教案（分教案）

课次：2 学时：3

章 节	第 2 章 数据类型与运算符 2.1 变量与基本数据类型 2.2 运算符
教学目的和教学要求	了解 Python 的保留字，切记保留字不能用作变量； 掌握基本数据类型，具体包括整型、浮点型、字符类型和布尔类型，并熟练操作不同数据类型之间的转换。 掌握算术运算符、赋值运算符、比较运算符、逻辑运算符、位运算符、成员运算符和身份运算符，并理解不同运算符之间的优先级。合理运用运算符，实现程序的内容运算。
教学重点难点	**教学重点：** 保留字、变量、基本数据类型、数据类型转换，算术运算符、赋值运算符、比较运算符、逻辑运算符、位运算符、成员运算符、身份运算符 **教学难点：**基本数据类型、数据类型转换，不同运算符的合理运用
教学方法和辅助手段	**教学方法：**课堂讲解、案例分析，实际操作，提问讨论，反思质疑 **辅助手段：**电脑、投影仪、教科书、教案、网络资料

第 2 章 数据类型与运算符

所有编程语言的第一个功能肯定是定义变量，因为总是要处理数据。变量是编程的起始点，程序用到的各种数据都是存储在变量内的。Python 是一门弱类型语言，弱类型包含两方面的含义：

①所有的变量无须声明即可使用，或者说对从未用过的变量赋值就是声明了变量；

②变量的数据类型可以随时改变，同一个变量可以一会儿是数值型，一会儿是字符串型。

形象地看，变量就像一个个小容器，用于“盛装”程序中的数据。常量同样也用于“盛装”程序中的数据。常量与变量的区别是：常量一旦保存某个数据之后，该数据就不能发生改变；但变量保存的数据则可以多次发生改变，只要程序对变量重新赋值即可。

2.1 变量与基本数据类型

1、Python 保留字符

这些保留字不能用作常数或变数，或任何其他标识符名称。所有 Python 的关键字只包含小写字母。

表 2-1 python 保留字符

and	exec	not	else
assert	finally	or	except
break	for	pass	is
class	from	print	lambda
continue	global	raise	with
def	if	return	yield

del	import	try
elif	in	while

如打印关键字列表：

```
import keyword
print(keyword.kwlist)  # 打印关键字列表
```

打印结果：

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

2、标识符

标识符是电脑语言中允许作为名字的有效字符串集合。其中，有一部分是关键字，构成语言的标识符。这样的标识符是不能做它用的标识符的，否则会引起语法错误（SyntaxError 异常）。

Python 标识符字符串规则和其他大部分用 C 编写的高级语言相似：

- ✧ 第一个字符必须是字母或下划线（_）
- ✧ 剩下的字符可以是字母和数字或下划线
- ✧ 大小写敏感

注意：Python 中以下划线开头的标识符有特殊含义，一般应避免使用相似的标识符。

- ✧ 以单下划线开头的标识符（如_width）表示不能直接访问的类属性。另外，也不能通过“from xxx import *”导入；
- ✧ 以双下划线开头的标识符（如__add）表示类的私有成员；
- ✧ 以双下划线开头和结尾的是 Python 里专用的标识，例如“__init__()”表示构造函数。

3、变量

- ✧ Python 中的变量赋值不需要类型声明，type()函数可以查看变量的数据类型。
- ✧ 变量名必须是一个有效的标识符，且慎用小写字母 l 和大写字母 O。
- ✧ 每个变量在内存中创建，都包括变量的标识，名称和数据这些信息。
- ✧ 每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。
- ✧ 等号（=）用来给变量赋值。
- ✧ 等号（=）运算符左边是一个变量名，等号（=）运算符右边是存储在变量中的值。

```
import numpy as np

age = 18  # 变量 = 整型数据
# sex 单独一个变量名称，而无赋值，则会报错
salary = 6889.0  # 变量 = 浮点数
native_place = "河南省信阳市"
print(type(age), type(salary), type(native_place))  # 结果：<class 'int'> <class 'float'> <class 'str'>

# 如果想要存储九九乘法表的结果值，则虚先定义一个数组，如 ndarray
mul_table = np.zeros((9, 9), dtype=np.int)  # 默认为 float 类型，此处指定存储整型数据
for i in range(1, 10):
    for j in range(1, i + 1):
```

```
        print("%2d * %1d = %2d" % (j, i, i * j), end="  ")
        mul_table[i - 1, j - 1] = i * j
    print()
```

打印输出结果：

```
<class 'int'> <class 'float'> <class 'str'>
 1 * 1 =  1
 1 * 2 =  2   2 * 2 =  4
 1 * 3 =  3   2 * 3 =  6   3 * 3 =  9
 1 * 4 =  4   2 * 4 =  8   3 * 4 = 12   4 * 4 = 16
 1 * 5 =  5   2 * 5 = 10   3 * 5 = 15   4 * 5 = 20   5 * 5 = 25
 1 * 6 =  6   2 * 6 = 12   3 * 6 = 18   4 * 6 = 24   5 * 6 = 30   6 * 6 = 36
 1 * 7 =  7   2 * 7 = 14   3 * 7 = 21   4 * 7 = 28   5 * 7 = 35   6 * 7 = 42   7 * 7 = 49
 1 * 8 =  8   2 * 8 = 16   3 * 8 = 24   4 * 8 = 32   5 * 8 = 40   6 * 8 = 48   7 * 8 = 56   8 * 8 = 64
 1 * 9 =  9   2 * 9 = 18   3 * 9 = 27   4 * 9 = 36   5 * 9 = 45   6 * 9 = 54   7 * 9 = 63   8 * 9 = 72   9 * 9 = 81
```

4、基本数据类型

Python3 中有六个标准的数据类型：

- ✧ Number（数字或数值）
- ✧ String（字符串）
- ✧ List（列表）
- ✧ Tuple（元组）
- ✧ Set（集合）
- ✧ Dictionary（字典）

Python3 的六个标准数据类型中：

- ✧ 不可变数据（3 个）：Number（数字）、String（字符串）、Tuple（元组）；
- ✧ 可变数据（3 个）：List（列表）、Dictionary（字典）、Set（集合）。

（1）数值类型

Python 支持三种不同的数值类型：

- ✧ 整型(int) - 通常被称为是整型或整数，是正或负整数，不带小数点。Python3 整型是没有限制大小的，可以当作 Long 类型使用，所以 Python3 没有 Python2 的 Long 类型。
- ✧ 浮点型(float) - 浮点型由整数部分与小数部分组成，浮点型也可以使用科学计数法表示（2.5e2 = 2.5 ×10^2 = 250）；Python 的数学运算多数情况下使用浮点型。
- ✧ 复数((complex)) - 复数由实数部分和虚数部分构成，可以用 $a+bj$ 或者 complex(a,b)表示， 复数的实部 a 和虚部 b 都是浮点型。

Python 语言中有关负数的概念：

（1）虚数不能单独存在，它们总是和一个值为 0.0 的实数部分一起构成一个复数

（2）复数由实数部分和虚数部分构成

（3）表示虚数的语法：real+imagej

（4）实数部分和虚数部分都是浮点数

（5）虚数部分必须有后缀 j 或 J

复数的内建属性：

（1）复数对象拥有数据属性，分别为该复数的实部和虚部。

（2）复数还拥有 conjugate 方法，调用它可以返回该复数的共轭复数对象。

（3）复数属性：real(复数的实部)、imag(复数的虚部)、conjugate()（返回复数的共轭复数）

有时候，需要对数据内置的类型进行转换，数据类型的转换，只需要将数据类型作为函数名即可。

- ✧ int(x) 将 *x* 转换为一个整数。
- ✧ float(x) 将 *x* 转换到一个浮点数。
- ✧ complex(x) 将 *x* 转换到一个复数，实数部分为 *x*，虚数部分为 0。
- ✧ complex(x, y) 将 *x* 和 *y* 转换到一个复数，实数部分为 *x*，虚数部分为 *y*。*x* 和 *y* 是数字表达式。

```
import math  # 数学运算，包含有常见的数学函数和数学常量
import cmath  # 复数运算

x1 = 1.258  # 定义一个浮点数
print("转换为复数：", complex(x1))
y1 = math.sin(x1)  # 数学运算，不含有复数
y2 = cmath.sin(x1)  # 复数运算，虚部为 0
print("math 模块：", y1, "，cmath 模块：", y2)
x2 = 1.258 + 0.589j  # 定义一个复数
# y3 = math.sin(x2)  # TypeError: can't convert complex to float
y3 = cmath.sin(x2)
print("复数运算：", y3, "，其共轭复数：", y3.conjugate())
```

打印输出结果：

```
转换为复数： (1.258+0j)
math 模块： 0.9514768040014665 ，cmath 模块： (0.9514768040014665+0j)
复数运算： (1.1213473916764998+0.1919104041432118j) ，其共轭复数： (1.1213473916764998-0.1919104041432118j)
```

（2）字符串类型

字符串就是连续的字符串序列，可以是计算机所能表示的一切字符的集合。在 Python 中，字符串属于不可变序列，即字符串一旦创建，不可修改，一旦修改或者拼接，都会重新生成新字符串。通常使用单引号、双引号或三引号括起来，三者在语义上没有差别，其中三引号内字符序列可以分布在连续的多行上，而单引号和双引号内字符序列必须在一行。

注：字符串处理函数很多，将单独列出一章进行讲解，这里只谈字符串类型。

Python 中的字符串支持转义字符。所谓转义字符是指使用反斜杠“\”对一些特殊字符进行转移。主要包括：

- ✧ \：续行符
- ✧ \n：换行符
- ✧ \0：空
- ✧ \a：响铃
- ✧ \b：退格
- ✧ \t：水平制表符，用于横向跳到下一制表位
- ✧ \’’：双引号
- ✧ \’：单引号
- ✧ \\：一个反斜杠
- ✧ \f：换页

- ✧ \0dd：八进制数，dd 代表的字符，如\012 代表换行
- ✧ \xhh：十六进制数，hh 代表的是字符，如\x0a 代表换行

在字符串界定符前面加上字母 r（或 R），字符串原样输出，不进行转义。

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:string_demo.py
@time:2021-09-06 14:51
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""



def string_type():
    """
    字符串类型示例
    :return:
    """
    title = '我喜欢的名言警句'  # 使用单引号，字符串内容必须在一行
    # 使用双引号，字符串内容必须在一行，不在一行则自动添加\
    mot_cn = "命运给予我们的不是失望之酒，而是机会之杯。"
    # 使用三引号，字符串内容可以分布在多行，如果多行，则输出也是多行
    mot_en = '''Our destiny offers not the cup of despair,
    but the chance of opportunity.'''
    print(title)
    print(mot_cn)
    print(mot_en)
    print("失望之\"酒\"\t 机会之杯")
    print(r"失望之\"酒\"\t 机会之杯")
    print('''
                          ▶ 学编程，你不是一个人在战斗~~
                          |
                      __\--__|_
    II=======00000[/ ★101___|
              _____\_____|/-----.
            /___mingrisoft.com___|
             \◎◎◎◎◎◎◎◎◎/
               ~~~~~~~~~~~~~~~~

    ''')

if __name__ == '__main__':
    string_type()
```

执行结果截图：

```
我喜欢的名言警句
命运给予我们的不是失望之酒，而是机会之杯。
Our destiny offers not the cup of despair,
    but the chance of opportunity.
失望之"酒"  机会之杯
失望之\"酒\"\t机会之杯
```

（3）布尔类型

python 中布尔值使用常量 **True 和 False 来表示；注意大小写**；比较运算符< > == 等返回的类型就是 bool 类型；布尔类型通常在 if 和 while 语句中应用。数据分析中常使用布尔索引。

需要注意的是，python 中 bool 是 int 的子类（继承 int），故 True==1 False==0 是会返回 Ture，如要切实判断用：xxx is True

True or False 判定，以下会被判定为 False：

- None
- False
- zero of any numeric type, for example, 0, 0.0, 0j.
- any empty sequence, for example, '', (), [].
- any empty mapping, for example, {}.
- instances of user-defined classes, if the class defines a __bool__() or __len__() method, when that method returns the integer zero or bool value False.

除了以上的，其他的表达式均会被判定为 True，这个需要注意，与其他的语言有比较大的不同。

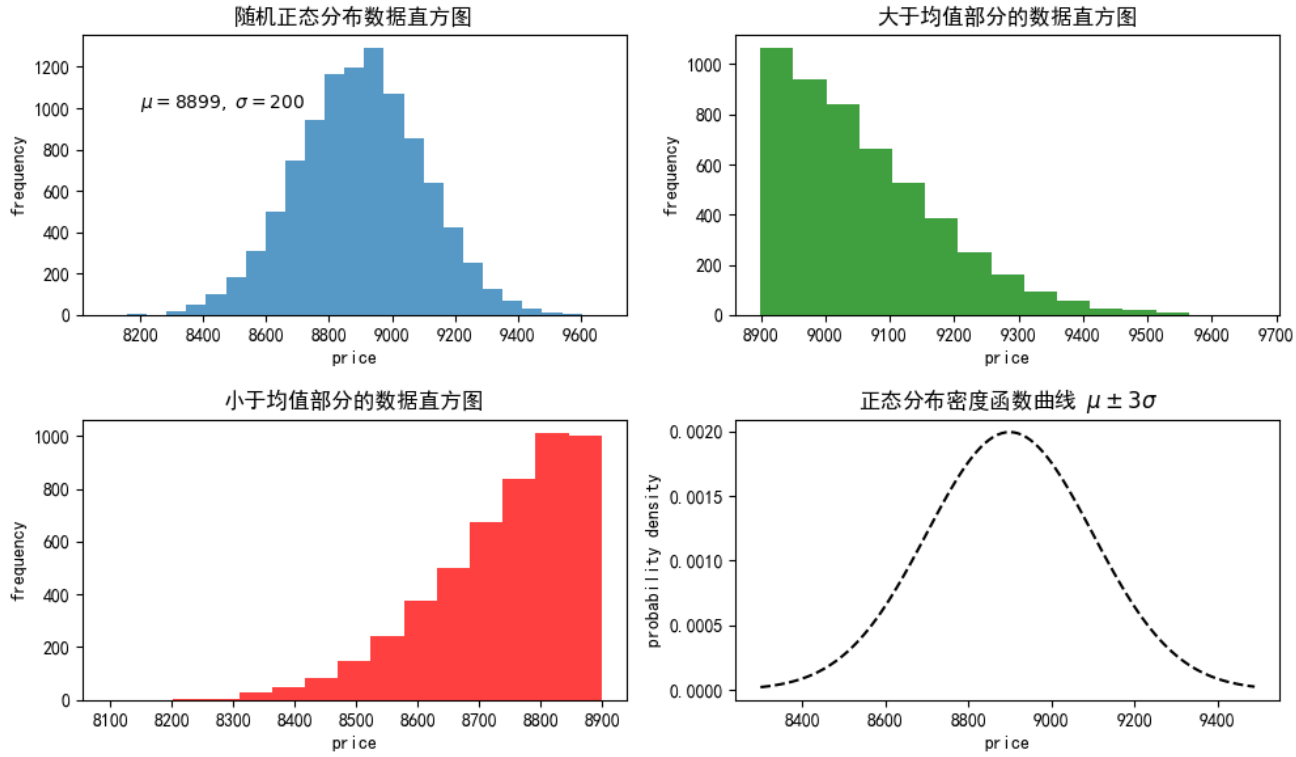
案例：布尔索引

随机生成正态分布随机数，然后绘制图像，其中大于均值部分和小于均值部分数据分别绘制，并计算概率密度函数值，其密度函数 $f(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$，绘制图像。

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:bool_type.py
@time:2021-09-06 14:56
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""
import numpy as np
import matplotlib.pyplot as plt
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False


def price_random(mu=0, sigma=1):
    """
    布尔类型示例：布尔索引，功能：
    1. 随机生成正态分布，服从(mu, sigma)
    2、获取大于均值部分的数据，绘制直方图
    3、获取小于均值部分的数据，绘制直方图
    4、绘制随机数据的正态分布概率密度曲线
    :param mu: 正态分布均值
    :param sigma: 正态分布标准方差
    :return:
    """
    # loc：float 类型，表示此正态分布的均值（对应整个分布中心）
    # scale：float 类型，表示此正态分布的标准差（scale 越大数据越分散；scale 越小数据越集中）
    # size：输出的 shape，size=(k,m,n) 表示输出 k 维，m 行，n 列的数，默认为 None，只输出一个值，size=100，表示输出 100 个值
    price_data = np.random.normal(loc=mu, scale=sigma, size=(10000, 1))
    price_gt_mu = price_data[price_data >= mu]  # 大于均值部分的数据
    price_lt_mu = price_data[price_data < mu]  # 大于均值部分的数据
    val = np.arange(mu - 3 * sigma, mu + 3 * sigma, 10)
    # 正态分布概率密度函数计算
    left = 1 / (np.sqrt(2 * np.pi) * sigma)
    right = np.exp(-(val - mu) ** 2 / (2 * sigma ** 2))
    pdf_val = left * right
    return price_data, price_gt_mu, price_lt_mu, val, pdf_val


def ax_label(fig_obj):
    """
    设置子图的坐标轴名称
    :param fig_obj: 当前子图对象
    :return:
    """
    fig_obj.xlabel("price")
    fig_obj.ylabel("frequency")


def plt_price_hist(data, gt_data, lt_data, x, pdf):
    """
    绘制房价价格直方图
    :return:
    """
    plt.figure(figsize=(10, 6))
    plt.subplot(221)  # fig.add_subplot(2, 2, 1)
```

```
    plt.hist(data, 25, alpha=0.75)
    plt.title("随机正态分布数据直方图")
    ax_label(plt)
    plt.text(8200, 1000, r'$\mu=8899,\ \sigma=200$')  # 为图像添加标注文本
    plt.subplot(222)  # fig.add_subplot(2, 2, 2)
    plt.hist(gt_data, 15, color="green", alpha=0.75)
    plt.title("大于均值部分的数据直方图")
    ax_label(plt)
    plt.subplot(223)  # fig.add_subplot(2, 2, 3)
    plt.hist(lt_data, 15, color="red", alpha=0.75)
    plt.title("小于均值部分的数据直方图")
    ax_label(plt)
    plt.subplot(224)  # fig.add_subplot(2, 2, 4)
    plt.plot(x, pdf, 'k--')
    plt.title("正态分布密度函数曲线 " + "$\mu\pm3\sigma$")
    plt.xlabel("price")
    plt.ylabel("probability density")
    plt.show()


if __name__ == '__main__':
    price_data, price_gt_mu, price_lt_mu, val, pdf_val = price_random(8899, 200)
    plt_price_hist(price_data, price_gt_mu, price_lt_mu, val, pdf_val)
```



（4）数据类型转换

表 2-2 数据类型转换函数格式表

函数格式	使用示例	描述
int(x [,base])	int("8")	可以转换的包括 String 类型和其他数字类型，但是会丢失精度。
float(x)	float(1)或者 float("1")	可以转换 String 和其他数字类型，不足的位数用 0 补齐，例如 1 会变成 1.0
complex(real ,imag)	complex("1") 或者 complex(1,2)	第一个参数可以是 String 或者数字，第二个参数只能为数字类型，第二个参数没有时默认为 0
str(x)	str(1)	将数字转化为 String
repr(x)	repr(Object)	返回一个对象的 String 格式
eval(str)	eval("12+23")	执行一个字符串表达式，返回计算的结果，如例中返回 35
tuple(seq)	tuple((1,2,3,4))	参数可以是元组、列表或者字典，wie 字典时，返回字典的 key 组成的集合
list(s)	list((1,2,3,4))	将序列转变成一个列表，参数可为元组、字典、列表，为字典时，返回字典的 key 组成的集合
set(s)	set(['b', 'r', 'u', 'o', 'n']) 或者 set("asdfg")	将一个可以迭代对象转变为可变集合，并且去重复，返回结果可以用来计算差集 x - y、并集 x | y、交集 x & y
frozenset(s)	frozenset([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])	将一个可迭代对象转变成不可变集合，参数为元组、字典、列表等.
chr(x)	chr(0x30)	chr()用一个范围在 range（256）内的（就是 0～255）整数作参数，返回一个对应的字符。返回值是当前整数对应的 ascii 字符。
ord(x)	ord('a')	返回对应的 ASCII 数值，或者 Unicode 数值
hex(x)	hex(12)	把一个整数转换为十六进制字符串
oct(x)	oct(12)	把一个整数转换为八进制字符串

密码加密登录案例演示：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:datatype_transform.py
@time:2021-09-09 9:34
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""



# 功能：
# 1、对输入的密码进行加密处理
# 2、登录时，输入用户名和密码，并对密码进行解密



def encoding_ps(password):
```

```
    """
    对输入的密码进行加密运算：用到数据类型转换
    :param password: 输入的数字密码
    :return:
    """
    ps8 = oct(password)  # 把整型数字转换为 8 进制
    ps16 = hex(password)  # 把整型数字转换为 16 进制
    ps = ps8 + ps16
    ps = ps[2:]  # 截取字符串，即不取八进制的 0o
    print("密码加密后是：", ps)
    return ps


def login(username, ps_encoding, password):
    """
    模拟简单登录功能：用户输入用户名和密码，对密码和解密后的密码进行判断，相等，则登陆成功！
    :param username: 用户名字符串
    :param ps_encoding: 加密密码
    :param password: 输入的密码
    :return:
    """
    # 1、对已经注册的密码进行解密
    ind = ps_encoding.find("x")  # 在加密字符串数字中查找十六进制字符 x
    ps = ps_encoding[ind + 1:]  # 字符串截取
    ps = int(ps, 16)  # 讲十六进制转换为八进制
    if isinstance(username, str) and ps == password:
        print("登录成功！欢迎：" + username)
        love_nf = 520.1314
        love_nd = 5201314
        love = "我爱你一生一世"
        print(str(love_nf) + ", " + str(love_nd) + ", " + love)
    else:
        raise ValueError("用户名或者密码错误！")


def input_info():
    """
    接受用户输入，并进行简单校验
    """
    username = input("1. 请输入用户名：")
    password = input("2. 请输入一个数字密码：")  # 接受输入的类型始终为 string
    if password.isdigit():  # 判断是否时数字密码
        if 6 <= len(password) <= 15:
            password = int(password)  # 把输入的数字密码（字符串）转换为整型
            ps = encoding_ps(password)  # 调用函数，进行密码加密运算
            login(username, ps, password)  # 调用函数，模拟登录
        else:
```

```
            raise ValueError("密码长度必须介于[6, 15]位！")
    else:
        raise ValueError("您输入的密码非数字密码，请重新输入！")


if __name__ == '__main__':
    input_info()
```

运行结果：

请输入用户名：zhouxingchi
请输入一个数字密码：987456654
密码加密后是：72666622160x3adb648e
登陆成功！欢迎：zhouxingchi
520.1314，5211314，我爱你一生一世

2.2 运算符

1、算术运算符

表 2-3 算术运算符表

运算符	描述	实例
+	加——两个对象相加	a + b 输出结果 30
-	减——得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘——两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除——x 除以 y	b / a 输出结果 2
%	取模——返回除法的余数	b % a 输出结果 0
**	幂——返回 x 的 y 次幂	a**b 为 10 的 20 次方， 输出结果 100000000000000000000
//	取整除——返回商的整数部分（**向下取整**）	print(9//2) 4 print(-9//2) -5

数字特征案例分析：

```
# -*- coding: UTF-8 -*-
"""
@file_name:arithmetic_operator.py
@time:2021-09-09 10:13
@copyright: http://maths.xynu.edu.cn
"""
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns


def cal_numerical_characteristics(house_price):
    """
    计算房价的数字特征
    :param house_price: 价格序列数据
    :return:
    """
    price_range = max(house_price) - min(house_price)  # np.ptp(house_price)
    print('北京房价整套出售价格极差（单位：万元）：' + str(price_range))  # 转换为字符串

    price_avg = np.mean(house_price)  # 调用函数计算均值
    price_std = np.std(house_price)  # 调用函数计算标准方差
    low, up = price_avg - 3 * price_std, price_avg + 1 * price_std  # mu±3sigma 外的数据删除
    house_price_ = house_price[low <= house_price]  # 取范围内的数据
    house_price_ = house_price_[house_price <= up]
    print("删除[mu-3 * sigma, mu + sigma]外的数据共%d 条" % (len(house_price) - len(house_price_)))

    price_avg = sum(house_price_) / len(house_price_)  # 均值
    price_std = np.sqrt(sum((house_price_ - price_avg) ** 2) / len(house_price_))  # 标准方差
    print('北京房价整套售价均值：%.1f 万元' % price_avg)
    print('北京房价整套售价标准方差为：%.1f' % price_std)

    price_cv = price_std / price_avg
    print('北京房价整套售价变异系数为：%.5f' % price_cv)

    price = house_price_ // 1  # 向下取整
    p_mode, count = price_mode(price)
    print("北京房价整套售价众数：%d，频数：%d" % (p_mode, count))

    mdval = price_median(house_price_)
    print("北京房价整套售价的中位数为：%.1f" % mdval)

    # 绘制小提琴图
    sns.violinplot(x=house_price_, color="c")
    plt.title("The violinplot of house price with " + r"$[\mu - 3\sigma, \mu + \sigma]$")
    plt.xlabel("price")
    plt.show()


def price_mode(price):
    """
    计算价格众数
    :param price: 价格序列数据
    :return:
```

```
    """
    mode_dict = dict()  # 字典存储
    for p in price:
        if p not in mode_dict:
            mode_dict[p] = 1  # 新的数，频数为 1
        else:
            mode_dict[p] += 1  # 已存在，则加一
    mode_val = list(mode_dict.values())  # 获取频数，并转换为列表
    idx = int(np.argmax(mode_val))  # 在频数列表中查询最大频数的索引
    price_key = list(mode_dict.keys())  # 获取价格，并转换为列表
    return price_key[idx], mode_val[idx]


def price_median(house_price):
    """
    求价格中位数
    :param house_price: 房价序列数据
    :return:
    """
    price = house_price // 1
    price.sort()  # 进行原列表排序
    size = len(price)  # 数据的个数
    if size % 2 == 0:  # 判断为偶数取中间两位数
        med = (price[size // 2 - 1] + price[size // 2]) / 2  # //为整除，/为浮点数
    else:  # 或为奇数，切片总长度除 2，取中间一位
        med = price[size // 2]
    return med


def read_data():
    """
    读取文件，获取指定数据
    :return:
    """
    house_data = pd.read_csv("../data/house_price.csv")
    price = np.asarray(house_data.loc[:, "Price"], dtype=np.float)
    return price


if __name__ == '__main__':
    house_price = read_data()
    cal_numerical_characteristics(house_price)
```

程序运行结果：

北京房价整套出售价格极差（单位：万元）：5940.0

删除[mu-3 * sigma, mu + sigma]外的数据共 2525 条

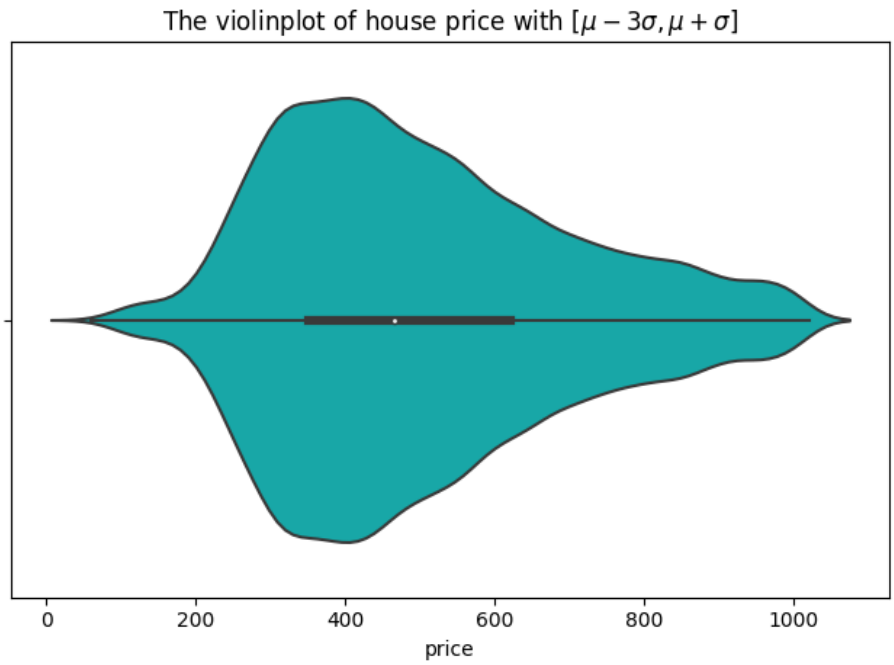
北京房价整套售价均值：501.0 万元

北京房价整套售价标准方差为：197.2

北京房价整套售价变异系数为：0.39360

北京房价整套售价众数：450，频数：319

北京房价整套售价的中位数为：465.0



基本数学运算

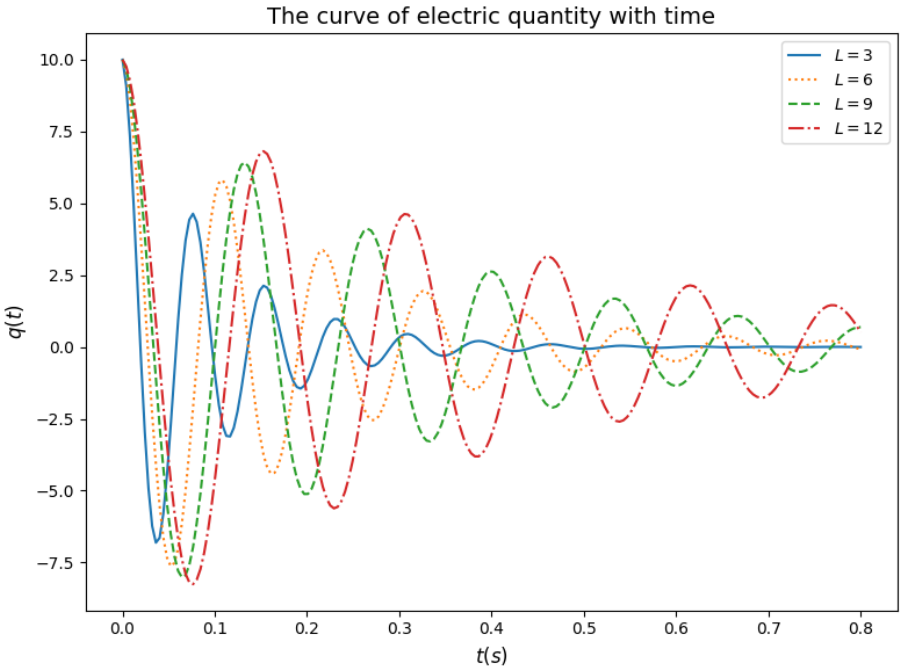
例：电容上的电量$q(t)$为时间的函数，可以根据下式计算：

$$q(t)=q_0e^{-Rt/(2L)}\cos\left[\sqrt{\frac{1}{LC}-\left(\frac{R}{2L}\right)^2}t\right]$$

其中q_0为初始电量，R 为电阻，L 为电感，C 为电容。绘制该函数从$t=0$到$t=0.8$之间的图形。假定$q_0=10$，$R=60$，$C=0.00005$，L 取值范围 3~12，增量为 3.

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



q0, R, C = 10, 60, 0.00005  # 基本参数定义
t = np.linspace(0, 0.8, 200)  # 等分时间
plt.figure(figsize=(8, 6))
line_style = ["-", ":", "--", "-."]
for L, line in zip(np.arange(3, 13, 3), line_style):
    qt = q0 * np.exp(-R * t / 2 / L) * np.cos(np.sqrt(1 / L / C - (R / 2 / L) ** 2) * t)
    plt.plot(t, qt, ls=line, label=r"$L = %d$" % L)

plt.legend()
plt.title('The curve of electric quantity with time', fontdict={"fontsize": 14})
plt.xlabel(r"$t(s)$", fontdict={"fontsize": 12})
plt.ylabel(r"$q(t)$", fontdict={"fontsize": 12})
```

```
plt.show()
```



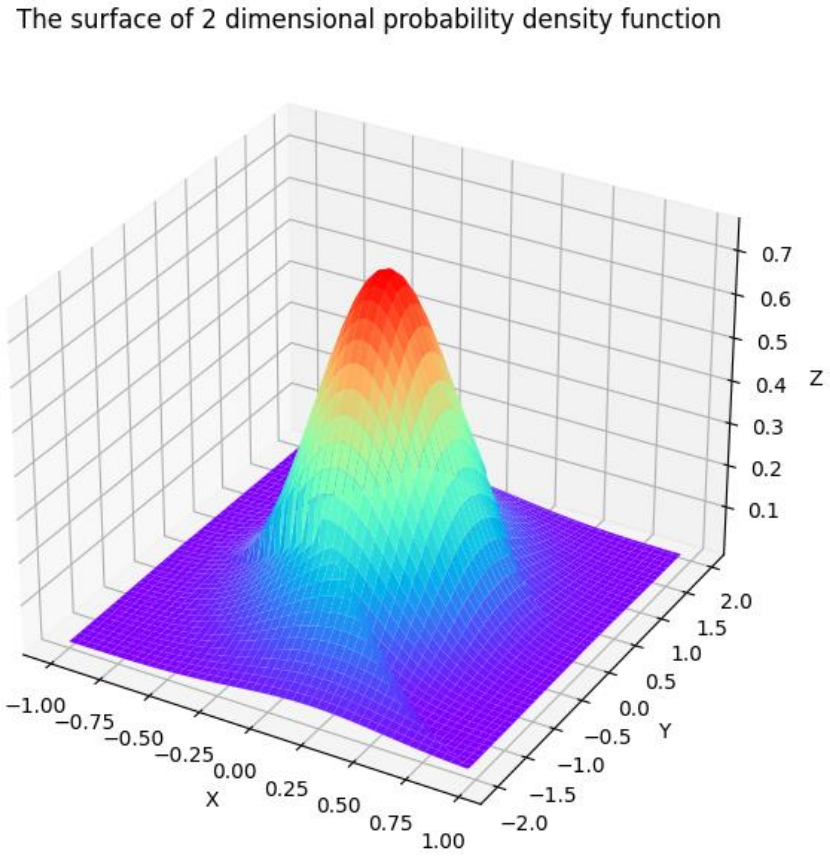
例：假设某概率密度函数由下面分段函数表示

$$p(x_1,x_2)=\begin{cases}0.5457\,e^{-0.75x_2^2-3.75x_1^2-1.5x_1}, & x_1+x_2>1\\ 0.7575\,e^{-x_2^2-6x_1^2}, & -1<x_1+x_2\le 1\\ 0.5457\,e^{-0.75x_2^2-3.75x_1^2+1.5x_1}, & x_1+x_2\le -1\end{cases}$$

代码如下：

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


def prob_density_fun(x, y):
    """
    定义密度函数
    :param x: x 轴坐标数值
    :param y: y 轴坐标数值
    :return:
    """
    zi = 0.5457 * np.exp(-0.75 * y ** 2 - 3.75 * x ** 2 - 1.5 * x) * (x + y > 1) + \
         0.7575 * np.exp(-y ** 2 - 6 * x ** 2) * ((x + y > -1) & (x + y <= 1)) + \
         0.5457 * np.exp(-0.75 * y ** 2 - 3.75 * x ** 2 + 1.5 * x) * (x + y <= -1)
    return zi


def plt_3d():
    """
```

```
    绘制二维概率密度函数曲面
    :return:
    """
    x = np.arange(-1, 1, 0.04)
    y = np.arange(-2, 2, 0.04)
    [xi, yi] = np.meshgrid(x, y)  # 生成网格点
    # 调用函数计算二维密度函数
    Z = prob_density_fun(xi, yi)
    # 可视化
    plt.figure(figsize=(8, 6))
    ax = plt.gca(projection='3d')
    ax.plot_surface(xi, yi, Z, cmap='rainbow')
    ax.set_xlabel('X')
    ax.set_ylabel("Y")
    ax.set_zlabel('Z')
    plt.title("The surface of 2 dimensional probability density function")
    plt.show()


if __name__ == '__main__':
    plt_3d()
```



2、赋值运算符

表 2-4 赋值运算符表

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

3、比较运算符

表 2-5 比较运算符表

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 true.
<>	不等于 - 比较两个对象是否不相等	(a <> b) 返回 true。这个运算符类似 != 。
>	大于 - 返回 x 是否大于 y	(a > b) 返回 False。
<	小于 - 返回 x 是否小于 y。所有比较运算符返回 1 表示真，返回 0 表示假。这分别与特殊的变量 True 和 False 等价。	(a < b) 返回 true。
>=	大于等于 - 返回 x 是否大于等于 y。	(a >= b) 返回 False。
<=	小于等于 - 返回 x 是否小于等于 y。	(a <= b) 返回 true。

折纸游戏

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:paper_folding.py
@time:2021-09-09 11:21
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""



def paper_folding(distance):
```

```
    """
    纸对折计算，假设一张纸的厚度位 0.1mm
    :param distance:
    :return:
    """
    num = 0  # 记录纸对折的次数
    while True:
        thickness = 2 ** num * 0.0001  # 纸对折的厚度，单位米
        num += 1
        if thickness >= distance:
            break
    return num


if __name__ == '__main__':
    print("1、成年男性的平均身高为 1.75 米")
    distance = 1.75
    print("\t 需要对折【%d】次" % paper_folding(distance))
    print("2、珠穆朗玛峰的高度位 8848 米")
    distance = 8848
    print("\t 需要对折【%d】次" % paper_folding(distance))
    print("3、地球到月球之间的平均距离为 38.4 万千米")
    distance = 38.4 * 10 ** 7
    print("\t 需要对折【%d】次" % paper_folding(distance))
    print("4、地球到太阳之间的平均距离为 1.496*10^8 千米")
    distance = 1.496 * 10 ** 11
    print("\t 需要对折【%d】次" % paper_folding(distance))
```

程序运行结果：

1、成年男性的平均身高为 1.75 米
　　需要对折【16】次
2、珠穆朗玛峰的高度位 8848 米
　　需要对折【28】次
3、地球到月球之间的平均距离为 38.4 万千米
　　需要对折【43】次
4、地球到太阳之间的平均距离为 1.496*10^8 千米
　　需要对折【52】次

4、逻辑运算符

表 2-6 逻辑运算符表

运算符	逻辑表达式	描述	实例
and	x and y	布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。	(a and b) 返回 20。
or	x or y	布尔"或" - 如果 x 是非 0，它返回 x 的值，否	(a or b) 返回 10。

		则它返回 y 的计算值。	
not	not x	布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not(a and b) 返回 False

分段线性插值的原理（额外要求：插值节点 x 是等距的）：

设已知点 $a=x_0<x_1<\cdots<x_n=b$ 上的函数值 $f(x_0),f(x_1),\cdots,f(x_n)$，，若有一折线 $y(x)$ 函数满足：

（1）在[a,b]上连续,

（2） $y(x_i)=f(x_i),i=0,1,2,\cdots,n$

（3）在每个子区间 $[x_i,x_{i+1}]$ 上是线性函数,

则称 $y(x)$ 是 $f(x)$ 的分段线性插值函数。由插值多项式的惟一性，在每个小区间上可表示为：

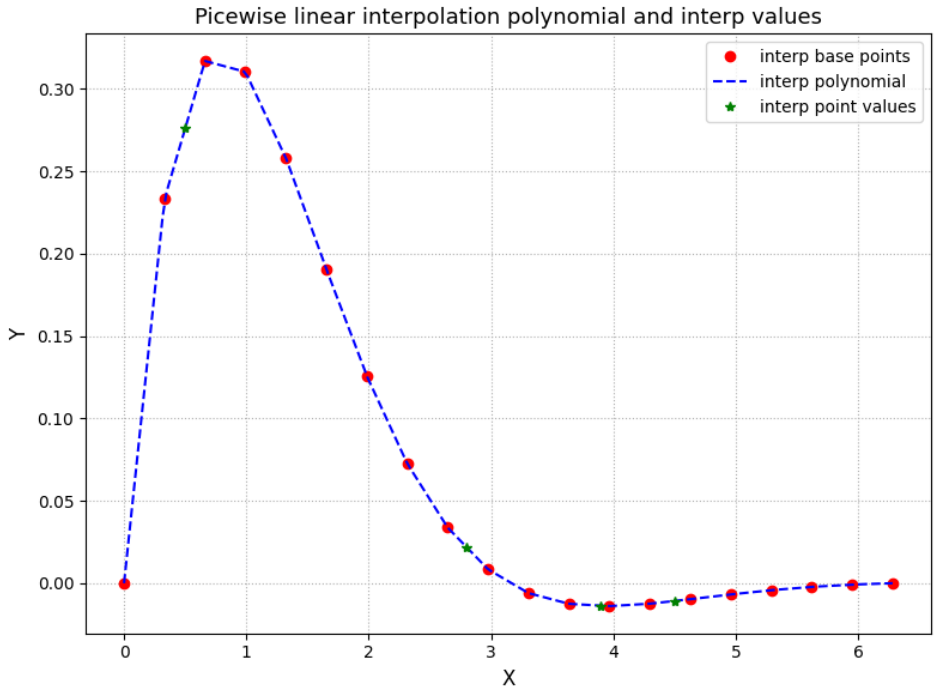
$$y(x)=\frac{x-x_{i+1}}{x_i-x_{i+1}}f(x_i)+\frac{x-x_i}{x_{i+1}-x_i}f(x_{i+1}),\ \ x_i\le x\le x_{i+1}$$

分段线性插值函数 $L_n(x)$ 的余项函数 $R(x)=f(x)-L_n(x)$ 满足 $|R(x)|\le\frac{h^2M}{8}$，其中 $h=\max\limits_{0\le i\le n-1}|x_{i+1}-x_i|$，$M=\max\limits_{a\le x\le b}|f''(x)|$。

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:picewise_linear_interpolation.py
@time:2021-09-09 22:00
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""
import numpy as np
import matplotlib.pyplot as plt


# 已知离散数据点，推测未知的离散点，存在缺失值，需要插补...
# 统计一座新建的桥梁一天的车流量
# 时间：每个一小时，统计一分钟内过桥的车量，如 8 点—8 点 01 分，车通过桥梁 152 辆
# t = np.array([6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21])
# y = np.array([34, 58, 189, 211, 120, 100, 123, 99, 157, 111, ...])

def check_equidistant(x):
    """
    判断离散数据点 x 是否是等距节点
    """
    n = len(x)  # 已知数据点的个数
    if n > 1:
```

```
        xx = np.linspace(min(x), max(x), n, endpoint=True)
        if (x == xx).all() or (x == xx[::-1]).all():  # [True, True, True, ....]
            return x[1] - x[0], n  # 返回等距步长和数据点个数
        else:
            raise ValueError("非等距节点，不适用该插值法！")
    else:
        raise ValueError("插值点的个数少于 2 个！")


def fit_interp(x, y, x0):
    """
    根据已知离散数据节点 x、y，求解分段线性插值函数，并求解 x0 的对应插值点 y0
    """
    h, n = check_equidistant(x)  # 判断是否等距
    if x0 is not None:
        x0 = np.asarray(x0, dtype=np.float)
        n0 = len(x0)  # 所求插值点的个数
        y0 = np.zeros(n0)  # 用于存储 x0 所对应的插值点
        # 查找 x0 所在的子区间索引号
        for i in range(n0):
            idx = np.infty  # 区间索引号初始化为无穷
            for j in range(n - 1):
                if x[j] <= x0[i] <= x[j + 1] or x[j] >= x0[i] >= x[j + 1]:
                    idx = j  # idx 标记 x0 所在子区间索引号
                    break
            if idx is np.infty:
                raise ValueError("分段线性插值不能进行外插值！")
            # 分段线性插值函数计算
            y0[i] = ((x0[i] - x[idx]) * y[idx + 1] - (x0[i] - x[idx + 1]) * y[idx]) / h
        return y0


def plt_interp(x, y, x0=None, y0=None):
    """
    绘制插值图像
    """
    plt.figure(figsize=(8, 6))
    plt.plot(x, y, "ro", label="interp base points")
    # 模拟细化数值，进行插值曲线的绘制
    xi = np.linspace(min(x), max(x), 200)  # 模拟 200 个值，用于绘图
    yi = fit_interp(x, y, xi)
    plt.plot(xi, yi, "b--", label="interp polynomial")
    if x0 is not None and y0 is not None:
        plt.plot(x0, y0, "g*", label="interp point values")
    plt.legend()
    plt.xlabel("X", fontdict={"fontsize": 12})
    plt.ylabel("Y", fontdict={"fontsize": 12})
```

```
    plt.title("Picewise linear interpolation polynomial and interp values", fontdict={"fontsize": 13})
    plt.grid(ls=":")
    plt.show()


if __name__ == '__main__':
    t = np.array([6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21])
    np.random.seed(0)
    y = np.random.randint(50, 200, len(t))
    t0 = np.array([8.5, 13.4, 18.7])
    y0 = fit_interp(t, y, t0)
    print(y0)
    plt_interp(t, y, t0, y0)

    # x = np.linspace(0, 2 * np.pi, 20)
    # y = np.exp(-x) * np.sin(x)
    # x0 = np.array([0.5, 2.8, 3.9, 4.5])
    # y0 = fit_interp(x, y, x0)
    # print("插值点值：", y0, "精确值：", np.exp(-x0) * np.sin(x0))
    # plt_interp(x, y, x0, y0)
```



打印结果：

插值点值： [0.27614687 0.02192052 -0.01362281 -0.01078866]

精确值： [0.29078629 0.02037065 -0.0139217 -0.01085938]

5、位运算符

表 2-7 位运算符表

运算符	描述	实例

&	按位与运算符：参与运算的两个值，如果两个相应位都为 1，则该位的结果为 1，否则为 0	(a & b) 输出结果 12 ，二进制解释：0000 1100
|	按位或运算符：只要对应的二个二进位有一个为 1 时，结果位就为 1。	(a | b) 输出结果 61 ，二进制解释：0011 1101
^	按位异或运算符：当两对应的二进位相异时，结果为 1	(a ^ b) 输出结果 49 ，二进制解释：0011 0001
~	按位取反运算符：对数据的每个二进制位取反，即把 1 变为 0，把 0 变为 1。	(~a) 输出结果 -61 ，二进制解释：1100 0011，在一个有符号二进制数的补码形式。
<<	左移动运算符：运算数的各二进位全部左移若干位，由 << 右边的数字指定了移动的位数，高位丢弃，低位补 0。	a << 2 输出结果 240 ，二进制解释：1111 0000
>>	右移动运算符:把">>"左边的运算数的各二进位全部右移若干位，>> 右边的数字指定了移动的位数	a >> 2 输出结果 15 ，二进制解释：0000 1111

6、成员运算符

表 2-8 成员运算符表

运算符	描述	实例
in	如果在指定的序列中找到值则返回 True，否则返回 False。	x 在 y 序列中 ，如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值则返回 True，否则返回 False。	x 不在 y 序列中 ，如果 x 不在 y 序列中返回 True。

7、身份运算符

身份运算符 is 用于判断两个变量引用对象是否为同一个(同一块内存空间)，比较运算符的 == 用于判断引用变量的值是否相等。

表 2-9 身份运算符表

运算符	描述	实例
is	is 是判断两个标识符是不是引用自一个对象	**x is y**, 类似 **id(x) == id(y)** , 如果引用的是同一个对象则返回 True，否则返回 False
is not	is not 是判断两个标识符是不是引用自不同对象	**x is not y** ， 类似 **id(a) != id(b)**。如果引用的不是同一个对象则返回结果 True，否则返回 False。

案例分析：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:identity operator.py
@time:2021-09-09 16:36
```

```
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""
string1 = '大江东去，浪淘尽，千古风流人物。'
string2 = '大江东去，浪淘尽，千古风流人物。'
string3 = '北国风光，千里冰封，万里雪飘。'
print("string1: ", id(string1))
print("string2: ", id(string2))
print("string3: ", id(string3))
if string1 is string2:
    print('string1 和 string2 两个标识符是引用自一个对象。')
if string2 is not string3:
    print('string2 和 string3 两个标识符不是引用自一个对象。')
listA = [1, 2, 3, 4, 5, 6]  # 列表为可变对象
listB = [1, 2, 3, 4, 5, 6]
print("列表 A 和列表 B 是否引用同一个对象：", listA is listB)
print("列表 A 和列表 B 的值是否相等：", listA == listB)
listC = (1, 2, 3, 4, 5, 6)  # 元组为不可变对象
listD = (1, 2, 3, 4, 5, 6)
print("元组 C 和元组 D 是否引用同一个对象：", listC is listD)
print("元组 C 和元组 D 的值是否相等：", listC == listD)
```

打印输出结果：

```
string1:  2992096693808
string2:  2992096693808
string3:  2992096693584
string1 和 string2 两个标识符是引用自一个对象。
string2 和 string3 两个标识符不是引用自一个对象。
列表 A 和列表 B 是否引用同一个对象： False
列表 A 和列表 B 的值是否相等： True
元组 C 和元组 D 是否引用同一个对象： True
元组 C 和元组 D 的值是否相等： True
```

8、运算符优先级

表 2-10 运算符优先级表

运算符	描述
**	指数 (最高优先级)
~ + -	按位翻转，一元加号和减号 (最后两个的方法名为 +@ 和 -@)
* / % //	乘，除，取模和取整除
+ -	加法减法
>> <<	右移，左移运算符

&	位 'AND'
^ |	位运算符
<= < > >=	比较运算符
<> == !=	等于运算符
= %= /= //= -= += *= **=	赋值运算符
is / is not	身份运算符
in / not in	成员运算符
not / and / or	逻辑运算符

本章小结：

本章讲解了 Python 语言的数据类型，具体包括数值型、字符串、布尔类型，并以案例的形式讲解各种数据类型的具体含义和操作方法。接着，讲解了常见的操作运算符，如数值运算符，比较运算符、赋值运算符，逻辑运算符、成员运算符、身份运算符，位运算符等。运算符是数据的基本运算操作，是 Python 的基础知识，需要掌握且熟练操作，是后期学习的基础。

作　业	课下完成教案案例内容，并完成实验指导书实验二的内容，书写实验报告。
主要 参考资料	[1] 明日科技,王国辉,李磊,冯春龙. Python 从入门到项目实践[M]. 吉林：吉林大学出版社，2018. [2] 明日科技. Python 项目开发案例集锦[M]. 吉林：吉林大学出版社，2019. [3] 李刚. 疯狂 Pyhton 讲义[M]. 电子工业出版社，2019
课后自我总结分析	
备注	