## 第一章

### Python自述

1. Web全栈开发

掌握：前段开发、数据库管理、后台技术框架

职业：Web权栈开发工程师

1. 数据科学方向

掌握：数据库管理、数据分析、数据可视化、能够制作数据看板、实现数据指标监控

职业：数据产品经理、量化交易、初级BI商业分析师

1. 人工智能方向-机器学习

掌握：机器学习常用算法思想、利用Python建立机器学习模型并对应用场景进行智能化

职业：数据分析工程师、机器学习算法工程师、搜索算法工程师、推荐算法工程师

1. 人工智能方向-深度学习

掌握：深度学习常用框架、掌握GNN网络相关算法 实现图像自动上色 图像美化等

职业：人工智能工程师

### Python简介

1. 跨平台的计算机程序设计语言
2. 解释型语言---无需编译
3. 交互式语言---可以在提示符>>>后直接执行代码
4. 面向对象语言

### 搭建Python开发环境

1. 安装Python解释器

<https://www.python.org/> 官网

<https://www.python.org/downloads/release/python-381/>

### 第三方开发环境Pycharm的下载与安装

Pycharm安装教程：<https://www.run>oob.com/w3cnote/pycharm-windows-install.html

Pycharm下载地址：<http://www.jetbrains.com/pycharm/download/#section=windows>

1. 编辑者：File--Settings--Editor--File and Code Templates--Python Script

### Python中的输出函数print( )

1. print( )函数

①输出内容可以是数字、字符串、含有运算符的表达式

②可以将输出内容输出至显示器、文件

③print函数的输出形式：换行/不换行

### 转义字符与原字符

1. 什么是转义字符

反斜杠+想要实现的转移功能首字母

1. 为什么需要转义字符

①当字符串中包含反斜杠、单引号、双引号等有特殊用途的字符时，必须使用反斜杠对这些字符进行转义：反斜杠 \\，单引号 \’，双引号 \”

②当字符串中包含换行、回车、水平制表符、退格等无法表示的特殊字符时，也可以使用转义字符对这些字符进行转义：换行 \n，回车 \r，水平制表符 \t，退格 \b

## 第二章

### 二进制与字符编码

（1）8bit（位）=1byte（字节）

1024byte=1KB

1024KB=1MB

1024MB=1GB

1024GB=1TB

（2）二进制0、1----ASCII----GB2312----GBK----GB18030----Unicode----UTF-8

### Python中的标识符和保留字

1. 保留字：一些单词被python赋予了特定意义，这些单词在给任何对象起名字的时候都不能用。
2. 标识符：变量、函数、类、模块和其他对象起的名字就叫标识符。

规则：字母、数字、下划线；不能用数字开头；不能是保留字；区分大小写

### 变量的定义和使用

1. 变量：其值可以改变的量

eg：name=’玛丽亚’ （name：变量名，=：赋值运算符；玛丽亚：值）

1. 变量由三部分组成

标识：表示对象所存储的内存地址，使用内置函数id(obj)来获取

类型：表示的是对象的数据类型，使用内置函数type(obj)来获取

值：表示对象所存储的具体数据，使用print(obj)可以将值进行打印输出

### 变量的多次赋值

当多次赋值之后，变量名会指向新的空间

### Python中常见的数据类型

整数类型 ---- int ---- 98

浮点数类型 ---- float ---- 3.14159

布尔类型 ---- bool ---- True、False

字符串类型 ---- str ---- ‘人生苦短，我用Python’

### 数据类型---整数类型

（1）英文为integer，简写int，可以表示正数、负数、零

（2）整数的不同进制表示方式

十进制：默认的进制

二进制：0b开头

八进制：0o开头

十六进制：0x开头

### 数据类型---浮点类型

1. 浮点数由整数部分和小数部分组成
2. 浮点数存储不精确性：使用浮点数进行计算时，可能会出现小数位数不确定的情况

解决方案：导入模块decimal

### 数据类型---布尔类型

1. 用来表示真或假的类型：True表示真，False表示假
2. 布尔值可以转化为整数：True----1，False----0

### 数据类型---字符串类型

1. 字符串类型有被称为不可变的字符序列
2. 可以使用单引号、双引号、三引号来定义：前两者定义的字符串必须在一行，三引号定义的字符串可以分布在连续的多行。

### 数据类型转换---str( )函数与int( )函数

为什么需要数据类型转换：将不同数据类型的数据拼接在一起

str()：将其他数据类型转换为字符串

int()：将其他数据类型转换为整数

float()：将其他数据类型转换为浮点数

### 数据类型转换---float( )函数

### Python中的注释

单行注释：#开头，直到换行结束

多行注释：一对三引号

中文编码声明注释：在文件开头加上中文声明注释，用以指定源码文件袋编码格式

## 第三章

### Python的输入函数input( )

present = input（‘大圣想要什么礼物呢？’）

present：变量

=：赋值运算符--将输入函数的结果赋值给吧变量present

input( )：输入函数--需要输入回答

### input函数的高级使用

### Python中的运算符---算数运算符

算数运算符：标准算数运算符（+、—、\*、/、//整除）、取余运算符(%)、幂运算符(\*\*)

### Python中的运算符---赋值运算符

赋值运算符：执行顺序(右--->左)、支持链式赋值(a=b=c=1)、支持参数赋值(+=、—=、\*=、/=、//=、%=)、支持系列解包赋值(a,b,c=20,30,40)

### Python中的运算符---比较运算符

比较运算符：对变量或表达式的结果进行大小、真假的比较

>、<、>=、<=、!=

==：对象value的比较

is、is not：对象id的比较

### Python中的运算符---布尔运算符

布尔运算符：and、or、not、in、not in

### Python中的运算符---位运算符

位运算符：将数据转成二进制进行计算

位与&：对应数位都是1，结果数位才是1，否则位0

位或|：对应数位都是0，结果数位才是0，否则是1

左移位运算符<<：高位溢出舍弃，低位补0，向左移动一位，相当于乘2

右移位运算符>>：低位溢出舍弃，低位补0，向右移动一位，相当于除以2

### 运算符的优先级

（）--->算数运算--->位运算--->比较运算符--->布尔运算--->赋值运算

先乘除、后加减、有幂运算先进行幂运算

## 第四章

### 程序的组织结构---顺序结构

1. 任何简单或复杂的算法都可以由顺序结构、选择结构（if语句）和循环结构（while语句、for-in语句）这三种基本结构组合而成。
2. 程序从上到下顺序的执行代码，中间没有任何的判断和跳转，直到程序结束。

### 对象的布尔值

1. Python中一切皆对象，所有对象都有一个布尔值。

使用内置函数bool( )获取对象的布尔值

1. 以下对象的布尔值为False：False、数值0、None、空字符串、空列表、空元组、空字典、空集合

### 分支结构---单分支结构

1. 选择结构：程序根据判断条件的布尔值选择性的执行部分代码，明确的让计算机知道在什么条件下，该去做什么
2. 单分支结构：中文语义：如果.....就.....

### 分支结构---双分支结构

双分支结构：中文语义：如果.....不满足......就.....

### 分支结构---多分支结构

多分支结构：中文语义：多选一，执行

### 分支结构---嵌套if的使用

if 条件表达式1：

if 内层条件表达式：

内层条件执行体1

else：

内层条件执行体2

else：

条件执行体

### 条件表达式

条件表达式是if....else的简写

语法结构：x if 判断条件 else y

运算规则：如果判断条件的布尔值为True，条件表达式返回值为x，否则条件表达式的返回值为y

### pass语句

pass语句：语句什么都不做，只是一个占位符，用在语法上需要语句的地方

什么时候用：先搭建语法结构，还没想好代码怎么写的时候

哪些语句一起使用：if语句的条件执行体、for-in语句的循环体、定义函数时代函数体

## 第五章

### 内置函数range( )函数的使用

1. range( )函数：用于生成一个整数序列
2. 创建range( )对象的三种方式：

①range (stop)：创建一个(0，stop)之间的整数序列，步长为1

②range (start，stop)：创建一个(start，stop)之间的整数序列，步长为1

③range (start，stop，step)：创建一个(start，stop)之间的整数序列，步长为step

1. 返回值是一个迭代器对象
2. range类型的优点：不管range对象表示的整数序列有多长，所有range对象占用的内存空间都是相同的，因为仅仅需要存储start、stop、step；只有当用到range对象时，才会去计算序列中的相关元素
3. in与not in判断整数序列中是否存在（不存在）指定的整数

### while循环结构

1. 反复做同一件事情多情况，称为循环
2. 循环的分类：while、for-in
3. 语法结构：while条件表达式：（条件表达式内容位True，则执行条件执行体）

条件执行体（循环体）

1. 选择结构的if与循环结构的while的区别：

if是判断一次，条件为True执行一次

while是判断N+1次，条件为True执行N次

1. while循环的执行流程：四步循环法：初始化变量、条件判断、条件执行体、改变变量

### while练习题—1-100之间的偶数和

### for-in循环

1. for-in循环：in表达从（字符串、序列等）中依次取值，又称为遍历
2. for-in遍历的对象必须是可迭代对象
3. for-in的语法结构：for 自定义的变量 in 可迭代对象：

循环体

1. 循环体内不需要访问自定义变量，可以将自定义变量替代为下划线

### for-in练习题—100-999之间的水仙花数

### 流程控制语句break

break语句：用于结束循环结构，通常与分支结构if一起使用

### 流程控制语句continue

continue语句：用于结束当前循环，进入下一次循环，通常与分支结构中的if一起使用

### else语句

else语句：与else语句配合使用的三种情况：

else的搭配：if...else...：if表达式不成立时执行else

while...else...：没有碰到break时执行else

for...else....：没有碰到break时执行else

### 嵌套循环

嵌套循环：嵌套循环中又嵌套了另外的完整的循环结构，其中内层循环做为外层循环的循环体执行

### 二重循环中的break与continue

二重循环中的break和continue用于控制本层循环

## 第六章

### 为什么需要列表

（1）变量可以存储一个元素，而列表是一个“大容器”可以存储N多个元素，程序可以方便的对这些数据进行整体操作

（2）列表相当于其他语言中的数组

### 列表的创建

1. 第一种方式：列表需要使用[ ]，元素之间使用**英文的逗号**进行分割
2. 第二种方式：使用中括号，调用内置函数list( )

### 列表的特点

1. 列表元素按顺序有序排序
2. 索引映射唯一一个数据
3. 列表可以存储重复数据
4. 任何数据类型混存
5. 根据需要动态分配和回收内存

### 列表的查询操作-获取指定元素的索引

获取指定元素的索引：index

①如果列表中存在N个相同元素，只返回相同元素中的第一个元素的索引

②如果查询到元素在列表中不存在，则会抛出ValueError

③还可以在指定的strat和stop之间进行查找

### 列表的查询操作-获取列表中的单个元素

获取列表中的单个元素：

①正向索引从0到N-1，举例：lst [ 0 ]

②逆向索引从-N到-1，举例：lst [ -N ]

③指定索引不存，抛出indexError

### 列表的查询操作-获取列表中的多个元素

1. 语法格式：列表名[start：stop：step]
2. 切片操作

①切片的结果：原列表片段的拷贝

切片的范围：[start，stop)

step默认为1，简写为[start：stop]

②step为正数：从start开始往后计算切片

[:stop:step]：切片的第一个元素默认是列表的第一个元素

[start::step]：切片的最后一个元素默认是列表的最后一个元素

③step为负数：从start开始往前计算切片

[:stop:step]：切片的第一个元素默认是列表的最后一个元素

[start::step]：切片的最后一个元素默认是列表的第一个元素

### 列表元素的判断及遍历

（1）判断指定元素在列表中是否存在：in、not in

（2）列表元素的遍历：for 迭代变量 in 列表名：

### 列表元素的增加操作

（1）append( )：在列表末尾添加一个元素

（2）extend( )：在列表的末尾至少添加一个元素

（3）insert( )：在列表的任意位置添加一个元素

（4）切片：在列表的任意位置添加至少一个元素

### 列表元素的删除操作

（1）remove( )：一次性删除一个元素、重复元素只删除第一个、元素不存在的元素抛出ValueError

（2）pop( )：删除一个指定索引位置上的元素、指定索引不存在就会抛出ValueError、不指定元素则删除列表中的最后一个元素

（3）切片：一次删除列表中至少一个元素，但是将产生一个新的列表对象

（4）clear( )：清空列表

（5）del：删除列表

### 列表元素的修改操作

为指定索引的元素赋予一个新值

为指定的切片赋予一个新值

### 列表元素的排序操作

两种常见的方式：

1. 调用sort( )方法，列中所有的元素按照默认从小到大的顺序进行排序，可以指定reserve=True，进行降序排序
2. 调用内置函数sorted( )，可以指定reserve=True，进行降序排序，原列表不发生改变

### 列表生成式

列表生成式简称“生成列表的公式”

[i\*i for i in range(1,10)]

注意事项：表示列表元素的表达式，中，通常包含自定义变量

## 第七章

### 什么是字典

字典{ }：Python内置的数据结构之一，与列表[ ]一样是一个可变序列(可进行增删改操作)；以键值对的方式存储数据，字典是一个无序的序列。

### 字典的实现原理

字典的实现原理：字典的实现原理与查字典类似，查字典是先根据部首或拼音查找相应的页码，Python中的字典是根据key值查找value所在的值。

### 字典的创建

1. 最常用的方式：使用花括号，scores={ ‘张三’:100, ‘李四’:98, ‘王五’:45 }
2. 使用内置函数
3. ( )：dict（name=’jack’, age=20）

### 字典的常用操作-字典元素的获取

1. [ ]：举例，scores[‘张三’]，如果字典中不存在指定的Key，抛出keyError
2. get方法：举例，scores.get(‘张三’)，如果字典中不存在指定的Key，并不会抛出keyerror，而是返回None，可以通过参数设置默认的value，以便指定的key不存在时返回。

### 字典元素的增删改操作

1. key的判断：in，指定的key在字典中存在返回True，’张三’ in scores

not in，指定的key在字典中不存在返回True，’Marry’ not in scores

1. 字典元素的删除：del scores[‘张三’]
2. 字典元素的新增/修改：scores[‘jack’]=90

### 获取字典视图

获取字典视图的三个方法：

1. keys( )：获取字典中的所有key
2. values( )：获取字典中的所有value
3. items( )：获取字典中所有的key、value对

### 字典元素的遍历

for item in scores： print（item）

### 字典的特点

1. 字典中所有元素都是一个key-value对，key不允许重复，value可以重复
2. 字典中的元素是无序的
3. 字典中的key必须是不可变对象
4. 字典也可以根据需要动态的伸缩
5. 字典会浪费较大的内存，是一种使用空间换时间段数据结构

### 字典生成式

（1）内置函数zip( )：用于将可迭代的对象作为参数，将对象中对应的元素打包成一个元组，然后返回由这些元组组成的列表

items=[‘Fruits’, ‘Books’, ‘Others’]

prices=[96, 78, 45]

------>{‘Fruits’:96, ‘Books’:78, ‘Others’:45}

代码：lst=zip(items, prices)

print(list(lst))

1. 字典生成式：{item.upper(): price for item, price in zip(items,prices)}

## 第八章

### 什么是元组

1. 元组：Python内置的数据结构之一，是一个不可变序列，t=(‘Python’, ’hello’, 98)
2. 不可变序列与可变序列

①不可变序列：字符串、元组（没有增删改的操作）

②可变序列：列表、字典（可以对序列执行增删改操作，对象地址不发生更改）

### 元组的创建方式

1. 直接小括号

t=(‘Python’, ’hello’, 98)

1. 使用内置函数tuple( )

t=tuple(‘Python’, ‘hello’, 98)

1. 只包含一个元组的元素需要使用逗号和小括号

t=(10, )

### 为什么要将元素设计成不可边序列

1. 在多任务环境下，同时操作对象时不需要加锁；因此，在程序中尽量使用不可变序列
2. 注意事项：元组中存储的是对象的引用

如果元组中对象本身是不可变对象，则不能再引用其他对象

如果元组中的对象时可变对象，则可变对象的引用不允许改变，但数据可以改变

### 元组的遍历

元组是可迭代对象，所以可以使用for...in进行遍历

### 集合的概述与创建

1. Python语言提供的内置数据结构
2. 与列表、字典一样都属于可变类型的序列
3. 集合是没有value的字典
4. 集合的创建方式

①直接{}：s={‘Python’, ‘hello’, 90}

②使用内置函数set( )：

1. set集合特点：无序、不重复、无索引

### 集合的相关操作

1. 集合元素的判断操作

in 或 not in

1. 集合元素的新增操作

调用add( )方法，一次添加一个元素

调用update( )方法，一次至少添加一个元素

1. 集合元素的删除操作

remove( )：一次删除一个指定元素，如果指定的元素不存在抛出Keyerror

discard( )：一次删除一个指定元素，如果指定的元素不存在不抛异常

pop( )：一次只删除一个任意元素

clear( )：清空集合

### 集合之间的关系

1. 两个集合是否相等：可以使用运算符==或者！=进行判断
2. 一个集合是否是另一个集合的子集：可以调用方法issubset进行判断，B是A的子集
3. 一个集合是否是另一个集合的超集：可以调用方法issuperset进行判断，A是B的超集
4. 两个集合是否没有交集：可以调用方法isdisjoint进行判断

### 集合的数学操作

1. 交集intersection、&
2. 并集union、|
3. 差集difference、—
4. 对称差集symmetric\_difference

### 集合生成式（有一个总结）★

用于生成集合的公式：{i\*i for i in range(1,10)}

将{}修改为[ ]就是列表生成式

没有元组生成式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列表、字典、元组、集合总结 | | | | |
| 数据结构 | 是否可变 | 是否重复 | 是否有序 | 定义符号 |
| 列表（list） | 可变 | 可重复 | 有序 | [ ] |
| 元组（tuple） | 不可变 | 可重复 | 有序 | （） |
| 字典（dict） | 可变 | key不可重复  value可重复 | 无序 | {key：value} |
| 集合（set） | 可变 | 不可重复 | 无序 | { } |

## 第九章

### 字符串的创建与驻留机制

1. 字符串：在Python中字符串是基本数据类型，是一个不可变的字符序列
2. 什么叫字符串驻留机制呢？

仅保存一份相同且不可变字符串的方法，不同的值被存放在字符串的驻留池中，Python的驻留机制对相同的字符串只保留一份拷贝，后续创建相同字符串时，不会开辟新空间，而是把该字符串的地址赋给新创建的变量。

1. 驻留机制的几种情况（交互模式）

字符串的长度为0或1时

符合标识符的字符串

字符串只在编译时进行驻留，而非运行时

[-5,256]之间的整数数字

sys中的intern方法强制2个字符串指向同一个对象，Pycharm对字符串进行了优化处理。

1. 字符串驻留机制的优缺点

当需要值相同的字符串时，可以直接从字符串池里拿来使用，避免频繁的创建和销毁，提升效率和节约内存，因此拼接字符串和修改字符串时会比较影响性能的。

在需要进行字符串拼接时建议使用str类型的join方法，而非+，因为join（）方法是先计算出所有字符中的长度，然后再拷贝，只new一次对象，效率要比”+”高

### 字符串的常用操作-字符串的查询操作

1. index( )：查找子串substr第一次出现的位置，如果查找的子串不存在时，会抛出ValueError
2. rindex( )：查找子串substr最后一次出现的位置，如果查找的子串不存在时，会抛出ValueError
3. find( )：查找子串substr第一次出现的位置，如果查找的子串不存在时，则返回-1
4. rfind( )：查找子串substr最后一次出现的位置，如果查找的子串不存在时，则返回-1

### 字符串的常用操作-字符串的大小写转换操作方法

1. upper( )：把字符串中所有字符都转成大写字母
2. lower( )：把字符串中的所有字符都转成小写字母
3. swapcase( )：把字符串中所有大写字母转成小写字母，把所有小写字母转成大写字母
4. capitalize( )：把第一个字符转换为大写，把其余字符转换为小写
5. title( )：把每个单词的第一个字符转换为大写，把每个单词的剩余字符转换为小写

### 字符串的常用操作-字符串的内容对齐操作的方法

1. center( )：居中对齐，第一个参数指定宽度，第二个参数指定填充符，第二个参数是可选的，默认是空格，如果设置宽度小于实际宽度，则返回原字符串。
2. ljust( )：左对齐，第一个参数指定宽度，第二个参数指定填充符，第二个参数是可选的，默认是空格，如果设置宽度小于实际宽度，则返回原字符串。
3. rjust( )：右对齐，第一个参数指定宽度，第二个参数指定填充符，第二个参数是可选的，默认是空格，如果设置宽度小于实际宽度，则返回原字符串。
4. zfill( )：右对齐，左边用0填充，该方法只接受一个参数，用于指定字符串的宽度，如果设置宽度小于等于字符串的长度，则返回原字符串。

### 字符串的常用操作-字符串的拆分

1. split( )

①从字符串的左边开始拆分，默认的拆分字符是空格字符串，返回的值都是一个列表

②通过参数sep指定拆分字符串的拆分符

③通过参数maxsplit指定拆分字符串时代最大拆分次数，在经过最大次拆分之后，剩余的子串会单独做为一部分

1. rsplit( )

①从字符串的右边开始拆分，默认的拆分字符是空格字符串，返回的值都是一个列表

②通过参数sep指定拆分字符串的拆分符

③通过参数maxsplit指定拆分字符串时代最大拆分次数，在经过最大次拆分之后，剩余的子串会单独做为一部分

### 字符串的常用操作-字符串判断的相关方法

1. isidentifier( )：判断指定的字符串是不是合法的字符串
2. isspace( )：判断指定的字符串是否全部由空白字符组成（回车、换行、水平制表符）
3. isalpha( )：判断指定的字符串是否全部由字母组成
4. isdecimal( )：判断指定的字符串是否全部由十进制的数字组成
5. isnumeric( )：判断指定的字符串是否全部由数字组成
6. isalnum( )：判断指定的字符串是否全部由字母和数字组成

### 字符串的常用操作-替换与合并

1. relpace( )：第一个参数指定被替换的字符串，第二个参数指定替换子串的字符串，该方法返回替换后得到的字符串，替换前的字符串不发生变化，调用该方法时可以通过第三个参数指定最大替换次数
2. join( )：将列表或元组中的字符串合并成一个字符串

### 字符串的比较操作

1. 运算符：>、>=、<、<=、==、!=
2. 比较规则：首先比较两个字符串的第一个字符，如果相等则继续比较下一个字符，依次比较下去，直到两个字符串中的字符不相等时，其比较结果就是两个字符串的比较结果，两个字符串中的所有后续字符将不再比较
3. 比较原理：两个字符进行比较时，比较的是其ordinal value(原始值)，调用内置函数ord可以得到指定字符的oridinal value。与内置函数ord对应的是内置函数chr，调用内置函数chr时，指定ordinal value可以得到其对应的字符

### 字符串的切片操作

字符串时不可变类型：不具备增删改操作，切片操作将产生新的对象

### 格式化字符串

为什么需要格式化字符串：按一定格式输出的字符串

①%作为占位符：%s（字符串）、%i或%d（整数）、%f（浮点数）

②{ }作占位符：

③f-string

### 字符串的编码与解码

1. 为什么需要字符串的编码转换

A计算机(str在内存中以unicode表示)---->编码---->byte字节传输---->解码---->B计算机(显示)

1. 编码与解码的方式

编码：将字符串转换为二进制数据（bytes）

解码：将bytes类型的数据转换为字符串类型

## 第十章

### 函数的定义与调用

1. 什么是函数：函数就是执行特定任务、完成特定功能的一段代码
2. 为什么需要函数：复用代码、隐藏实现细节、提高可维护性、提高可读性便于调试
3. 函数的创建

def 函数名（输入参数）： def calc（a，b）：

函数体 c=a+b

return xxx return c

1. 函数的调用

函数名（实际参数）

result=calc（10,20）

print（result）

### 函数的参数传递

1. 位置实参：根据形参对应的位置进行实参传递
2. 关键字实参：根据形参的名称进行实参传递

### 函数参数传递的内存分析

### 函数的返回值

函数返回多个值时，结果为元组

### 函数的参数定义-默认值参数

默认值参数：函数定义时，给形参设置默认值，只有与默认值不符的时候才需要传递实参

### 函数的参数定义-个数可变的位置参数、关键字形参

1. 个数可变的位置参数

定义函数时，可能无法事先确定传递的位置实参的个数，可以使用可变的位置参数

使用\*定义个数可变的位置形参

结果为一个元组

1. 个数可变的关键字形参

定义函数时，无法事先确定传递的关键字实参的个数时，使用可变的关键字形参

使用\*\*定义个数可变的关键字形参

结果为一个字典

### 函数的参数总结

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数的参数总结 | | | | |
| 序号 | 参数的类型 | 函数的定义 | 函数的调用 | 备注 |
| 1 | 位置实参 |  | √ |  |
| 将序列中的每个元素都转换为位置实参 |  | √ | 使用\* |
| 2 | 关键字实参 |  | √ |  |
| 将字典中的每个键值对都转换为关键字实参 |  | √ | 使用\*\* |
| 3 | 默认值形参 | √ |  |  |
| 4 | 关键字形参 | √ |  | 使用\* |
| 5 | 个数可变的位置形参 | √ |  | 使用\* |
| 6 | 个数可变的关键字形参 | √ |  | 使用\*\* |

### 变量的作用域

1. 变量的作用域：程序代码能访问该变量的区域
2. 根据变量的有效范围可分为：局部变量、全局变量

①局部变量：在函数内定义并使用的变量，只在函数内部有效，局部变量使用global声明，这个变量就会变成全局变量

②全局变量：函数体外定义的变量，可作用于函数内外

### 递归函数

1. 什么是递归函数

如果在一个函数的函数体内调用了该函数本身，这个函数就称为递归函数

1. 递归的组成部分

递归调用与递归终止条件

1. 递归的调用过程

每递归调用一次函数，都会在栈内存分配一个栈帧

每执行完一次函数，都会释放相应的空间

1. 递归的优缺点

缺点：占用内存多，效率低下

优点：思路和代码简单

### 斐波那契数列-本章总结

1 1 2 3 5 8

## 第十一章

### Bug的由来

1. Bug的由来

世界上第一部万用计算机的进化版-马克2号（Mark II）

1. Debug

### Bug的常见类型-粗心导致的错误

粗心导致的语法错误SyntaxError：

age = input('请输入你的年龄：')  
print(type(age))  
  
if int(age) >= 18:  
 print('成年人....')

i=0  
while i<10:  
 print(i)  
 i+=1

粗心导致错误的自查宝典：

1. 漏了末尾的冒号，如if语句，循环语句，else子句等
2. 缩进错误，该缩进的没缩进，不该缩进的瞎缩进
3. 把英文符号写成中文符号，比如说：引号，冒号，括号
4. 字符串拼接的时候，把字符串和数字拼接在一起
5. 没有定义变量，比如说while的循环条件的变量
6. “==”比较运算符和“=”赋值运算符的混用

### Bug的常见类型-知识点不熟练导致的错误

1. 索引越界问题IndexError

lst=[11,22,33,44]

print(lst[4]) #错误list index out of range

1. append（）方法的使用掌握不熟练

lst=[]

lst=append(‘A’, ‘B’, ‘C’)

print(lst)

lst.append('a')  
lst.append('b')  
lst.append('c')

### Bug的常见类型-思路不清导致的错误

思路不清导致的问题解决方案：

1. 使用print( )函数
2. 使用“#”暂时注释部分代码

### Bug的常见类型-被动掉坑-try-except

1. 被动掉坑：程序代码逻辑没有错误，只是因为用户错误操作或者一些“例外情况”而导致的程序崩溃。
2. 被动掉坑问题的解决方案：

①Python提供了异常处理机制，可以在异常出现时即时捕获，然后内部“消化”，让程序继续运行。

②多个except结构：捕获异常的顺序按照先子类后父类的顺序，为了避免遗漏可能出现的异常，可以在最后增加BaseException

### try-except-else结构与try-except-else-finally结构

1. try-except-else结构

如果try块没有抛出异常，则执行else块，如果try块抛出异常，则执行except块

1. try-except-else-finally结构

finally块无论是否发生异常都会执行，能常用来释放try块中申请的资源

### Python中常见的异常类型

|  |  |  |
| --- | --- | --- |
| 序号 | 异常类型 | 描述 |
| 1 | ZeroDivisionError | 除（或取模）零（所有数据类型） |
| 2 | IndexError | 序列中没有此索引（index） |
| 3 | KeyError | 映射中没有这个键 |
| 4 | NameError | 未声明/初始化对象（没有属性） |
| 5 | SyntaxError | Python语法错误 |
| 6 | ValueError | 传入无效的参数 |

### Python的异常处理机制-Traceback模块的使用

使用traceback模块打印异常信息

### Pycharm开发环境的调试

1. 断点

程序运行到此处，暂时挂起，停止执行。此时可以详细观察程序的运行情况，方便做出进一步的判断。

1. 进入调试模式的三种方式

①单击工具栏上的按钮

②右键单击编辑区：点击‘debug’模块名

③快捷键：shift+F9

## 第十二章

### 编程的两大思想-面向过程-面向对象

|  |  |  |
| --- | --- | --- |
|  | 面向过程 | 面向对象OOP |
| 区别 | 事物比较简单，可以用线性的思维去解决 | 事物比较复杂，使用简单的线性思维无法解决 |
| 共同点 | 面向过程和面向对象都是解决实际问题的一种思维方式 | |
|  | 二者相辅相成，并不是对立的  解决复杂问题，通过面向对象方式便于我们从宏观上把握事物之间复杂的关系，方便我们分析整个系统；具体到微观操作，仍然使用面向过程方式来处理 | |

### 类与对象

1. 类

类别，分门别类，物以类聚，人类，鸟类，动物类......

类似多个类似事物组成的群体的统称。能够帮我们快速理解和判断事物的性质。

1. 数据类型

不同的数据类型属于不同的类；使用内置函数type( )查看数据类型

1. 对象

100、99、520都是int类之下包含的相似度不同个例，这个个例专业术语称为实例或对象。

Python中一切皆对象

### 定义Python中的类

1. 创建类的语法

class Student：

pass

1. 类的组成：类属性、实例方法、静态方法、类方法

### 对象的创建

1. 对象的创建又称为类的实例化
2. 语法：

实例名=类名（）

1. 例子

stu=Student（）

1. 意义：有了实例，就可以调用类中的内容

### 类属性、类方法、静态方法

1. 类属性：类中方法外地变量称为类属性，被该类的所有对象所共享
2. 类方法：使用@classmethod修饰的方法，使用类名直接访问的方法
3. 静态方法：使用@staticmethod修饰的方法，使用类名直接访问的方法
4. print（Student.native\_place） #访问类属性

Student.cm（） #调用类方法

Student.sm（） #调用静态方法

### 动态绑定属性和方法

Python是动态语言，在创建对象后，可以动态地绑定属性和方法

## 第十三章

### 面向对象的三大特征-封装的实现方式

1. 封装：提高程序的安全性

①将数据（属性）的行为（方法）包装到类对象中。在方法内部对属性进行操作，在类对象的外部调用方法。这样，无需关心方法内部的具体实现细节，从而隔离了复杂度。

②在Python中没有专门的修饰符用于属性的私有，如果该属性不希望在类对象外部被访问，前边使用两个“\_”

1. 继承：提高代码的复用性
2. 多态：提高程序的可扩展性和可维护性

### 继承及其实现方式

1. 语法格式

class 子类类名（父类1, 父类2...）：

pass

1. 如果一个类没有继承任何类，则默认继承object
2. Python支持多继承
3. 定义子类时，必须在其构造函数中调用父类的构造函数

### 方法重写

（1）如果子类对继承自父类的某个属性或方法不满意，可以在子类中对其（方法体）进行重新编写

（2）子类重写后的方法中可以通过super( ).xxx( )调用父类中被重写的方法

### object类

1. object类是所有类的父亲，因此所有类都有object类的属性和方法。
2. 内置函数dir( )可以查看指定对象的所有属性
3. Objcect有一个\_\_str\_\_( )方法，用于返回一个对于“对象的描述”，对应于内置函数str( )经常用于print( )方法，帮我们查看对象的信息，所以我们经常会对\_\_str\_\_( )进行重写。

### 多态

1. 简单的说，多态就是“具有多种形态”，它指的是：即便不知道一个变量所引用的对象到底是什么类型，仍然可以通过这个变量调用方法，在运行过程中根据变量所引用对象的类型，动态决定调用哪个对象中的方法。
2. 静态语言和动态语言关于多态的区别

静态语言实现多态的三个必要条件：继承、方法重写、父类引用指向子类对象

动态语言的多态崇尚“鸭子类型”，当看到一只鸟走起来像鸭子、游泳起来像鸭子、那么这只鸟就可以被称为鸭子。在鸭子类型中，不需要关心对象时什么类型，到底是不是鸭子，只关心对象的行为。

### 特殊属性和特殊方法

|  |  |  |
| --- | --- | --- |
|  | 名称 | 描述 |
| 特殊属性 | \_\_dict\_\_ | 获得类对象或实例对象所绑定的所有属性和方法的字典 |
| 特殊方法 | \_\_len\_\_( ) | 通过重写\_\_len\_\_( )方法，让内置函数len( )的参数可以是自定义类型 |
| \_\_add\_\_( ) | 通过重写\_\_add\_\_( )方法，可以使用自定义对象具有“+”功能 |
| \_\_new\_\_( ) | 用于创建对象 |
| \_\_init\_\_( ) | 对创建的对象进行初始化 |

### \_\_new\_\_与\_\_init\_\_演示创建对象的过程

### 类的赋值与浅拷贝

1. 变量的赋值操作

只是形成两个变量，实际上还是指向同一个对象

1. 浅拷贝

Python拷贝一般都是浅拷贝，拷贝时，对象包含的子对象内容不拷贝。因此，源对象与拷贝对象会引用同一个子对象（主对象变了，子对象没变）

1. 深拷贝

使用copy模块和deepcopy模块，递归拷贝对象中包含的子对象，源对象和拷贝对象所有的子对象也不相同（主对象和子对象都变化）

### 深拷贝

## 第十四章

### 什么叫模块-模块化编程的好处

1. 模块英文为Modules
2. 函数与模块的关系：一个模块中可以包含N多个函数
3. 在Python中一个扩展名为.py的文件就是一个模块
4. 使用模块的好处：方便其它程序和脚本的导入并使用、避免函数名和变量名冲突、提高代码的可维护性、提高代码的可重用性

### 模块的导入

1. 创建模块

新建一个.py文件，名称尽量不要与Python自带的标准模块名称相同

1. 导入模块

import 模块名称 [as 别名]

from 模块名称 import 函数/变量/类

### 以主程序形式运行

在每个模块的定义中都包括一个记录模块名称的变量\_\_name\_\_，程序可以检查该变量，以确定他们在哪个模块中执行。如果一个模块不是被导入到其它程序中执行，那么它可能在解释器的顶级模块中执行。顶级模块的\_\_name\_\_变量的值为\_\_main\_\_

if \_\_name\_\_ = ‘\_\_main\_\_’ :

pass

### Python中的包

1. 包是一个分层次的目录结构，它将一组功能相近的模块组织在一个目录下
2. 作用：代码规范、避免模块名称冲突
3. 包与目录的区别：包含\_\_init\_\_.py文件的目录称为包，目录里通常不包含\_\_init\_\_.py文件
4. import 包名.模块名

### Python中的内置模块

|  |  |
| --- | --- |
| 模块名 | 描述 |
| sys | 与Python解释器及其环境操作相关的标准库 |
| time | 提供与时间相关的各种函数的标准库 |
| os | 提供了访问操作系统服务功能的标准库 |
| calendar | 提供了与日期相关的各种函数的标准库 |
| urllib | 用于读取来自网上（服务器）的数据标准库 |
| json | 用于使用JSON序列化和反序列化对象 |
| re | 用于在字符串中执行正则表达式匹配和替换 |
| math | 提供标准算数运算函数的标准库 |
| decimal | 用于进行精确控制运算精度、有效数位和四舍五入操作的十进制运算 |
| logging | 提供了灵活的记录事件、错误、警告和调试信息等日志信息的功能 |

### 第三方模块的安装及使用

1. 第三方模块的安装

pip install 模块名

1. 第三方模块的使用

import 模块名

## 第十五章

### 编码格式的介绍

1. Python的解释器使用的是Unicode（内存）
2. .py文件在磁盘上使用UTF-8存储（外存）
3. 各种编码

ASCII：7位表示1个字符，最高位为0,只能表示128个字符

ISO8859-1：8位表示1个字符，能表示256个字符，兼容ASCII

GB2312、GBK、GB18030：兼容ISO8859-1，汉字2个字节

Unicode：定长编码，2个字节表示1个字符，不兼容ISO8859-1

UTF-8：变长编码，1-4个字节表示1个字符，英文1个字节，汉字3个字节，是Unicode的实现

### 文件的读写原理-读取磁盘文件中的内容

1. 文件的读写俗称“IO操作”
2. 文件读写操作流程：Python操作文件--->打开或新建文件--->读、写文件--->关闭资源
3. 操作原理
4. 内置函数open( )创建文件对象
5. 语法规则：file = open（filename [,mode, encoding]）

file：被创建的文件对象

open：创建文件对象的函数

filename：要创建或打开的文件名称

mode：打开模式，默认为只读

encoding：默认文本文件中字符的编写格式为gbk

### 常用的文件打开模式

按照文件中数据的组织形式，文件分为以下两大类：

（1）文本文件：存储的是普通“字符”文本，默认为unicode字符集，可以使用记事本程序打开

（2）二进制文件：把数据内容用“字节”进行存储，无法用记事本打开，必须使用专用的软件打开，举例：mp3音频文件，jpg图片，.doc文档等

|  |  |
| --- | --- |
| 打开模式 | 描述 |
| r | 以只读模式打开文件  文件的指针将会放在文件的开头 |
| w | 以只写模式打开文件  如果文件不存在则创建  如果文件存在，则覆盖原有内容，文件指针在文件的开头 |
| a | 以追加模式打开文件  如果文件不存在则创建，文件指针在文件开头；  如果文件存在，则在文件末尾追加内容，文件指针在原文件末尾 |
| b | 以二进制方式打开文件  不能单独使用，需要与其它模式一起使用，rb或者wb |
| + | 以读写方式打开文件  不能单独使用，需要与其他模式一起使用，a+ |

### 文件对象的常用方法

|  |  |
| --- | --- |
| 方法名 | 说明 |
| read([size]) | 从文件中读取size个字节或字符的内容并返回，若省略[size]，则读取到文件末尾，即一次读取文件所有内容 |
| readline( ) | 从文本文件中读取一行内容 |
| readlines( ) | 把文本文件中每一行都作为独立的字符串对象，并将这些对象放入列表返回 |
| write(str) | 将字符串str内容写入文件 |
| writelines(s\_list) | 将字符串列表s\_list写入文本文件，不添加换行符 |
| seek(offset[,whence]) | 把文件指针移动到新的位置，offset表示相对于whence的位置：  offset：为正往结束方向移动，为负往开始方向移动  whence不同的值代表不同含义：  0：从文件头开始计算（默认值）  1：从当前位置开始计算  2：从文件尾开始计算 |
| tell( ) | 返回文件指针的当前位置 |
| flush( ) | 把缓冲区的内容写入文件，但不关闭文件 |
| close( ) | 把缓冲区的内容写入文件，同时关闭文件，释放文件对象相关资源 |

### with语句（上下文管理器）

1. with语句可以自动管理上下文资源，不论什么原因跳出with块，都能确保文件正确的关闭，以此来达到释放资源的目的
2. with open(‘logo.png’, ‘rb’) as src\_file

src\_file.read( )

### os模块的常用函数

1. os模块是Python内置的与操作系统功能和文件系统相关的模块。该模块中的语句的执行结果通常与操作系统有关，在不同的操作系统上运行，得到的结果可能不一样。
2. os模块与os.path模块用于对目录或文件进行操作

|  |  |
| --- | --- |
| os模块操作目录相关函数 | |
| 函数 | 说明 |
| getcwd( ) | 返回当前的工作目录 |
| listdir(path) | 返回指定路径下的文件和目录信息 |
| mkdir(path[,mode]) | 创建目录 |
| makedirs(path1/path2...[,mode]) | 创建多级目录 |
| rmdir(path) | 删除目录 |
| removedirs(path1/path2...) | 删除多级目录 |
| chdir(path) | 将path设置为当前工作目录 |

### os.path模块的常用方法-课堂案例

|  |  |
| --- | --- |
| os.path模块操作目录相关函数 | |
| abspath(path) | 用于获取文件或目录的绝对路径 |
| exists(path) | 用于判断文件或目录是否存在，存在返回True，不存在返回False |
| join(path,name) | 将目录与目录或者文件名拼接起来 |
| splitext( ) | 分离文件名和扩展名 |
| basename(path) | 从一个目录中提取文件名 |
| dirname(path) | 从一个路径中提取文件路径，不包括文件名 |
| isdir(path) | 用于判断是否为路径 |

## 实战-学生管理系统

### 学生信息管理系统-需求分析

1. 添加学生及成绩信息
2. 将学生信息保存到文件中
3. 修改和删除学生信息
4. 查询学生信息
5. 根据学生成绩进行排名’
6. 统计学生的总分

### 学生信息管理系统-系统设计

（1）学生信息管理系统的7大模块：录入学生信息模块、查找学生信息模块、删除学生信息模块、修改学生信息模块、学生成绩排名模块、统计学生总人数模块、显示全部学生信息模块

（2）学生信息维护：录入学生信息、删除学生信息、修改学生信息

查询/统计：按学生姓名、ID查找，查询并显示所有学生信息，统计学生总人数

排序：升序/降序（根据英语成绩、Python成绩、Java成绩、总成绩排序）

1. 系统开发环境

操作系统：win10

Python解释器版本：Python3.8

开发工具：Pycharm

Python内置模块：os，re

1. 项目目录结构：studentsys

### 学生信息管理系统-主函数设计

|  |  |
| --- | --- |
| 实现主函数 | |
| 编号 | 功能 |
| 10 | 退出系统 |
| 1 | 录入学生信息，调用insert（）函数 |
| 2 | 查找学生信息，调用search（）函数 |
| 3 | 删除学生信息，调用delete（）函数 |
| 4 | 修改学生信息，调用modify（）函数 |
| 5 | 对学生成绩排序，调用sort（）函数 |
| 6 | 统计学生总人数，调用total（）函数 |
| 7 | 显示所有的学生信息，调用show（）函数 |

### 学生信息管理系统-录入学生信息功能

save(student)函数，用于将学生信息保存到文件

insert( )函数，用于录入学生信息

### 学生信息管理系统-删除学生信息功能

从控制台录入学生id，到磁盘文件中找到对应的学生信息，并将其删除

编写主函数中调用的删除学生信息的函数delete( )

调用了show( )函数显示学生信息，该函数的功能将在后面完成

### 学生信息管理系统-修改学生信息功能

从控制台录入学生id，到磁盘文件中找到对应的学生信息，将其进行修改

编写主函数中调用的修改学生信息的函数modify( )

调用了show( )函数显示学生信息，该函数的功能将在后面完成

### 学生信息管理系统-查找学生信息功能

从控制台录入学生id或姓名，到磁盘文件中找到对应的学生信息

编写主函数中调用的修改学生信息的函数search( )

定义显示查询结果的函数show\_student(query\_student)

### 学生信息管理系统-统计学生人数

统计学生总人数：统计学生信息文件中保存到学生信息个数

### 学生信息管理系统-显示所有学生信息

将学生信息文件中保存的全部学生信息获取并显示

### 学生信息管理系统-排序模块设计

主要对学生信息按照英语成绩、Python成绩、Java成绩、总成绩进行升序/降序排序

### 学生信息管理系统-项目打包生成exe

在线安装方式：pip install Pyinstaller

## 实操案例