1 打印hello world

>>> print ('hello world')

hello world

>>> print("hello world")

hello world

>>> print ("hello world")

hello world

>>>

2 python 有交互式启动文件启动，交互启动在dos界面中输入python就可或者win+r 输入python输入一个命令返回一个结果就像命令行一样

文件启动就像写c语言等静态语言这些语言使用 文件写完后直接解释器解释 可用记事本写完后把后缀名改为.py或者在idle中按ctrl+n运行出一个文件解释

基本的程序编写方法IPO（input,process,output）

计算机只能完确定性的成计算问题

如果计算机能完成计算以外的问题比如如何思考问题等那是人工智能AI实现

解决计算机问题大概6个步骤

1 分析问题

2 划分边界（计算机类问题）

3 设计算法

4 编写程序

5 调试测试

6 升级维护

>>> str1 = input("请输入一个名字")

请输入一个名字张浩然

>>> str2 = input("请输入内容：")

请输入内容：牛逼

>>> print("{}是不是很{}".format(str1,str2))

张浩然是不是很牛逼

>>>

format()方法是将变量和字符串结合到一起输出

from turtle import \*

color("red", "yellow") #选择颜色

begin\_fill()

while True:

forward(200) #长度

left(460) # 旋转460度

if abs(pos()) < 1:

break

end\_fill()

done()

python程序包括格式框架，注释，变量，表达式，分支语句，循环语句，函数等语法元素。

python语言采用严格的缩进来表明程序的格式框架，缩进就是每一行代码开始前的空白区域，用来表示代码之间的包含和层次关系，不需要缩进就顶行编写不留空格

python和C语言一样用 #注释

python可以用大小写字母，数字，汉字组合成变量名但是名字首写字符不能是数字

标识符：对变量、常量、函数、函数、语句块起的名字叫标识符

标识符在python区分大小写 而且在python 3 x 中有33个保留字标识符不能和他一样同样保留字符也对大小写敏感 for是保留字符但是For不是保留字符

python有33个保留字符

False def if raise None del import return True elif in

try and else is while as except lambda with assert finally

nonlocal yield break for not class from or continue

global pass

文本在程序中用字符串类型表示（string）python中，字符串是用两个引号或者单个引号括起来的一个或多个字符

字符串是字符的序列，可以按照单个字符或者片段进行索引 字符串包括两种序号体系 正向递增序号 和反响递减序号

正向递增序号 是从左往右开始第一个为0向后一次次增1个

反向递减序号 是从右往左开始第一个为-1向左一次减一个

比如标识符tempstr[-1]表示最后一个字符 tempsrr[0]表示第一个字符 #一个空格也是一个字符

python字符串也提供区间访问方式 采用[N:M]格式，表示字符串从N到M（不包含M）的子字符串 N和M为字符串的索引号

>>> zhr = "1234567zhr"

>>> print(zhr[2:5])

345

>>>

>>> print(zhr[1])

2

>>>

>>> print(zhr[0:-1])

1234567zh

>>>

#表示打印从第一个开始到最后一个但是不包括最后一个字符

表达式就是程序中产生或者计算新数据值的代码，有点像数学中的计算公式。

python的赋值是 = （等号左右有没有空格无所谓空格但是bash中=号左右要有空格）

python也可给多个变量赋值 先从左到右运算等号右边的表达式然后把结果从左到右赋给等号左边的表达式

>>> x=3

>>> y=5

>>> x,y=y,x

>>> print(x)

5

>>> print(y)

3

>>>

input（）函数

从控制台获得用户输入，无论用户输入什么内容，input（）函数都以字符串类型返回结果

zhr=input("请输入：")

请输入：张浩然

>>> print(zhr)

张浩然

>>>

变量 = input（“请输入：”）

input（）函数可以包含一些提示性的文字

无论用户输入的是数字，字符，input（）统一按照字符串类型输出

if elif else

if <条件1>：

<语句块1>

elif <条件2>：

<语句块2>

。。。

else:

<语句块N>

在条件 Tempstr[-1] in ["f", "F"]:

中该表达式由保留字in组成，表示判断字符串tempstr的最后一个字符是否在一个由F，f的集合中如果在就返回true 如果不在就返回false

eval（）函数是一个非常重要的函数作用是将字符串str当成有效的表达式来求值并返回计算结果。

>>> tempstr=str(12345)

>>> c=(eval(tempstr[0:-1])-32)/1.8

>>> print(c)

667.7777777777777

>>>

首先定义一个字符串tempstr类容是“12345”

然后用eval函数把字符串第一个到最后一个（不包含最后一个）的字符串转化为python语句（上面转化成了整型也就是数字）然后做数学运算然后

赋值给c 然后打印

注意eval（）括号中不能直接输入一个字符因为eval会把他理解为一个没有定义的变量然后报错

输入字符串"'hello'"时 eval("'hello'")eval会把双引号去掉然后把'hello'当成一个字符串

eval（）也可和input（）调用

>>> x=eval(input("请输入一个数字:"))

请输入一个数字:123

>>> y=3\*x

>>> print(y)

369

>>>

也可直接写成

>>> print(3\*eval(input("请输入一个数字:")))

请输入一个数字:123

369

python支持+ - \* / \*\*（幂）5种基本的算数运算操作 在表达式内部加空格将被解释器去掉功能不变所

while语句

while （<条件>）:

<语句块1>

<语句块2>

当条件为true是执行语句块1 当条件为假时退出循环，执行循环后的语句块2语句。

def保留字可以定义一个函数

调用python的turtle库去画图

import turtle

turtle.setup(650,350,200,200)#绘制框框和电脑屏幕的关系

turtle.penup()#提起画笔这时到pendown的动作不会被画出来

turtle.fd(-250)#画笔向左移动250个像素向右移动是250 （向左向右前提是没有做seth指定移动的绝对度数如果指定了seth那么只能在度数指向的方向前行或者后退）

turtle.pendown()#落下画笔开始画画

turtle.pensize(25)#画笔的尺寸为25

turtle.pencolor("purple")#画笔的颜色为purple紫色

turtle.seth(-40)#以当前为水平位-40度的方向准备移动（turtle.fd（...）开始向旁边指定的方向移动指定的像素）

for i in range(4):

turtle.circle(40,80)#下有详解

turtle.circle(-40,80)#....

turtle.circle(40,80/2)#...

turtle.fd(40)#前进40个像素

turtle.circle(16,180)#...

turtle.fd(40 \* 2/3)#前进40X三分之二个像素

上述代码用了python编程的经典表达式<a>.<b>()

意思是调用一个对象<a>的方法<b>() ,也可以表示调用函数库<a>中的函数<b>()

面向对象: 举个例子，盖一座大楼，你想到的是楼怎么盖，哪里要有柱子，哪里要有梁，哪里楼梯等等（这就是面向对象），

至于柱子该怎么建，用什么建，方的圆的，等等，这就是面向过程。

面向过程和面对对象是一种编程思想，不能说某某语言是不是面对对象或是面向过程，

而是某某语言是否支持面对对象或面向过程。（python即可面向对象也可面向过程）

import引用函数库有两种方式，对函数的使用方式略有不同

第一种import <库名>

此时程序可调用库名中所有函数，使用库中的函数格式如下；

<库名>.<函数名>(<函数参数>）

第二种引用函数库的方法如下

from turtle import \* （表示调用turtle库中的所有函数）

此时不再用库名直接写函数名和参数

<函数名>(<函数参数>)

第一种解释器把<a>.<b>()整体作为函数名

第二种解释器会把<b>作为函数名如果自己定义的函数和<b>发生冲突时python会以最近的函数定义为准

避免冲突最好用第一种<a>.<b>()调用函数

turtle库是一个是一个直观有趣的图形绘制函数库 turtle图形绘制概念诞生与1969年，并且用于logo编程语言，python接受了这个概念

并成为python标准库之一

turtle绘图库基本框架是一个小海龟在坐标系中爬行爬行的轨迹形成绘制图形，对于小海龟来说有前行，后退，旋转，等爬行行为

刚开始时小海龟处于画布正中央，此处坐标为(0.0),

turtle.setup(width,height,startx,starty) (turtle库中的setup函数所有函数)

width：窗口宽度如果是整数，表示像素值：如果是小数表示窗口宽度与屏幕的比例。（这里的宽度只是一个相对的概念不是正方形的长宽高，只是指窗口左右的距离）

height：窗口高度，如果是整数，表示像素值：如果是小数表示窗口高度与屏幕的比例。（就是表示窗口上下高度）

startx：窗口左侧与屏幕左侧的像素距离，如果是None，窗口左右距离位于屏幕正中央

starty：窗口顶部与屏幕顶部的像素距离，如果是None，窗口位于屏幕垂直中央

（详情见python语言程序设计基础50页）

turtle.penup() 和turtle.pendown()是一组函数，函数表示画笔控制函数

turtle.penup() 别名

turtle.pu(),turtle.up() 作用首先抬起画笔，之后移动画笔不绘制图案 无参数

turtle.pendown() 别名

turtle.pd(),turtle.down() 作用首先落下画笔，之后移动画笔绘制图案 无参数

turtle.pensize()函数 turtle.pensize()函数用来设置画笔尺寸，函数定义如下：

turtle.width()作用：设置画笔宽度，当无参数输入时则返回当前画笔宽度。（实验感觉可以直接定义画笔宽度不定义尺寸 尺寸包含宽度 但是宽度不等于尺寸 尺寸还有长度）

turtle.pencolor()函数 定义了画笔的颜色

turtle.pencolor()函数的参数定义是 turtle.pencolor(colorstring) 直接跟颜色的字符串也就是参数要打双引号或者单引号

或者turtle.pencolor((r,g,b)) r,g,b对应的是RGB数值

很多RGB颜色都有固定的英文名字，这些英文名字可以作为colorstring输入到函数中如turtle.pencolor（"purple"）

也可以采用（r,g,b）形式直接输入颜色值 几种典型的RGB颜色如下

英文名称 RGB 十六进制 中文名称

white 255 255 255 #FFFFFF 白色

black 0 0 0 #000000 黑色

grey 190 190 190 #BEBEBE 灰色

darkgreen 0 100 0 #006400 深绿色

gold 255 215 0 #FFD700 金色

violet 238 130 238 #EE82EE 紫罗兰

purple 160 32 240 #A020F0 紫色

RGB颜色是计算机系统常用的颜色系统值一 R（red）G（green）B（blue）由三原色组成每个颜色的值不同组成新的颜色最小0最大255

turtle.fd()函数是当前水平坐标小乌龟向右还是向左移动多少个像素 200是向右移动200个像素再开始画画 -200是像左移动200个像素开始画画

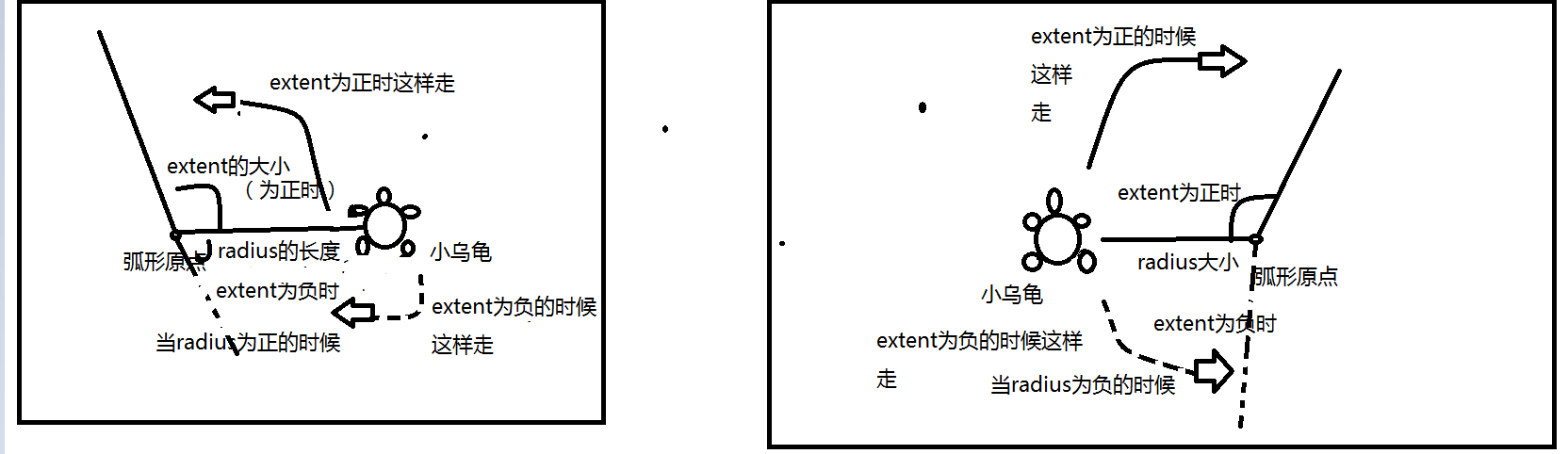
turtle.seth()函数是让小乌龟按照角度爬行 例turtle.seth(90) 以当前为水平坐标90度的方向爬行也就是向上爬也可写负数-90就是向下爬

turtle.circle也是一个确定小乌龟移动放向的函数他不光确定了方向也确定了移动的长度

turtle.circle(radius,extent) circle函数就是按照弧形移动

radius为正就是让移动的其实半径在小乌龟的左侧（不是原点的那一面）radius为负就是让移动的半径在小乌龟的右侧radius的大小就是圆弧的半径 （图解如下）

extent为正的时候往起始长度正的方向走为负的时候往其实方向负的方向走



数字类型

计算机对数字类型要求是准确性和高效性

正确性表示正确无误的解读数据所代表的类型含义，例如输入数字1010 计算机要知道他是表示运算的数字类型还是像门牌号一样的字符串，即使他是数字还要确认他是二进制还是十进制还是其他的进制

高效性表示程序可以为数字提供较高的运算速度，同时具备较少的存储空间代价。

整数和带有小数点的数字分别由计算机中央处理器中不同的硬件逻辑操作，如整数加法和小数加法，前者运算速度 比后面的块5到20倍 为了提升运算速度所以要区分不同运行速度的数字类型

Python有三种不同的数字类型

整数、浮点数、复数，分别对应数学中的

整数、实数、复数

（1010表示整数“1010”表示字符串）

整形类型有四种进制表达式：十进制、二进制、八进制、十六进制默认十进制其他进制需要增加引导符号 如，二进制以0b或者0B引导 0b101、0B101

八进制数由0o或者0O引导

十六进制由0x或者0X引导

大小写字母均可使用

整数类型理论上的取值范围是[负无穷大，正无穷大]，实际上取值范围受限与python程序的计算机内存大小

Pow(x,y)函数是python语言的一个内置函数 就是用来计算x的y次方pow()也可以嵌套使用

浮点数类型

就是带有小数的数值python语句要求所有的浮点数必须带有小数部分小数部分可以使0，它可以区分整数和浮点数的类型

浮点数可以用十进制计数法和科学计数法表示

科学计数法表示浮点数

9.6e5=9.6\*10\*10\*10\*10\*10

4.3e-3=4.3\*—（10\*10\*10）

e或者E为10后面的数字表示10有机个0

浮点数和整形的的计算由不同的计算机硬件执行处理方法不同

0.0和0都是表示一样但是在计算机内部表示不同

计算机只能提供浮点数15位的准确性

整数运算可以完全准确的运算结果

用整数做浮点数的精确运算然后再加上小数点是浮点数精准运算的做法之一

复数分实部和虚部他解决了j=根号 —1

实数是12.3+4j 、 -5.6+7j 、 1.23e-4+5.67+89j

实部是12.3 虚部通过后缀j表示所以虚部是4

在python中z=12.3+4j

z.real可以看出实部z.imag可以看出虚部

python提供了9个内置运算符

x+y x加y

x – y x减y

x\*y x乘y

x/y x与y的商

x//y x与y之整数商，即不大于x与y之商的最大整数

x%y x与y之商的余数

-x 负x也就是 x\*（-1）

+x x的本身

X\*\*y x的y次方 = pow（x，y）

操作符运算的结果可能会改变数字类型

整数之间运算，如果数学意义上的结果是小数，结果是浮点数

整数之间运算，如果数学意义上的结果是整数，结果是整数

整数和浮点数混合运算，输出结果是浮点数

整数或者浮点数与复数运算，输出结果是复数

Python有6个内置函数与计算有关

abs(x) x的绝对值

divmod(x,y) (x//y,x%y),输出结果是二元组形式

pow（x，y[,z]） 表示x的y次方然后除与z余的数 其中[…]可以不写

例如求3的(3的999次方)次方然后求最后四位一般写为 pow(3，POW（3，999）)%10000但是这样做运算非常慢一般计算机算不出来

pow（3，Pow（3，999），10000）这样幂运算和模块运算一起来速度非常块

round(x[,ndigits])对x四舍五入，保留ndigits位小数。 Round(x)返回四舍五入的整数

max(x1,x2….xn) 取x1 到xn的最大值

min（x1，x2……xn） 取x1,x2，x3….xn的最小值

内置函数数字转换函数

Int(x) 将x转换成为整数，x可以是浮点数或者字符串

>>> x=2.33

>>> float(x)

2.33

>>> int (float(x))

2

Float(x) 将x转换成为浮点数，x可以是字符串或者整数

Int（10.99）

10

Complex（re[，im]） 生成一个复数实部为re，虚步为im，re可以是整数、浮点数或者字符串，im可以是整数或者浮点数但是不能是字符串

Python运算字符的优先级



Ascii码

我们都知道在计算机内部所有的信息都将表示成一个二进制字符串，每一个二进制位有0，1两种状态，因此8个二进制可以组合出256种状态，这种被称为一个字节。也就是说，一个字节一共有256种不同的状态，没一个状态对应一个符号，256个字符就是从00000000到11111111

上世纪60年代美国制定了一套字符编码，对应英语与二进制之间的关系，做了统一的规定，这被称为ascii码，一直沿用至今

Unicode

世界上存在着多种的编码方式，同一个二进制数字可以被解释为不同的符号，因此，要打开一个文本文件，就必须要知道他的编码方式，否则用错位的编码方式解读就会出现乱码

可以想象，如果有一种编码，将世界上所有的字符都容纳其中。每一个字符都给予一个独一无二的编码，那么乱码问题就会消失。这就是unicode

Unicode是一个很大的集合没，现在的规模可以容纳100多万个字符号。每个符号的编码方式都不一样比如U+0639表示的是英语的大写字母A，U+4E25表示的是汉字“严“。。。

Python就是使用unicode

其中python的内置函数chr(x) 和ord(x)专门用看做unicode转换的

chr(x)其中x是输入unicode编码 通过chr函数转换成其特殊字符

ord(x)其中x是输入特殊的单字符通过ord函数转换成unicode编码

>>> chr(10004)

'✔'

>>>

>>> ord("张")

24352

>>>

Python的内置函数hex（x）和oct(x)分别返回整数x对应的 16进制和八进制值的字符串，字符串以小写形式表示。

例如

>>> hex(255)

'0xff'

>>> oct(255)

'0o377'

>>>

在python解释器内部，所有数据类型都是采用面向对象的方式实现，封装成为一个类。字符串也是一个类，他具有和类似<a>.<b>()形式的字符串处理函数。在面向对象中这类函数被称为方法。字符串类共包含43个内置方法。

Str.lower() 返回字符串str的副本，全部字符为小写

Str.upper() 返回字符串str的副本 全部字符为大写

Str.islower() 当字符串都都是小写的时候，返回true 否则返回false

Str.isupper() 当字符串都是大写的时候，返回true 否则返回false

Str.isprintable() 当字符都是可打印的时候返回true否则返回false

Str.isnumeric() 当str所有的字符是数字的时候返回true 否则返回fales

Str.isspace() 当str所有的字符都是空格，返回true 否则返回false

Str.count(sub[start[,end]]) 返回字符sub在[start:end]中sub出现的次数

Str.replace(old,new[,count]) 返回字符串的副本 所有old的字符被替换成new，如果count给出，则前count次old出现被替换

>>> a="a666666666666b"

>>> a.replace("6","1",4)

'a111166666666b'

类，对象，方法，函数

类(class)是一个非常抽象的概念，类是对象的抽象的表达方式，，也就是拥有同一属性的对象封装成一个类，而对象是类的具体表达方式也可以说对象是实例化后的类。类包含了函数和数据成员的定义，而函数只有操作，方法的实现方式是函数，可以把方法看作函数

对象就是类实力化后的东西 可以把猫或者狗看成对象，而他们的抽象化后都是动物，动物就是他们的类

而方法是这个对象可以做什么，属性就是这个对象有什么，

<a><b>()这类处理函数在面对对象中就是方法

上面的srt.count() 这个就是一个很好的例子str是一个类我们用的时候可以把他具体化到一个对象count就是这个类的方法

字符串类型格式化

Format（）方法

**<模板字符串>**.format(逗号分隔的参数)

format()方法用来控制修改字符串中嵌入值出现的位置，其基本的思想是将format()方法中逗号分隔的参数按照顺序替换到模板的字符串槽中。槽中用大括号表示

>>> print("{}:计算机{}的cpu占有率是{}%".format("2017-6-11","zhr","10"))

2017-6-11:计算机zhr的cpu占有率是10%

其中format后面的字符串可以用标识符（变量名称表示）

**<模板字符串>**.format(逗号分隔的参数) 前面的模板字符串也可以写成“{参数序号:格式控制标记}”

>>> print("0:30".format(z))

0:30

>>> print("{0:30}".format(z))

zhr

>>> print("{0:>30}".format(z))

zhr

>>> print("{0:-^30}".format(z))

-------------zhr--------------

>>> print("{0:#^30}".format(z))

#############zhr##############

>>> print("{0:\_>30}".format(z))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_zhr

0是开始的标记30是结束的标记可以不管他们

其中>是像右对齐 <是向左对齐 ^是居中前面的#-是用这些符号填充其余空白的地方

格式控制标记也可以用来表示整形的二进制unicode符10

进制8进制16进制等等

>>> print("{:b}".format(1004)) #b是输出二进制

1111101100

>>> print("{:c}".format(1004)) #c是输出unicode编码

Ϭ

>>> print("{:d}".format(1004)) #d是10进制

1004

>>> print("{:o}".format(1004)) #o是8进制

1754

>>> print("{:x}".format(1004)) #x是输出小写的16进制

3ec

>>> print("{:X}".format(1004)) #X是输出大写的16进制

3EC

>>> print("{:e}".format(1004)) #e是输出小写e的指数形式

1.004000e+03

>>> print("{:E}".format(1004)) #E是输出大写E的指数形式

1.004000E+03

>>> print("{:f}".format(1004)) #f输出浮点数的标准形式

1004.000000

>>> print("{:%}".format(1004)) #%输出百分号形式

100400.000000%

输出389的2进制，8进制，10进制，16进制，以及unicode码

>>> print("{0:b},{0:o},{0:d},{0:x},{0:c}".format(389))

110000101,605,389,185,ƅ

格式化输出0.002178对应的科学表示法形式，保留4位有效的标准浮点型式以及百分形式

>>> print("{0:.4e},{0:%}".format(0.002178))

2.1780e-03,0.217800%

>>>

用format编写进度条

**import time #调用time库**

**scl=10 #设置变量scl（它包含了运行的次数放在range里使用）**

**print("------执行开始------") #输出进度条第一行执行开始**

**for i in range(scl+1): #建立一个循环循环次数是scl的数字 因为range只能到规定的数前一个所以要加一**

**a, b="\*\*" \* i,".." \*(scl - i) #先运算等号右边的\*\*乘i和..乘于scl减去i，然后把右边第一赋值给等号左边第一个标识符，再把等号右边第二个赋值给等号左边第二个标识符**

**c=(i / scl) \* 100 #c是百分比**

**print("%{:3.0f}[{}->{}]".format(c,a,b)) #用format输出三个标识符 “”里最左边的{}里的参数规定了宽度为3 .0代表精确到小数点后面0位（四舍五入） f是输出浮点数 后面两个的{} {}没有规定直接输出。**

**time.sleep(0.2) #调用time库循环一次停顿0.2秒**

**print("------执行结束------") #输出最后一行**

int a=0  
printf("%d",a++); //输出0,执行完后a=1 和a=a+1一样  
  
int a=0  
printf("%d",++a);//输出1，执行完后a=1 **和a+=1一样**

**循环中的break和continue**

**他们用来辅助控制循环执行。**

**Break用来跳出这个for或者while循环本体**

**而continue用来跳出这一次循环继续下一次循环**

**例如：**

**for s in "PYTHON":**

**if s == 'T':**

**break**

**print(s,end = "")**

**PY**

**for s in "PYTHON":**

**if s == 'T':**

**continue**

**print(s,end = "")**

**PYHON**

**Random库**

**Python内置的random库主要用于产生各种分布的伪随机数序列。Random库采取梅森旋转算法生成伪随机数列**

**9个常用的随机数生成的函数**

**Seed(a=none) #初始化随机数种子，默认值为当前系统时间**

**random（） #生成0.0 和0.1之间的随机小数**

**randint（a，b） #生成a和b之间的整数**

**getrandbits(k) #生成一个kbit长度的随机整数**

**randrange(start,stop[,step]) #生成一个以start开始stop结束其中start和stop或者和生成的随机数中间可以整除4**

**例如**

**>>> random.randrange(3,9,3)**

**3**

**>>> random.randrange(3,9,3)**

**6**

**>>> random.randrange(3,9,3)**

**6**

**uniform（a，b） #生成一个a和b之间随机的小数**

**choice（seq） #从序列类型中随机生成一个元素**

**shuffle（seq） #将序列类型中的元素随机排序，返回打乱后的序列**

**sample（pop,k） # 从pop类型中随机选取k个元素**

**try-except语句**

**try:**

**<语句一>**

**except <异常类型> :**

**<语句块二>**

**例子**

**try:**

**a = eval(input("请输入数字：")) #把需要的语句包含到try-except语句里 输入的结果用eval函数把字符转换成数字如果输入其他字符会报nameerror错误 如果直接输出也会报另一个错误**

**num = a \* 3 # 输入的结果乘三**

**print(num) #输出这个标识符**

**except NameError: #当报错的信息为NameError时就会执行下面的语句如果报其他的错误就不会执行下面的语句**

**print("NameError!!!!") #当报错为NameError时执行print函数**

**try:**

**a = eval(input("请输入数字：")) #把需要的语句包含到try-except语句里 输入的结果用eval函数把字符转换成数字如果输入其他字符会报nameerror错误 如果直接输出也会报另一个错误**

**num = a \* 3 # 输入的结果乘三**

**print(num) #输出这个标识符**

**except NameError: #当报错的信息为NameError时就会执行下面的语句如果报其他的错误就不会执行下面的语句**

**print("NameError!!!!") #当报错为NameError时执行print函数**

**except: #当报其他除NmaeError错误之外的错误输出以下语句**

**print("其他错误")**

**try-except也可以用else finally等字样**

**try:**

**<语句块1>**

**except <异常类型1>:**

**<语句块2>**

**else:**

**<语句块3>**

**finally:**

**<语句块4>**

**此次的else语句和for循环和while循环中的else一样，当 try中的语句块1正常执行结束而且没有发生异常的时候执行else中的语句块3 可以看成是对try语句块的正常执行后的一种追加处理，finally则不同无论try语句块是否执行成功他都会执行语句块4**

**函数和代码复用**

**函数是一段具有特定功能的，可充重用的语句组，用函数名来表示并通过函数名来进行功能调用。函数也可以看成一段具有名字的子程序，可以在需要的地方调用执行，不需要再每个执行的地方重复编写这些语句。每次使用函数可以输出不同的参数，以实现对不同数据的处理：函数执行后，还可以反馈相应的处理结果**

**有些函数时用户自己编写的称为自定义函数，python安装包也自带了一些函数和方法，包括python的内置函数**

**函数有最主要两个目的：降低编程的难度和代码复用。**

**Python使用def保留字定义一个函数，语法如下**

**def <函数名> (<参数列表>):**

**<函数体>**

**return <返回值列表>**

**函数名可以是任何有效的python标示符;参数列表是调用该函数时传递给他的值，可以有0个,一个,或者多个，当传递多个参数是各个参数由逗号隔开，当没有参数传递是也要保留圆括号。函数定义中参数列表里面的参数是形式参数，简称形参（**全称为"形式参数" 由于它不是实际存在变量，所以又称虚拟变量。是在定义函数名和函数体的时候使用的参数,目的是用来接收调用该函数时传入的参数.在调用函数时，实参将赋值给形参。因而，必须注意实参的个数，类型应与形参一一对应，并且实参必须要有确定的值。**）**

**函数体是函数每次被调用的时候执行的代码，由一行或者多行语句组成。**

**当需要返回值的时候使用保留字段return和返回值列表没，如果函数可以没有return语句，在函数体结束位置将控制权返回给调用者**

**lambda函数**

**lambda函数是33个保留字之一，该函数用来定义一 种特殊的函数——匿名函数，又称为lambda函数。匿名函数并非没有名字，而是将函数名作为函数结果返回，语法格式如下:**

**<函数名> = lambda <参数列表>: <表达式>**

**Lambda函数和正常函数一样， 等价于下面的形式：**

**def <函数名> (<参数列表>) :**

**return <表达式>**