Python基础语法

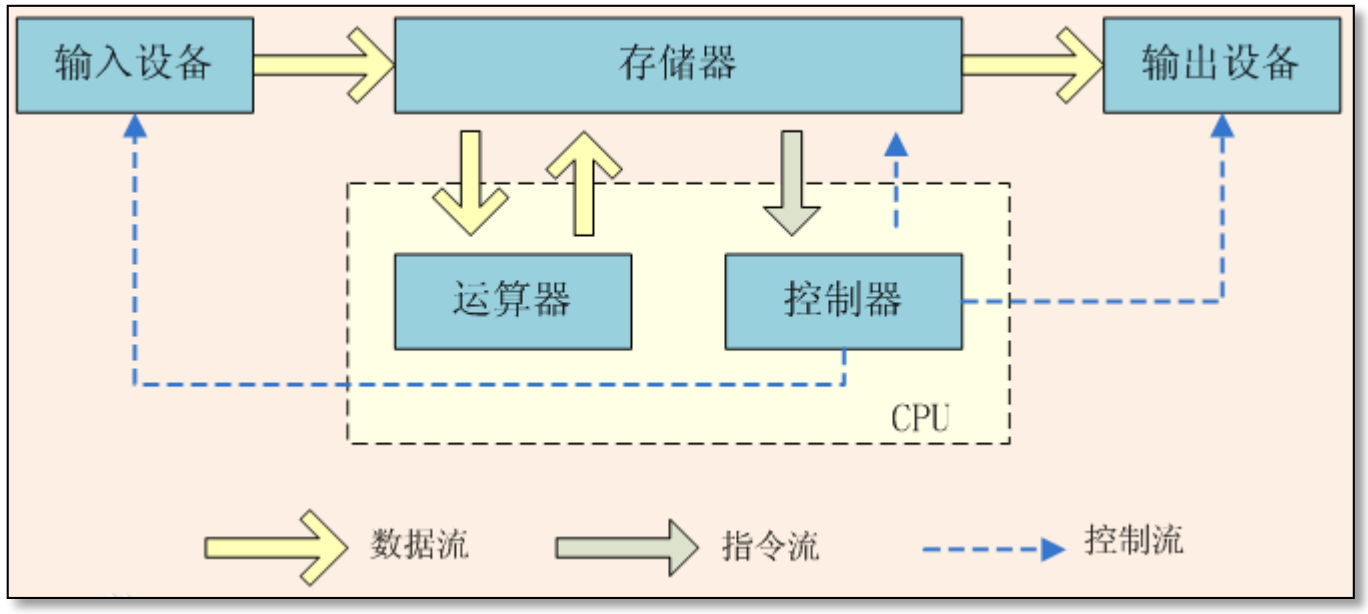
讲师：Wayne

从业十余载，漫漫求知路

编程基础

- 程序
  - 一组能让计算机识别和执行的指令
- 电子计算机
  - 能够执行程序的机器
- 现代计算机
  - 艾伦·麦席森·图灵（Alan Mathison Turing，1912年6月23日 - 1954年6月7日），英国数学家、逻辑学家，被称为计算机科学之父，人工智能之父。图灵提出的著名的图灵机模型为现代计算机的逻辑工作方式奠定了基础
  - 冯·诺依曼著名匈牙利裔美籍犹太人数学家、计算机科学家、物理学家和化学家 ，数字计算机之父。他提出了以二进制作为数字计算机的数制基础，计算机应该按照程序顺序执行，计算机应该有五大部件。

冯诺依曼体系架构



- CPU由运算器和控制器组成
- 运算器，完成各种算数运算、逻辑运算、数据传输等数据加工处理
- 控制器，控制程序的执行
- 存储器，用于记忆程序和数据，例如内存
- 输入设备，将数据或者程序输入到计算机中，例如键盘、鼠标
- 输出设备，将数据或程序的处理结果展示给用户，例如显示器、打印机等

- CPU中还有寄存器和多级缓存Cache

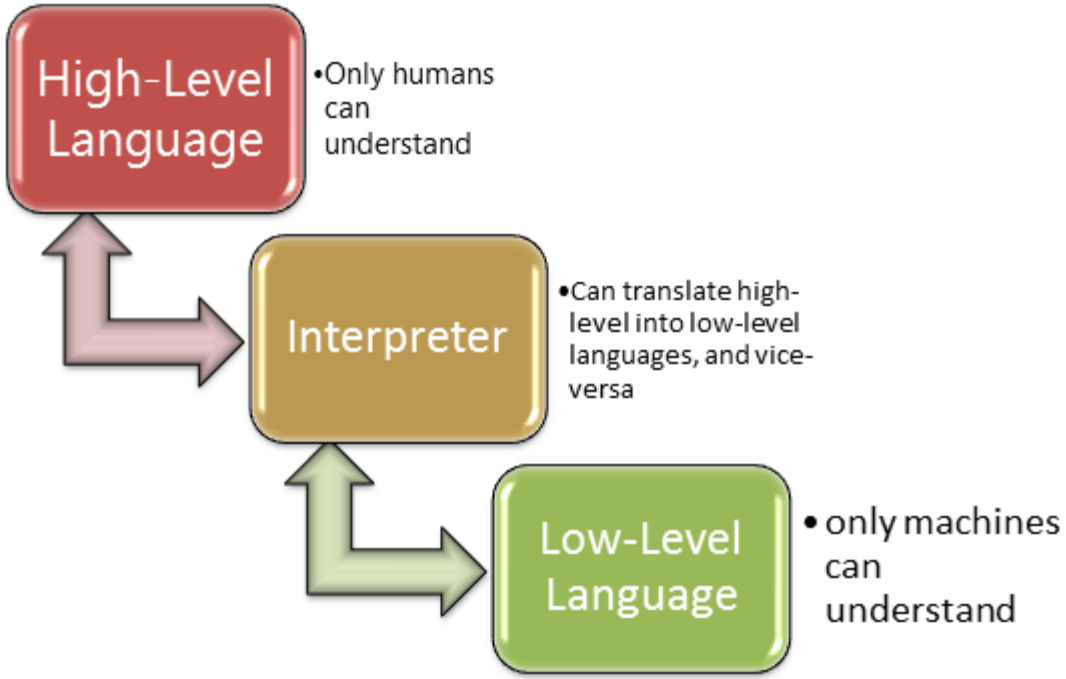
编程基础

- 计算机语言
  - 人与计算机之间交互的语言
- 机器语言
  - 一定位数组成二进制的0和1的序列，称为机器指令。机器指令的集合就是机器语言
  - 与自然语言差异太大，难学、难懂、难写、难记、难查错
- 汇编语言
  - 用一些助记符号替代机器指令，称为汇编语言。ADD A,B 指的是将寄存器A的数与寄存器B的数相加得到的数放到寄存器A中
  - 汇编语言写好的程序需要汇编程序转换成机器指令
  - 汇编语言只是稍微好记了些，可以认为就是机器指令对应的助记符。只是符号本身接近自然语言

语言分类

- 低级语言
  - 面向机器的语言，包括机器语言、汇编语言
  - 不同的机器不能通用，不同的机器需要不同的机器指令或者汇编程序
- 高级语言
  - 接近自然语言和数学语言的计算机语言
  - 高级语言首先要书写源程序，通过编译程序把源程序转换成机器指令的程序
  - 1954年正式发布的Fortran语言是最早的高级语言，本意是公式翻译
  - 人们只需要关心怎么书写源程序，针对不同机器的编译的事交给编译器关心处理

低级语言到高级语言



- 语言越高级，越接近人类的自然语言和数学语言
- 语言越低级，越能让机器理解
- 高级语言和低级语言之间需要一个转换的工具：编译器、解释器

- C、C++等语言的源代码需要本地编译
- Java、Python、C#的源代码需要被解释器编译成中间代码（Bytecode），在虚拟机上运行

- 编译语言，把源代码转换成目标机器的CPU指令
- 解释语言，解释后转换成字节码，运行在虚拟机上，解释器执行中间代码

高级语言的发展

- 非结构化语言
  - 编号或标签、GOTO，子程序可以有多个入口和出口
  - 有分支、循环
- 结构化语言
  - 任何基本结构只允许是唯一入口和唯一出口
  - 顺序、分支、循环，废弃GOTO
- 面向对象语言
  - 更加接近人类认知世界的方式，万事万物抽象成对象，对象间关系抽象成类和继承
  - 封装、继承、多态
- 函数式语言
  - 古老的编程范式，应用在数学计算、并行处理的场景。引入到了很多现代高级语言中
  - 函数是“一等公民”，高阶函数

程序Program

- 程序
  - 算法 + 数据结构 = 程序
  - 数据一切程序的核心
  - 数据结构是数据在计算机中的类型和组织方式
  - 算法是处理数据的方式，算法有优劣之分

- 写程序难点
  - 理不清数据
  - 搞不清处理方法
  - 无法把数据设计转换成数据结构，无法把处理方法转换成算法
  - 无法用设计范式来进行程序设计
  - 世间程序皆有bug，但不会debug

Python解释器

- 官方CPython
  - C语言开发，最广泛的Python解释器
- IPython
  - 一个交互式、功能增强的Cpython
- PyPy
  - Python语言写的Python解释器，JIT技术，动态编译Python代码
- Jython
  - Python的源代码编译成Java的字节码，跑在JVM上
- IronPython
  - 与Jython类似，运行在.Net平台上的解释器，Python代码被编译成.Net的字节码

Python基础语法

- 注释—— # 标注的文本
- 数字
  - 整数
    - Python3开始不区分long和int，long被重命名为int，所以只有int了
    - 进制0xa、0o10、0b10
    - bool，2个值True、False
  - 浮点数
    - 1.2、3.1415、-0.12，1.46e9等价于$1.46*10^9$
    - 本质上使用了C语言的double类型
  - 复数，1+2j

Python基础语法

- 字符串
  - 使用 ' " 单双引号引用的字符的序列
  - '''和""" 单双三引号，可以跨行、可以在其中自由的使用单双引号
  - 在字符串前面加上r或者R前缀，表示该字符串不做特殊的处理
  - 3.6版本开始，新增f前缀，格式化字符串

基础语法

- 转义序列
  - \\ \t \r \n \' \"
  - 前缀r，把里面的所有字符当普通字符对待
- 缩进
  - 未使用C等语言的花括号，而是采用缩进的方式表示层次关系
  - 约定使用4个空格缩进
- 续行
  - 在行尾使用 \
  - 如果使用各种括号，认为括号内是一个整体，内部跨行不用 \

基础语法

- 标识符
  1. 一个名字，用来指代一个值
  2. 只能是字母、下划线和数字
  3. 只能以字母或下划线开头
  4. 不能是python的关键字，例如def、class就不能作为标识符
  5. Python是大小写敏感的

  约定：

  不允许使用中文

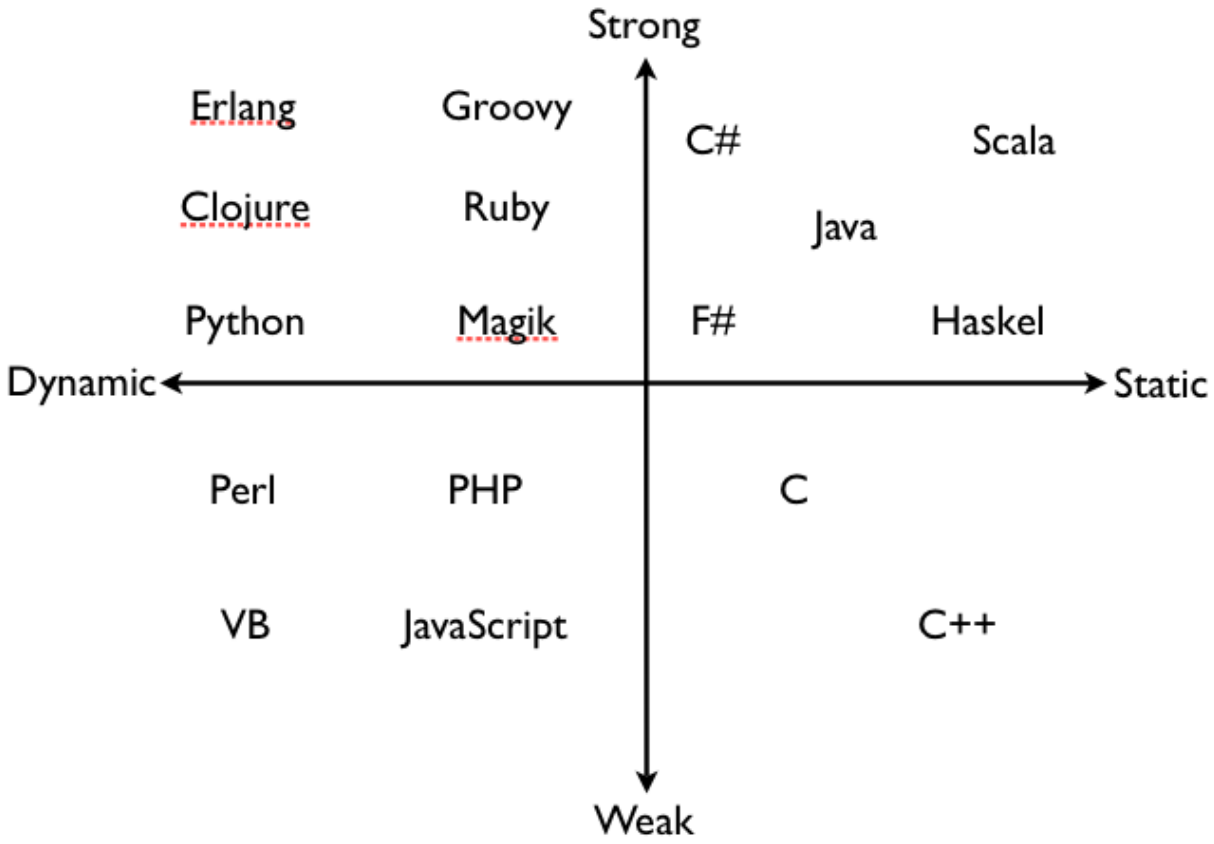
  不允许使用歧义单词，例如class_

  在python中不要随便使用下划线开头的标识符

基础语法

- 常量
  - 一旦赋值就不能改变值的标识符
  - python中无法定义常量
- 字面常量
  - 一个单独的量，例如 12、"abc" 、'2341356514.03e-9'
- 变量
  - 赋值后，可以改变值的标识符

Python的语言类型



- Python是动态语言、强类型语言
- 静态编译语言
  - 实现声明变量类型，类型不能再改变
  - 编译时检查
- 动态编译语言
  - 不用事先声明类型，随时可以赋值为其他类型
  - 编程时不知道是什么类型，很难推断
- 强类型语言
  - 不同类型之间操作，必须先**强制类型转换**为同一类型。print('a'+1)
- 弱类型语言
  - 不同类型间可以操作，自动隐式转换，JavaScript中console.log(1+'a')

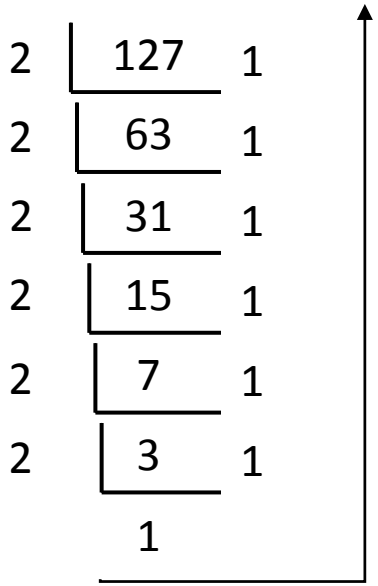
运算符Operator

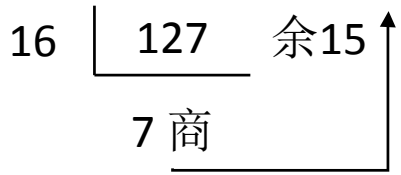
- 算数运算符
  - + - * / % **
  - 自然除/结果是浮点数，整除//。注：2.x中/和//都是整除

- 位运算符
  - & | ~ ^ << >>
  - 常用方式：乘除2的倍数，32 // 8相当于 32 >> 3
  - 12，0xc，0o14，0b1100
    - ~12等于多少，为什么

进制

- 常见进制有二进制、八进制、十进制、十六进制。应该重点掌握二进制、十六进制
  - 十进制逢十进一；十六进制逢十六进一；二进制逢二进一
- 转为十进制——按位乘以权累加求和
  - 0b1110 计算为 1* 2**3 + 1* 2**2 + 1* 2**1 = 14
  - 0x41 计算为 4 * 16 + 1 * 1 = 65
- 转为二进制
  - 0xF8 按位展开即可，得到 0b1111 1000
  - 127 除以基数2，直到商为0为止，反向提取余数
- 转为十六进制
  - 127 除以基数16，直到商为0为止，反向提取余数





原码、反码、补码，负数表示法

- 原码
  - 5 => 0b101，1 => 0b1 ，-1 => -0b1， bin(-1)
- 反码
  - 正数的反码与原码相同；负数的反码符号位不变其余按位取反
- 补码
  - 正数的补码与原码相同；负数的补码符号位不变其余按位取反后+1
- 负数表示法
  - 早期数字电路的CPU中的运算器实现了加法器，但是没有减法器，减法要转换成加法
  - 负数在计算机中使用补码存储，-1的补码为1111 1111
  - 5-1 => 5+(-1)直觉上是0b101-0b1，其实计算机中是0b101+0b11111111，溢出位舍弃
  - ~12为什么是-13？
  - 10^9 等于？ 10^-9等于？为什么

运算符

- 比较运算符
  - == != > < >= <=
  - 返回一个bool值
  - 1 < '1' 1 == '1'
  - 链式比较操作符
    - 4>3>2 4>mynumber >=1
- 逻辑运算符
  - 与或非 and or not
  - 短路运算符
    - and 如果第一个表达式为False，后面就没有必要计算了，这个逻辑表达式一定是False
    - or 如果第一个表达式True，后面没有必要计算了，这个逻辑表达式一定是True

运算符

- 赋值运算符
  - a = min(3, 5)
  - += -= *= /= %= 等
  - x = y = z = 10
- 成员运算符
  - in、not in
- 身份运算符
  - is 、is not

运算符优先级（由高到低）

运算符	描述
'expr'	字符串转换
{key:expr,...}	字典
[expr1,expr2...]	列表
(expr1,expr2,...)	元组
function(expr,...)	函数调用
x[index:index]	切片
x[index]	下标索引取值
x.attribute	属性引用
~x	按位取反
+x，-x	正，负
x**y	幂
x*y，x/y，x%y	乘，除，取模

运算符	描述
x+y，x-y	加，减
x<<y，x>>y	移位
x&y	按位与
x^y	按位异或
x|y	按位或
x<y，x<=y，x==y，x!=y，x>=y，x>y	比较
x is y，x is not y	等同测试
x in y，x not in y	成员判断
not x	逻辑否
x and y	逻辑与
x or y	逻辑或
lambda arg,...:expr	Lambda匿名函数

- 算数运算符 > 位运算符 > 身份运算符 > 成员运算符 > 逻辑运算符
- 记不住，用括号
- 长表达式，多用括号，易懂、易读

表达式Expression

- 由数字、符号、括号、变量等的组合
  - 算数表达式
  - 逻辑表达式
  - 赋值表达式
    - Python中，赋值即定义，如果一个变量已经定义，赋值相当于重新定义

内存管理

- 变量无须事先声明，也不需要指定类型，这是动态语言的特性
- Python编程中一般无须关心变量的存亡，一般也不用关心内存的管理
- python使用引用计数记录所有对象的引用数
  - 当对象引用数变为0，它就可以被 垃圾回收GC
  - 计数增加：
    - 赋值给其它变量就增加引用计数，例如x=3; y=x; z=[x, 1]
    - 实参传参，如foo(y)
  - 计数减少：
    - 函数运行结束时，局部变量就会被自动销毁，对象引用计数减少
    - 变量被赋值给其它对象。例如x=3; y=x; x=4
- 有关性能的时候，就需要考虑变量的引用问题，但是该释放内存，还是尽量不释放内存，看需求

内存管理

- 查看引用计数（选学）

```
import sys
x = []
print(sys.getrefcount(x))
print(sys.getrefcount([]))

y = x
print(sys.getrefcount(y), sys.getrefcount(x))
x = 5 # 注意字面常量
print(sys.getrefcount(y))
print(sys.getrefcount(x))

z = 5
print(sys.getrefcount(x))
```

程序控制

- 顺序
  - 按照先后顺序一条条执行
  - 例如，先洗手，再吃饭，再洗碗
- 分支
  - 根据不同的情况判断，条件满足执行某条件下的语句
  - 例如，先洗手，如果饭没有做好，玩游戏，如果饭做好了，就吃饭，如果饭都没有做，叫外卖
- 循环
  - 条件满足就反复执行，不满足就不执行或不再执行
  - 例如，先洗手，看饭好了没有，没有好，一会来看一次是否好了，一会儿来看一次，直到饭好了，才可是吃饭。这里循环的条件是**饭没有好**，饭没有好，就循环的来看饭好了没有。

单分支结构

- if语句

  if condition:

  　　代码块

  condition必须是一个bool类型，这个地方有一个隐式转换bool(condition)

  if 1<2: # if True:

  　　print('1 less than 2')

- 代码块
  - 类似于if语句的冒号后面的就是一个语句块
  - 在if、for、def、class等关键字后使用代码块

真值表

对象/常量	值
""	假
"string"	真
0	假
>=1	真
<=-1	真
()空元组	假
[]空列表	假
{}空字典	假
None	假

- **False等价**布尔值，相当于bool(value)
  - 空容器
    - 空集合set
    - 空字典dict
    - 空列表list
    - 空元组tuple
  - 空字符串
  - None对象
  - 0

多分支结构

- if...elif...else语句

```
if condition1:
    代码块1
elif condition2:
    代码块2
elif condition3:
    代码块3
.....
else:
    代码块
```

- 举例

```
a = 5
if a<0:
    print('negative')
elif a==0:
    print('zero')
else:
    print('positive')
```

分支嵌套

- 举例

```
score = 80
if score<0:
    print('wrong')
else: # score >=0
    if score==0:
        print('egg')
    elif score <= 100: # > 0
        print('right')
    else: # score>100
        print('too big')
```

- 嵌套结构，可以是分支、循环的嵌套
- 可以互相嵌套多层

- 多分支结构，只要有一个分支被执行，其他分支都不会被执行
- 前一个条件被测试过，下一个条件相当于隐含着这个条件

练习：

- 输入2个数字，输出最大数
- 给定一个不超过5位的正整数，判断其有几位

- 使用input函数
  - input函数获取键盘输入 input([prompt])，返回一个输入的字符串
- int函数
  - 把给定类型的数值转换为整数

练习：

- 给定一个不超过5位的正整数，判断其有几位
  - 使用input函数

```
val = input('>>>')
val = int(val)
if val >= 1000:  #fold
    if val>=10000:
        print(5)
    else:
        print(4)
else:
    if val>=100:
        print(3)
    elif val >= 10:
        print(2)
    else:
        print(1)
```

循环——while语句

- 语法

  while condition:

  　　block

  - 当条件满足即condition为True，进入循环体，执行block
- 举例

  flag=10

  while flag:

  　print(flag)

  　flag -= 1

  - 执行结果是什么？为什么？
  - 如果flag=-10可以吗？如何改造？

循环——for语句*

- 语法

  for element in iterable:

  block

  - 当 **可迭代对象** 中有元素可以迭代，进入循环体，执行block
- range函数
- 举例：打印1~10

  for i in range(10):

  print(i+1)

  - 执行结果是什么？为什么？
  - 如果想倒序打印呢？

循环 continue语句

- 中断当前循环的当次执行，继续下一次循环
- 举例：计算10以内的偶数（for循环）

```
for i in range(10):
    if not i%2:
        print(i)
```

- 还有其它的实现吗？

循环 continue语句

- 举例：计算10以内的偶数（for循环）

```
for i in range(0,10,2):  # 减少迭代次数
    print(i)

for i in range(0,10): # 使用位与
    if i & 1:
        continue
    print(i)
```

循环 break语句

- 终止当前循环
- 举例：计算1000以内的正整数被7整除的前20个数（for循环）

```
count = 1
for i in range(7, 1000, 7):
    print(i)
    if count == 2:
        break
    count += 1
```

循环 continue、break语句

- 总结
  - continue和break是**循环**的控制语句，只影响当前循环，包括while、for循环
  - 如果循环嵌套， continue和break也只影响语句所在的那一层循环
  - continue和break 不是跳出语句块，所以 if cond: break 不是跳出if，而是终止if外的break所在的循环

练习：

- 给定一个不超过5位的正整数，判断该数的位数，依次打印出万位、千位、百位、十位、个位的数字

练习：

- 给定一个不超过5位的正整数，判断该数的位数，依次从**万位打印到个位**的数字。分析如下：

难点在于如何对一个正整数进行迭代处理（暂不考虑使用字符串处理方式）。假设有数字54321，目标打印出5、4、3、2、1。

第一趟　54321//10000=5　54321%10000=4321
第二趟　4321 //1000 =4　4321 %1000 =321
第三趟　321 //100 =3　321 %100 =21
第四趟　21 //10 =2　21 %10 =1
第五趟　1 //1 =1　1 %1 =0

总结：

每一趟中，整除就能获得最高位置的数，取模就能获得出去最高位的数的剩余数字，且取模后的数字作为下一趟的待处理数字。

c = 54321　w = 10000
第一趟 out = c // w　c = c % w　w = w // 10
第二趟 out = c // w　c = c % w　w = w // 10
……

```
c = int('54321')
w = 10000

for i in range(5):
    print(c // w)
    c %= w
    w //= 10
```

练习：

- 给定一个不超过5位的正整数，判断该数的位数，依次打印出个位、十位、百位、千位、万位的数字

```
val = input('>>>')
val = int(val)
print(val)
if val >= 1000: # 折半
    if val>=10000:
        num = 5
    else:
        num = 4
else:
    if val>=100:
        num = 3
    elif val >= 10:
        num = 2
    else:
        num = 1
print(num)
c = val
for i in range(num):
    n = c // 10
    print(c - n*10)
    c = n  # 如果打印顺序是从万位到个位，如何实现？
```

假设输入为54321

第一趟：54321 // 10 = 5432

54321 - 5432*10 = 1

第二趟：5432 // 10 = 543

5432 - 543*10 = 2

依次类推

练习：

- 给定一个不超过5位的正整数，判断该数的位数，依次从万位打印到个位的数字

```
val = input('>>>')
val = int(val)
print(val)
if val >= 1000: # 折半
    if val>=10000:
        num = 5
    else:
        num = 4
else:
    if val>=100:
        num = 3
    elif val >= 10:
        num = 2
    else:
        num = 1
print(num)
pre = 0
for i in range(num):
    w = 10 ** (num - i - 1)
    cur = val // w
    print(cur - pre*10)
    pre = cur
```

假设输入为54321

第一趟：cur = 54321 // 10000 = 5

打印5 - 0

pre = 5

第二趟：cur = 54321 // 1000 = 54

打印54 - 5 * 10 = 4

pre = 54

第二趟：cur = 54321 // 100 = 543

打印543 - 54 * 10 = 3

pre = 543

依次类推

练习：

- 给定一个不超过5位的正整数，判断该数的位数，依次从万位打印到个位的数字

```
c = int('54001') # 00451, 01230
w = 10000
length = 5
flag = False

while w:
    t = c // w
    if not flag:
        if t:
            flag = True # 第一个非0
        else:
            length -= 1
    if flag:
        print(t)
    c %= w
    w //= 10

print("The length of number is", length)
```

为什么length能够算出来？

循环 else子句

- 语法

```
while condition:
    block
else:
    block

for element in iterable:
    block
else:
    block
```

  - 如果循环正常的执行结束，就执行else子句；如果使用break终止，else子句不会执行

练习：

- 给一个半径，求圆的面积和周长。圆周率3.14
- 输入两个数，比较大小后，从小到大升序打印
- 依次输入若干个整数，打印出最大值。如果输入为空，则退出程序
- 输入n个数，求每次输入后的算数平均数
- 打印一个边长为n的正方形
- 求100内所有奇数的和（2500）
- 判断学生成绩，成绩等级A~E。其中，90分以上为'A'，80~89分为'B'，70~79分为'C'，60~69分为'D'，60分以下为'E'
- 求1到5阶乘之和
- 给一个数，判断它是否是素数（质数）
  - 质数：一个大于1的自然数只能被1和它本身整除

谢谢

咨询热线 400-080-6560