**Python语言程序设计 课堂笔记** 北京理工大学

绿色字体代表亟需实践的代码

蓝色字体代表运行结果

红色字体代表重点内容

# 第0周 课堂导学

# 第1周 Python基本语法元素

## 1.1 程序设计的基本方法

**计算机与程序设计**

**编译与解释**

编译：将源代码全部转为目标代码（静态语言）

解释：逐行解释为目标代码（脚本语言）

**程序的基本编写方法**

iPO（input process output）

## 1.2 Python开发环境配置

**Python程序的编写与运行**

交互式

文件式

**实例1 圆面积的计算**

|  |
| --- |
| >>> r=25  >>> area=3.1415\*r\*r  >>> print(area)  >>> print(“{:.2f}”.format(area)) |

**实例2 同切圆绘制**

|  |
| --- |
| >>> import turtle  >>> turtle.pensize(2)  >>> turtle.circle(10)  >>> turtle.circle(40)  >>> turtle.circle(80) |

**实例3 绘制五角星**

|  |
| --- |
| >>> from turtle import \*  >>> color (‘red’,’red’)  >>> begin\_fill()  >>> for i in range(5)  fd(200)  Rt(144)  >>> end fill |

## 1.3 实例：温度转换

**输入输出格式设计**

标识放在温度最后，f表示华氏度，C表示摄氏度

*82f表示华氏82度*

**设计算法**

转换公式

C=(f-32)/1.8 f=C\*1.8+32

|  |
| --- |
| tempstr=input("please input a value of temperate with signal:")  if tempstr[-1] in ['f','f']:  C=(eval(tempstr[0:-1])-32)/1.8  print("the value after being calculated is {:.2f}C".format(C))  elif tempstr[-1] in ['C','c']:  f=eval(tempstr[0:-1])\*1.8+32  print("the value after being calculated is {:.2f}f".format(f))  else :  print("format error!") |

## 1.4 Python程序语法元素分析

**程序的格式框架**

缩进：单层缩进，多层缩进

*严格明确：缩进是语法的一部分，缩进不正确则程序运行错误*

*所属关系：缩进是表达戴马建包含与层次关系的唯一手段*

*长度一致*

注释：单行注释#，多行注释’’’ ‘’’

**命名与保留字**

变量：用来保存和表示数据的占位符号

命名规则：大小写字母，数字，下划线，汉字等字符及组合

注意事项：大小写敏感，首字符不能是数字，不与保留字相同

保留字：被编程语言内部定义并保留使用的标识符

**数据类型**：供计算机程序理解的数据形式

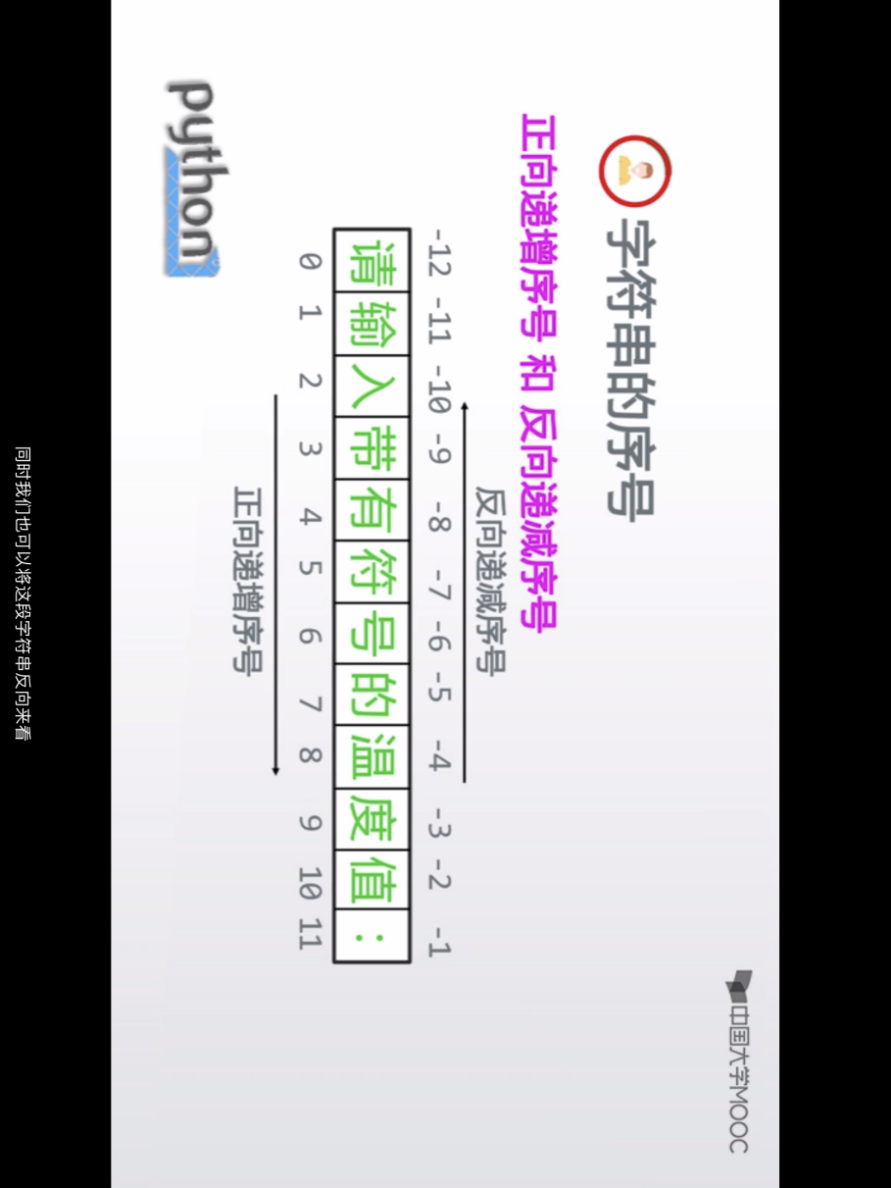
10,011,101

-整数类型：10011101

-字符串类型：”10,011,101”

-列表类型：[10,011,101]

字符串的序号 正向递增序号和反向递减序号



字符串的使用：使用[ ]获取字符串中的一个或多个字符

-索引：返回字符串中的单个字符 <string> [M]

-切片：返回字符串中的一段字符子串 <string> [M:N]

**#表示从M到N，但是取不到N**

|  |
| --- |
| l **=** [1,2,3,4,5,6,7,8]  l[:] *# 取全部元素*  [1, 2, 3, 4, 5, 6, 7, 8]  l[::**-**1] *# 逆序*  [8, 7, 6, 5, 4, 3, 2, 1]  l[::2] *# 取下标为偶数位的元素*  [1, 3, 5, 7]  l[1::2] *# 下标为奇数位*  [2, 4, 6, 8]  l[2:4] *# 下标为 [2,4)*  [3, 4]  l[:3] *# 取前3位，下标为 [0,3)*  [1, 2, 3]  l[:100] *# 超过数组长度时，尾部截断*  [1, 2, 3, 4, 5, 6, 7, 8]  l[100:0] *# 起始位置大于长度时，返回空list*  []  l[**-**3:] *# 取后3位，下标为 [n-3,n)*  [6, 7, 8]  l[3:] *# 下标为 [3，n)*  [4, 5, 6, 7, 8]  l[:**-**3] *# 下标为 [0, n-3)*  [1, 2, 3, 4, 5]  l[3] *# 取下标为3的数*  4  l[**-**3] *#l[n-3]=l[5] #取下标为n-3的数*  6  l[**-**1] *# 最后一个数，即下标为n-1的数*  8  l[2::3] *# 每隔3取一个数*  [3, 6]  **for** i **in** range(3):  **print**(l[i:i**+**3]) *# 窗口式读取，每次读3个数，循环3次*  [1, 2, 3][2, 3, 4][3, 4, 5] |

**语句与函数**

赋值语句 分支语句-使用if elif else表示条件分支语句

函数

**Python程序的输入输出**

输入函数 input( )

-input函数的使用格式：<变量>=input(<提示信息字符串>)

输出函数 print( )

-print函数的**格式化**：print(“转换后的温度是{:.2f}C”.format(C))

{ }表示槽，后续变量填充到槽中

-eval( )函数 去掉参数最外侧引号并执行余下语句的函数

|  |
| --- |
| >>> eval(“1”)  1  >>> eval(“1+2”)  3  >>> eval(‘”1+2”’)  “1+2”  >>> eval(‘print(“hello”)’)  hello |

# 第2周 Python基本图形绘制

## 2.1 深入理解Python语言

**计算机技术的演进**

**编程语言的设计初心**

**Python语言的特点**

通用语言，强制可读性，语法简洁，支持中文字符，生态高产

## 2.2 实例：Python蟒蛇绘制

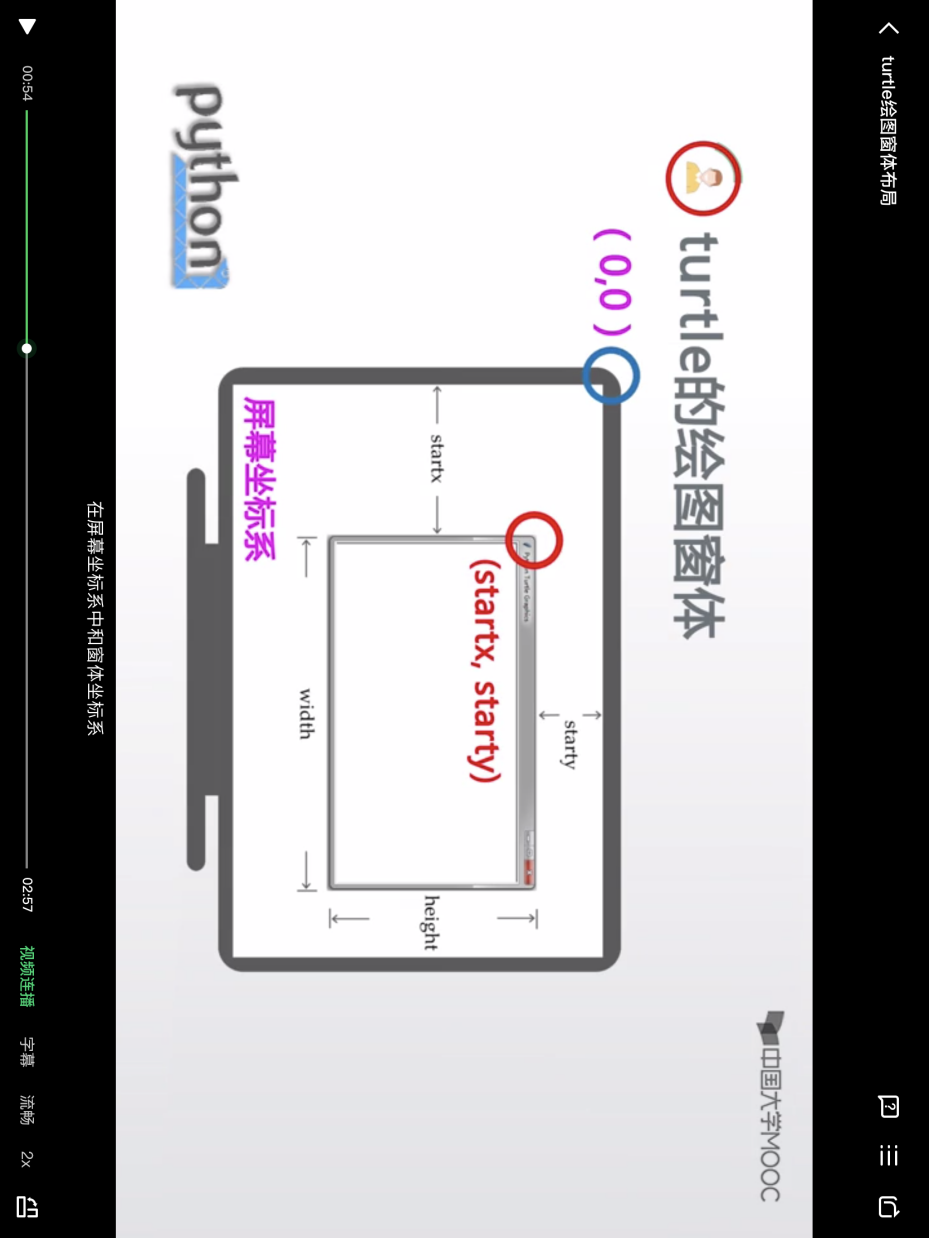
|  |
| --- |
| import turtle  turtle.setup(650,350,200,200)  turtle.penup()  turtle.fd(-250)  turtle.pendown()  turtle.pensize(25)  turtle.pencolor("purple")  turtle.seth(-40)  for i in range(4):  turtle.circle(40,80)  turtle.circle(-40,80)  turtle.circle(40,80/2)  turtle.fd(40)  turtle.circle(16,180)  turtle.fd(40\*2/3)  turtle.done() |

## 2.3 turtle库的使用

标准库

第三方库

**turtle的绘图窗体**



turtle.setup(width,height,startx,starty)设置窗体的大小和位置

*后两个参数不设置，默认在屏幕中央*

**turtle空间坐标体系**

-绝对坐标：以窗体中心为原点的坐标系

turtle.goto(x,y)

-海龟坐标：以海龟当前的位置和方向为原点（初始海龟处于窗体中心）

turtle.fd(d) 向海龟的前进方向移动，可以为负数

turtle.bk(d) 向海龟的后方向移动

turtle.circle(r,angle) 以左侧一个点为圆心做曲线运行（右侧则r取负）

**turtle角度坐标体系**

-绝对角度



turtle.seth(angle) 改变海龟的前进方向

-海龟角度

turtle.left(angle)

turtle.right(angle)

**RGB色彩体系（Python默认小数表示通道）**



## 2.4 turtle程序语法元素分析

**库引用** 扩充Python程序功能的方式

[1]

import <库名>

<库名>.<函数名>(<函数参数>)

[2]

from <库名> import <函数名>

from <库名> import \*

<函数名>(<函数参数>)

[3]

import <库名> as <库别名>

<库别名>.<函数名>(<函数参数>)

**turtle画笔控制函数**

turtle.penup( )将画笔抬起

turtle.pendown( )将画笔落下

turtle.pensize(width)画笔宽度 = turtle.width(width)

turtle.pencolor(color) 画笔颜色，使用颜色字符串或RGB小数值

**turtle运动控制函数**

*turtle.fd(d) turtle.bk(d) turtle.circle(r,angle)*

**turtle方向控制函数**

*turtle.seth(angle) turtle.left(angle) turtle.right(angle)*

**循环语句与range( )函数**

***for <变量> in range(<参数>):***

***<被循环执行的语句>***

|  |
| --- |
| >>> for i in range(5):  print("Hello: ",i)  Hello: 0  Hello: 1  Hello: 2  Hello: 3  Hello: 4 |

range( )产生循环计数序列

-range(N) 产生0到N-1的整数序列，共N个

-range(M,N) 产生M到N-1的整数序列，共N-M个

# 第3周 基本数据类型

## 3.1 数字类型及操作

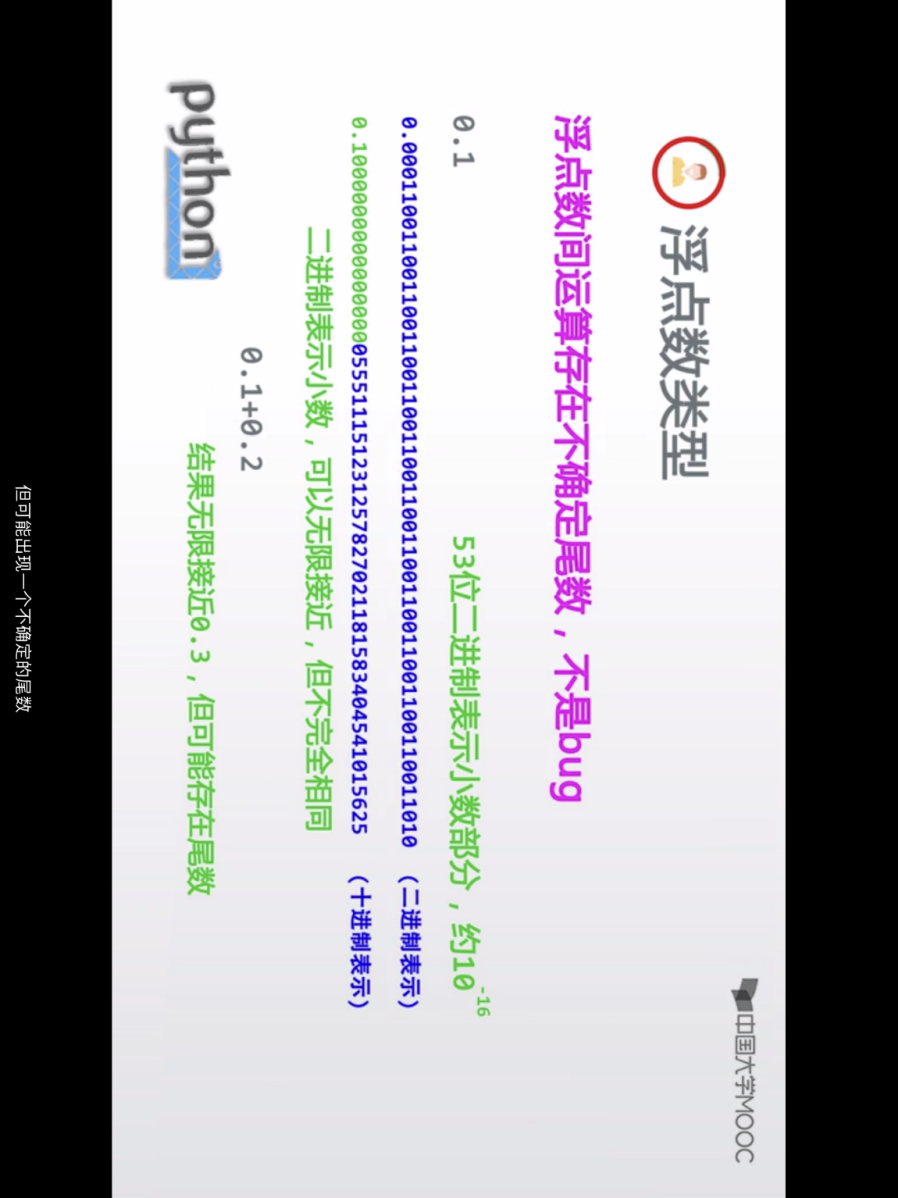
**整数类型** 可正可负，没有取值范围限制

pow(x,y) 计算，想算多大算多大

四种表示形式：十进制，二进制（0B/0b），八进制（0o/0O），十六进制（0x/0X）

**浮点数类型**

*浮点数间运算存在不确定尾数*



|  |
| --- |
| >>> 0.1+0.3  0.4  >>> 0.1+0.2  0.30000000000000004  >>> 0.1+0.2==0.3  False  >>> round(0.1+0.2,1)==0.3  True |

round(x,d) 对x四舍五入，d是小数截取位数

<a>E<b> 使用字母E作为幂的符号，以10为基数

**复数类型**

|  |
| --- |
| >>> z=1+2j  >>> z.real  1.0  >>> z.imag  2.0 |

**数值运算操作符**

x+y 加

x-y 减

x\*y 乘

x/y 除，结果为浮点数

x//y 整数除，结果为整数部分

+x x本身

-y y的负值

x%y 余数

x\*\*y 幂运算=pow(x,y)

x+=y x-=y x\*=y x/=y x//=y x%=y x\*\*=y

**数值运算函数**

abs(x) x的绝对值

divmod(x,y) 商余，同时输出商和余数

max( ) min( ) 最大最小值函数

int(x) 舍弃小数部分

float(x) 变成浮点数类型

## 3.2 实例：天天向上的力量

|  |
| --- |
| #工作日努力1%，休息日后退1%  dayup=1  dayfactor=0.01  for i in range(365):  if i%7 in[6,0]:  dayup\*=(1-dayfactor)  else:  dayup\*=(1+dayfactor)  print("工作日的力量为{:.2f}".format(dayup)) |

|  |
| --- |
| #工作日需要努力多少才能与每天努力1%相同  def dayUP(df):  dayup=1  for i in range(365):  if i%7 in[6,0]:  dayup\*=(1-0.01)  else:  dayup\*=(1+dayfactor)  #print("工作日的力量为{:.2f}".format(dayup))  return dayup  dayfactor=0.01  while dayUP(dayfactor)<37.78:  dayfactor+=0.001  print("需要工作日努力{:.3f}%".format(dayfactor\*100)) |

## 3.3 字符串类型及操作

**字符串的表示方法**

单行字符串：使用一对单引号或者一对双引号

多行字符串：使用三个引号来表示

|  |
| --- |
| >>> string=''' "这是一个字符串" '''  >>> print(string)  "这是一个字符串" |

转义符 \

-转义符表达特定字符的本意

-转义符形成一些组合，表示不可打印的含义

\b 回退 \n换行（光标移动到下行首） \r 回车（光标移动到本行首）

**字符串操作符**

x+y 连接两个字符串x和y

n\*x x\*n 复制n次字符串x

x in s 如果x是s的子串，则返回True

|  |
| --- |
| weekstr="星期一星期二星期三星期四星期五星期六星期日"  weekid=eval(input("请输入星期数字（1-7）："))  pos=(weekid-1)\*3  print(weekstr[pos:pos+3]) |

**字符串处理函数**

len(x) 返回字符串的长度

str(x) 任意类型x所对应的字符串形式

hex(x) oct(x) 整数x的十六进制和八进制字符串

chr(u) u为**Unicode**编码，返回对应的字符

ord(x) x为字符，返回其对应的Unicode编码

|  |
| --- |
| >>> " 1+1=2 "+chr(10004)  ' 1+1=2 ✔'  >>> "这个字符♉的Unicode值是："+str(ord("♉"))  '这个字符♉的Unicode值是：9801'  >>> for i in range(12):  print(chr(9800+i),end=" ")  ♈ ♉ ♊ ♋ ♌ ♍ ♎ ♏ ♐ ♑ ♒ ♓ |

**字符串的处理方法**

str.lower(s) 返回字符串的副本，全部字符小写

str.upper(s) 返回字符串的副本，全部字符大写

str.split(sep=None) 返回一个列表，有str根据sep被分割的部分组成

|  |
| --- |
| >>> string="a,b,c"  >>> string.split(",")  ['a', 'b', 'c'] |

str.count(sub) 返回子串sub在str中的出现次数

str.replace(old,new) 所有old子串被替换为new子串

str.center(width,[fillchar]) 字符串str根据宽度width居中，两侧填充fillchar

str.strip(chars) 从str中去掉其左侧和右侧chars中列出的字符

str.join(iter) 在iter变量除最后元素外每个元素后增加一个str

|  |
| --- |
| >>> string=","  >>> string.join("12345")  '1,2,3,4,5' |

**字符串的格式化**

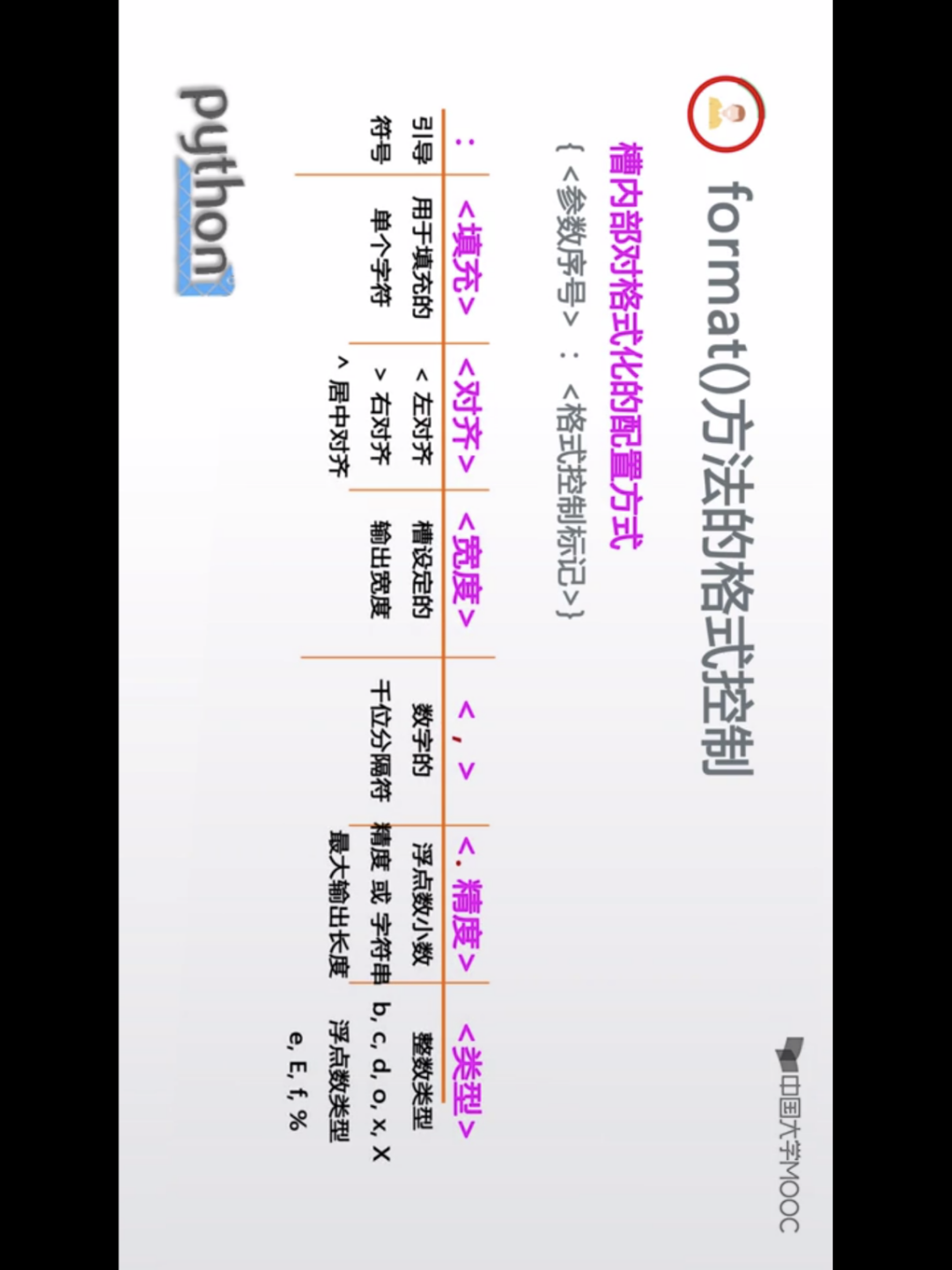
-格式化方法使用.format( )方法

<模板字符串>.format(<逗号分隔的参数>)

-槽

|  |
| --- |
| >>> print("{0} : 计算机{1}的CPU占用率为{2}%".format("2018-10-10","C",10))  2018-10-10 : 计算机C的CPU占用率为10% |

槽内部的格式化配置方式 {<参数序号>:<格式控制标记>}



|  |
| --- |
| >>> "{0:=^20}".format("python")  '=======python======='  >>> "{0:\*>20}".format("jlu")  '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*jlu'  >>> "{:10}".format("jlu")  'jlu '  >>> "{0:,.2f}".format(12345.6789)  '12,345.68'  >>> "{0:b},{0:c},{0:d},{0:o},{0:x},{0:X}".format(425)  '110101001,Ʃ,425,651,1a9,1A9'  >>> "{0:e},{0:E},{0:f},{0:%}".format(3.14)  '3.140000e+00,3.140000E+00,3.140000,314.000000%' |

## 3.4 time库的使用

**时间获取**

time.time( ) 获取当前时间戳，计算机的内部值

time.ctime( ) 获取当前时间并以易读方式表示

time.gmtime( ) 获取当前时间，表示为计算机可处理的时间格式

|  |
| --- |
| >>> import time  >>> time.time()  1592989600.0137954  >>> time.ctime()  'Wed Jun 24 17:06:48 2020'  >>> time.gmtime()  time.struct\_time(tm\_year=2020, tm\_mon=6, tm\_mday=24, tm\_hour=9, tm\_min=9, tm\_sec=7, tm\_wday=2, tm\_yday=176, tm\_isdst=0) |

**时间格式化**

time.strftime(tpl,ts) 将时间格式化为字符串，tpl是格式化模板字符串，ts是计算机内部时间类型变量





time.strptime(ts,tpl) 将字符串解析为时间格式

|  |
| --- |
| >>> t=time.gmtime()  >>> time.strftime("%Y-%m-%d %H:%M:%S",t)  '2020-06-24 09:13:04'  >>> timestr='2020-06-24 09:13:04'  >>> time.strptime(timestr,"%Y-%m-%d %H:%M:%S")  time.struct\_time(tm\_year=2020, tm\_mon=6, tm\_mday=24, tm\_hour=9, tm\_min=13, tm\_sec=4, tm\_wday=2, tm\_yday=176, tm\_isdst=-1) |

**程序计时**

time.perf\_counter( ) 测量时间

time.sleep(s) 产生时间

|  |
| --- |
| >>> start=time.perf\_counter( )  >>> end=time.perf\_counter( )  >>> end-start  4.599590634999913  >>> print(start,end)  5743.553951846 5748.153542481  >>> def wait():  time.sleep(3)    >>> wait() |

## 3.5 实例：文本进度条

|  |
| --- |
| #一个简单的开始  import time  scale=10  print("------执行开始------")  for i in range(scale+1):  a='\*'\*i  b='.'\*(scale-i)  c=(i/scale)\*100  print("{:^3.0f}%[{}->{}]".format(c,a,b))  time.sleep(0.1)  print("------执行结束------")    ------执行开始------  0 %[->..........]  10 %[\*->.........]  20 %[\*\*->........]  30 %[\*\*\*->.......]  40 %[\*\*\*\*->......]  50 %[\*\*\*\*\*->.....]  60 %[\*\*\*\*\*\*->....]  70 %[\*\*\*\*\*\*\*->...]  80 %[\*\*\*\*\*\*\*\*->..]  90 %[\*\*\*\*\*\*\*\*\*->.]  100%[\*\*\*\*\*\*\*\*\*\*->]  ------执行结束------ |

|  |
| --- |
| #单行动态刷新  import time  for i in range(101):  print("\r{:3}%".format(i),end="")  time.sleep(0.1) |

|  |
| --- |
| #完整效果  import time  scale=50  print("执行开始".center(scale//2,"-"))  start=time.perf\_counter( )  for i in range(scale+1):  a='\*'\*i  b='.'\*(scale-i)  c=(i/scale)\*100  dur=time.perf\_counter( )-start  print("\r{:^3.0f}%[{}->{}]{:.2f}s".format(c,a,b,dur),end="")  time.sleep(0.1)  print("\n"+"执行结束".center(scale//2,'-')) |

# 第4周 程序的控制结构

## 4.1 程序的分支结构

**单分支结构**

if <条件>:

<语句块>

**二分支结构**

[1]

if <条件>:

<语句块>

else:

<语句块>

[2]

<表达式1> if <条件> else <表达式2>

**多分支结构**

if <条件1>:

<语句块1>

elif <条件2>:

<语句块2>

else:

<语句块n>

**条件判断及组合**

< <= >= > == !=

x and y x or y not x

**异常处理**

try :

<语句块1>

except <异常类型> :

<语句块2>

else :

<语句块3> #不发生异常时执行

finally :

<语句块4> #一定会被执行

|  |
| --- |
| >>> try:  num=eval(input("please input an integer:"))  print(num\*\*2)  except:  print("it is not an integer!")    please input an integer:7  49  please input an integer:abc  it is not an integer! |

## 4.2 实例：身体质量指数BMI

|  |
| --- |
| height,weight=eval(input("请输入身高（米）和体重（公斤）[用逗号隔开]："))  bmi=weight/pow(height,2)  print("BMI的数值为:{:.2f}".format(bmi))  if bmi<18.5:  who="偏瘦"  elif 18.5<=bmi<25:  who="正常"  elif 25<=bmi<30:  who="偏胖"  else :  who="肥胖"  print("BMI指标为：国际‘{0}’".format(who)) |

## 4.3 程序循环结构

**遍历循环**

for <循环变量> in <遍历结构>:

<语句块>

- 计数循环 range(N)

- 字符串遍历循环

for c in s:

<语句块>

|  |
| --- |
| >>> for c in "python123":  print(c,end=",")  p,y,t,h,o,n,1,2,3, |

- 列表遍历循环

for item in ls:

<语句块>

|  |
| --- |
| >>> for item in [123,"python",456]:  print(item,end=" + ")  123 + python + 456 + |

- 文件遍历循环

**无限循环**

while <条件>:

<语句块>

**循环控制保留字**

break 跳出当前的循环

continue 跳出当次的循环，继续执行后续循环

|  |
| --- |
| >>> for c in "python":  if c=="t":  continue  print(c,end=" ")  p y h o n  >>> s="python"  >>> while s!="":  for c in s:  if c=="h":  break  print(c,end=" ")  s=s[:-1]    p y t p y t p y t p y t p y p |

**循环的高级用法**

- 循环+else 如果没执行break语句，则最后执行一次语句块2

[1]

for <循环变量> in <遍历结构> :

<语句块1>

else :

<语句块2>

[2]

while <条件> :

<语句块1>

else :

<语句块2>

|  |
| --- |
| >>> for c in "python ":  if c=="t":  continue  print(c,end=" ")  else:  print("\n正常退出")    p y h o n  正常退出  >>> for c in "python ":  if c=="t":  break  print(c,end=" ")  else:  print("\n正常退出")    p y |

## 4.4 random库的使用

基本随机数函数import random

random.seed(a=None) 初始化给定的随机数种子，默认为系统时间

*种子相同产生的随机数相同，可以数据复现*

random.random( ) 生成一个[0.0,1.0)之间的随机小数

|  |
| --- |
| >>> import random  >>> random.random()  0.16087634729275258  >>> random.seed(10)  >>> random.random()  0.5714025946899135  >>> random.random()  0.4288890546751146  >>> random.seed(10)  >>> random.random()  0.5714025946899135 |

**拓展随机数函数**

random.randint(a,b) 生成一个[a,b]之间的整数

random.randrange(m,n[,k]) 生成[m,n)之间以k为步长的随机整数

random.getrandbits(k) 生成一个k比特长的随机整数

random.uniform(a,b) 生成[a,b]之间的随机小数

random.choice(seq) 从序列seq中随机选择一个元素

random.shuffle(seq) 将序列seq中元素随机排列，返回打乱后的序列

|  |
| --- |
| >>> random.randint(10,100)  71  >>> random.randrange(10,100,10)  10  >>> random.getrandbits(16)  13506  >>> random.uniform(10,100)  51.632245728365426  >>> seq=[1,2,3,4,5,6,7,8,9]  >>> random.choice(seq)  8  >>> random.shuffle(seq)  >>> print(seq)  [2, 6, 7, 8, 4, 9, 1, 3, 5] |

## 4.5 实例：圆周率的计算

|  |
| --- |
| #公式求解方法  pi=0  n=100  for k in range(n):  pi+=1/pow(16,k)\*(4/(8\*k+1)-2/(8\*k+4)-1/(8\*k+5)-1/(8\*k+6))  print("圆周率的值为：{}".format(pi))  圆周率的值为：3.141592653589793 |

|  |
| --- |
| #蒙特卡洛方法  from random import random  from time import perf\_counter  darts=1000\*1000  hit=0  start=perf\_counter()  for i in range(1,darts+1):  x,y=random(),random()  dist=pow(x\*\*2+y\*\*2,0.5)  if dist<=1:  hit+=1  pi=4\*(hit/darts)  print("圆周率的值为：{}".format(pi))  print("运行时间是：{:.5f}s".format(perf\_counter()-start))  圆周率的值为：3.140504  运行时间是：1.63398s |

# 第5周 函数和代码复用

## 5.1 函数的定义与使用

def <函数名> (<参数(0个或多个)>) :

<函数体>

return <返回值>

|  |
| --- |
| >>> def fact(n):  s=1  for i in range(1,n+1):  s\*=i  return s  >>> fact(7)  5040 |

def <函数名> (<非可选参数><可选参数>) :

<函数体>

return <返回值>

|  |
| --- |
| >>> def fact (n,m=1):  s=1  for i in range(1,n+1):  s\*=i  return s//m  >>> fact(10)  3628800  >>> fact(10,5)  725760 |

def <函数名> (<参数>，\*b) : #不确定参数的总数量

<函数体>

return <返回值>

|  |
| --- |
| >>> def fact (n,\*b):  s=1  for i in range(1,n+1):  s\*=i  for item in b:  s\*=item  return s  >>> fact(10,3)  10886400  >>> fact(10,3,5,8)  435456000 |

**局部变量和全局变量**

global 在函数中声明变量为全局变量

|  |
| --- |
| >>> n,s=10,100  >>> def fact(n):  s=1  for i in range(1,n+1):  s\*=i  return s  >>> print(fact(n),s)  3628800 100  >>> n,s=10,100  >>> def fact(n):  **global s**  for i in range(1,n+1):  s\*=i  return s  >>> print(fact(n),s)  362880000 362880000 |

*局部变量为****组合数据类型****且未在函数中创建，等同于全局变量*

|  |
| --- |
| >>> ls=["F","f"]  >>> def func(a):  ls.append(a)  return  >>> func("c")  >>> print(ls)  ['F', 'f', 'c']  >>> ls=["F","f"]  >>> def func(a):  **ls=[]**  ls.append(a)  return  >>> func("c")  >>> print(ls)  ['F', 'f'] |

**lambda函数**

<函数名> = lambda <参数> : <表达式>

|  |
| --- |
| >>> f=lambda x,y:x+y  >>> f(1,2)  3  >>> f=lambda :"this is a function"  >>> f()  'this is a function' |

## 5.2 实例：七段数码管绘制

|  |
| --- |
| import turtle  import time  def drawgap():  turtle.penup()  turtle.fd(5)  def drawline (draw):  drawgap()  turtle.pendown() if draw else turtle.penup()  turtle.fd(40)  drawgap()  turtle.right(90)  def drawdigit(digit):  drawline(True) if digit in [2,3,4,5,6,8,9] else drawline(False)  drawline(True) if digit in [0,1,3,4,5,6,7,8,9] else drawline(False)  drawline(True) if digit in [0,2,3,5,6,8,9] else drawline(False)  drawline(True) if digit in [0,2,6,8] else drawline(False)  turtle.left(90)  drawline(True) if digit in [0,4,5,6,8,9] else drawline(False)  drawline(True) if digit in [0,2,3,5,6,7,8,9] else drawline(False)  drawline(True) if digit in [0,1,2,3,4,7,8,9] else drawline(False)  turtle.left(180)  turtle.penup()  turtle.fd(20)  def drawdate(date):  turtle.pencolor("red")  for i in date:  if i=='-':  turtle.write('年',font=("Arial",30,"normal"))  turtle.pencolor("green")  turtle.fd(55)  elif i=='=':  turtle.write('月',font=("Arial",30,"normal"))  turtle.pencolor("blue")  turtle.fd(55)  elif i=='+':  turtle.write('日',font=("Arial",30,"normal"))  else:  drawdigit(eval(i))    def main():  turtle.setup(900,350,200,200)  turtle.penup()  turtle.fd(-350)  turtle.pensize(5)  drawdate(time.strftime('%Y-%m=%d+',time.gmtime()))  turtle.hideturtle()  turtle.done()  main()  捕获 |

## 5.3 代码复用与函数递归

## 5.4 PyInstaller库的使用

将.py源代码转换成无需源代码的可执行文件.exe

(cmd) pip install pyinstaller

## 5.5科赫雪花小包裹

|  |
| --- |
| import turtle  def koch(size,n):  if n==0:  turtle.fd(size)  else:  for angle in [0,60,-120,60]:  turtle.left(angle)  koch(size/3,n-1)  def main():  turtle.setup(600,600)  turtle.penup()  turtle.goto(-200,100)  turtle.pendown()  turtle.pencolor("blue")  turtle.pensize(3)  level=3  koch(400,level)  turtle.right(120)  koch(400,level)  turtle.right(120)  koch(400,level)  turtle.hideturtle()  main()  捕获 |

# 第6周 组合数据类型

## 6.1 集合类型及操作

**集合类型定义**

无序，唯一，不可更改

使用{ }或set( )函数建立

|  |
| --- |
| >>> a={"python",123,("python",123)}  >>> a  {('python', 123), 123, 'python'}  >>> b=set("pypy123123")  >>> b  {'p', '1', 'y', '3', '2'}  >>> c={"python",123,"python",123}  >>> c  {123, 'python'} |

**集合操作符**

S | T 返回一个集合，包括所有元素

S - T 返回一个集合，在S中但不在T中

S & T 返回一个集合，同时在S和T中

S ^ T 返回一个集合，非相同元素

S <= T S < T 子集关系

S >= T S > T 包含关系

S |= T 更新集合S，包括所有元素

S -= T 更新集合S，在S中但不在T中

S &= T 更新集合S，同时在S和T中

S ^= T 更新集合S，非相同元素

|  |
| --- |
| >>> a={"p","y",123}  >>> b=set("pypy123")  >>> a-b  {123}  >>> b-a  {'2', '1', '3'}  >>> a&b  {'p', 'y'}  >>> a|b  {'p', '1', 'y', '3', '2', 123}  >>> a^b  {'2', '1', 123, '3'} |

**集合的处理方法**

S.add(x) x如果不在集合中，则增加到S

S.discard(x) 移除S中的x，如果x不在S中，不报错

S.remove(x) 移除S中的x，如果x不在S中，产生异常

S.clear( ) 移除所有元素

S.pop( ) 随机返回一个元素，并在S中删除该元素，若S空返回异常

S.copy( ) 返回集合的副本

len(S) 集合元素的个数

x in S 判断是否包含

x not in S 判断是否包含

set(x) 将其他类型变量转变为集合类型

## 6.2 序列类型及操作

序列是具有先后关系的一组元素

分为：字符串类型，元组类型，列表类型

正向递增序号，反向递减序号

**序列的基本操作**

x+y 连接两个序列x和y

n\*x x\*n 复制n次序列x

x in s 如果x是s的元素，则返回True

x not in s 如果x不是s的元素，则返回True

s[i] 索引，返回s中的第i个元素

s[i:j] s[i:j:k] 切片

len(x) 返回序列中元素的个数

s.index(x) s.index(x,i,j) 返回序列s中从i到j的位置中第一次出现元素x的位置

s.count(x) x出现的次数

**元组类型及操作** 元组一旦创建就不能修改

使用( )或tuple( )创建，元素间使用逗号分隔，可以使用或不使用小括号

|  |
| --- |
| >>> def func():  return 1,2  >>> func()  (1, 2)  >>> creature="cat","dog","tiger","human"  >>> creature  ('cat', 'dog', 'tiger', 'human')  >>> color=(0x001100,"blue",creature)  >>> color  (4352, 'blue', ('cat', 'dog', 'tiger', 'human'))  >>> color[-1][2]  'tiger' |

**列表类型及操作**

使用[ ]或list( )创建

ls[i]=x 替换列表ls第i个元素为x

ls[i:j:k]=lt lt替换切片后的子列表

del ls[i] del ls[i:j:k] 删除元素

ls+=lt 将lt元素增加到ls中

ls\*=n 是ls的元素重复n次

|  |
| --- |
| >>> ls=["cat","dog","tiger",1024]  >>> ls[1:2]=[1,2,3,4]  >>> ls  ['cat', 1, 2, 3, 4, 'tiger', 1024]  >>> del ls[::3]  >>> ls  [1, 2, 4, 'tiger']  >>> ls\*2  [1, 2, 4, 'tiger', 1, 2, 4, 'tiger'] |

ls.append(x) 在最后增加一个元素x

ls.clear( ) 删除所有元素

ls.copy( ) 生成新列表，赋值为ls中所有元素

ls.insert(i,x) 在第i个位置增加元素x

ls.pop(i) 将第i位置的元素取出并删除该元素

ls.remove(x) 出现的第一个x删除

ls.reverse( ) 将列表中的元素反转

## 6.3 实例：基本统计值计算

sorted(ls) 对列表进行排序

|  |
| --- |
| def getNum(): #获取用户不定长度的输入  nums = []  iNumStr = input("请输入数字(回车退出): ")  while iNumStr != "":  nums.append(eval(iNumStr))  iNumStr = input("请输入数字(回车退出): ")  return nums    def mean(numbers): #计算平均值  s = 0.0  for num in numbers:  s = s + num  return s / len(numbers)    def dev(numbers, mean): #计算方差  sdev = 0.0  for num in numbers:  sdev = sdev + (num - mean)\*\*2  return pow(sdev / (len(numbers)-1), 0.5)    def median(numbers): #计算中位数  sorted(numbers)  size = len(numbers)  if size % 2 == 0:  med = (numbers[size//2-1] + numbers[size//2])/2  else:  med = numbers[size//2]  return med    n = getNum() #主体函数  m = mean(n)  print("平均值:{},方差:{:.2},中位数:{}.".format(m, dev(n,m),median(n))) |

## 6.4 字典类型及操作

-映射 是一种索引与数据的对应

序列类型由0-N的整数作为数据的默认索引

映射类型则由用户为数据定义索引

采用{ }和dict( )创建，键值对用：表示

{<键1>:<值1>,<键2>:<值2>,<键3>:<值3>, ... ,<键n>:<值n>}

|  |
| --- |
| >>> d={"中国":"北京","美国":"华盛顿","法国":"巴黎"}  >>> d  {'中国': '北京', '美国': '华盛顿', '法国': '巴黎'}  >>> d["中国"]  '北京'  >>> de={}  >>> type(de)  <class 'dict'> |

**字典处理函数及方法**

del d[k] 删除键k对应的数据值

k in d 判断键k是否在字典d中

d.keys( ) 返回字典中所有键的信息

d.values( ) 返回字典中所有值的信息

d.items( ) 返回字典中所有键值对的信息

|  |
| --- |
| >>> d.keys()  dict\_keys(['中国', '美国', '法国'])  >>> d.items()  dict\_items([('中国', '北京'), ('美国', '华盛顿'), ('法国', '巴黎')]) |

d.get(k,<default>) 键k存在，则返回相应值，不存在则返回<default>

d.pop(k,<default>) 键k存在，则取出相应值，不存在则返回<default>

d.popitem( ) 随机从字典中取出一个键值对，以元组形式返回

d.clear( ) 删除所有键值对

len(d) 字典中的元素个数

|  |
| --- |
| >>> d.get("中国","伊斯兰堡")  '北京'  >>> d.get("巴基斯坦","伊斯兰堡")  '伊斯兰堡'  >>> d.popitem()  ('法国', '巴黎')  >>> d  {'中国': '北京', '美国': '华盛顿'} |

## 6.5 jieba库的使用

(cmd) pip install jieba

精确模式，全模式，搜索引擎模式

jieba.lcut(s) 精确模式，返回一个列表的分词结果

jieba.lcut(s,cut\_all=True) 全模式，返回列表，存在冗余

jieba.lcut\_for\_search(s) 搜索引擎模式，返回列表，存在冗余

jieba.add\_word(w) 向分词词典增加新词w

## 6.6 实例：文本词频统计

|  |
| --- |
| #统计哈姆雷特中出现频率最多的单词  def getText():  **txt = open("hamlet.txt", "r").read()**  txt = txt.lower()  for ch in '!"#$%&()\*+,-./:;<=>?@[\\]^\_‘{|}~':  txt = txt.replace(ch, " ") #将文本中特殊字符替换为空格  return txt    hamletTxt = getText()  words = hamletTxt.split() #words为列表类型  counts = {} #字典类型  for word in words:  counts[word] = counts.get(word,0) + 1  items = list(counts.items())  **items.sort(key=lambda x:x[1], reverse=True)**  for i in range(10):  word, count = items[i]  print ("{0:<10}{1:>5}".format(word, count)) |

|  |
| --- |
| import jieba  excludes = {"将军","却说","荆州","二人","不可","不能","如此"}  txt = open("threekingdoms.txt", "r", encoding='utf-8').read()  words = jieba.lcut(txt)  counts = {}  for word in words:  if len(word) == 1:  continue  elif word == "诸葛亮" or word == "孔明曰":  rword = "孔明"  elif word == "关公" or word == "云长":  rword = "关羽"  elif word == "玄德" or word == "玄德曰":  rword = "刘备"  elif word == "孟德" or word == "丞相":  rword = "曹操"  else:  rword = word  counts[rword] = counts.get(rword,0) + 1  for word in excludes:  del counts[word]  items = list(counts.items())  items.sort(key=lambda x:x[1], reverse=True)  for i in range(10):  word, count = items[i]  print ("{0:<10}{1:>5}".format(word, count)) |

# 第7周 文件和数据格式化

## 7.1 文件的使用

文本文件，二进制文件

- 文本文件 由单一特定编码组成的文件

- 二进制文件 直接由0和1组成，没有统一的编码

|  |
| --- |
| #文本形式打开文件  tf=open(“f.txt”,”rt”)  print(tf.readline())  tf.close()  #二进制形式打开文件  tf=open(“f.txt”,”rt”)  print(tf.readline())  tf.close() |

**文件的打开和关闭**

<变量名> = open(<文件名>,<打开模式>)

- 文件名 绝对路径 D:/PYE/F.txt D:\\PYE\\F.txt

相对路径 ./PYE/F.txt F.txt

- 打开模式

‘r’ 只读模式，默认值

‘w’ 覆盖写模式，文件不存在则创建，存在则完全覆盖

‘x’ 创建写模式，文件不存在则创建，存在则返回错误

‘a’ 追加写模式，文件不存在则创建，存在则在文件最后追加内容

‘b’ 二进制文本模式

‘t’ 文本文件模式，默认值

‘+’ 与r/w/x/a一同使用，在原功能的基础上增加同时读写功能

|  |
| --- |
| f=open(“f.txt”)  f=open(“f.txt”,”rt”)  f=open(“f.txt”,”wb”) |

<变量名>.close( )

**文件内容的读取**

<f>.read(size=-1) 读入全部内容，如果给出参数，读入前size长度

<f>.readline(size=-1) 读入一行内容，如果给出参数，读入前size长度

<f>.readlines(hint=-1) 读入文件所有行，以每行元素为列表，给出参数，则读入前hint行

**数据的文件写入**

<f>.write(s) 向文件中写入字符串或字节流

<f>.writelines(lines) 将元素全为字符串的列表写入，无缝拼接

<f>.seek(offset) 改变当前文件操作指针的位置

0-文件开头 1-当前位置 2-文件结尾

## 7.2 实例：自动轨迹绘制

|  |
| --- |
| import turtle as t  t.title('自动轨迹绘制')  t.setup(800, 600, 0, 0)  t.pencolor("red")  t.pensize(5)  #数据读取  datals = []  f = open("data.txt")  for line in f:  line = line.replace("\n","")  **datals.append(list(map(eval, line.split(","))))**  # map()将前一个操作作用于后面数据的每一个元素  f.close()  #自动绘制  for i in range(len(datals)):  t.pencolor(datals[i][3],datals[i][4],datals[i][5])  t.fd(datals[i][0])  if datals[i][1]:  t.rt(datals[i][2])  else:  t.lt(datals[i][2])  捕获 |

## 7.3 一维数据的格式化和处理

数据间有序使用列表表达

数据间无序使用集合表达

**一维数据的存储**

- 空格分隔

- 逗号分隔

- 特殊符号分隔

**一维数据的处理**

## 7.4 二维数据的格式化和处理

csv数据存储格式 每行数据由逗号隔开的数据组织方式

一般索引习惯：ls[row][column]

**从csv格式的文件中读入数据**

|  |
| --- |
| fo=open(fname)  ls=[]  for line in fo:  line=line.replace(“\n”,””) #将回车替换为空字符串  ls.append(line.split(“,”))  fo.close() |

**将数据写入csv格式的文件**

|  |
| --- |
| f=open(fname,’w’)  for item in ls:  f.write(‘,’.join(item)+’\n’)  f.close() |

**数据的逐一处理**

|  |
| --- |
| ls=[[1,2],[3,4],[5,6]]  for row in ls:  for column in row:  print(column) |

## 7.5 wordcloud库的使用

(cmd) pip install wordcloud

w=wordcloud.WordCloud(<参数>) 生成词云

w.generate(txt) 向WordCloud对象w中加载文本txt

w.to\_file(filename) 将词云输出为图像文件，.png或.jpg格式

- 参数

**width** 生成图片的宽度，以像素为单位，默认400

**height** 生成图片的高度，以像素为单位，默认200

**min\_font\_size** 指定词云中字体的最小字号，默认4号

**max\_font\_size** 指定词云中字体的最大字号，默认根据高度自动调节

**font\_step** 字体字号的步进间隔，默认1

**max\_words** 词云中的最大单词数量，默认200

**stop\_words** 指定词云的排除词列表，既不显示的单词列表

**background\_color** 词云的背景颜色

## 7.6 实例：政府工作报告词云

# 第8周 程序设计方法学

# 第9周 Python计算生态概览

## 9.1 从数据处理到人工智能

**数据分析**

Numpy

Pandas

SciPy

**数据可视化**

Matplotlib.pyplot

Seaborn

Mayavi

**文本处理**

PyPDF2

NLTK

Python-docx

**机器学习**

Scikit-learn

TensorFlow

MXNet