**第一台计算机**

**诞生时间**：1946年2月14日

**地点**：宾夕法尼亚大学 计算机内部以二进制数形式进行存储

**计算机之父**：约翰.冯·诺依曼

基本工作原理是存储程序和程序控制

没有安装任何软件的计算机称为裸机

**IDE：**集成开发环境

**包括**：解释器(编译器)和编辑器

**例如**：

解释器(解析代码、环境)：python3.9

编辑器(用来编写代码的软件)：pycharm

**变量**

概念：变量是计算机程序中用于存储数据的一种命名方式，可以使用变量名对数据进行存储和操作(本质上就是给数据起名)

定义变量：变量名 = 变量值

=：赋值运算符

将等号右边的值赋值给等号左边的变量

**变量名命名规则：**

1.由数字、字母、下划线组成

2.数字不能开头

3.尽量做到见名识义

4.尽量避免单独使用字母 l、o、z

5.不能使用python内置关键字作为变量名

6.严格区分大小写

7.命名规范推荐使用 大驼峰、小驼峰、下划线命名法

大驼峰命名法：变量名中每一个单词的首字母都大写，其余字母都小写

小驼峰命名法：变量名中第一个单词的首字母不大写，其余单词首字母都大写

下划线命名法：变量名中单词与单词之间使用下划线进行连接

**python内置关键字**：

['False', 'None', 'True', '\_\_peg\_parser\_\_', 'and', 'as', 'assert', 'async', 'await', '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']

**python基础数据类型**：

整型 int 整数

浮点型 float 小数

字符串 str “ ” 引号（单引号、双引号、三引号）

列表 list [ ] 由一系列元素组成的可变序列容器

元组 tuple ( ) 由一系列变量组成的不可变序列容器

集合 set { }

字典 dict { } 由一系列键值对组成的可变映射容器

布尔 只有两个值 True/False

**字符串：**

由一系列字符组成的不可变序列容器

字符：字符串中的每一个字母、数字、特殊符号、中文都称之为是一个字符

序列：有顺序的排列；支持索引和切片操作

不可变：1.没有增删改方法 2.所有的操作都不会作用于原字符串

**字符串格式化输出：**

**%格式化法：**

%s给字符串类型占位 %d给整型类型占位 %f给浮点类型占位(默认保留6位小数)

%.2f 保留2位小数

使用方式：在字符串后%(变量名1,变量名2,变量3……) 如果只有一个变量则可以省略()

**format格式化法：**

占位符：{ } 不区分数据类型

使用方式：在字符串后 .format(变量名1,变量名2,变量3……)

**f格式化法：**

占位符：{数据} 不区分数据类型

使用方式：在字符串前 f

**字符串输入：**

输出函数 print(数据)

查看变量的数据类型 type(数据)

输入函数 input("提示语") 接收过来的数据都是字符串

**索引**

语法格式：字符串[索引]

从左往右，从0开始，从右往左，从-1开始

**切片**

语法格式：字符串[起始下标:结束下标:步长]

取头不取尾，取不到结束下标对应的数据，只能取到前一位

步长不写默认为1，步长可以是负数

下标可以正负数混用

[m:]：从下标为m的位置取到字符串结束

[:n]：从字符串开头取到下标为n的前一位

[:]、[::]：取整个字符串

[::-1]：将字符串反转

**字符串方法：**

**调整类方法：**

**capitalize**

语法格式：字符串.capitalize()

作用：将字符串中第一个单词的首字母大写，其余字母全部小写

**title**

语法格式：字符串.title()

作用：将字符串中每一个单词的首字母大写，其余字母全部小写

**upper**

语法格式：字符串.upper()

作用：将字符串中所有的字母都大写

**lower**

语法格式：字符串.lower()

作用：将字符串中所有的字母都小写

**center** --中间

语法格式：字符串.center(宽度,填充物)

作用：让字符串在指定宽度中进行居中,左右空余使用指定字符进行填充；如果不指定填充物，默认用空格填充

如果指定宽度小于字符串的长度，则字符串不变

**ljust**

语法格式：字符串.ljust(宽度,填充物)

作用：让字符串在指定宽度中进行居左,右边空余使用指定字符进行填充；如果不指定填充物，默认用空格填充

如果指定宽度小于字符串的长度，则字符串不变

**rjust**

语法格式：字符串.rjust(宽度,填充物)

作用：让字符串在指定宽度中进行居右,左边空余使用指定字符进行填充；如果不指定填充物，默认用空格填充

如果指定宽度小于字符串的长度，则字符串不变

**操作类方法**：

**replace** ---替换

语法格式：字符串.replace(old,new,count)

作用：可以通过count参数指定要修改的字符数量；如果不指定count参数，默认替换所有的指定字符.

**split** ---分割

语法格式：字符串.split(字符)

作用：将字符串以指定字符进行分割，将分割之后的数据存放在列表中返回；如果不指定字符，默认以空格分割

**join** ---拼接

语法格式："字符".join(列表)

作用：将列表中的元素以指定字符拼接回字符串

**strip**

语法格式：字符串.strip(字符)

作用：去除字符串两边指定的字符；如果不指定字符，默认去除字符串两边的空格

**lstrip**  l-->left-->左

语法格式：字符串.strip(字符)

作用：去除字符串左边指定的字符；如果不指定字符，默认去除字符串左边的空格

**rstrip** r-->right-->右

语法格式：字符串.strip(字符)

作用：去除字符串右边指定的字符；如果不指定字符，默认去除字符串右边的空格

**判别类方法：**

**startswith**

语法格式：字符串.startswith(字符)

作用：判断字符串是否以指定字符开头，如果是则返回True，否则返回False

**endswith**

语法格式：字符串.endswith(字符)

作用：判断字符串是否以指定字符结尾，如果是则返回True，否则返回False

**isalpha**

语法格式：字符串.isalpha()

作用：判断字符串是否全由字母组成，如果是则返回True，否则返回False

**isalnum**

语法格式：字符串.isalnum()

作用：判断字符串是否全由字母 或 数字 或 字母和数字 组成，如果是则返回True，否则返回False

**isdigit**

语法格式：字符串.isdigit()

作用：判断字符串是否全由数字组成，如果是则返回True，否则返回False

**isupper**

语法格式：字符串.isupper()

作用：判断字符串中是否所有的字母都大写，如果是则返回True，否则返回False

**islower**

语法格式：字符串.islower()

作用：判断字符串中是否所有的字母都小写，如果是则返回True，否则返回False

**istitle**

语法格式：字符串.istitle()

作用：判断字符串中是否所有的单词首字母都大写，其余字母都小写，如果是则返回True，否则返回False

**isspace** ---空格

语法格式：字符串.isspace()

作用：判断字符串是否全由空白字符组成(空格、\n回车、\t制表符、\r回车符)

**查询类方法**：

**index** --- 索引

语法格式：字符串.index(字符,起始位置下标,结束位置下标)

作用：不指定范围,默认查询该字符在整个字符串中第一次出现的位置的下标；

如果指定范围，则找范围内第一次出现的位置的下标

如果查询到字符不存在则报错

**rindex** --- 索引

语法格式：字符串.rindex(字符,起始位置下标,结束位置下标)

作用：不指定范围,默认查询该字符在整个字符串中最后一次出现的位置的下标；

如果指定范围，则找范围内最后一次出现的位置的下标

如果查询到字符不存在则报错

**find**

语法格式：字符串.find(字符,起始位置下标,结束位置下标)

作用：不指定范围,默认查询该字符在整个字符串中第一次出现的位置的下标；

如果指定范围，则找范围内第一次出现的位置的下标

如果查询到字符不存在则返回-1

**rfind**

语法格式：字符串.rfind(字符,起始位置下标,结束位置下标)

作用：不指定范围,默认查询该字符在整个字符串中最后一次出现的位置的下标；

如果指定范围，则找范围内最后一次出现的位置的下标

如果查询到字符不存在则返回-1

**count**---数量

语法格式：字符串.count(字符,起始位置下标,结束位置下标)

作用：统计字符在字符串指定范围内出现的次数，如果字符不存在则返回0；如果不指定范围则默认在整个字符串中进行查询

**补充方法：**

**in**

语法格式：字符 in 字符串

作用：判断字符是否在字符串中，如果在则返回True,否则返回False

**len**

语法格式：len(字符串)

作用：统计字符串的长度 统计字符串中字符的个数

**循环遍历字符串的两种方式：**

**while循环**---有限循环

初始值 --- 索引的第一个值0

while 条件(判断初始值): --- 索引的范围：len(str1)-1

代码块 --- 通过索引获取字符串中的字符 字符串[索引]

步长(修改初始值) --- 索引每次加1

i = 0

while i <= len(str1)-1:

print(str1[i])

i += 1

**for循环**

for i in str1:

临时变量i接收字符串中每一个字符

print(i)

**数据类型转换**

整型 int 整数

浮点型 float 小数

字符串 str

**总结：**

将数据转换成字符串 str(数据)

将数据转换成整型 int(数据) 不涉及四舍五入，仅实现取整效果

将数据转换成浮点 float(数据)

**运算符的分类及符号：**

**算术运算符：**

+ - \* / % 取余 // 取整 \*\* 幂

python2中除法结果是整数 python3中除法结果为小数

字符串与字符串相加 将多个字符串中的数据进行拼接

字符串与整型相乘 将字符串中内容重复整型次

**复合赋值运算符：**

=：赋值运算符

+= -= \*= /= //= %= \*\*=

**比较运算符：**

最终的结果都是布尔类型值 --- 如果条件成立则返回True，否则返回False

> < >= <= == !=

**逻辑运算符：**

and 和、并且 两个条件需要同时满足

or 或者 满足其中一个即可

not 非 取相反值

**逻辑运算符的优先级：** () > not > and > or

**判断语句：**

if 条件1:

条件1成立时执行的代码

elif 条件2:

条件2成立时执行的代码

elif 条件3:

条件3成立时执行的代码

else:

以上条件都不成立时执行的代码

**关键字总结：**

**判断分支语句：**

if 如果 在一个完整的分支语句中if必须只能出现一次

elif 否则如果 elif出现的次数不限制，必须搭配if使用

else 否则 else可有可无，有且仅有一次，必须搭配if使用

**if嵌套 （一层if判断语句中再嵌套另外一层if判断语句）**

**循环：重复不断的去做某一件相同的事情**

**if和while的区别：**

while循环与if判断的语法格式基本类似

如果if后的条件成立，则代码执行一次

如果while后的条件成立，就不断重复去执行代码块，直到条件不成立

**While循环：**

**死循环：**

while 条件:

条件成立时执行的代码

条件一直成立时，就会实现死循环

**有限循环**：

初始值

while 条件(对值进行判断):

条件成立执行的代码

步长(修改初始值)---位置不一定要写在代码块下方

获取当前数据的布尔类型值 **bool(数据)**

**for循环:**

循环变量：要循环的变量

临时变量：自定义一个变量名，一般临时变量为i

for 临时变量 in 循环变量:

要执行的代码块

**for循环循环字符串：**

临时变量i接收字符串中的每一个字符，循环的次数是由字符串中字符的数量为准

**for循环循环数字--range()**

for循环循环获取数字，需要与range()方法搭配使用

range(起始值,结束值,步长)

1.取头不取尾，获取不到结束值，只能拿到结束值的前一位

2.步长不写默认为1，步长可以是负数

3.起始值不写默认从0开始

4. range()只传递一个参数--当前参数代表结束值(指定循环次数)

**while循环和for循环区别：**

while循环能实现 死循环 、有限循环

for循环只能实现有限循环

一般情况下，while循环更多使用于死循环，for循环更多使用于有限循环

**break和continue的异同：**

相同点：都用在循环里

不同点：

break 结束循环

continue 跳过本次循环，进行下一次循环

**列表**：由一系列元素组成的可变序列容器

序列：支持索引和切片操作

可变：1.有增删改方法 2.所做的所有操作都会直接作用于原列表

**定义**：

1.直接定义 [ ]

list1 = [1,2,3,4,5]

2. 列表推导式 本质通过for循环实现往列表中增加数据

语法格式：[要增加的数据 循环 条件]

3. 构造法 本质数据类型转换 强转（不能转数字） list(可迭代对象)

可迭代对象：字符串、列表

**索引**：又称之为下标、序号

语法：列表[索引]

从左往右从0开始，从右往左从-1开始

**切片**：

语法：列表[起始下标:结束下标:步长]

1.取头不取尾

2.下标可以正负数结合

3.步长不写默认为1,可以是负数

4.[n:]：从下标为n的位置取到列表结尾

5.[:m]：从列表开头取到下标为m的前一位

6.[:]、[::]：取整个列表

7.[::-1]:将整个列表进行反转

**循环遍历列表：**

**while循环**：

num = 0

while num < len(list1):

通过下标取值 列表[下标]

print(list1[num])

num += 1

**for循环**：

for i in list1:

print(i)

**列表增加数据方法：**

**append**

语法格式：列表.append(数据)

作用：将数据作为一个整体添加到列表尾部

**insert**

语法格式：列表.insert(索引,数据)

作用：将数据添加到指定下标对应的位置

**extend**

语法格式：列表.extend(可迭代对象)

作用：将可迭代对象中的数据拆分依次添加到列表尾部

**列表删除数据方法：**

**pop**

语法格式：列表.pop(索引)

作用：不指定索引，默认删除列表尾部的元素；指定索引则删除索引对应的数据；如果索引不存在则报错

**remove**

语法格式：列表.remove(数据)

作用：从列表中删除指定的数据，如果数据不存在则报错

**clear**

语法格式：列表.clear()

作用：清空列表，返回一个空列表

**del**

语法格式：del 列表 del 列表[索引]

作用：删除列表，列表不存在，打印列表会报错； 删除列表指定下表对应的数据

**列表修改数据方法：**

通过下标找到要修改的数据，直接重新赋值即可

语法格式：列表[下标] = 新值

**列表查询数据方法：**

index(元素)：

默认查询元素在整个列表种第一次出现的位置的下标

count(元素)：

统计元素在整个列表种出现的次数，如果元素不存在则返回0

**补充方法：**

**in**

语法格式：元素 in 列表

作用：判断元素是否在列表中存在，如果是则返回True，否则返回False

**len**

语法格式：len(列表)

作用：统计列表的长度 统计列表中元素的个数

**列表排序及反转：**

**sort()**

语法格式：列表.sort(reverse=True)

作用：默认将列表中的元素进行升序排序，如果使用参数reverse=True，则降序排序

**reverse()**

语法格式：列表.reverse()

作用：将列表中的元素进行反转，效果与切片中[::-1]一致

**列表嵌套：层层递找**

list1 = [1.2,6,"hello",[1,2,3,"kk",5],8,"python"]

列表修改数据 列表[下标] = 新值

**元组**

由一系列变量组成的不可变序列容器

序列：支持索引和切片操作

不可变：1.没有增删改方法 2.所有的操作都不会直接作用于原数据

**定义元组**：

定义多数据元组

tuple1 = (1,"hello",1.2,[1,2])

print(tuple1)

print(type(tuple1))

定义单数据元组 要求在单个数据后必须加逗号

如果不加逗号，则整个变量的类型与数据的类型保持一致

**索引** 元组[索引] 从左往右从0开始，从右往左从-1开始

**切片** 元组[起始下标:结束下标:步长]

1.取头不取尾

2.下标可以正负数结合

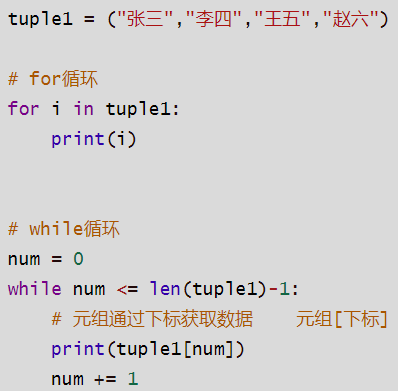
3.步长不写默认为1,可以是负数

4.[n:]：从下标为n的位置取到列表结尾

5.[:m]：从列表开头取到下标为m的前一位

6.[:]、[::]：取整个列表

7.[::-1]:将整个列表进行反转

**循环遍历元组：**

**查询数据方法：**

**index**

语法格式：元组.index(数据,开始下标,结束下标)

作用：默认查询数据在整个字符串中第一次出现的位置的下标，如果数据不存在则报错

**count**

语法格式：元组.count(数据)

作用：统计数据在元组中出现的次数。如果数据不存在则次数为0

**in**

语法格式：变量 in 元组

作用：判断变量是否在元组内存在，如果存在则返回True，否则返回False

**not in**

语法格式：变量 not in 元组

作用：判断变量是否不在元组内存在，如果确实不存在则返回True，否则返回False

**len**

语法格式：len(元组)

作用：统计元组的长度 统计元组中变量的个数

**字典：**

由一系列键值对组成的可变映射容器

键值对：键与值搭配组成键值对 "name":"张三"

映射：一一对应的关系

可变：1.有增删改方法 2.对字典所作的所有的操作会直接修改原字典

**键：**只能是不可变的数据类型 例如： 整型、浮点型、字符串、元组

**值：**没有任何要求，任何一个数据类型都可以

**定义**：

1.直接定义

dict1 = {"name":"张三","age":18,"gender":"男"}

键(key)：name、age、gender

值(value)：张三、18、男

键值对："name":"张三"、"age":18、"gender":"男"

2.构造法 本质是数据类型转换 dict(key=value)

dict2 = dict(school="希望小学",student\_c=10)

print(dict2)

**增加数据**：字典名[键] = 值

**修改数据**：字典名[键] = 值

**有则修改，无则增加**

**删除数据方法：**

**clear()**

语法格式：字典.clear()

作用：清空字典，字典仍然存在

**del**

语法格式：del 字典 del 字典[键]

作用：删除整个字典，字典不存在； 删除指定键值对，如果键不存在则报错

**查询数据方法：**

直接通过键获取值

语法格式：字典名[键名]

作用：找到键名对应的值，如果键不存在则报错

get()方式获取

语法格式：字典名.get(键名)

作用：找到键名对应的值，如果键不存在则返回None

**字典常见的操作：**

**len()**

语法格式：len(字典)

作用：统计字典的长度，统计字典中键值对的个数

**keys()**

语法格式：字典.keys()

作用：获取字典中所有的键，存放在列表中返回

**values()**

语法格式：字典.values()

作用：获取字典中所有的值，存放在列表中返回

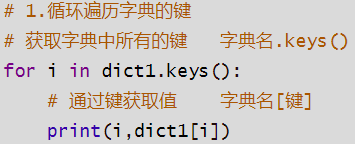
**items()**

语法格式：字典.items()

作用：获取字典中所有的键值对，将每一对键值对存放在元组中，将所有的元组存放在列表中返回

**循环遍历字典：**

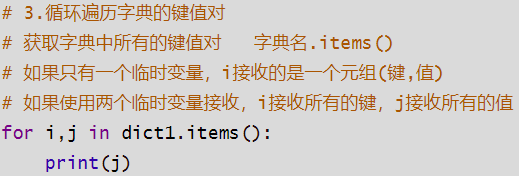
**1.循环遍历字典的键**



**2.循环遍历字典的值**



**3.循环遍历字典的键值对**



**集合：**

由一系列不可变不重复数据组成的可变容器

不可变：集合中只能存放所有不可变类型的数据：数字、字符串、元组

不重复：集合中的数据不重复，集合能够对数据进行自动去重

无序：集合中数据是无序的

可变：1.有增删方法 2.对集合进行操作都会直接作用于原集合

为什么集合没有修改方法？

列表：列表[下标] = 新值 通过下标获取要修改的数据，直接赋予新的值

字典：字典[键] = 新值 通过找到存在的键，重新给当前键赋予新的值

集合：由于集合中元素是无序的，无法获取元素，所以没有修改方法

**定义集合：**

定义有数据的集合

支持：整型、浮点型、字符串、元组

不支持：列表、字典、集合

定义空集合 构造法 set()

**增加数据方法：**

**add**

语法格式：集合.add(元素)

作用：将元素作为一个整体添加到集合中

**update**

语法格式：集合.update(可迭代对象)

作用：将可迭代对象中的数据拆分添加到集合中

**集合删除方法：**

**pop**

语法格式：集合.pop()

作用：随机删除集合中的一个数据

**discard**

语法格式：集合.discard(元素)

作用：从集合中删除指定元素；如果元素不存在则不进行删除操作，也不会报错

**remove**

语法格式：集合.remove(元素)

作用：从集合中删除指定元素；如果元素不存在则直接报错

**clear**

语法格式：集合.clear()

作用：清空集合中的元素，集合仍然存在

**集合交集与并集:**

交集 &

并集 |

**公共方法:**

+ 除数字以外，实现拼接效果(字符串、列表、元组) 字典、集合不支持+操作

\* 数据类型与数字相乘，实现重复n次的效果(字符串、列表、元组) 字典、集合不支持\*操作

in 判断指定元素是否在数据中，支持所有的数据类型(字典只能判断键是否存在)

not in 判断指定元素是否不在数据中，支持所有的数据类型(字典只能判断键是否不存在)

**python中的函数分类**

1.内置函数：python解释器提供，直接使用的函数

# 例如：del len max min

2.自定义函数：程序员根据自己程序的需求编写的函数

max(数据)：找到数据中的最大值--列表、元组、集合数字中的最大值

min(数据)：找到数据中的最小值--列表、元组、集合数字中的最小值

len(数据)：统计数据的长度，统计数据中元素的个数(支持所有的数据类型，字典统计键值对的数量)

del 数据：删除指定数据(所有的数据类型都可以使用)

**枚举函数**

enumerate(可迭代对象,start)：获取可迭代对象中的每一个元素的序号

start参数指定序号的起始值；不指定start，序号默认从0开始

一般多用于与for循环搭配使用

如果只有一个临时变量，临时变量i是一个元组(序号,元素)

如果有两个临时变量接收，临时变量i接收所有的序号，j接收所有的元素

**函数：**

将一段具有功能性的代码进行封装，封装到函数中，在需要使用时调用函数即可

优点：提高代码重复使用率，降低代码冗余量，提高代码可读性

**分类：**

1.内置函数：python解释器自带的，可以直接进行使用

max() min() len() input() print()

2.自定义函数：程序员根据自己程序的需求编写的函数

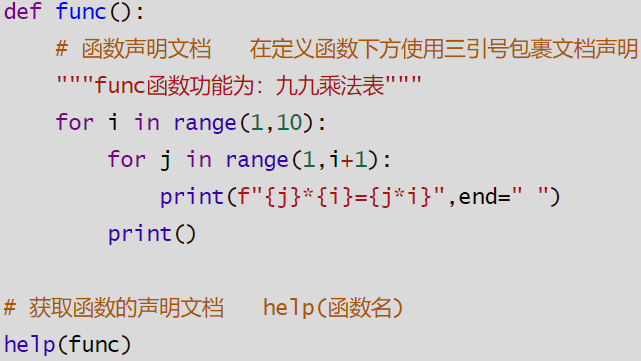
**函数的定义及调用**

def 函数名(参数)：

代码块

函数名(参数)

**函数声明文档定义及获取**



**函数参数的分类**

**函数的参数**

**形参：**形式上的参数，定义函数时传递的参数叫做形参

**实参：**实际上的参数，调用函数时传递的参数叫做实参

形参与实参数量与位置一一对应

求和函数

def func(a,b):

print(a + b)

func(10,20)

求差函数

def func1(a,b):

print(a - b)

func1(60,20)

全局变量：定义在函数外，在函数外和函数内都能使用

在函数内对全局可变数据类型数据进行操作，全局变量会随之变化

在函数内对全局不可变数据类型数据进行操作，不能进行操作

**修改全局变量**-->*不可变数据类型* **global 变量名** 换行实现修改

局部变量：定义在函数内，只能在当前函数内使用，不能在函数外使用

**参数的分类**(根据形参实参数量位置进行分类)

**位置参数**：形参与参数数量和位置一一对应的关系

**关键字参数**：在传递实参时以键=值的形式传递参数，多个关键字参数之间可以打乱顺序

**缺省参数**：在传递形参时以键=值的形式传递参数

位置参数必须放在缺省参数之前，否则报错

此时传递实参时，如果不指定该参数的值则使用默认值，指定值则使用该值

**混合使用：**

位置参数在关键字参数和缺省参数之前，否则报错

关键字参数和缺省参数之间可以打乱顺序

**不定长参数**：用户传递的实参数量不确定，无法确定需要使用多少个形参来接受数据

不定长位置参数：

\*args：\*接收所有的位置参数，将所有的数据存放在args这个元组中

不定长关键字参数：

\*\*kwargs：\*\*接收所有的关键字参数，将所有的关键字参数存放kwargs这个字典中

**return的作用**：

1.返回数据 2.结束函数

**函数的返回值**：

1.函数返回一个值 return 值

2.函数返回多个值 return 值1,值2…… 将所有的值存放在元组里返回

3.函数返回None 1.return 2.return None 3.函数中没有return

4.函数返回函数 了解--涉及闭包知识点 外函数返回内函数对象

**函数根据参数和返回值分类**

无参数无返回值

def func():

print("无参数无返回值的函数")

func()

有参数无返回值

def func1(a):

print(a\*a)

func1(10)

无参数有返回值

def func2():

return "无参数有返回值的函数"

print(func2())

有参数有返回值

def func3(a):

return a\*2

print(func3(20))

**函数嵌套调用** 在一个函数中调用另外一个函数

**拆包**：要拆的数据个数与变量的个数保持一致

**可拆包的数据类型**：元组、列表、集合、字典、字符串

对字典进行拆包，拆出来的数据是键

**模块分类：**

1.内置模块：python解释器自带的模块，提供给程序员直接进行使用的模块

2.自定义模块：程序员根据自己程序的需求编写的模块

3.第三方模块：别人编写好上传到官方的模块，如果我们想要使用，需要下载安装

**导包的两种方式：**

1.from 模块名 import 方法名 方法名()

2.import 模块名 模块名.方法名()

**正则：**

import re

match()方法

语法格式：re.match(r"正则表达式",字符串)

作用：从字符串开头开始匹配，一旦匹配成功返回一个re.match对象，匹配失败返回None

**元字符** -->单个的字符

. : 匹配除\n以外的任意一个字符

\d : 匹配数字0-9之间的任意一个

\D : 匹配除数字0-9以外的任意一个字符

\w : 匹配数字、字母、下划线、中文中的任意一个字符

\W : 匹配除数字、字母、下划线、中文以外的任意一个字符

\s : 匹配空白字符(空格、\n换行符、\t制表符、\r换行符)

\S : 匹配非空白字符

^ : 以……开头（match方法本身就是从字符串开头开始匹配，^对match方法来说没有什么作用）

$ : 以……结尾（match方法使用$频率较高，$用于正则限制字符串要求）

**字符组 []**

[2-4] : 匹配数字2-4 2 3 4

[2-47-9] : 匹配数字2-4 7-9 234 789

[0-9a-zA-Z] : 匹配所有的数字0-9字母a-zA-Z

[0-9a-zA-Z\_] : 匹配数字、字母、下划线

**量词**-->必须与元字符搭配使用，指定量词前的元字符匹配次数

\*：匹配0次或无数次 可有可无

?：匹配0次或1次 非贪婪

+：匹配1次或无数次 贪婪

**{}**

{n}：匹配n次

{n,m}：最少匹配n次，最多匹配m次

{n,}：最少匹配n次

{,m}：最多匹配m次

**贪婪匹配和非贪婪匹配**

贪婪匹配：越多越好，尽可能多的匹配字符

非贪婪匹配：越少越好，尽可能少的匹配字符

**分组与或**

分组 ()

或 |

获取正则中第n个分组中匹配成功的数据 group(n)

**正则方法**：

import re

match

语法格式：re.match(r"正则表达式",字符串)

作用：从字符串开头开始匹配；一旦匹配成功立马返回对象；匹配失败返回None

可以使用group()，匹配成功则返回匹配成功的数据；匹配失败则报错

search

语法格式：re.search(r"正则表达式",字符串)

作用：从字符串任意位置开始匹配；一旦匹配成功立马返回对象；匹配失败返回None

可以使用group()，匹配成功则返回匹配成功的数据；匹配失败则报错

findall

语法格式：re.findall(r"正则表达式",字符串)

作用：将字符串中所有符合规则的数据存放在列表中返回

**给分组起别名并使用分组**

起别名：(?P<别名>正则表达式)

使用分组：(?P=别名)

简单的方式：使用分组的序号

默认分组按照序号排序：第一个分组序号为1 第n个分组序号为n

使用分组：使用第一个分组 \1 使用第n个分组 \n

**文件**

操作文件的流程：

1.打开文件

2.对文件内容进行操作(读、写)

3.关闭文件

**打开文件**：

第一种方式：直接使用**open()方法**

open(file,mode,encoding):

file:指定要打开的文件(文件名、文件路径)

mode：指定文件的打开模式 r-read只读 w-write只写 a-add追加

encoding:指定编码方式 utf-8 如果不指定，读取中文，写入中文都会有问题

当以只读的模式打开文件，mode参数可以省略，r是文件的默认访问方式

打开文件的第二种方式：**with语句**

推荐使用with语句打开文件

优势：不需要用户手动关闭文件句柄，一旦跳出with语句，文件会自动关闭

with open(file,mode,encoding="utf-8") as 文件对象:

对文件进行操作

**关闭文件**：文件对象.close()

**读取文件内容方法**:

read()

语法格式：文件对象.read(n)

作用：不指定参数n，默认读取文件中所有的内容;指定参数n，从文件中读取n个字符(换行也属于一个字符)

readline()

语法格式：文件对象.readline()

作用:依次读取一行数据，如果读取到空字符串，说明文件没有内容了

readlines()

语法格式：文件对象.readlines()

作用：以列表的形式读取文件中所有的内容，列表的每一项是文件的每一行

**往文件中写入内容**:

w模式：如果文件不存在，则创建文件；如果已存在的文件中有数据，则覆盖 w-write只写

a模式：如果文件不存在，则创建文件；如果文件中有内容，则往后追加数据 a-add追加

往文件写入数据的方式

文件对象.write(数据)

要求数据必须是字符串类型的数据

**文件操作方法**:

import os

rename

语法格式：os.rename(old,new)

作用：给文件重命名

remove

语法格式：os.remove(文件路径)

作用：删除指定文件

**对文件夹进行操作**:

mkdir mk->make制作 dir-->direction目录、文件夹

语法格式：os.mkdir(文件夹)

作用：创建单级空目录

rmdir rm->remove删除 dir目录

语法格式：os.rmdir(文件夹)

作用：删除单级空目录

listdir

语法格式：os.listdir(文件夹)

作用：获取当前文件夹下所有的子文件(包括隐藏文件)，以列表的形式返回

getcwd

语法格式：os.getcwd()

作用：获取当前文件的父级目录路径

拷贝文件

**定义类**

函数首字母小写，类首字母大写--->类名推荐使用大驼峰命名法

大驼峰命名法：每一个单词的首字母都大写，其余字母都小写

是否继承父类需要根绝程序需求决定-->如果没有要继承的父类，类名后的()可以省略

object在类中-->顶级父类

定义类

将具有相同特性和行为的事物抽象出一个类

特征即属性 行为即方法

在类中编写实例方法

以函数的形式定义方法---参数中默认携带self

self-->实例对象本身

创建对象-->实例化对象 对象名 = 类名()

打印不同的对象，获取不同的内存地址

实例化对象可以调用实例方法 实例对象名.实例方法名()

给对象增加实例属性:

对象名.属性名 = 属性值 print(实例对象名.属性名)

定义\_\_init\_\_初始化方法:给实例对象增加实例属性

在实例方法中获取使用实例属性 需要使用self.属性名

**魔法方法**:

**\_\_init\_\_**

定义\_\_init\_\_初始化方法:给实例对象增加实例属性

init方法调用：当对象一旦创建成功会立马调用init

在\_\_init\_\_中定义实例属性： self.属性名 = 属性值

由于每一个对象的属性都不一样，所以属性值不固定，通过传参指定

\_\_init\_\_方法的形参需要在类名()传递实参

**\_\_new\_\_**

new方法作用：创建对象，给实例对象开辟内存空间(空间用来存放当前对象的属性)

如果new方法执行成功，说明当前对象创建成功(对象的内存空间开辟成功)

init方法作用：给对象增加实例属性(将对象的属性增加到创建好的空间中)

所以new方法在init前被执行，只有new执行成功，对象创建成功才执行init

**\_\_str\_\_**

被调用：print(对象名)

不写str方法，打印对象获取对象的内存地址；指定str返回值，打印对象获取str的返回值

要求：str方法需要返回一个字符串 一般用于返回对象描述信息

没有编写str方法时，打印的数据为：对象的内存地址

编写完str方法以后，打印的数据为：str方法中指定的对象描述信息

\_\_ del \_\_

被调用：当销毁对象时调用当前方法

当类及类中方法、对象操作都执行结束以后；对象被销毁

手动删除对象时，del方法会被调用

当程序执行结束以后，对象就会被视为垃圾，自动被销毁，del方法会被调用

**P1理论考试**

**break和continue的异同**

相同点：都用在循环里

不同点： break:结束循环

coutinue:跳过本次循环，进行下一次循环

**列举5个python内置关键字**

for、if、elif、else、as、False、True、while、and、or、coutinue、def、class、break、from

**字符串方法及作用(至少五个)？**

字符串.capitalize()：将字符串中第一个单词的首字母大写，其余字母全部小写

字符串.title()：将字符串中每一个单词的首字母大写，其余字母全部小写

字符串.upper()：将字符串中所有的字母都大写

字符串.lower()：将字符串中所有的字母都小写

字符串.center(宽度，填充物)：让字符串在指定宽度中进行居中,左右空余使用指定字符进行填充；如果不指定填充物，默认用空格填充；如果指定宽度小于字符串的长度，则字符串不变

字符串.ljust(宽度，填充物): 让字符串在指定宽度中进行居左,右边空余使用指定字符进行填充；如果不指定填充物，默认用空格填充；如果指定宽度小于字符串的长度，则字符串不变

字符串.rjust(宽度，填充物): 让字符串在指定宽度中进行居右,左边空余使用指定字符进行填充；如果不指定填充物，默认用空格填充；如果指定宽度小于字符串的长度，则字符串不变

字符串.replace(old,new,count): 可以通过count参数指定要修改的字符数量；如果不指定count参数，默认替换所有的指定字符

字符串.split(字符)：将字符串以指定字符进行分割，将分割之后的数据存放在列表中返回；如果不指定字符，默认以空格分割

“字符”.join(列表)：将列表中的元素以指定字符拼接回字符串

字符串.strip(字符)：去除字符串两边指定的字符；如果不指定字符，默认去除字符串两边的空格

字符串.lstrip(字符)：去除字符串左边指定的字符；如果不指定字符，默认去除字符串左边的空格

字符串.rstrip(字符)：去除字符串右边指定的字符；如果不指定字符，默认去除字符串右边的空格

字符串.startswith(字符)：判断字符串是否以指定字符开头，如果是则返回True，否则返回False

字符串.endswith(字符)：判断字符串是否以指定字符结尾，如果是则返回True，否则返回False

字符串.isalpha()：判断字符串是否全由字母组成，如果是则返回True，否则返回False

字符串.isalnum()：判断字符串是否全由字母 或 数字 或 字母和数字组成，如果是则返回True，否则返回False

字符串.isdigit()：判断字符串是否全由数字组成，如果是则返回True，否则返回False

字符串.isupper()：判断字符串中是否所有的字母都大写，如果是则返回True，否则返回False

字符串.islower()：判断字符串中是否所有的字母都小写，如果是则返回True，否则返回False

字符串.istitle()：判断字符串中是否所有的单词首字母都大写，其余字母都小写，如果是则返回True，否则返回False

字符串.isspace()：判断字符串是否全由空白字符组成(空格、\n回车、\t制表符、\r回车符)

字符串.index(字符,起始位置下标,结束位置下标)：不指定范围,默认查询该字符在整个字符串中第一次出现的位置的下标；如果指定范围，则找范围内第一次出现的位置的下标；如果查询到字符不存在则报错。

字符串.rindex(字符,起始位置下标,结束位置下标)：不指定范围,默认查询该字符在整个字符串中最后一次出现的位置的下标；如果指定范围，则找范围内最后一次出现的位置的下标；如果查询到字符不存在则报错。

字符串.find(字符,起始位置下标,结束位置下标)：不指定范围,默认查询该字符在整个字符串中第一次出现的位置的下标；如果指定范围，则找范围内第一次出现的位置的下标；如果查询到字符不存在则返回-1

字符串.rfind(字符,起始位置下标,结束位置下标)：不指定范围,默认查询该字符在整个字符串中最后一次出现的位置的下标；如果指定范围，则找范围内最后一次出现的位置的下标；如果查询到字符不存在则返回-1。

字符串.count(字符,起始位置下标,结束位置下标)：统计字符在字符串指定范围内出现的次数，如果字符不存在则返回0；如果不指定范围则默认在整个字符串中进行查询。

**列表增加数据的三种方式？**

append():将数据作为一个整体添加到列表尾部

insert():将数据作为一个整体添加到指定下标位置

extend():将数据进行拆分依次添加到列表尾部

**循环遍历字典的三种方式？**

① 循环遍历字典的键：

for i in dict.keys():

print(i)

② 循环遍历字典的值：

for i in dict.values():

print(i)

③ 循环遍历字典的键值对：

for i,j in dict.items():

print(I,j)

**对全局变量和局部变量的理解？**

全局变量：定义在函数外，在函数内和函数外均可使用

局部变量：定义在函数内，在函数内可以使用，函数外不可使用

**对不定长参数的理解？**

不定长位置参数：（\*args）接受所有的不定长位置参数，将所有的位置参数存放在args元组中

不定长关键字参数：（\*kwargs）接收所有的不定长关键字参数，将所有的关键字参数存放在kwargs字典中

**return的作用？**

函数的返回值、结束函数

**导包的两种方式**

① import 模块名 模块名.方法名.（）

② from 模块名 import 方法名 方法名.（）

**正则的三个方法及作用**

① match():从字符串开头开始匹配，一旦匹配成功立马返回对象，否则返回None

② search():从字符串任意位置开始匹配，一旦匹配成功立马返回对象，否则返回None

③ findall():找出字符串中所有符合条件的数据，存放在列表中

**文件操作的理解？**

open():打开文件

close():关闭文件

write():给文件中写入数据

read(num):读取文件中num个字节的数据；如果未指定num，则默认读取整个文件的数据

readline():按行读取文件内容，一次读取一行，如果读到空字符串说明文件所有的内容读取完毕

readlines():以列表的形式读取文件所有内容，列表中的每一项是文件内容的每一行

**self和cls的区别**

self:实例对象本身

cls:类对象本身

**面向对象三要素：**

①继承：子类继承父类，子类就拥有了父类的属性和方法，提高了代码的重用性，方便代码维护更新

②封装：隐藏对象的属性和实现细节，进对外提供公共访问方式，通过私有方法和私有属性实现封装

③多态：不同的派生类调用基类用相同的方法产生不同的效果。多态的两个必要条件：继承、方法重写

**python垃圾回收机制**

以引用计数为主，标记清除、分代回收为辅

**对引用计数的理解？**

每一个对象都会维护一个ob\_ref，用来记录当前对象的被引用次数

当对象的被引用次数为0时，当前对象会被视为垃圾进行回收

对象被创建、被引用、被当作元素使用、被当作参数使用，都会使当前对象的被引用次数+1

当对象的值被改变、不做参数、元素使用时，都会让当前对象的被引用次数-1

**标记清除的理解？**

分为两个阶段：

① 标记阶段：将所有的【活动对象】打上标记

② 清除阶段：将所有未打上标记的【非活动对象】作为垃圾进行清除回收

**四大高阶函数及其作用？**

map: map(函数名，可迭代对象)

将函数依次作用于可迭代对象中的每一个数据

返回结果是一个迭代器

可以通过for循环、list强转、tuple强转获取数据

reduce: reduce(函数名，可迭代对象)

使用时需要导包，from functools import reduce

对可迭代对象中的数据进行累积操作，最终结果是一个具体的值

filter: filter(函数名，可迭代对象)

过滤获取可迭代对象中满足条件的数据，返回的结果是一个迭代器

sorted:

对可迭代对象中的数据进行排序

不指定参数默认升序排序；指定参数reverse = True实现降序排序

**迭代器的功能？**

返回可迭代对象中的数据

记录当前迭代位置

迭代完成后再次迭代抛出停止迭代（StopIteration）的异常

**闭包定义**

内外函数嵌套

内函数引用外函数作用域下的非全局变量

外函数返回内函数对象

**装饰器定义**

在不改变原函数的基础上，给原函数增加新的功能

利用闭包的思路，本质上是一个闭包内函数

break和continue的异同

列举5个python内置关键字

字符串方法及作用(至少五个)？

列表增加数据的三种方式？

循环遍历字典的三种方式？

对全局变量和局部变量的理解？

对不定长参数的理解？

return的作用？

导包的两种方式

正则的三个方法及作用:

文件操作的理解？

self和cls的区别

面向对象三要素：

python垃圾回收机制：

对引用计数的理解？

标记清除的理解？

四大高阶函数及其作用？

迭代器的功能？

闭包定义

装饰器定义