(f"{lst1 = }")
lst1.append("666")  #追加元素到列表中
print(f'After lst1.append("666"), {lst1 = }')
lst1.insert(1, "is")  #插入元素到列表中
print(f'After lst1.insert(1, "is"), {lst1 = }')
lst1.remove("is")  #从列表中移除指定的元素
print(f'After lst1.remove("is"), {lst1 = }')
#从指定位置弹出列表元素
# 默认弹出列表的最后一个元素
print(f'{lst1.pop(0) = }')
print(f'After lst1.pop(0), {lst1 = }')
#将参数转换成列表元素后
#连接到原列表的末尾
lst1.extend("Love")
print(f'After lst1.extend("Love"), {lst1 = }')
#对列表元素按照ASCII码从小到大的顺序进行排序
lst1.sort()
print(f'After lst1.sort(), {lst1 = }')
#在列表中对指定的内容进行计数
print("'o'在lst1中出现了", lst1.count("o"), "次")
#在列表中查找指定的内容，返回其索引值
print("'t'在lst1中的索引值是：", lst1.index("t"))
lst1 = [1,2,3]
#通过copy()方法可以复制原列表对象
#得到一个新的列表对象
lst2 = lst1.copy()
lst2[0] = 100
print(f"{lst1 = }, {lst2 = }")
```

列表推导式

- 列表推导式将列表元素的生成规则放置在一对方括号内完成列表的创建

```
#代码2.18 列表推导式的使用
lst1 = [n for n in range(1,10) if n%2==1]   #创建列表包含10以内的奇数
print(f"{lst1 = }")
lst2 = [5*n for n in range(5)]    #创建等差数列构成的列表
print(f"{lst2 = }")
lst3 = [3**n for n in range(5)]   #创建等比数列构成的列表
print(f"{lst3 = }")
```

组合数据类型——元组

- 元组（tuple）也是一种序列数据类型，但是它是一种不可变数据类型，其元素不可增删改

```
#代码2.19 元组对象的定义和基本运算
#定义空元组、一元组、二元组
tup1,tup2,tup3 = tuple(),(1,),(2,3)
print(f"{tup1 = }, {tup2 = }, {tup3 = }")
#通过+运算连接两个旧元组，得到一个新元组
tup1 = tup2 + tup3
print(f"{tup1 = }")
tup1 = tuple("Python!")
#元组也支持索引和切片运算
print(f"{tup1[2] = }, {tup1[3:6] = }")
#由于元组是不可变对象，元素不可以修改
#所以该语句报错
tup1[2] = 'a'
```

```
#代码2.20 元组对象的常用方法
#tuple()函数可以将参数对象转换成元组
tup1 = tuple("Beautiful is better than ugly.")
print(f"{tup1 = }")
#统计指定的参数在元组中出现的次数
print(f"{tup1.count('t') = }")
#查找指定的参数，并返回其在元组中的索引值
print(f"{tup1.index('a') = }")
```

组合数据类型——字典

- 字典（dict）是表示映射关系的类型，由**键-值**对构成字典中的元素，是一种可变数据类型
- 字典不是序列类型，其中元素不能使用索引值进行访问，而是通过键访问对应元素，**键必须唯一**

```
#代码2.21 字典对象的定义和基本运算
dct1 = {}  #定义一个空字典
#字典元素的键可以是整数、字符串和元组等不可变对象
dct1 = {1:"Python", "Tom":"Male", (3,4):"Red"}
print(f"{dct1[1] = }, {dct1['Tom'] = }, {dct1[(3,4)] = }")
dct1[(3,4)] = "Blue"  #字典的元素是可以被修改的
del dct1[1] #字典的元素可以被删除
dct1[2] = "Java"  #字典中可以随时添加元素
print(f"After change：{dct1 = }")
```

```
#如果一个容器对象中的每个元素都是
#包含2个元素的对象，则可以被转换成字典
dct2 = dict([[1,"Python"],["Tom","Male"],[(3,4),"Red
"]])
print(f"{dct2 = }")
#使用zip()函数可以在两个对象之间
#建立元素的对应关系，从而创建字典
dct3 = dict(zip(["Tom","Jack","Rose"],["Male","Ma
le","Female"]))
print(f"{dct3 = }")
```

字典的常用方法

```
#代码2.22 字典对象的常用方法
dct1 = dict(zip(["Tom","Jack","Rose"],["Male","Male","Female"]))
#keys()方法可以返回字典中的所有键
print(f"{dct1.keys() = }")
#values()方法可以返回字典中的所有值
print(f"{dct1.values() = }")
#items()方法可以返回字典中的所有键-值对
print(f"{dct1.items() = }")
#get()方法用来返回以参数1作为键的元素值，如果找不到则返回参数2的内容
print(f"{dct1.get('Tom','Unknown') = }")
print(f"{dct1.get('Jerry','Unknown') = }")
#copy()方法通过拷贝原字典数据创建一个新的字典对象
dct2 = dct1.copy()
print(f"{dct2 = }")
```

```
#popitem()方法用于弹出最后被加入字典的元素
print(f"{dct2.popitem() = }")
#pop()方法用于弹出字典中的指定元素，并返回其值
print(f"{dct2.pop('Tom') = }")
print(f"After pop: {dct2 = }")
#update() 作用是使用参数对象的内容对字典进行更新
dct2.update({'Jerry':'Male'})
print(f"After update: {dct2 = }")
#clear()方法可以将字典中的元素清空
print(f"{dct2.clear() = }")
#fromkeys()用参数1作为键，参数2作为值，创建字典
dct3 = dict.fromkeys(["Alice","Mary"],"Female")
print(f"{dct3 = }")
#setdefault()用于以参数作为键-值对插入元素
print(f"{dct3.setdefault('Alice','Male') = }")
print(f"{dct3.setdefault('John','Male') = }")
print(f"After setdefault: {dct3 = }")
```

组合数据类型——集合

- 集合是一种非序列容器对象，其中的每个元素都是唯一的，可变集合（set）表示该集合创建后依然可以对其中元素进行增减或修改，不可变集合（frozenset）表示该集合一旦创建，元素便无法修改

```
#代码2.24 集合对象的定义和基本运算
set1 = {1,2,3,4,5} #定义可变集合对象
set2 = {3,4,5,6,7}
print(f"{3 in set1 = }") #判断某个元素是否在集合中
print(f"{set1|set2 = }") #求两个集合的并集
print(f"{set1&set2 = }") #求两个集合的交集
print(f"{set1-set2 = }") #求两个集合的差集
print(f"{set2-set1 = }")
set2 = set1
print(f"{set1 = }, {set2 = }")
```

```
#判断集合set1是否为集合set2的真子集
print(f"{set1<set2 = }")
#判断集合set1是否为集合set2的子集
print(f"{set1<=set2 = }")
set3 = frozenset({1,2,3}) #创建不可变集合
print(f"{set3 = }")
#不可变集合是无法对元素进行修改的，所以程序报错
set3.add(4)
```

集合的常用方法


```
#代码2.25 集合对象的常用方法
set1 = {1,2,3,4,5}  #定义可变集合对象
set1.add(6) #向可变集合中添加元素
print(f"After add: {set1 = }")
#从可变集合中删除元素，如果该元素不存在就报错
set1.remove(6)
#从可变集合中删除元素
#如果该元素不存在，什么也不做
set1.discard(6)
print(f"After remove: {set1 = }")
set2 = {3,4,5,6,7}
print(f"{set1 = }, {set2 = }")
```

```
#求两个集合的交集
print(f"{set1.intersection(set2) = }")
#求两个集合的差集
print(f"{set1.difference(set2) = }")
print(f"Before update: {set1 = }, {set2 = }")
#用参数中的对象更新可变集合
set1.update(set2)
print(f"After update: {set1 = }")
#判断set2是否是set1的子集
print(f"{set2.issubset(set1) = }")
#判断set1是否是set2的父集
print(f"{set1.issuperset(set2) = }")
```

Python函数定义和调用

- Python函数与C语言函数基本相似，定义的关键字是**def**，
- Python函数的形式参数可以设置默认值

```
#代码2.26 从键盘上输入一个正整数，判断其是否为素数
def isPrime(n):
    for i in range(2,n):
        if n%i==0:
            return False
    else:
        return True
num = int(input("请输入一个正整数："))
print(f"{num}是素数" if isPrime(num) else f"{num}不是素数")
```

```
#代码2.27 参数的默认值
def fun(a=10, b=20, c=30):
    return f"{a} + {b} + {c} = {a+b+c}"
print(fun(15, 25, 35)) #没有使用参数默认值
print(fun(15, 20)) #参数c使用了默认值30
print(fun(b=15)) #参数a和参数c使用了默认值
print(fun()) #所有参数都采用默认值
```

匿名函数与lambda关键字

- **lambda**表达式的功能是创建一个非常短小的函数应用，将参数和返回值的关系进行约定

```
#代码2.28 匿名函数lambda表达式的应用
ages = {"Tom":20, "Jack":19, "Rose":18, "Mary":21}
#默认的排序方式是按照键的从小到大顺序
print(f"{sorted(ages) = }")
#在sorted函数中，可以通过指定key参数对应的函数来改变排序的依据
#以lambda表达式的返回值，即年龄进行排序
print(f"{sorted(ages, key=lambda x:ages[x]) = }")
```

Python异常处理

- 程序在执行过程中会遇到未知的错误，Python语言通过异常处理机制在运行程序发生错误时

```
#代码2.29 异常处理机制的使用
lst1 = [2,0]
for i in range(3):
    print(f"循环的第{i+1}次运行：")
    try:
        print(f"{36/lst1[i] = }")
    except Exception as e:
        print("程序产生了异常：", e)
    else:
        print("程序没有产生异常")
    finally:
        print("无论有没有产生异常，这里的代码都会被执行")
```

Python文件处理

- 与C语言相同，Python支持以文本文件或者二进制文件操作，一般过程包括：
  - 使用open()函数打开文件对象
    - 文件路径
    - 打开方式：w,r,a,w+,r+,a+,wb,rb,ab,wb+,rb+,ab+
  - 读/写文件
    - write()、writelines()
    - read()、readline()、readlines()
  - 使用close()函数关闭文件对象
    - 若使用关键字with包含文件操作的程序，则关闭文件的代码可以省略。

Python读写文件

```
#代码2.30 将数据写入文件
with open("data.txt", "w") as file:
    #写入单个字符串
    file.write("人生苦短，我用Python。\n")
    poem = [
        "静夜思 (唐 李白)\n",
        "床前明月光，\n",
        "疑是地上霜。\n",
        "举头望明月，\n",
        "低头思故乡。\n"
    ]
    #写入容器对象中的字符串
    file.writelines(poem)
```

```
#代码2.31 从文件中读取数据
with open("data.txt", "r") as file:
    #从文件中读取所有内容
    str1 = file.read()
    print("str1 =", repr(str1))
    #将读取位置恢复到文件的首部
    file.seek(0)
    #从文件中读取一行内容
    str2 = file.readline()
    print("str2 =", repr(str2))
    file.seek(0)
    #从文件中读取所有内容到列表中
    str3 = file.readlines()
    print("str3 =", repr(str3))
```

Python面向对象基础

- 面向对象程序设计中，先定义类再使用该类实例
- 关键字：**class**

```
#代码shopTest.py 使用自定义的类创建对象，
import shop
shopName = 'the Berkeley Bowl'
fruitPrices = {'apples': 1.00, 'oranges': 1.50, 'pears': 1.75}
berkeleyShop = shop.FruitShop(shopName, fruitPrices)
applePrice = berkeleyShop.getCostPerPound('apples')
print(applePrice)
print(f'Apples cost ${ applePrice : .2f} at { shopName }.')
otherName = 'the Stanford Mall'
otherFruitPrices = {'kiwis':6.00, 'apples': 4.50, 'peaches': 8.75}
otherFruitShop = shop.FruitShop(otherName, otherFruitPrices)
otherPrice = otherFruitShop.getCostPerPound('apples')
print(otherPrice)
print(f'Apples cost ${ otherPrice : .2f} at { otherName }.')
print("My, that's expensive!")
```

```
#代码shop.py 定义一个类，用来描述水果商店
class FruitShop:
    def __init__(self, name, fruitPrices):
        #name: 水果商店的名称
        #fruitPrices: 水果店中每种水果的单价，用字典结构保存，
        #例如：{'apples':2.00, 'oranges': 1.50, 'pears': 1.75}
        self.fruitPrices = fruitPrices
        self.name = name
        print(f'Welcome to {name} fruit shop')
    def getCostPerPound(self, fruit):
        #fruit: 水果的名称
        #return: 返回水果店中对应水果的单价，
        #若找不到则返回None
        if fruit not in self.fruitPrices:
            return None
        return self.fruitPrices[fruit]
    def getPriceOfOrder(self, orderList):
        # orderList: 以元组(fruit, numPounds)为元素的列表
        # return: 返回该订单的总价
        totalCost = 0.0
        for fruit, numPounds in orderList:
            costPerPound = self.getCostPerPound(fruit)
            if costPerPound != None:
                totalCost += numPounds * costPerPound
        return totalCost
    def getName(self):
        return self.name
```

Python类的继承

- 面向对象程序设计的一个非常强的优势是在代码复用方面，通过扩展或修改一个已经定义好的类来建立新的类，不需要将已有类的功能再次实现，这种机制成为继承（Inheritance）

```
#代码2.32 类的继承实例
class Dog:
    def __init__(self,name):
        self.name = name
    def greet(self):
        print(f"I am {self.name}.")
class BarkingDog(Dog):
    def bark(self):
        print("汪汪汪~~")
dog = BarkingDog("旺财")
dog.greet()
dog.bark()
```

数值计算库NumPy

- NumPy是一个用于数值计算的第三方模块
  - 创建数组或矩阵
  - 对数组或矩阵对象进行函数运算
  - 完成数值积分或线性代数的运算
  - ……
- 安装命令：**pip install numpy**

NumPy多维数组创建

- 通过NumPy模块中的array()函数可以依据列表、元组、或其他序列类型对象中的数据，创建数组（ndarray）对象，ndarray对象和列表不一样，其中元素的**数据类型必须保持一致**

```
#代码2.33 使用NumPy中的array()函数创建数组
import numpy as np
#创建1维数组
arr1 = np.array([1,2,3])
print(f"{arr1 = }")
#创建2维数组
arr2 = np.array([[1,2],[3,4],[5,6]])
print(f"{arr2 = }")
#查看数组的元素类型
print(f"{arr2.dtype = }")
#以指定的数据类型创建数组
arr3 = np.array([1,2,3], dtype='float64')
print(f"{arr3 = }")
#数据类型float64指的是64位的双精度浮点数
print(f"{arr3.dtype = }")
```

```
#代码2.34 创建特征数组对象
import numpy as np
#使用arange()函数创建连续数值构成的数组，
#三个参数分别表示起始值、终止值（不包含）、步进值
print(f"{np.arange(10, 20, 2) = }")
print(f"{np.arange(0.1, 1, 0.2) = }")
#使用linespace()函数创建等差数列数组，
#三个参数分别表示起始值、终止值、元素数量
print(f"{np.linspace(0, 1, 5) = }")
#使用logspace()函数创建等比数列数组，
#前两个参数分别是以10的幂表示的起始值和终止值
print(f"{np.logspace(0, 1, 5) = }")
print(f"{np.zeros([3, 2]) = }")#使用zeros()函数创建全零数组
print(f"{np.ones([2, 3]) = }") #使用ones()函数创建全1数组
print(f"{np.identity(3) = }") #使用identity()函数创建单位数组
print(f"{np.diag([1, 2, 3, 4]) = }") #使用diag()函数创建对角线数组
```

NumPy数组常用属性

- NumPy数组对象具有一系列有用的属性

```
#代码2.35 ndarray对象的属性
import numpy as np
arr1 = np.array([[1,2,3],[4,5,6]])
print(f"{arr1.dtype = }")  #dtype属性表示元素的类型
print(f"{arr1.ndim = }")  #ndim属性表示数组的维度
print(f"{arr1.shape = }")  #shape属性表示数组每一维的大小
print(f"{arr1.size = }")  #size属性表示数组元素的数量
arr1.shape = 3,2  #通过设置shape属性，可以该表数组元素的排列
print(f"After shape: {arr1.shape = }")
#astype()方法可以按照指定的类型得到新数组
arr2 = arr1.astype(np.float64)
print(f"After astype: {arr1.dtype = }, {arr2.dtype = }")
```

NumPy生成随机数

- NumPy中包含了random模块，可生成随机数

```
#代码2.36 random模块中的函数举例
from numpy import random, arange
arr1 = arange(15)
random.shuffle(arr1)   #将有序的数组随机打乱
print(f"After shuffle: {arr1 = }")
#choice()函数用于在样本中随机进行抽取
print(f"{random.choice(arr1, size=(2,3)) = }")
#randint()函数用于在指定的范围内构成随机整数数组
print(f"{random.randint(10, 20, size=(3,3)) = }")
#permutation()函数用于直接生成一个乱序数组
print(f"{random.permutation(10) = }")
#normal()函数用于产生正态分布的随机数，
#第一个参数是期望值，第二个参数是标准差
print(f"{random.normal(100, 10, size=(2,3)) = }")
#uniform()函数用于产生均匀分布的随机数，前两个参数分别是区间的起始值和终止值
print(f"{random.uniform(10, 20, size=(2,3)) = }")
```

```
#poisson()函数用于产生泊松分布的随机数，
#第一个参数用于指定λ系数
print(f"{random.poisson(2.0, size=(2,3)) = }")
#beta()函数用于产生beta分布的随机数
print(f"{random.beta(1, 3, size=(2,3)) = }")
#chisquare()函数用于产生卡方分布的随机数，
#第一个参数表示自由度数
print(f"{random.chisquare(3 ,size=(2,3)) = }")
#gamma()函数用于产生gamma分布的随机数
print(f"{random.gamma(2, 2 ,size=(2,3)) = }")
```

NumPy数组变换

- 通过reshape()方法改变其数组维度，
- 数据散开ravel()方法
- 数据扁平化flatten()方法

```
#代码2.37 改变数组维度和数据散开示例
import numpy as np
arr1 = np.arange(12)
print(f"{arr1 = }")
arr2 = arr1.reshape(4,3)
print(f"{arr2 = }")
#参数-1表示该维度的元素个数通过元素总数进行推算
arr3 = arr1.reshape(2,-1)
print(f"{arr3 = }")
#arr4是arr3进行数据散开后的结果
arr4 = arr3.ravel()
print(f"{arr4 = }")
```

NumPy数组合并

- hstack()、vstack()和concatenate函数数组合并
- hsplit()、vsplit()和split()函数分别实现数组分隔
- 使用transpose()方法和T属性实现矩阵转置

```
#代码2.38 数组合并示例
import numpy as np
arr1 = np.arange(4).reshape(2,2)
arr2 = np.arange(4,8).reshape(2,2)
print(f"{arr1 = }")
print(f"{arr2 = }")
print(f"{np.hstack([arr1,arr2]) = }")
print(f"{np.vstack([arr1,arr2]) = }")
#concatenate()函数的参数axis为1时表示横向合并，
#为0时表示纵向合并
print(f"{np.concatenate([arr1,arr2], axis=1) = }")
print(f"{np.concatenate([arr1,arr2], axis=0) = }")
```

```
#代码2.39 数组分割示例
import numpy as np
from numpy.core.fromnumeric import reshape
arr1 = np.arange(16).reshape(4,4)
print(f"{arr1 = }")
print(f"{np.hsplit(arr1,2) = }")
print(f"{np.vsplit(arr1,2) = }")
#split()函数的参数axis为1时表示横向分割，
#为0时表示纵向分割
print(f"{np.split(arr1,2,axis=1) = }")
print(f"{np.split(arr1,2,axis=0) = }"
```

NumPy数组的索引和切片

- 一维数组的索引和切片方法与序列类型对象一样
- 多维数组各个维度的索引用逗号分隔即可

```
#代码2.41 一维数组的索引和切片示例
import numpy as np
arr1 = np.arange(10)
print(f"{arr1 = }")
print(f"{arr1[5] = }")
print(f"{arr1[-2] = }")
print(f"{arr1[2:7] = }")
```

```
#代码2.42 多维数组的索引和切片示例
import numpy as np
arr1 = np.arange(16).reshape(4,4)
print(f"{arr1 = }")
#选取行索引值为1，列索引值为2的元素
print(f"{arr1[1,2] = }")
#选取行、列索引值均在[1,3)区间内的元素
print(f"{arr1[1:3,1:3] = }")
print(f"{arr1[2,:] = }") #选取索引值为2的那一行的所有元素
print(f"{arr1[:,2] = }") #选取索引值为2的那一列的所有元素
print(f"{arr1[arr1>5] = }") #选取元素值大于5的元素
```

NumPy数组的运算

- 数组和标量间的运算，广播机制，条件逻辑运算

```
#代码2.43 数组与标量间的运算示例
import numpy as np
#使用循环结构完成10000次乘法运算
arr1 = np.arange(10000)
for i in range(len(arr1)):
    arr1[i] = arr1[i] * 5
print(f"计算完毕后数组的前5项为：{arr1[:5] = }")
#使用数组结构完成10000次与标量的乘法运算，
#其运行效率远远优于上述代码
arr1 = np.arange(10000)
arr1 = arr1 * 5
print(f"标量计算完毕数组前5项为：{arr1[:5] = }")
```

```
#代码2.44 数组运算中的广播机制示例
import numpy as np
arr1 = np.array([[0,0,0],[1,1,1],[2,2,2]])
arr2 = np.array([1,2,3])
print(f"{arr1 = }")
print(f"{arr2 = }")
print(f"{arr1 + arr2 = }")
```

```
#代码2.45 NumPy模块中where函数使用示例
import numpy as np
arr1 = np.array([[2,4],[6,7]])
arr2 = np.array([[1,5],[3,8]])
#数组arr3中的元素由arr1和arr2中对应位置较大的元素构成
arr3 = np.where(arr1>arr2, arr1, arr2)
print(f"{arr1 = }")
print(f"{arr2 = }")
print(f"{arr3 = }")
```

NumPy数组的读写操作

- 读/写文本文件
- 读/写二进制文件

```
#代码2.47 NumPy模块中的文本文件读写示例
import numpy as np
arr1 = np.arange(12).reshape(4,3)
#将数组arr1中的数据写入文本文件arr.txt，
#数据之间默认以空格分隔，此处指定以逗号分隔数据
np.savetxt("arr.txt", arr1, delimiter=',')
#将文本文件arr.txt中的数据读取出来并与变量arr2关联，
#并指定以逗号作为数据之间的分隔符
arr2 = np.loadtxt("arr.txt", delimiter=',')
print(f"{arr2 = }")
```

```
#代码2.46 NumPy模块中的二进制文件读写示例
import numpy as np
arr1 = np.arange(12).reshape(3,4)
#将数组arr1保存至arr.npy文件中
np.save("arr.npy", arr1)
print(f"{arr1 = }")
#将arr.npy中的数据读取出来
arr2 = np.load('arr.npy')
print(f"{arr2 = }")
```

NumPy数组排序

- 使用NumPy中数组对象自带的sort()方法与argsort()方法可以轻松地完成数组的排序运算

```
#代码2.48 使用sort()方法对数组进行排序示例
import numpy as np
arr1 = np.random.randint(10,20,size=10)
print(f"Before sort: {arr1 = }")
arr1.sort()
print(f"After sort: {arr1 = }")
#数组的sort()方法可以使用指定维度上的元素进行排序
arr2 = np.random.randint(10,20,size=(4,4))
print(f"Before sort: {arr2 = }")
arr2.sort(axis=0)
print(f"After sort by axis=0: {arr2 = }")
arr2.sort(axis=1)
print(f"After sort by axis=1: {arr2 = }")
```

```
#代码2.49 使用argsort()方法对数组排序示例
import numpy as np
arr1 = np.random.randint(10,20,size=10)
print(f"{arr1 = }")
arr2 = arr1.argsort()
print(f"{arr2 = }")
print(f"arr1中最小元素的索引值为：{arr2[0]},arr1中最大元素的索引值为：{arr2[-1]}")
```

数据去重和重复数据

- NumPy中unique()函数用于剔除数组的重复元素
- NumPy中tile()函数或repeat()函数完成复制运算

```
#代码2.50 使用unique函数对数组进行去重示例
import numpy as np
arr1 = np.random.randint(10,20,size=10)
print(f"{arr1 = }")
print(f"{np.unique(arr1) = }")
```

```
#代码2.51  使用tile()函数或者repeat()函数对数组进行重复示例
import numpy as np
arr1 = np.arange(6).reshape(2,3)
print(f"{arr1 = }")
print(f"{np.tile(arr1,3) = }")
#repeat()函数的使用与tile()函数类似，
#其参数axis用于指定重复的维度
print(f"{np.repeat(arr1,2,axis=0) = }")
print(f"{np.repeat(arr1,2,axis=1) = }")
```

Numpy常用统计函数

- NumPy中提供了很多统计分析函数，常见的有sum()、mean()、std()、var()、min()和max()函数等，需要注意在参数中指定其运算的维度

```
#代码2.52 NumPy中的常用统计函数示例
import numpy as np
arr1 = np.arange(10).reshape(2,5)
print(f"{arr1 = }")
#求和运算
print(f"{np.sum(arr1) = }")
print(f"{np.sum(arr1,axis=0) = }")
print(f"{np.sum(arr1,axis=1) = }")
#计算平均值的运算
print(f"{np.mean(arr1) = }")
print(f"{np.mean(arr1,axis=0) = }")
print(f"{np.mean(arr1,axis=1) = }")
#计算标准差的预算
print(f"{np.std(arr1) = }")
print(f"{np.std(arr1,axis=0) = }")
print(f"{np.std(arr1,axis=1) = }")
```

本章小结

- Python语言的基础编程知识
- Python的基本语法
- 各种类型对象的使用方法
- 程序的流程控制结构
- 定义和调用函数的方法
- 以及使用第三方科学计算模块NumPy进行数组运算和数据统计的基本方法。

输入理想的程序

输出快乐的人生