重要概念

Python 语言：是一种面向对象的解释型计算机程序设计语言

面向对象

- 面向对象是一种编程思想，是对现实世界中的事物进行抽象的方式。应用到代码编程设计中，是一种建立现实世界事物模型的方式。

解释型语言

- 执行程序时代码解释一行运行一行，解释器把源代码转换成称为字节码的中间形式，然后再把它翻译成计算机使用的机器语言并运行

Python 的优点

- 简单、易学、免费开源、高层语言、可移植性、解释性、面向对象、可扩展性、丰富的库、规范的代码

Python 的缺点

- 运行速度慢（可以使用 C++改写关键部分来提高运行速度）、国内市场较小、中文资料匮、构架选择太多

PyCharm 介绍

- PyCharm 是一种 Python IDE，带有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具。

注释的分类

- 分为多行注释（"""注释内容"""）和单行注释（# 注释内容）

注释的作用：

1. 注释的主要作用是为了提高代码的可读性
2. 注释可以成为函数、方法的文档
3. 特殊情况下，注释也会有额外的功能，如# coding=utf-8

pycharm 中的提示性信息：

1. 语法错误：文字底部红色波浪线

2. 语法不符合规范：文字底部灰色波浪线
3. 单词拼写提示：文字底部绿色波浪线

关键字

- 关键字是 Python 语言中已经占用，具有特殊含义的符号

标识符

- 标识符是指用来标识某个实体的符号

标识符规范：

1. 只能由数字，字母，_（下划线）组成
2. 不能以数字开头
3. 不能是关键字
4. 区分大小写

命名约束：

- 下划线分隔法（推荐）：多个单词组成的名称，使用全小写字母书写，中间使用_分隔

驼峰命名法：

1. 大驼峰：多个单词组成的名称，每个单词首字母使用大写字母书写，其余字母使用小写字母书写
2. 小驼峰：多个单词组成的名称，第一个单词首字母小写，其他单词首字母都大写，其余字母使用小写字母书写

变量：变量用于描述计算机中的数据存储空间

变量的作用：在计算机内存中保存数据

变量类型获取：使用 type(数据)即可获取数据对应的类型

输入数据的类型：input 函数接收的数据默认为字符串类型

转换函数：

- 通过转换函数实现接收其他类型的数据

  1.接收整数：字符串→整型数据： int(“整数格式的字符串”)

  2.接收小数：字符串→浮点型数据：float(“小数格式的字符串”)

比较运算

- 比较运算是发生在两个同类型数据间的一种操作，比较运算是使用固定的比对规则对两个数据进行比对，如果比较运算式子成立得到结果 True,如果不成立，得到结果 False

字符串比较运算规则：

1. 对两个字符串中对应位置每个字符逐一比对
2. 排序较小的整体字符串值小
3. 如果比较结果相等，比较下一个字母

4. 直到比对到最后一个字母
5. 若比对的一方没有字符，则整体字符串值小

关系运算

- 关系运算是发生在两个布尔型数据间的一种操作，关系运算是根据固定规则对两个布尔值进行运算，最终结果是布尔值

循环概念

- 循环是程序设计语言中反复执行某些代码的一种计算机处理过程

函数概念

- 函数(function)是将具有独立功能的代码块组织成为一个整体，使其具有特殊功能的代码集。（函数是能实现一定功能的代码块）

函数的作用

- 使用函数可以加强代码的复用性，提高程序编写的效率

函数定义和调用规则：

1. 定义规则：函数必须先定义，后调用。否则程序将报错
2. 调用规则：函数定义部分的代码仅用于声明函数，调用时才实际执行函数内容

函数文档注释：文档注释可以为函数添加功能说明，方便开发者查阅函数相关信息

形参和实参

1. 形参是函数定义时规定的参数，仅在函数体内有效
2. 实参是函数调用时使用的参数，该值将传递给函数

函数参数的作用域

- 函数参数（形参）的作用域是从函数定义位置开始到函数定义结束位置

变量的作用域

1. 局部变量：函数内部定义的变量从函数定义位置开始到函数定义结束位置有效
2. 全局变量：函数外部定义的变量，从变量定义位置开始，在整个文件中有效

面向对象和面向过程的区别

- 面向过程关注的是完成工作的步骤，面向对象关注的是谁能完成工作。面向对象是在完成工作的时候关注哪些个体能够完成对应的工作，找到对应的个体即可完成对应任务。

成员方法内可以调用类下的所有属性和方法

类方法下只能调用类属性

静态方法可以用类名调用，也可以用对象名来调用

封装的作用

- 封装操作可以对受访问保护的成员进行功能开放的控制，达到保护数据不被非法访问的目的

列表的概念

- 列表是一种存储大量数据的存储模型

列表的特点

- 列表具有索引的概念，可以通过索引操作列表中的数据。列表中的数据可以进行添加、删除、修改、查询等操作

列表常用方法

方法名	功能
append(data) 关键词：追加	在列表的末尾添加数据
insert(idx,data) 关键词：插入	在列表的指定位置插入数据，如果索引位置超过列表数据总量，数据将插入到列表末尾处
extend(model) 关键词：追加全部	在列表的末尾添加参数对象中的所有数据
remove(data) 关键词：删除	从列表中删除指定的数据，如果数据不存在将报错
pop(idx) 关键词：获取删除	从列表中获取并删除指定索引位置上的数据，如果索引值

方法名	功能
clear() 关键词：清空	清空列表中的数据
index(data) 关键词：查询位置	查询列表中指定数据对应的索引，如果数据不存在将报错
count(data) 关键词：统计数量	统计列表中指定数据出现的数量

元组的概念

- 元组是一种存储固定数据的存储模型

元组的特点

- 元组具有索引的概念，可以通过索引操作元组中的数据。元组中的数据可以进行查询操作，但不能进行添加、删除、修改操作。

元组常用方法

方法名	功能
index(data) 关键词：查询位置	查询元组中指定数据对应的索引，如果数据不存在将报错
count(data) 关键词：统计数量	统计元组中指定数据出现的数量

元组注意事项

1. 元组中的数据如果是非引用类型数据，不允许修改
2. 元组中的数据如果是引用类型对象，该对象不允许替换，而对象的属性值可以

集合的概念

- 集合是一种存储大量无序不重复数据的存储模型。

集合的特点

- 集合没有索引的概念。集合中的数据可以进行添加、删除等操作。

集合的作用

- 当需要存储较多数据，且保障不重复时，并且进行迭代取出操作时，推荐选择集合

集合常用方法

方法名	功能
add(data) 关键词：添加	在集合中添加数据
remove(data) 关键词：删除	从集合中删除指定的数据，如果数据不存在将报错
pop() 关键词：获取删除	从集合中获取并删除第一个数据
clear() 关键词：清空	清空集合中的数据

字典的概念

- 字典是一种使用“键值对结构”存储数据的存储模型

字典的特点

- 字典不具有索引的概念，字典使用键 key 代替索引，可以通过键操作字典中存储的数据值 value。字典可以根据键 key 进行数据的添加、删除、修改、查询操作。

字典的常用方法

方法名	功能
pop(key) 关键词：删除获取	从字典中删除指定键 key 对应的键值对，如果键 key 不存在将报错
popitem() 关键词：删除	从字典中删除指定键 key 对应的键值对，如果键 key 不存在将报错
clear() 关键词：清空	清空字典中的数据
setdefault(key,value) 关键词：检测添加	添加新的键值对，如果存在对应的键，则忽略该操作

方法名	功能
update(dict) 关 键词：更新数据	使用新字典中的数据对原始字典数据进行更新
get(key) 关键词：获取	根据键 key 查询字典中对应的值，如果键 key 不存在将返回 None
keys() 关键词：获取键列表	获取字典中所有的键 key 组成的列表数据
values() 关键词：获取值列表	获取字典中所有的值 value 组成的列表数据
items() 关键词：获取键值对列表	获取字典中所有的键值对列表数据

字典注意事项

- 字典中的键是唯一的

字典的作用

1. 当需要存储少量数据，并且期望在编程期以最快的速度获取单个数据， 推荐选择字典。
2. 当需要使用非对象格式保存单个对象的属性值，推荐选择字典

for 循环

- for 循环用于对数据存储模型进行访问遍历

Range 的功能

- 创建连续的整数

Range 的应用场景

- 配合 for 循环构造指定次数的循环
- 快速创建由连续的整数作为数据的列表、元组、集合对象

公共方法

方法名	功能
len(model) 关键词：数据总量	获取容器模型中的数据总量
max(model) 关键词：最大值	获取容器模型中的最大值，对于字典获取字典的键 key 的最大值
min(model) 关键词：最小值	获取容器模型中的最小值，对于字典获取字典的键 key 的最小值

切片的作用

- 获取列表、元组或字符串中的局部数据

字符串概念

- 字符串是一个容器，包含若干个字符并按照一定的顺序组织成一个整体。字符串支持索引操作。

字符串切片概念

- 字符串切片指获取字符串中的某一部分，形成新的字符串对象。

字符串操作

- 基本操作
-

操作格式/函数名称	功能
变量名[idx] 关键词： 获取单个字符	获取字符串中指定索引位置的字符
str1 in str2 关键词：包含	判断一个字符串 str2 是否包含另一个字符串 str1
str1 not in str2 关键词：不包含	判断一个字符串 str2 是否不包含另一个字符串 str1
len(str) 关键词：长度	本操作是一个函数，无须用字符串对象调用，用于获取一个字符串中所包含的字符数量
max(str) 关键词：最大值	本操作是一个函数，无须用字符串对象调用，用于获取字符串中排序最大的字符
min(str) 关键词：最小值	本操作是一个函数，无须用字符串对象调用，用于获取字符串中排序最小的字符

- 状态获取操作
-

方法	功能
isupper() 关键词：是否全大写	判断字符串是否是全大写字母组成
islower() 关键词：是否全小写	判断字符串是否是全小写字母组成
isdigit() 关键词：是否全数字	判断字符串是否是由纯数字组成
isalpha() 关键词：是否全字母	判断字符串是否是由纯字母组成
isalnum() 关键词：是否全数字字母	判断字符串是否是由纯数字和字母组成
istilte() 关键词：是否单词首字母大写	判断字符串是否是满足单词首字母大写格式
startswith(str) 关键词：判定前缀	判断字符串是否以指定字符串开始
endswith(str) 关键词：判定后缀	判断字符串是否以指定字符串结束

- 字符转换操作

方法名	功能
lower()	字符串中所有字母转小写字母，支持英文字母
casefold()	字符串中所有字母转小写字母，支持各种语言

方法名	功能
upper()	字符串中所有字母转大写字母
swapcase()	字符串中字母大写转小写，小写转大写
title()	字符串中每个单词首字母大写，其余字母小写（区分单词以空格区分）
capitalize()	字符串首个字母大写，其余字母小写

- **格式转换操作**

方法	功能
strip (str)	去掉字符串左右两侧在参数字符串中包含的所有字符
lstrip (str)	去掉字符串左侧在参数字符串中包含的所有字符
ljust (len,str)	使用指定字符在原始字符串右侧补充到长度为指定值

方法	功能
rjust (len,str)	使用指定字符在原始字符串左侧补充到长度为指定值
center (len,str)	使用指定字符在原始字符串两侧补充到长度为指定值，左侧补充数量≥右侧补充数量
zfill(len)	使用 0 在原始字符串左侧补充到长度为指定值，小数点占 1 位

- **拆分操作**

方法名	功能
partition(str)	从字符串左侧查找到参数后，将参数左侧、参数、参数右侧的三个字符串组成元组并返回
rpartition(str)	从字符串右侧查找到参数后，将参数左侧、参数、参数右侧的三个字符串组成元组并返回
split(str)	使用参数作为分割线将原始字符串拆分成若干个字符串并组织成列表返回
splitlines ()	使用换行符作为分割线将原始字符串拆分成若干个字符串并组织成列表返回

- 连接操作

方法名	功能
join(str)	将原始字符串填充到参数的每个字符之间组成新的字符串返回
str1 + str2	将两个字符串按照顺序拼接成一个新的字符串返回

- 查询操作

方法名	功能
find(str,begin,end)	从左侧查找字符串从指定开始位置到指定结束位置间第一次出现的索引位置
rfind(str,begin,end)	从右侧查找字符串从指定开始位置到指定结束位置间第一次出现的索引位置
rindex(str,begin,end)	从左侧查找字符串从指定开始位置到指定结束位置间第一次出现的索引位置
rindex(str,begin,end)	从右侧查找字符串从指定开始位置到指定结束位置间第一次出现的索引位置

方法名	功能
count(str)	查询指定字符串在原始字符串中出现的次数

- 替换操作

方法名	功能
replace(old_str,new_str,num)	使用新字符串替换原始字符串中的指定字符串信息
expandtabs()	使用空格替换原始字符串中的制表位\t

- 其他操作

方法名	功能
maketrans(str1,str2)	使用两个等长的字符串中的每个对应位置的字符生成一个字典对象
translate (dict)	使用字典对字符串信息进行转换

默认参数概念

- 默认参数指函数/方法在定义时为形参赋值，对应的形参称为默认参数。
- 默认参数是一个参数定义期的概念，与调用无关。

默认参数的作用

- 如果参数定义默认参数，在调用函数/方法时，未对该参数进行传值，则使用默认值作为该参数的值。

关键字参数的概念

- 函数/方法在调用时为指定名称的形参进行赋值，对应实参称关键字参数。
- 关键字参数是一个参数调用期的概念，与定义无关。

关键字参数的作用

- 按名称确认实参为指定的形参赋值。

可变参数概念

- 函数/方法在定义时，定义一个特殊的形参，用于接收调用时传入的任意数量的实参，对应的形参称为可变参数。
- 可变参数是一个参数定义期与调用期都有效的概念。

可变参数的作用

- 简化函数定义过程，定义可以接收无数个实参的形参
- 形参接受到的数据，无论数量多少，包装成一个元组对象

字典参数概念

- 函数/方法在定义时，定义一个特殊的形参，用于接收未定义直接使用的关键字参数，对应的形参称为字典参数。

字典参数的作用

- 简化函数定义过程，接收未定义直接使用的关键字参数。
- 在多层级调用间进行关键字参数的逐级传递。
- 未定义的关键字参数，无论数量多少，接收后包装成一个字典对象。

递归函数的概念

- 函数/方法在执行过程中出现了对自身函数/方法的调用，称该过程为递归调用，称这样的函数为递归函数。

递归函数调用要求

- Python 语言中函数调用最大层级为 1000 层，超过该层级，程序将报错（不同语言设置不同）

匿名函数概念

- 匿名函数即没有名称的函数，在 python 语言中，匿名函数是使用 lambda 关键字定义的一个特殊表达式，也称为 lambda 表达式。

引用

- **引用概念**
-
  - 引用是一种变量指向数据存储空间的现象
-
  - 内存地址是数据在物理内存中的存储位置

- 引用地址是对象在内存中的描述性地址，该地址与内存地址有区别

引用特征

- 相同的数据在内存空间中仅占用一个存储空间，不同的变量使用相同的数据则指向相同的存储空间。

获取内存存储地址（编号）

- 语法格式：
- id(数据名)
- id(变量名)

地址存储特殊性

- 使用固定内存地址存储数据如下：

-5 到 256 的整数

True 和 False

由字母、数字、下滑线组成的字符串

-
  - 使用临时内存地址存储数据如下：
  - 小于-5 后大于 256 的整数
  - 所有小数
  - 包含字母、数字、下滑线之外的字符组成的字符串

可变类型与不可变类型

- ### 可变类型
-
  - 列表 集合 字典 对象
- ### 不可变类型
-
  - 数值 字符串 布尔 元组

- 函数的形参如果接收到的实参是不可变类型，函数内部的操作不会对外部的实参产生影响
- 函数的形参如果接收到的实参是可变类型，函数内部的操作会对外部的实参产生影响