### Chap2

ipo程序编写方法：

input, process, output

print函数

语法结构：print(输出内容)

chr(): 将数字转换为字符（ASCII码或unicode码）

ord(): 将字符转换为数字（转换为ASCII码或unicode码均可）

完整语句：print(value,…,sep=’’,end=’\n’,file=None)

Input函数

X=input(‘提示文字’)

无论输入的数据是什么类型，x的数据类型都是字符串

注释：

中文文档声明注释一定要写在第一行 coding=utf-8

多行注释用三引号

### Chap3 数据类型和运算符

保留字：

35个，严格区分大小写

标识符：

字符、数字、下划线，第一个字符不能是数字

不能使用保留字

下划线开头的标识符有特殊意义

常量命名全部用大写，可以使用下划线

变量：

变量名=value (等号是赋值运算符)

动态语言：类型可以变化

多个变量指向同一个值 no=number=1024 (内存地址相同)

常量：

程序运行中不允许改变的值

所有字母都大写

数据类型：

数值型：整型int：没有小数部分

十进制

二进制（0b或0B）0b10101

八进制（0o或0O）0o776

十六进制（0x或0X）0x987A

浮点型float：有小数点的数值

科学计数法表示 x=1.99E3

不确定的尾数问题

Round(a+b,1) 保留一位小数

复数：实部.real 虚部.imag

字符串型：转义字符：\n: 换行 \t: 水平制表符 \”: 双引号 \’: 单引号 [\\：一个反斜杠](file:///\\：一个反斜杠)

原字符：使转义字符失效，r或R （会保留\n，写在最前面）

索引：正向递增0-9 反向递减(-10)-(-1)

切片：字符串[n:m] 不包含m

操作符：x+y 将两个字符串连接

x\*n 字符串复制n次

x in s 如果x是s的子串，记过为True

布尔型：True为1， False为假

为False的情况：False或None

数值中的0,0.0，虚数0

空序列

自定义对象的实例

数据类型之间的转换：

隐式转换

显示转换：int(x), float(x), str(x), chr(x), ord(x), hex(x)（转成十六进制字符串）,oct(x)（八进制）, bin(x)（二进制）

Int(‘3.14’) 会报错因为字符串3.14不是整数，str转成int只有字符串里是整数才可以

Eval函数：

去掉字符串最外侧的引号

变量=eval(字符串)

算术运算符：

/ 除法 // 整除 % 取余 \*\*幂乘

赋值运算符：

+= x+=y x=x+y

a,b=b,a

比较运算符：

！=不等于

逻辑运算符：

And与 or或 not非（对与取反）

按位与运算：&

按位或运算：|

按位异或运算：^ 不同为1 相同为0

按位取反运算：~

移位运算：<<2 向左移动两位，溢出的舍弃，不足的补0， 移动一位\*2

>>2 向右移动两位，溢出的舍弃，不足的原来是0补0是1补1，移动一位//2

优先级：

幂运算>取反，正负号>算术运算符>位运算的移位>按位与>异或>按位或>比较>赋值

### Chap4 程序的流程控制

顺序结构：按语句自然顺序从上到下依次执行

选择结构：

单分支结构：if 表达式：

语句块

（true执行语句块，false跳过）

如果语句块只有一句代码，可以将语句块直接写在冒号后面

双分支结构：if 表达式:

语句1

Else:

语句2

多分支结构：if 表达式1：

语句块1

Elif 表达式2：

语句块2

Else：

语句块3

嵌套if：if 表达式1：

If 表达式2：

语句块1

Else：

语句块2

Else：

If表达式3：

语句块4

Else：

语句块4

选择结构：

使用and连接多个条件，只有全部满足才会执行if后面的语句块

使用or连接多个条件，只要有一个正确就会执行

循环结构：

遍历循环结构for：for 循环变量in 遍历对象:

语句块1

Else：

语句块2

只有for正常进行完（没有break）才会执行else

无限循环结构while：while 表达式：

语句块1

Else：

语句块2

表达式结果为false不再执行

程序跳转语句：

While 表达式1：

执行代码

If 表达式2：

Break（结束执行）

While 表达式1：

执行代码

If 表达式2：

Continue（跳过本次循环的后续代码）

空语句

起到占位符的作用

### Chap5 组合数据类型

数据容器：

可以容纳多分数据的数据类型

列表list，元组tuple，字符串str，集合set，字典dict

序列

用于存储多个值的连续空间，每个值都对应一个整数的编号，称为索引

正向递增：0~N-1

反向递减：-N~-1

切片：[strat:end:step] （不包含end，步长默认为1）

相加相乘

（’X’ in s），(‘x’ not in s)

s.index(x) 序列s中第一次出现x的位置

s.count(x) 序列s中出现x的总次数

列表：一系列的按特定顺序排列的元素组成

内置的可变序列，用[ ]定义，元素之间用逗号分开

创建方式：1. 直接用[ ] 列表名=[element1, element2…]

2. 使用内置函数List: 列表名=list(序列)

列表的删除 del列表名

列表的下标索引：print(lst [0]) 取出对应的元素

Enumerate函数：

For index,item in enumerate(lst):

Lst.append(x) 最后增加一个元素

Lst.insert(index,x) 在第index位置增加一个元素

Lst.clear() 清除所有元素

Lst.pop(index) 将第index位置的元素取出并删除

Lst.remove(x) 将列表中出现的第一个元素x删除

Lst.reverse(x) 将列表中的元素反转

Lst.copy() 拷贝列表中所有元素，生成一个新列表

列表排序：1. 列表对象sort

Lst.sort( key=None, reverse=False) key是排序规则，reverse是排序方式，默认升序

False是升序，true是降序

2. 内置函数sorted()

Sorted(iterable, key=None,reverse=False) iterable是排序对象

列表生成式：

Lst=[expression for item in range]

Lst=[expression for item in range if condition]

二维列表：

遍历：for row in 二维列表:

For item in row:

Pass

元组：

不可变序列，用()定义，用逗号分隔

只有一个元素的时候逗号也不能省略

创建方式：1. 使用()直接创建

元组名=(element1,element1,…)

2. 使用内置函数tuple()

元组名=tuple(序列)

删除元组: del 元组名

字典类型：

是可变数据类型

Key是不可变序列

根据一个信息查找另一个信息的方式构成了“键值对”，表示索引用的键key和对应的值value构成的成对关系

创建方式：1. 使用{}直接创建

D={key1:value1, key2:value2, …} Dict(key1=value1, key2=value2, …)

2. 使用内置函数dict()

1) 映射 z=zip(lst1,lst2) d=dict(z)

字典中的key是无序的

取值：d[key]或d.get(key)

遍历：1. 遍历出key与value的元组：for element in d.items():

2. 分别遍历出key和value：for key,value in d.items():

字典相关操作：

d.keys() 获取所有的key数据

d.values.() 获取所有的value数据

d,pop(key,default) key存在获取相应的value，同时删除key-value对，否则获取默认值

d.popitem() 随机从字典中取出一个key-value对，结果为元组类型，同时将对删除

d,clear() 清空字典中所有的key-value对

字典的生成式：

D={ key:value for item in range}

D={ key:value for key,value in zip(lst1,lst2)}

集合：

无序的不重复的元素序列

只能储存不可变数据类型，用{}定义，不存储列表字典

是可变数据类型

常见方式：1. 使用{}直接创建

S={element1,element2,…}

2. 使用内置函数set()

S=set(可迭代对象)

交集 A&B

并集 A|B

差集 A-B A中去掉B也含有的元素剩下的元素

补集 A^B 交集去掉AB都有的元素

s.add(x) 如果x不在集合s中则添加

s.remove(x) 如果x在集合中则删除，如果不在集合中，程序报错

s.clear() 清除集合中所有元素

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据类型 | 序列类型 | 元素是否可重复 | 是否有序 | 定义符号 |
| 列表list | 可变序列 | 可重复 | 有序 | [ ] |
| 元组tuple | 不可变序列 | 可重复 | 有序 | ( ) |
| 字典dict | 可变序列 | Key不可重复  Value可重复 | 无序 | {key:value } |
| 集合set | 可变序列 | 不可重复 | 无序 | { } |

### Chap6 字符串及正则表达式

字符串：

不可变数据类型

Str.lower() 将str字符串全部转成小写字母，结果为一个新的字符串

Str.upper() 将str字符串全部转成大写字母，结果为一个新的字符串

Str.split(sep=None) 把str按照指定的分隔符sep进行分隔，结果为列表

Str.count(sub) 结果为sub这个字符串在str中出现的次数

Str.find(sub) 查询sub是否存在，不存在结果为-1，如果存在结果为sub首次出现的索引

Str.index(sub) 与find功能相同，如果不存在会报错

Str.startswith(s) 查询str字符串是否以子串s开头

Str.endswith(s) 查询str字符串是否以子串s结尾

Str.replace(old,news) 使用news替换字符串中所有的old字符，结果是一个新的字符串

Str.center(width,fillchar) 字符串str在指定的宽度范围内居中，可以使用fillchar填充

Str.join(iter) 在iter中的每个元素的后面都增加一个新的字符串str

Str.strip(chars) 在字符串中去掉左侧和右侧chars中列出的字符串

Str.lstrip(chars) 从字符串中去掉左侧chars中列出的字符串

Str.rstrip(chars) 从字符串中去掉右侧chars列出的字符串

格式化字符串的三种方式： (连接各种数据类型)

占位符：%s 字符串格式 %d 十进制整数格式 %f 浮点数格式

f-string {}标明被替换的字符

str.format() 模版字符串.format(逗号分隔的参数)

: 填充 对齐方式 宽度 ， .精度 类型

字符串的编码和解码：

Str ---> byte ---> str

编码：str.encode(encoding=’utf-8’, errors=’strict/ignore/replace’)

解码：bytes.decode(encoding=’utf-8’, errors=’strict/ignore/replace’)

数据的验证

程序对用户输入的数据进行合法性验证

Str.isdigit() 所有字符都是数字（阿拉伯数字）

Str.isnumeric 所有字符都是数字

Str.isalpha 所有字符都是字母（包含中文字符）

Str.isalnum() 所有字符都是数字或字母 （包含中文字符）

Str.islower() 小写

Str.isupper() 大写

Str.istitle() 首字母大写

Str.isspace() 空白字符（\n,\t等）

数据的处理

字符串的拼接：1. Str.join() 2. 直接拼接 3. 使用格式化字符串拼接

正则表达式

元字符：具有特殊意义的专用字符

^ 匹配的开始

$ 匹配的结束

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| . | \w | \W | \s | \S | \d |
| 任意字符除\n | 字母数字下划线 | 非字母数字下划线 | 任意空白字符 | 任意非空白字符 | 任意十进制数 |

空白字符：转义字符等

限定符：用于限定匹配的次数 （匹配前面的字符）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ? | + | \* | {n} | {n,} | {n,m} |
| 0次或1次 | 1次或多次 | 0次或多次 | N次 | 最少n次 | n-m次 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 区间字符[ ] | 排除字符^ | 选择字符| | 转义字符 | [\u4e00-\u9fa5] | 分组（） |
| [ ]中指定的字符 | 不在[ ]中指定的字符 | |左右的任意字符 | 同转义字符 | 任意一个汉字 | 改变作用范围 |

Ex：colou+r colour/ colouu…r

Re模块：实现正则表达操作

Re.match(pattern,string,flags=0)

从开始位置进行匹配，如果起始位置匹配成功，结果为match对象否则为none

Re.search(pattern,string,flags=0)

搜索第一个匹配的值，如果匹配成功，结果为match对象否则为none

Re.findall(pattern,string,flags=0)

搜索所有符合正则表达式的值，结果是一个列表

Re.sub(pattern,repl,string,count,flags=0) 对字符串中制定子串的替换

Re.split(pattern,string,maxsplit,flags=0) 分隔字符串

### Chap7 异常处理

Bug：检测并排除计算机程序/机器的故障

异常处理机制：

Try…except: try: 可能会抛出异常的代码

Except 异常类型：异常处理代码（报错后执行的代码）

Except 异常类型B：异常处理代码（报错后执行的代码）

能捕获最多异常类型的写在最后

Try…except…else: try: 可能会抛出异常的代码

Except 异常类型：异常处理代码（报错后执行的代码）

Else: 没有抛异常要执行的代码

Try…except…else…finally: try: 可能会抛出异常的代码

Except 异常类型：异常处理代码（报错后执行的代码）

Else: 没有抛异常要执行的代码

Finally：无论是否产生异常都要执行的代码

Raise: 抛出一个异常，从而提醒程序出现了异常情况，程序能够正确的处理这些异常情况

Raise [ Exception类型 (异常描述信息)]

常见的异常类型：

ZeroDivisionError: 除数为0

IndexError: 索引超出范围

KeyError: 字典取值是key不存在

NameError: 使用一个没有声明的变量

SyntaxError: Python中的语法错误

ValueError: 传入的值错误

AttributeError: 属性或方法不存在

TypeError: 类型不合适

IndentationError: 不正确的缩进

Pycharm调试：

设置断点，进入调试窗口

### Chap8 函数及常用的内置函数

自定义函数：

Def 函数名称 （参数列表）：

函数体

[return 返回值列表] (非必须部分)

函数的参数传递

位置参数：调用时的参数个数和顺序必须与定义的参数个数和顺序相同

关键字参数：使用“形参名称=值”的方式进行传参，顺序可以不同

默认值参数：直接对形参进行赋值，在调用时如果该参数不传值，将使用默认值

（遵循位置参数在前，关键字参数/默认值参数在后）

可变参数：个数可变的位置参数（\*para）（可以接收任意个数的实际参数）

个数可变的关键字参数（\*\*para）（可以接收多个“参数=值”形式的参数）

函数的返回值return：

如果函数的运行结果需要再其他函数中使用，那么这个函数就应该被定义为带返回值的函数

Return可以出现在函数中的任意一个位置，用于结束函数

返回值可以是一个值或多个值

变量的作用域：

变量起作用的范围

局部变量：在函数定义处的参数和函数内部定义的变量，仅在函数内部作用

全局变量：在函数外定义的变量或函数内部使用global关键字修饰的变量，整个程序作用

匿名函数lambda：

没有名字的函数，只能使用一次，一般是在函数的函数体只有一句代码且只有一个返回值时用匿名函数来简化

Result=lambda 参数列表：表达式

递归函数：

在一个函数的函数体内调用该函数本身

一部分是递归调用，一部分是递归终止条件，一般用if-else