Python语言基础与应用02

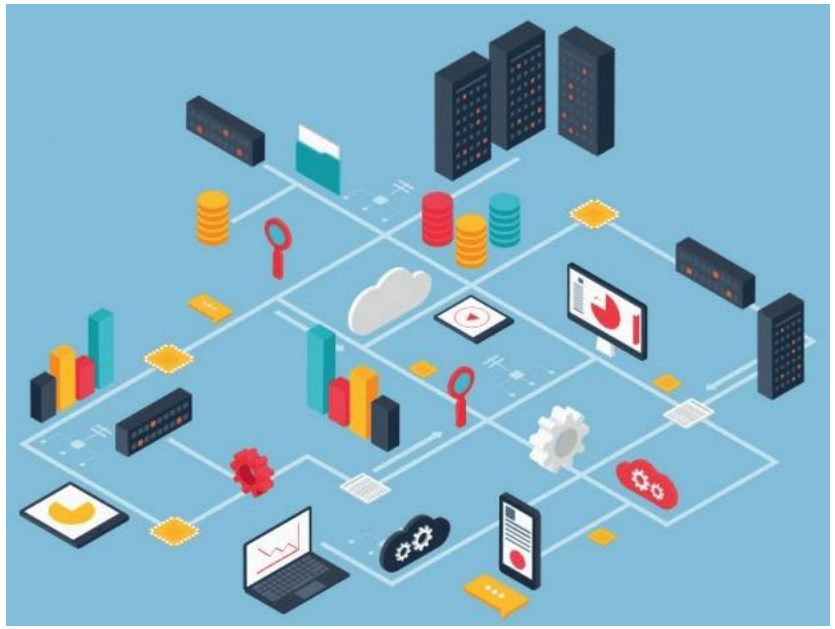
北京大学 陈斌

2019.07.02

目录

- 数据对象及其组织
- 自动计算过程
- 计算和控制流
- Python语言概览
- 基本数据类型
- 容器数据类型
- 输入-处理-输出
- OJ（Online Judge）系统

数据对象及其组织



- 什么是数据
- 多种多样的数据类型
- 数据类型归纳
- 对数据进行组织

什么是数据

- 要用计算机解决问题，首先要把问题表述为计算机能处理的形式。
- 现实世界中的万事万物蕴含着纷繁复杂的内容
- 我们只关注这些事物与所要求解问题相关的一些性质，表述其中关键的部分。
- 数据(data)是信息的表现形式和载体，是对现实世界实体和概念的抽象。

什么是数据

- 学生信息表
- 描述了一个学生的各方面属性
  - 70435
  - 张小明
  - 男
  - 19
  - 2016年9月1日
  - 照片图像

学号	70435
姓名	张小明
性别	男
年龄	19
入学日期	2016 年 9 月 1 日
照片	

大数据时代

- 计算机处理的数据越来越多
- 数据获取手段空前增多
- 人类开始广泛收集收据
- 大数据(big data)
  - Volume（大量）
  - Velocity（高速）
  - Variety（多样）
  - Value（低价值密度）
  - Veracity（真实性）

- Python语言是最热门的大数据分析处理语言

多种多样的数据类型

- 描述事物大小、次序的数值类型；
- 描述事物各方面特性的文本字符串类型；
- 描述事物时间属性的日期时间类型等；
- 每种数据类型都有自己的独特的运算。

```
>>> 12 * 34.5 + 23.4
437.4
>>> ('abc' + '123') * 3
'abc123abc123abc123'
>>>
>>> import math
>>> math.sqrt(12)
3.4641016151377544
```

多种多样的数据类型

- 复杂数据类型

图形、图像

音频

视频

对数据进行组织

- 对大量的数据进行处理的时候，需要建立各种各样的数据组织，以便提高计算效率
- 组织方式：
  - 没有组织
  - 顺序组织数据
  - 标签式组织数据

Python数据类型概览

- 简单类型用来表示值
  - 整数int、浮点数float
  - 复数complex
  - 逻辑值bool、字符串str
- 容器类型用来组织这些值
  - 列表list、元组tuple
  - 集合set、字典dict
- 数据类型之间几乎都可以转换

图书信息的数据分析

被引用指数0.0272
被图书引用册数5

计算机科学中的离散结构

作 者： 王元元，张桂芸编著

出版发行：北京：机械工业出版社，2004.01

ISBN号： 7-111-12939-3

页 数： 302

丛书名：高等院校计算机专业教育改革推荐教材

原书定价：28.00

开本：26cm

主题词：离散数学(学科: 高等学校) 离散数学

中图法分类号：O158 (数理科学和化学->数学->代数、数论、组合理论->离散数学)

内容提要: 高等院校计算机专业教育改革推荐教材:该教材涵盖了经典的"离散结构"或"离散数学"课程的主要内容，包括集合论基础、逻辑代数、形式系统与形式推理、组合论基础等内容。

参考文献格式：王元元，张桂芸编著. 计算机科学中的离散结构[M]. 北京：机械工业出版社, 2004.01.

- 有哪些数据项?
- 分别是什么数据类型?
- 有容器类型么?

序号	数据项	类型	示例
1	书名	字符串	"计算机科学中的离散结构"
2	作者	字符串的列表	["王元元","张桂芸"]
3	页数	整数	302

【H2】生活中的数据分析

- 指出有哪些数据项?
  - 10项以上
- 分别是什么数据类型?
- 指出其中的容器类型?

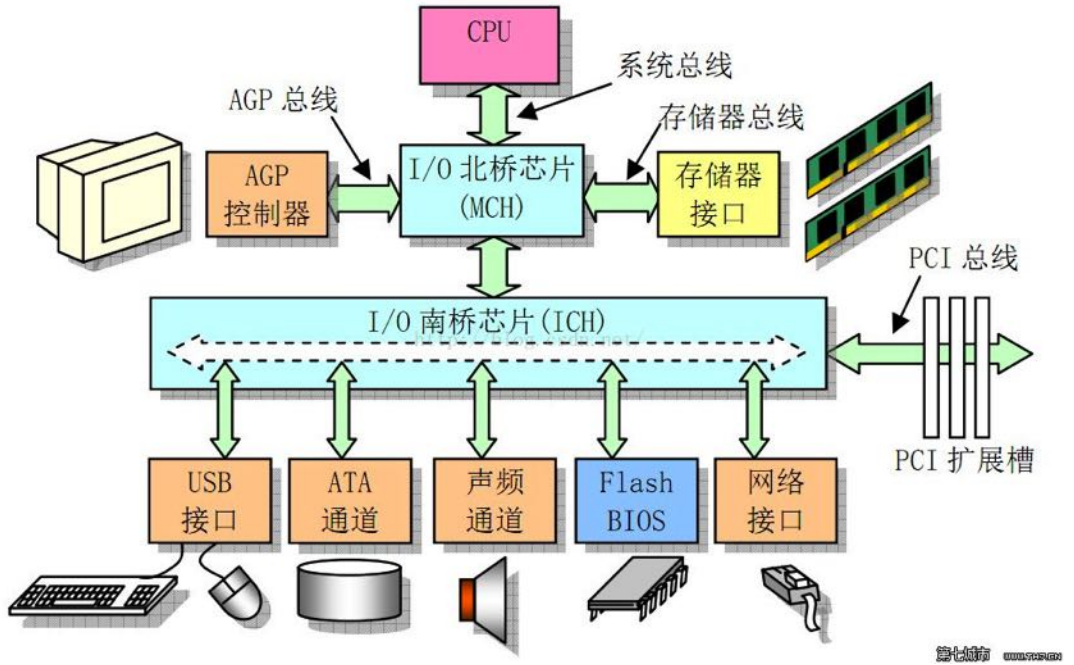
- 以京东商城为例
  - 对手机进行数据分析
  - 对鞋子进行数据分析
  - 对商品评价进行数据分析

序号	数据项	类型	示例

交一个doc/pdf文件

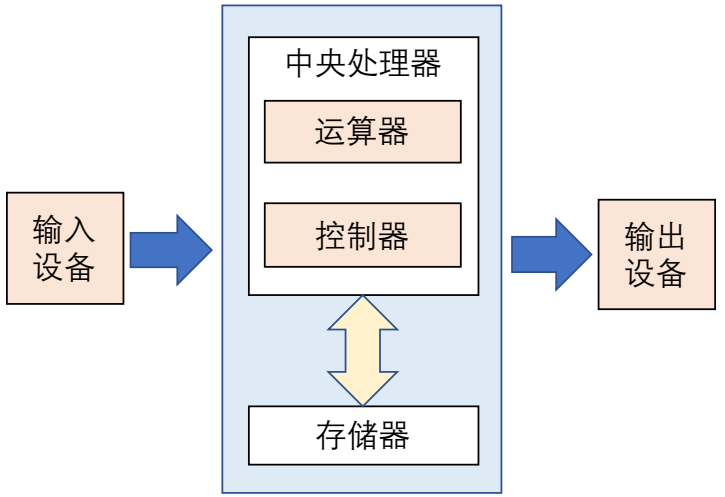
自动计算过程

- “冯·诺依曼结构”计算机
- 计算机内部运行过程
- 基本计算语句



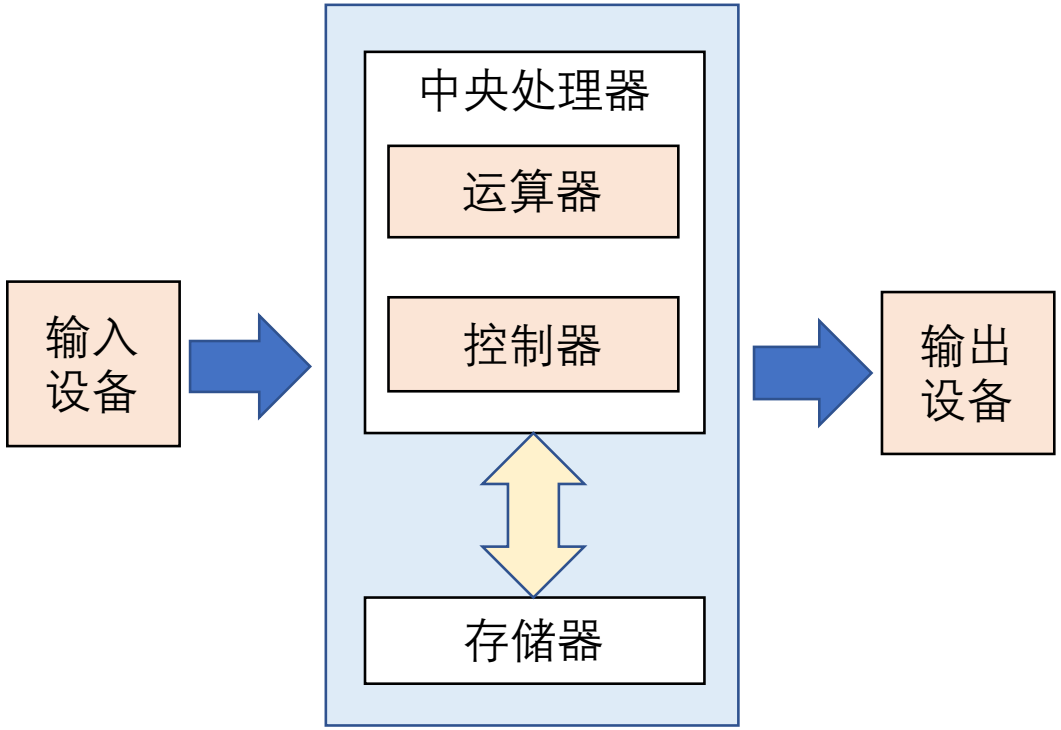
“冯·诺依曼结构”计算机

- “计算机之父”冯·诺依曼
  - 20世纪最重要的数学家之一
  - 现代计算机、博弈论、核武器和生化武器等领域的科　学全才
- 设计制造第一台电子计算机ENIAC时提出了“冯·诺依曼结构”



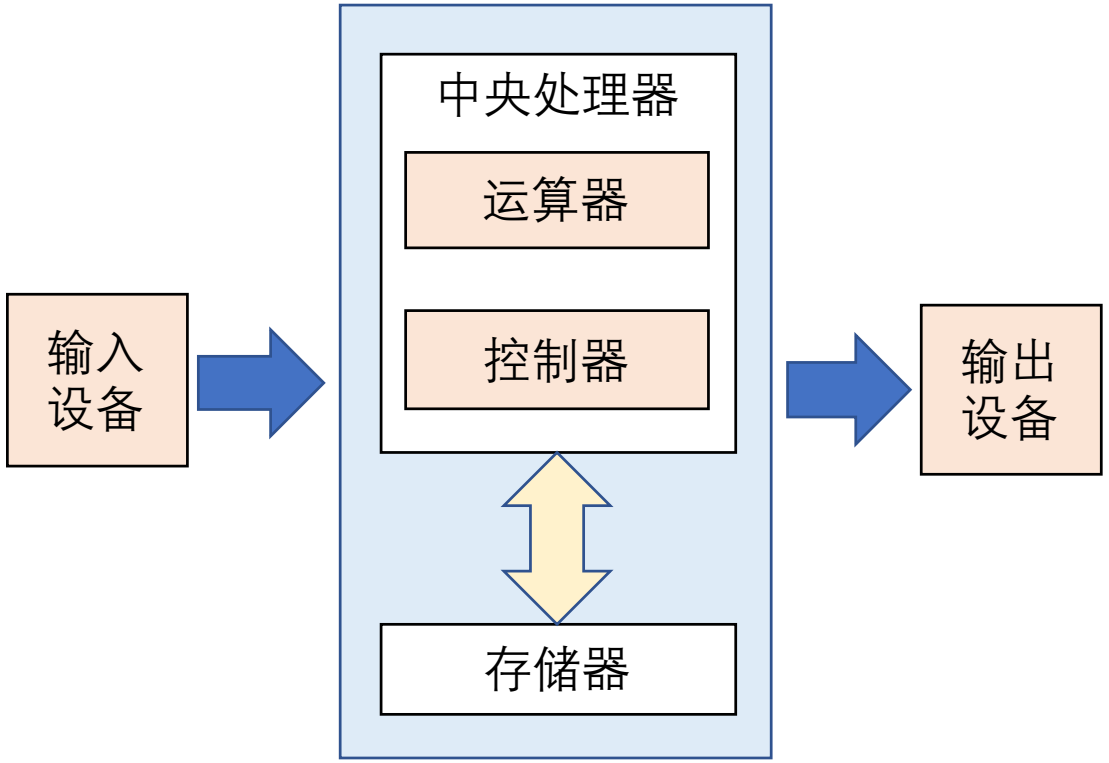
“冯·诺依曼结构”计算机

- 计算机硬件五大部件
  - 运算器：进行算术和逻辑运算
  - 控制器：控制计算机持续协调运行
  - 存储器：存储数据和程序
  - 输入设备：从计算机外部获取数据（如键盘、鼠标）
  - 输出设备：将计算结果反馈给外界（如显示器、打印机）



计算机内部运行过程

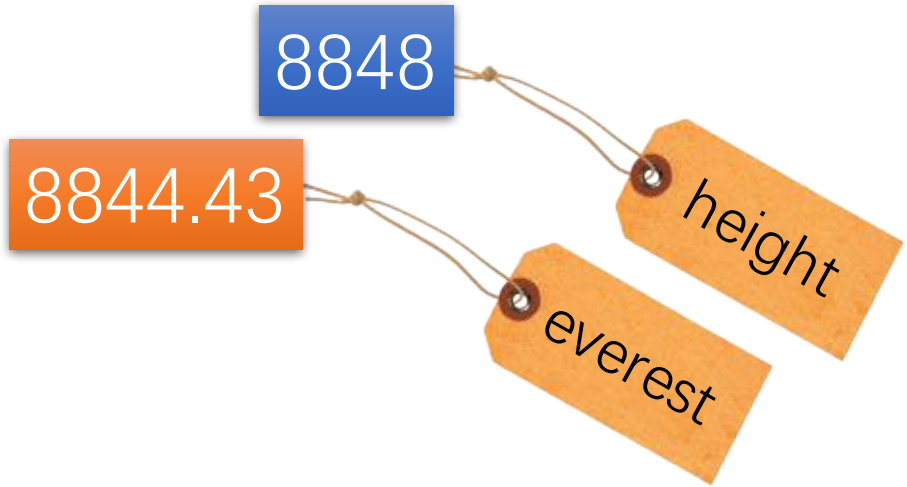
- 基本步骤
  - ① 控制器从存储器中取出程序语句，和所需的额外数据；
  - ② 数据齐全的语句交给运算器进行算术或者逻辑运算；
  - ③ 运算结果再存回存储器；
  - ④ 控制器确定下一条程序语句，回到步骤（1）继续。



基本计算语句

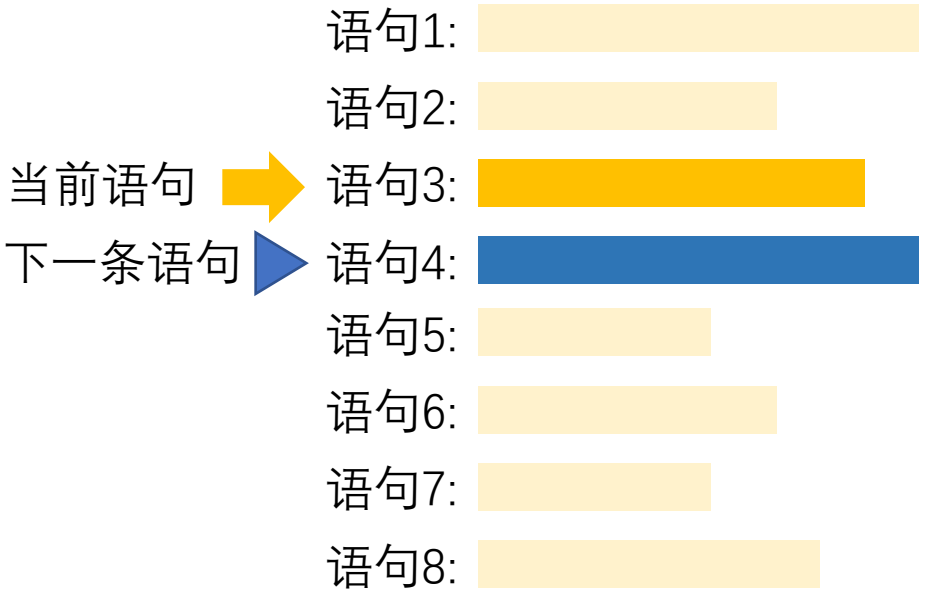
- 赋值语句
  - <变量> = <表达式>
- Python语言的赋值语句很好地结合了“计算”和“存储”
- 赋值语句的执行语义为：
  - 计算表达式的值，存储起来
  - 贴上变量标签以便将来引用
- 与计算机运行过程中的“计算”和“存储”相对应

```
height = 8848
everest = 8844.43
```

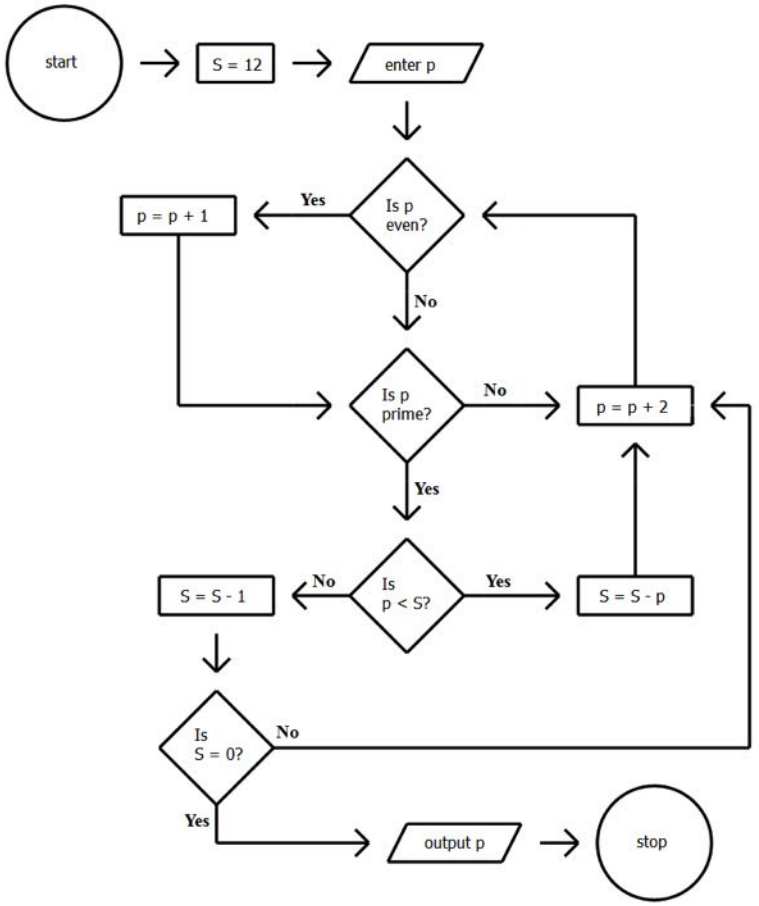


基本计算语句

- “控制器确定下一条程序语句”即对应“控制”
- 一个程序的很多语句，在存储器中的排列，就像在火车站买票一样排成一个队列
- 思考：下一条语句仅仅是“语句队列 中的后一条”一种情况吗?



计算和控制流



- 计算与流程
- 运算语句
- 控制流语句
  - 决定下一条语句

计算与流程

- 数据是对现实世界处理和过程的抽象
- 各种类型的数据对象
- 可以通过各种运算组织成复杂的表达式

```
>>> 12 * 34.5 + 23.4
437.4
>>> ('abc' + '123') * 3
'abc123abc123abc123'
>>>
>>> import math
>>> math.sqrt(12)
3.4641016151377544
```

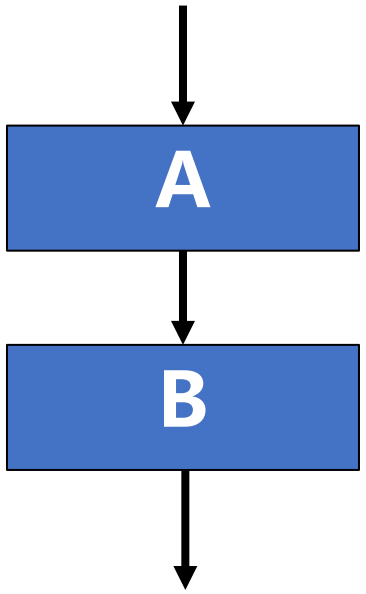
运算语句

- 将表达式赋值给变量进行引用
- 赋值语句用来实现处理与暂存
  - 表达式计算
  - 函数调用
  - 赋值

```
>>> n = 12 * 34
>>> n
408
>>> p2 = math.sqrt(2)
>>> p2
1.4142135623730951
>>> pfg = math.sqrt
>>> pfg
<built-in function sqrt>
>>> pfg(2)
1.4142135623730951
```

控制流语句

- 控制流语句用来组织语句描述过程



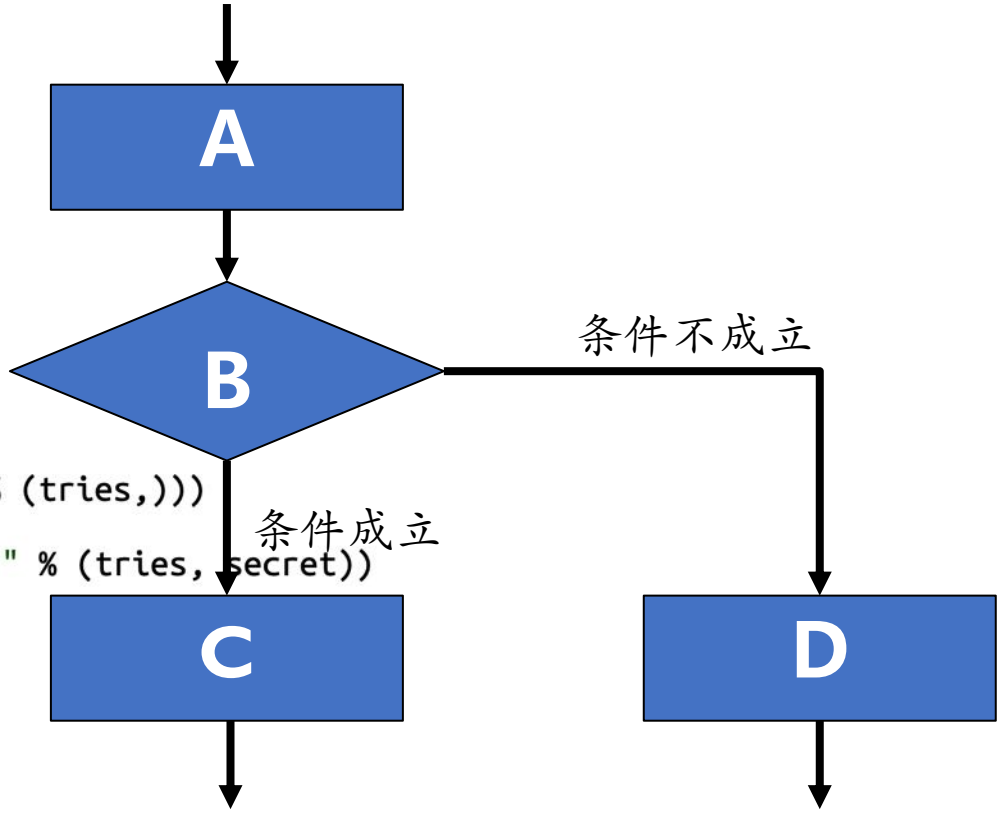
顺序结构

三角形.py

```
import turtle
t=turtle.Turtle()
t.color('green')
t.forward(100)
t.right(120)
t.forward(100)
t.right(120)
t.forward(100)
t.right(120)

t.hideturtle()
turtle.done()
```

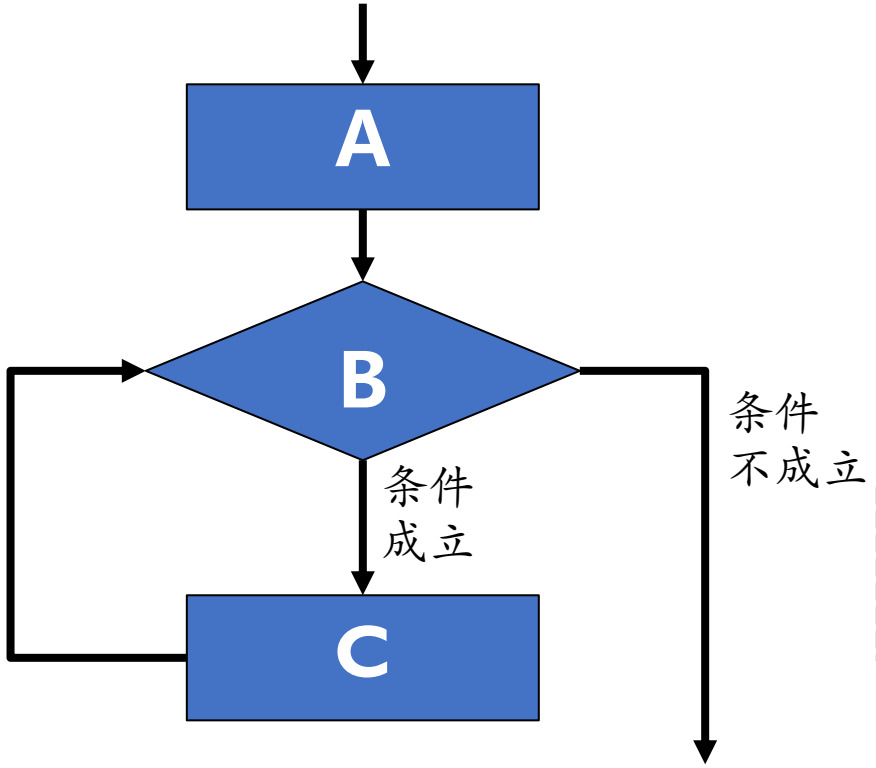
控制流语句

```
1   # 猜数字游戏
2   import random
3
4   secret = random.randint(1, 100)
5   print('''猜数游戏!
6   我想了一个1-100的整数，你最多可以猜6次，
7   看看能猜出来吗? ''')
8   tries = 1
9   while tries <= 6:
10      guess = int(input("1-100的整数，第%d次猜，请输入： " % (tries,)))
11      if guess == secret:
12          print("恭喜答对了！你只猜了%d次！\n就是这个：%d！" % (tries, secret))
13          break
14      elif guess > secret:
15          print("不好意思，你的数大了一点儿！")
16      else:
17          print("不好意思，你的数小了一点儿！")
18      tries += 1
19  else:
20      print("哎呀！怎么也没猜中！再见！")
```



条件分支：if

控制流语句



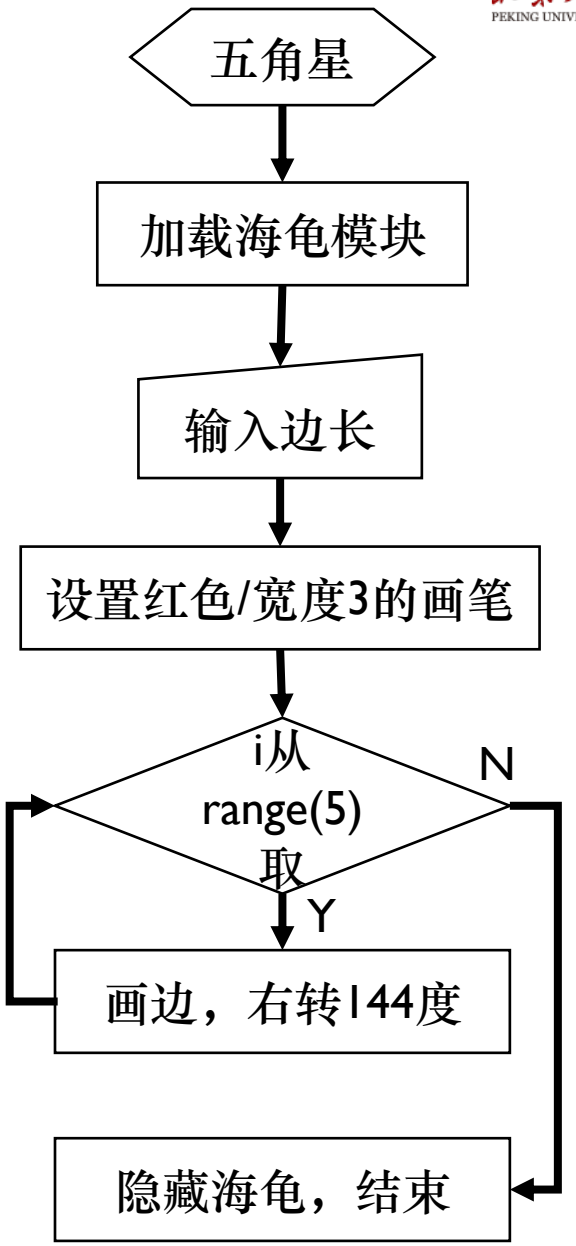

```
1   import turtle
2
3   size = int(input("Please input size:(20~200)"))
4
5   t = turtle.Turtle()
6   t.color('red')
7   t.pensize(3)
8
9   for i in range(5):
10      t.forward(size)
11      t.right(144)
12
13  t.hideturtle()
14  turtle.done()
```

循环结构for、while

分析程序流程

```
1   import turtle
2
3   size = int(input("Please input size:(20~200)"))
4
5   t = turtle.Turtle()
6   t.color('red')
7   t.pensize(3)
8
9   for i in range(5):
10      t.forward(size)
11      t.right(144)
12
13  t.hideturtle()
14  turtle.done()
```

打印显示



【H3】程序流程图绘制

```
1  import random
2
3  menu = ["coffee", "tea", "cola", "milk", "water"]
4  print("Menu:", menu)
5  name = input("Your name please:")
6  drink = random.choice(menu)
7  print("Hello", name, "! Enjoy your", drink)
```

【H3】程序流程图绘制

```
# 猜数字游戏
import random

secret = random.randint(1, 100)
print('''猜数游戏!
我想了一个1-100的整数，你最多可以猜6次，
看看能猜出来吗? ''')
tries = 1
while tries <= 6:
    guess = int(input("1-100的整数，第%d次猜，请输入： " % (tries,)))
    if guess == secret:
        print("恭喜答对了！你只猜了%d次！\n就是这个：%d！ " % (tries, secret))
        break
    elif guess > secret:
        print("不好意思，你的数大了一点儿！ ")
    else:
        print("不好意思，你的数小了一点儿！ ")
    tries += 1
else:
    print("哎呀！怎么也没猜中！再见！ ")
```

Python语言概览

数据对象和组织

- 对现实世界实体和概念的抽象
- 分为简单类型和容器类型
- 简单类型用来表示值
  - 整数int、浮点数float、复数complex、逻辑值bool、字符串str
- 容器类型用来组织这些值
  - 列表list、元组tuple、集合set、字典dict
- 数据类型之间几乎都可以转换

赋值和控制流

- 对现实世界处理和过程的抽象
- 分为运算语句和控制流语句
- 运算语句用来实现处理与暂存
  - 表达式计算、函数调用、赋值
- 控制流语句用来组织语句描述过程
  - 顺序、条件分支、循环
- 定义语句也用来组织语句，描述一个包含一系列处理过程的计算单元
  - 函数定义、类定义

Python数据类型：整数int、浮点数float

- 整数最大的特点是不限制大小
- 浮点数受到17位有效数字的限制
  - IEEE 754标准
- 常见的运算包括加、减、乘、除、整除、求余、幂指数等
- 浮点数的操作也差不多
  - 判断相等要特别注意
- 一些常用的数学函数如sqrt/sin/cos等都在math模块中
  - import math
  - math.sqrt(2)

```
>>> 5
5
>>> -100
-100
>>> 5 + 8
13
>>> 90 - 10
80
>>> 4 * 7
28
>>> 7 / 2
3.5
>>> 7 // 2
3
>>> 7 % 3
1
>>> 3 ** 4
81
>>> 2 ** 100
1267650600228229401496703205376
>>> divmod(9, 5)
(1, 4)
>>>
```

整数的进制：用几个符号表示数?

进制	表示	例子
十进制decimal	无前缀数字	367
二进制binary	0b前缀	0b101101111
八进制octal	0o前缀	0o557
十六进制 hexadecimal	0x前缀	0x16f

- 可以用各种进制表示整数
- 也可以转为字符串
  - `str(), bin(), oct(), hex()`
- 浮点数可以转为十六进制
  - `float.hex()`

```
>>> float.hex(1.23)
'0x1.3ae147ae147aep+0'
>>> (1.23).hex()
'0x1.3ae147ae147aep+0'
```

浮点数的精度问题

- 计算机内部用二进制保存数值，
- 十进制的有限小数转为二进制可能变成无限循环小数
  - $(0.1)_{10}=(0.0001100110011001\ldots)_2$
  - 四舍五入将产生误差
- 浮点数判断相等不能简单用相等关系符判断
- 可以视数值取小数点后固定位数进行四舍五入再判断相等

```
>>> 0.2+0.1
0.30000000000000004
>>> 0.2+0.1==0.3
False
>>> round(0.2+0.1, 10)==round(0.3, 10)
True
>>>
```

数值常见的运算和比较

运算符	功能	备注
m + n	加法	
m - n	减法	
m * n	乘法	
m // n	整数除法	结果是商的整数部分
m / n	除法	“真”除法，得到小数
m % n	求余数	
divmod(m, n)	求整数除法和余数	会得到两个整数，一个是 m // n，另一个是 m % n
m ** n	求乘方	整数 m 的 n 次方
abs(m)	求绝对值	
m == n	相等比较	m 是否等于 n
m > n	大于比较	m 是否大于 n
m >= n	大于或等于比较	m 是否大于或者等于 n
m < n	小于比较	m 是否小于 n
m <= n	小于或等于比较	m 是否小于或者等于 n

- 可以进行连续比较判断

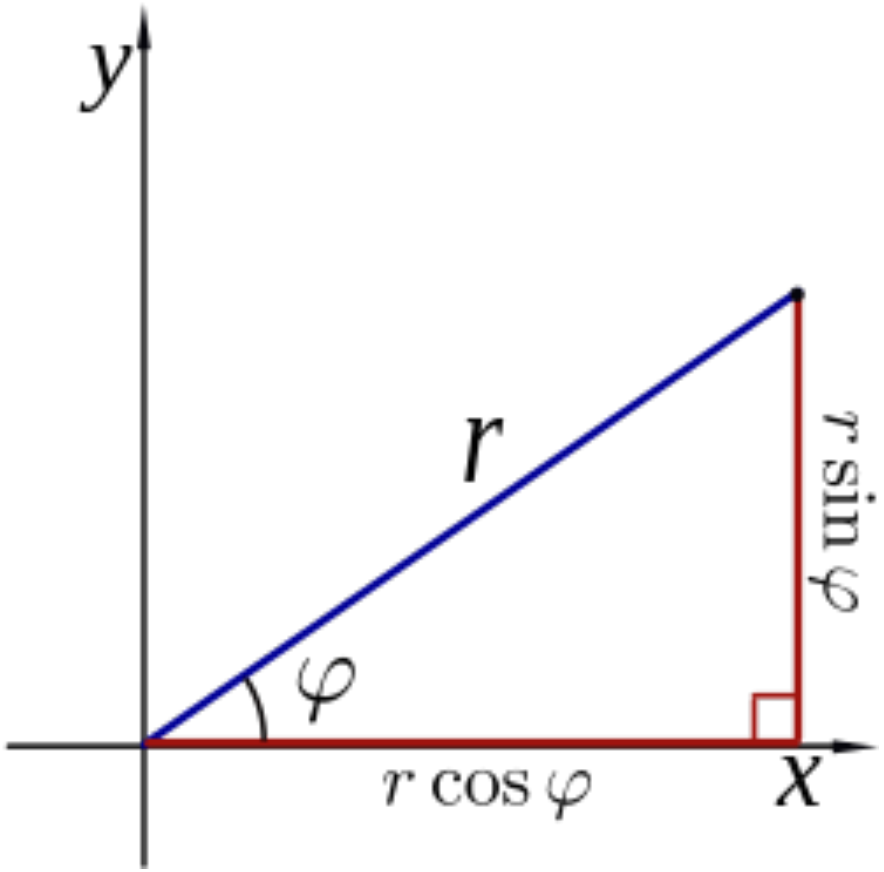
```
>>> 7 > 3 >= 3
True
>>> 12 < 23 < 22
False
>>> m, n = 4, 8
>>> 1 <= m < n <= 10
True
```

Python数据类型：复数

- Python内置复数类型
  - <class 'complex'>
- 支持所有常见的复数计算
  - abs函数支持复数取模运算
- 对复数处理的数学函数在模块cmath中
  - import cmath
  - cmath.sqrt(1+2j)

```
>>> 1+3j
(1+3j)
>>> (1+2j)*(2+3j)
(-4+7j)
>>> (1+2j)/(2+3j)
(0.6153846153846154+0.07692307692307691j)
>>> (1+2j)**2
(-3+4j)
>>> (1+2j).imag
2.0
>>> (1+2j).real
1.0
>>>
>>> abs(1+2j)
2.23606797749979
```

Python数据类型：复数的形式转换



polar：极坐标
rect：直角坐标

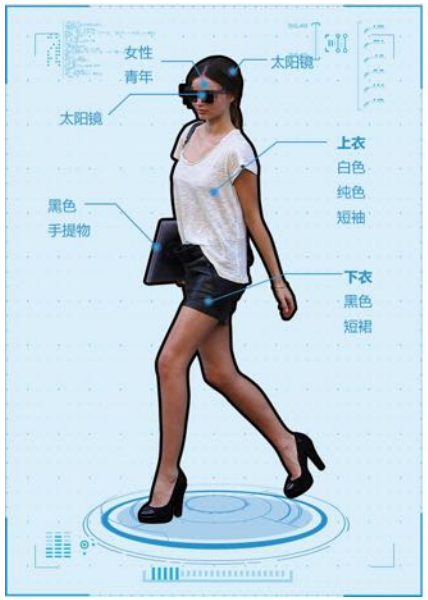
```
>>> import cmath
>>> cmath.polar(3+4j)
(5.0, 0.9272952180016122)
>>> cmath.rect(1, cmath.pi)
(-1+1.2246467991473532e-16j)
>>>
```

数据对象和命名

- Python语言中几乎所有的事物都是对象（Object）
  - 对象有类型（type）和值（value)
  - 对象有独一无二的标识（id）
  - 对象有一些属性（attribute）
  - 对象还有行为（方法method）

```
>>> type(2019)
<class 'int'>
>>> id(2019)
4452715952
>>> dir(2019)
['__abs__', '__add__', '__and__', '__bool__', '__
r__', '__dir__', '__divmod__', '__doc__', '__eq__
_floordiv__', '__format__', '__ge__', '__getattr
t__', '__hash__', '__index__', '__init__', '__in
vert__', '__le__', '__lshift__', '__lt__', '__mo
g__', '__new__', '__or__', '__pos__', '__pow__',
mod__', '__reduce__', '__reduce_ex__', '__repr__
```

- 例如：某个人（object）
  - 人类（type），物质躯体（value）
  - 此人有独特的生理标识（id）
  - 此人有一些特征（attribute）
  - 此人还可以做一些事（method）



给数据对象命名：赋值（assignment）

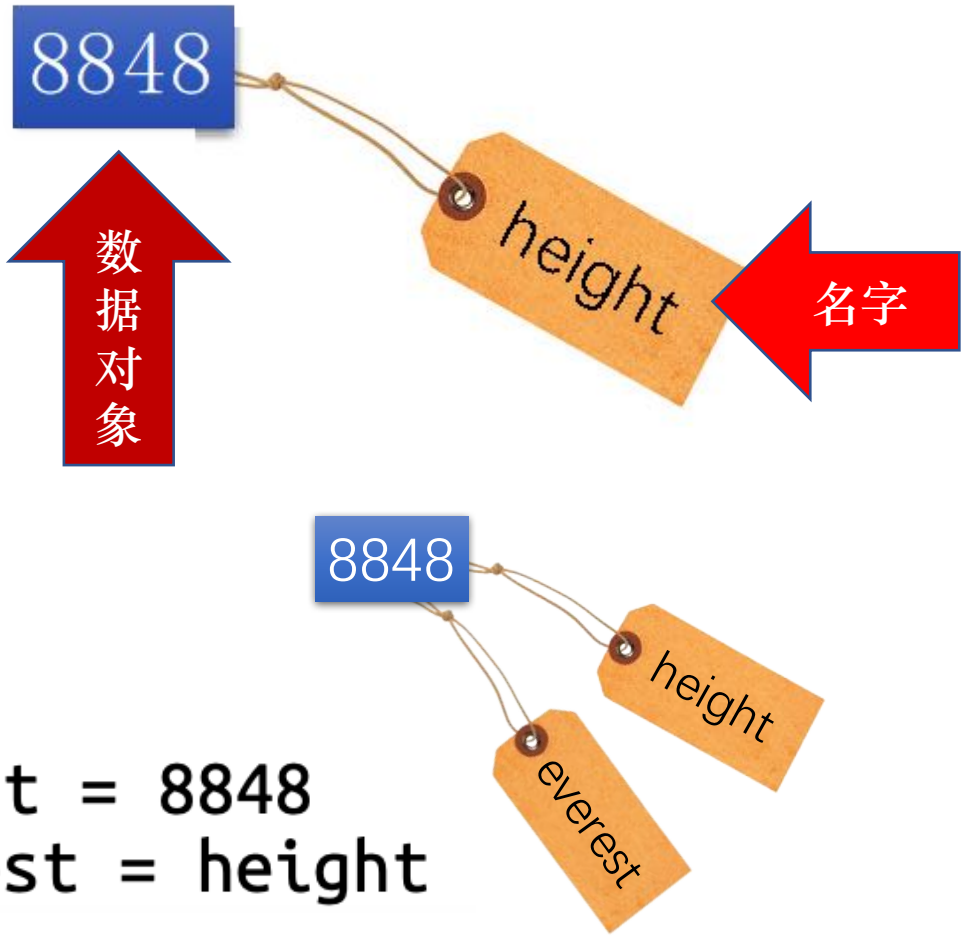
- 赋值语法
  - <名字> = <数据对象>

```
>>> n = 1
>>> m = 3.14
>>> msg = "hello"

>>> OK = 89
>>>  me = "chen"
>>> 身高 = 1.85
>>> 1k = 1000
SyntaxError: invalid syntax
```

- 名字规则
  - 字母和数字组合而成，下划线“_”算字母，字母区分大小写
  - 不带特殊字符（如空格、标点、运算符等）
  - 名字的第一个字符必须是字母，而不能是数字
  - （注：汉字算是字母）
- 起名的艺术

名字（Name）与变量（Variable）

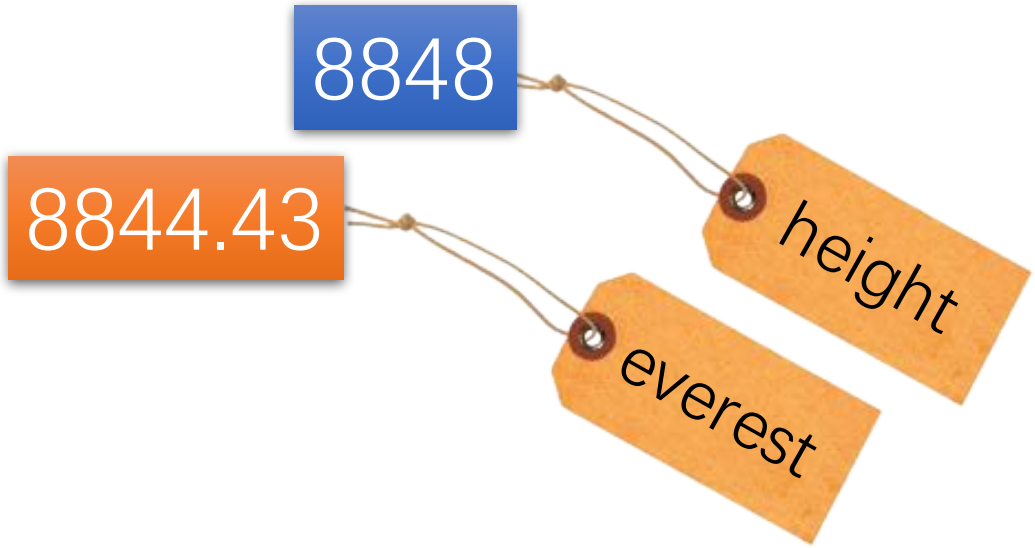
- 名字像一个标签，通过赋值来“贴”在某个数据对象上
- 名字和数据对象的关联，称为引用。
- 关联数值后的名字，就拥有了数据对象的值（value）、类型（type）和标识（id）
- 一个数据对象可以和多个名字关联



名字（Name）与变量（Variable）

- 与数值关联的名字也称作变量，表示名字的值和类型可以随时变化。

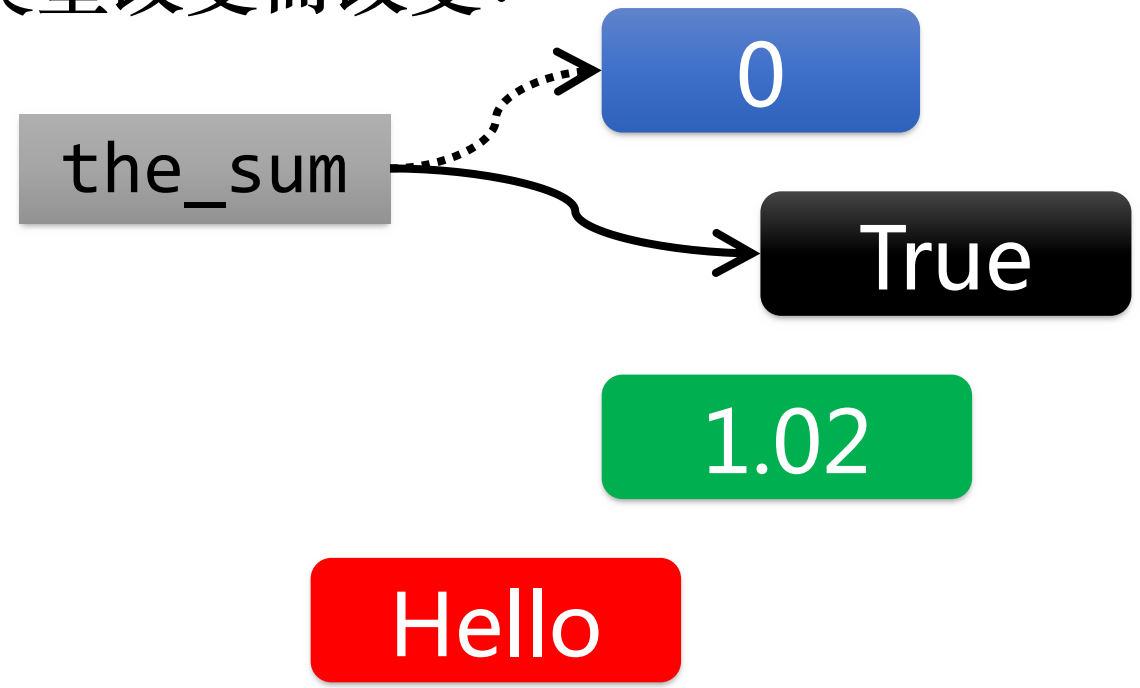
```
height = 8848
everest = height
everest = 8844.43
```



名字（Name）与变量（Variable）

- 变量可以随时指向任何一个数据对象
  - 比如True，1.02，或者"Hello"
- 变量的类型随着指向的数据对象类型改变而改变！

```
>>> the_sum = 0
>>> type(the_sum)
<class 'int'>
>>> the_sum = True
>>> type(the_sum)
<class 'bool'>
```



灵活多变的赋值语句

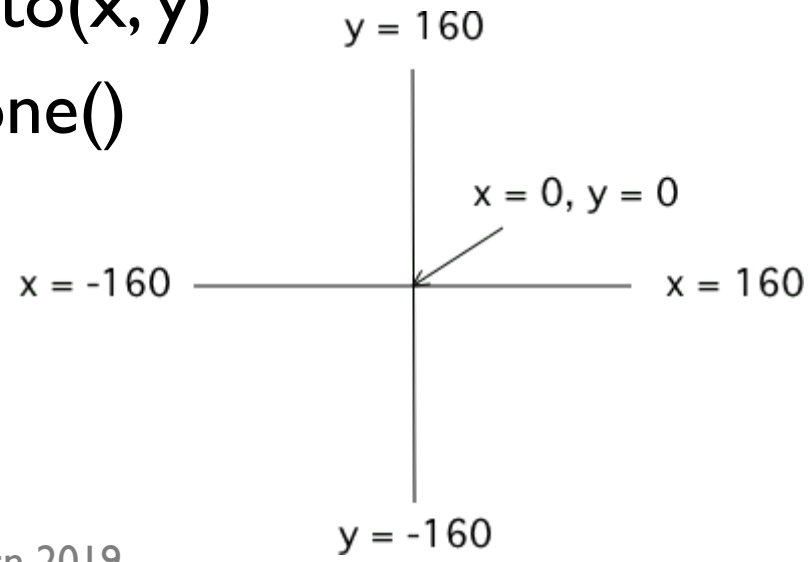
- 最基本的赋值语句形式
  - <名字> = <数据对象>
- 合并赋值
  - a = b = c = 1
- 按顺序依次赋值
  - a, b, c = 7, 8, 9

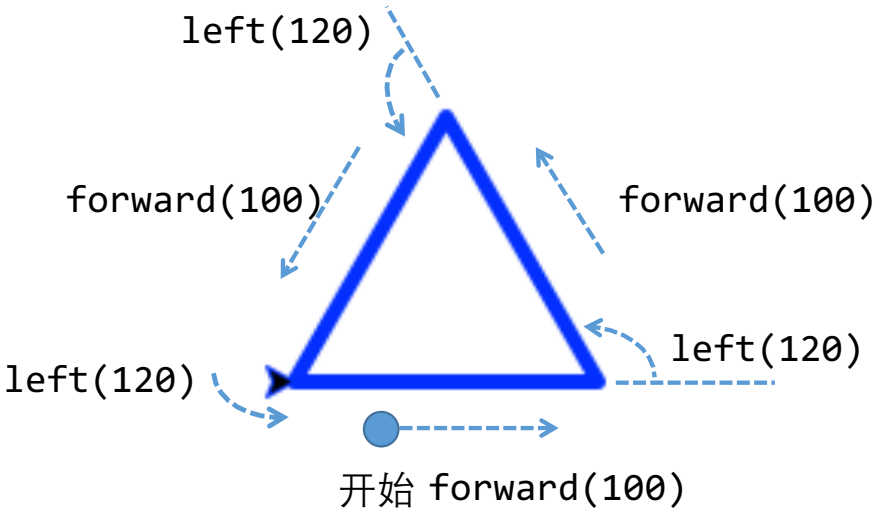
- 简写自操作赋值语句
  - price += 1
  - price *= 1.5
  - price /= 3 + 4

```
>>> a = b = c = 1
>>>
>>> a, b, c = 7, 8, 9
>>>
>>> price = 120
>>> price += 1
>>> price *= 1.5
>>> price /= 2 + 1
```

海龟做图：turtle

- 模拟海龟在沙滩上爬行所描绘的轨迹，从LOGO语言借鉴而来
  - 前进forward(n)；后退backward(n)；左转left(d)；右转right(d)
- 画笔：抬起落下、颜色、粗细
  - 抬起penup()；落下pendown()；笔色color()；笔粗细pensize(n)
- 直接定位：goto(x, y)
- 结束绘制：done()

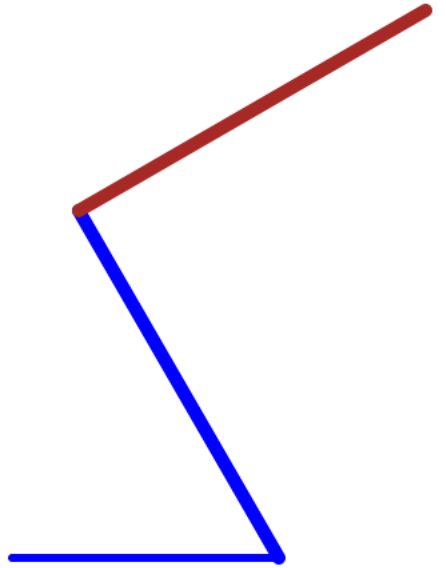




作图程序模版

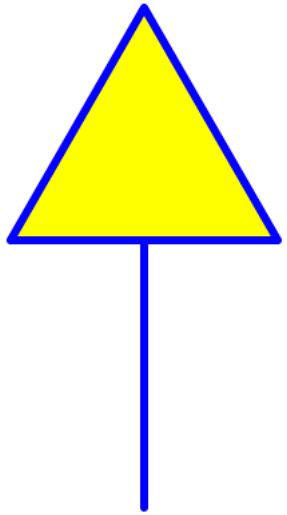
- 首先，导入turtle模块
- 然后，生成一只海龟
  - 可以做一些初始化设定
- 程序主体：用作图语句绘图
- 最后结束作图
  - 可选隐藏海龟：t.hideturtle()

```
1   # 1. 导入海龟模块
2   import turtle
3
4   # 2. 生成一只海龟，做一些设定
5   t = turtle.Turtle()
6   t.color("blue")
7   t.pensize(3)
8
9   # 3. 用海龟作图
10  t.forward(100)
11  t.right(60)
12  t.pensize(5)
13  t.backward(150)
14  t.left(90)
15  t.color("brown")
16  t.forward(150)
17
18  # 4. 结束作图
19  t.hideturtle()
20  turtle.done()
```



样例：警示牌

- 填充fill
  - 设定填充颜色fillcolor
  - 开始填充begin_fill
  - 结束填充end_fill



```
4   # 2. 生成一只海龟，做一些设定
5   t = turtle.Turtle()
6   t.pencolor("blue")
7   t.pensize(3)
8   t.fillcolor("yellow")
9
10  # 3. 用海龟作图
11  t.lt(90)
12  t.fd(100)
13  t.rt(90)
14  t.begin_fill()
15  t.fd(50)
16  t.left(120)
17  t.fd(100)
18  t.left(120)
19  t.fd(100)
20  t.left(120)
21  t.fd(50)
22  t.end_fill()
```

海龟函数的小结

- 前进forward(n)后退backward(n)
  - 缩写：fd(n)、bk(n)
- 左转left(n)、右转right(n)
  - 缩写：lt(n)、rt(n)
- 画笔
  - 笔画颜色pencolor(颜色名称)
  - 笔画粗细pensize(n)
- 抬笔penup()、落笔pendown()
  - 缩写pu()、pd()
- 画圆： circle(半径, 角度)
- 画点： dot(大小, 颜色)
- 填充
  - 填充颜色fillcolor(颜色名称)
  - 填充开始begin_fill()
  - 填充结束end_fill()
- 坐标控制
  - 直接到达goto(x,y)
  - 获取坐标position()
  - 计算距离distance(x,y)

随堂作业：绘制完整的三角警示牌

- 填充： begin_fill, end_fill
- 画圆： circle
- 画点： dot

逻辑类型：bool

- 用来作为判断条件，是逻辑推理的基础
  - 仅有两个值：True、False
- 数值的比较得到逻辑值
  - 3 > 4
- 逻辑值也有自己的运算
  - and，or，not
- 可以让计算机根据情况自动作出选择，更加聪明

```
>>> type(True)
<class 'bool'>
>>> 1 + 1 == 2
True
>>> 3 > 4
False
>>> n = 5
>>> 1 < n < 10
True
>>> (n > 1) and (n < 10)
True
>>> (1 + 1 == 2) or (3 > 4)
True
>>> not True
False
>>> not (3 > 4)
True
```

Python数据类型：逻辑值

- 逻辑值用来配合if/while等语句做条件判断
- 其它数据类型可以转换为逻辑值：
  - 数值：0与非0
  - 字符串：空串与非空串
  - 容器：空容器与非空容器
  - None是False

```
>>> True
True
>>> False
False
>>> 1>2
False
>>> 23<=34
True
>>> bool(0)
False
>>> bool(999)
True
>>> if (2>1):
        print ("OK")


OK
>>>
```

判断一个日期是否合法?

- m月d日
- 考虑合法的条件:
- m在1，3，5，7，8，10，12中，同时，d在1～31之间;

```
>>> m in (1,3,5,7,8,10,12) and (1<=d<=31)
```

- 或者，m在4，6，9，11中，同时，d在1～30之间;
- 或者，m是2，d在1～29之间。

随堂作业：判断日期是否合法

- 先写一个逻辑表达式，判断整数变量y是否闰年?
- 例如：y % 4 == 0
- 请继续补充，用and，or
- 可以与邻座同学讨论
- 再结合前面的日期判断

字符串类型：str

- 文字字符构成的序列（“串”）
  - 可以表示姓名、手机号、快递地址、菜名、诗歌、小说
- 用双引号或者单引号都可以表示字符串
  - 多行字符串用三个连续单引号表示
- 字符串操作：
  - +连接、*复制、len长度
  - [start:end:step]用来提取一部分（切片slice）

```
>>> 'abc'
'abc'
>>> "abc"
'abc'
>>> '''abc def
ghi jk'''
'abc def\nghi jk'
>>> "Hello\nWorld!"
'Hello\nWorld!'
>>> print ("Hello\nWorld!")
Hello
World!
>>> 'abc' + 'def'
'abcdef'
>>> 'abc' * 4
'abcabcabcabc'
>>> len('abc')
3
>>> 'abcd'[0:2]
'ab'
>>> 'abcd'[0::2]
'ac'
```

练习一下

- 用切片方法，提取出身份证号中的生日
- c="632622199907012570"
- 多试几个：
  - 350122200101076320
  - 13062319950406804X

```
1    c="632622199907012570"
2
3    # 13062319950406804X
4
5    # 350122200101076320
```

字符串： str

- 字符来自一个国际标准的大字符集Unicode
- 每种语言的字符都有一个编码
  - 包括表情符号😄👍
- 可以用函数在编码和字符之间转换
  - chr：编码到字符
  - ord：字符到编码

学校 School

二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形
0010 0000	32	20	(空格)	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(	0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29	)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k

```
>>> ord("A")
65
>>> ord("中")
20013
```

```
>>> chr(66)
'B'
>>> chr(20012)
'丬'
```

练习一下

- 用chr/ord函数找到你姓名的unicode编码，并且验证一下
- 再多试几个你关心的人名

```
>>> ord("A")
65
>>> ord("中")
20013
```

```
>>> chr(66)
'B'
>>> chr(20012)
'丬'
```

字符串str和字节串bytes

- Python语言中的字符串是unicode字符的串
- 可以通过encode()方法转换为各种字符编码的字节串bytes
  - 指定字符编码如gb2312, gbk等
- 而字节串则可以通过decode()方法转换为字符串str
  - 字节串属于特定字符编码

```
>>> type('中文')
<class 'str'>
>>> '中文'.encode()
b'\xe4\xb8\xad\xe6\x96\x87'
>>> '中文'.encode('gb2312')
b'\xd6\xd0\xce\xc4'
>>> type(b'\xd6\xd0')
<class 'bytes'>
>>> b'\xd6\xd0'.decode('gb2312')
'中'
>>>
```

Python数据类型：字符串

- 一些高级操作：
  - split：分割；join：合并
  - upper/lower/swapcase：大小写相关
  - ljust/center/rjust：排版左中右对齐
  - replace：替换子串

```
>>> 'You are my sunshine.'.split(' ')
['You', 'are', 'my', 'sunshine.']
>>> '-'.join(["One", "for", "Two"])
'One-for-Two'
>>> 'abc'.upper()
'ABC'
>>> 'aBC'.lower()
'abc'
>>> 'Abc'.swapcase()
'aBC'
>>> 'Hello World!'.center(20)
'    Hello World!    '
>>> 'Tom smiled, Tom cried, Tom shouted'.replace('Tom', 'Jane')
'Jane smiled, Jane cried, Jane shouted'
```

类型转换

- 可以把一个数据对象转换类型，得到新的数据对象
  - "8848"，"8844.43"：字符串
  - 8848：整数
  - 8844.43：浮点数
- 用类型名称可以直接转换
  - 字符串转数值：int()、float()
  - 数值转字符串：str()、bin()、oct()、hex()

```
>>> int("8848")
8848
>>> float("8844.43")
8844.43
>>> str(8848)
'8848'
>>> bin(8848)
'0b10001010010000'
>>> oct(8848)
'0o21220'
>>> hex(8848)
'0x2290'
```

```
>>> hex(8844.43)
Traceback (most recent call last):
  File "<pyshell#134>", line 1, in <module>
    hex(8844.43)
TypeError: 'float' object cannot be interpreted as an integer
>>> (8844.43).hex()
'0x1.146370a3d70a4p+13'
```

上机练习：基本数据类型

- 数值基本运算：33和7
  - +, -, *, /, //, %, **
  - hex(), oct(), bin()
- 类型转换
  - 1，0，'abc', None, 1.2, False, ''
  - str(), bool(), int(), float()
  - is None, ==, !=
- 字符串基本操作
  - +, *, len(), [], in
  - ord(), chr()
  - 含有中文的字符串
- 字符串高级操作
  - s='abcdefg12345'
    - 切片：获得defg12，获得fg12345，获得54321， 获得aceg2
  - t='Mike and Tom'
    - split拆分、
    - upper/lower/swapcase修改大小写、
    - ljust/center/rjust排版30位宽度左中右对齐
    - replace将Mike替换为Jerry

Python容器类型：列表和元组

- Python中有几种类型是一系列元素组成的序列，以整数作为索引
- 字符串str是一种同类元素的序列
- 列表list和元组tuple则可以容纳不同类型的元素，构成序列
- 元组是不可更新（不可变）序列
  - 字符串也是不能再更新的序列
- 列表则可以删除、添加、替换、重排序列中的元素
  - 可变类型

字符串str

0	1	2	3	4	5	6	7	8
H	*e*	*l*	*l*	*o*		*T*	*o*	*m*

列表list

0	1	2	3	4	5	6	7	8
123	2.4	'ab'	True	None	[1,2]	(2,3)	556	0

元组tuple

0	1	2	3	4	5	6	7	8
123	2.4	'ab'	True	None	[1,2]	(2,3)	556	0

容器类型：列表和元组

- 创建列表：[ ]或者list()
- 创建元组：( )或者tuple()
- 用索引[*n*]获取元素（列表可变）
- +：连接两个列表/元组
- *：复制n次，生成新列表/元组
- len()：列表/元组中元素的个数
- in：某个元素是否存在
- [start : end : step]：切片

```
>>> []
[]
>>> list()
[]
>>> alist = [1, True, 0.23
>>> alist[0]
1
>>> alist + ["Hello"]
[1, True, 0.234, 'Hello']
>>> alist * 2
[1, True, 0.234, 1, True,
>>> len(alist)
3
>>> 1 in alist
True
>>> alist
[1, True, 0.234]
>>> alist[1:3]
[True, 0.234]
>>> alist[0:3:2]
[1, 0.234]
>>> alist[::-1]
[0.234, True, 1]
```

```
>>> ()
()
>>> tuple()
()
>>> atuple = (1, True, 0.234)
>>> atuple[0]
1
>>> atuple + ("Hello",)
(1, True, 0.234, 'Hello')
>>> atuple * 2
(1, True, 0.234, 1, True, 0.234)
>>> len(atuple)
3
>>> 1 in atuple
True
>>> atuple
(1, True, 0.234)
>>> atuple[1:3]
(True, 0.234)
>>> atuple[0:3:2]
(1, 0.234)
>>> atuple[::-1]
(0.234, True, 1)
```

```
>>> alist[0] = False
>>> alist
[False, True, 0.234]
```

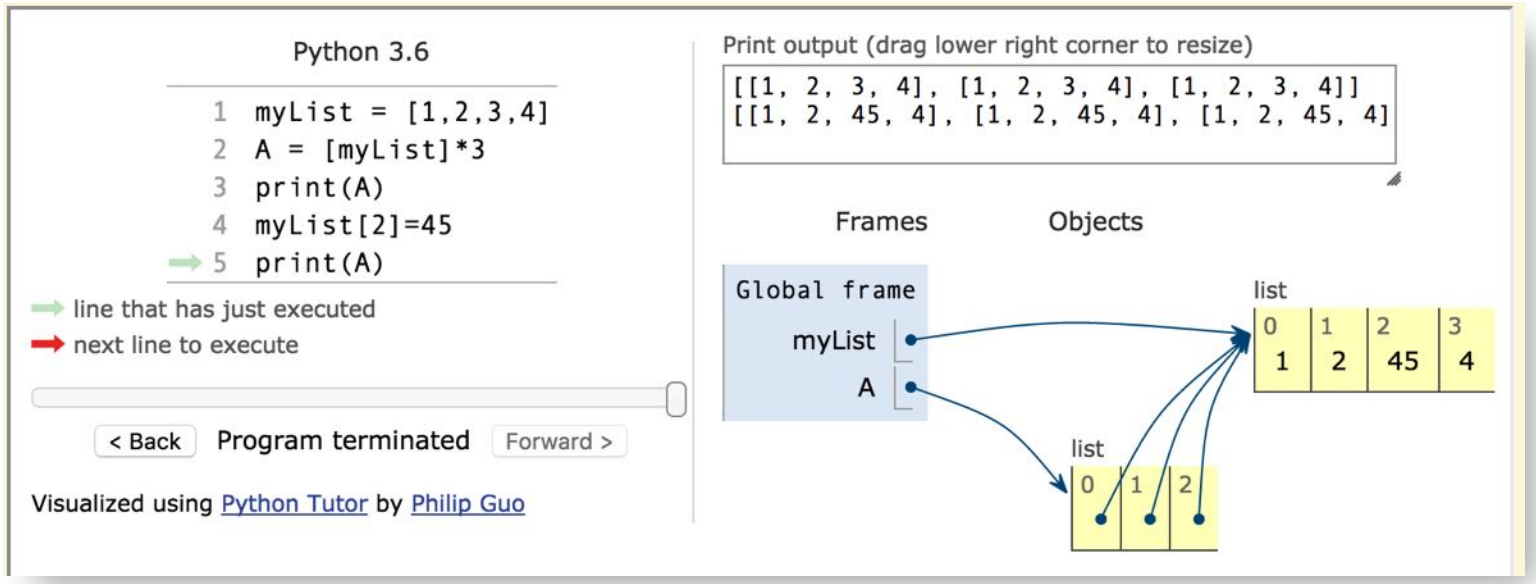
```
>>> atuple[0] = False
Traceback (most recent call last):
  File "<pyshell#93>", line 1, in <module>
    atuple[0] = False
TypeError: 'tuple' object does not support item assignment
```

列表list的其它方法

方法名称	使用例子	说明
append	alist.append(item)	列表末尾添加元素
insert	alist.insert(i,item)	列表中i位置插入元素
pop	alist.pop()	删除最后一个元素，并返回其值
pop	alist.pop(i)	删除第i个元素，并返回其值
sort	alist.sort()	将表中元素排序
reverse	alist.reverse()	将表中元素反向排列
del	del alist[i]	删除第i个元素
index	alist.index(item)	找到item的首次出现位置
count	alist.count(item)	返回item在列表中出现的次数
remove	alist.remove(item)	将item的首次出现删除

可变类型的变量引用情况

- 由于变量的引用特性，可变类型的变量操作需要注意
- 多个变量通过赋值引用同一个可变类型对象时
- 通过其中任何一个变量改变了可变类型对象
- 其它变量也看到了改变
  - alist = [1,2,3,4]
  - blist = alist
  - blist[0] = 'abc'
  - clist = alist[:]
  - Clist[0] = None



常用的连续序列生成器：range函数

- range(n)
  - 从0到n-1的序列
- range(start, end)
  - 从start到end-1的序列
- range(start, end, step)
  - 从start到end-1，步长间隔step
  - step可以是负数
- range函数返回range类型的对象，可以直接当做序列用，也可以转换为list或者tuple等容器类型

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(5, 10))
[5, 6, 7, 8, 9]
>>> list(range(1, 10, 2))
[1, 3, 5, 7, 9]
>>> list(range(10, 1, -2))
[10, 8, 6, 4, 2]
>>> range(10)
range(0, 10)
>>> tuple(range(10))
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

Python容器类型：集合set

- 集合是不重复元素的无序组合
- 用set()从其它序列转换生成集合
- 集合的常见操作
  - in：判断元素是否属于集合
  - |, union()：并集
  - &, intersection()：交集
  - -, difference()：差集
  - ^, symmetric_difference()：异或
  - <=,<,>=,>：子集/真子集/超集/真超集

```
>>> set()
set()
>>> aset = set('abc')
>>> aset
{'c', 'a', 'b'}
>>> 'a' in aset
True
>>> aset | set('bcd')
{'c', 'd', 'a', 'b'}
>>> aset & set(['b', 'c', 'd'])
{'c', 'b'}
>>> aset - set(('b', 'c', 'd'))
{'a'}
>>> aset ^ set('bcd')
{'a', 'd'}
>>> aset <= set('abcd')
True
>>> aset > set('abcd')
False
```

Python容器类型：集合set

- add(x)：集合中添加元素
- remove(x)：删除指定元素
- pop()：删除集合中任意元素并返回其值
- clear()：清空集合成为空集
- 如果经常需要判断元素是否在一组数据中，这些数据的次序不重要的话，推荐使用集合，可以获得比列表更好的性能

```
>>> aset
{'c', 'a', 'b'}
>>> aset.add(1.23)
>>> aset
{'c', 1.23, 'a', 'b'}
>>> aset.remove('b')
>>> aset
{'c', 1.23, 'a'}
>>> aset.pop()
'c'
>>> aset
{1.23, 'a'}
>>> aset.clear()
>>> aset
set()
```

Python容器类型：字典dict

- 字典是通过键值key来索引元素value，而不是象列表是通过连续的整数来索引
- 字典是可变类型，可以添加、删除、替换元素
- 字典中的元素value没有顺序，可以是任意类型
- 字典中的键值key须是不可变类型（数值/字符串/元组）

```
>>> student = {'name':'Tom', 'age':20,
           'gender':'Male', 'course':['math', 'computer']}
>>> student
{'name': 'Tom', 'age': 20, 'course': ['math', 'computer'], '
gender': 'Male'}
>>> student['name']
'Tom'
>>> student['age']
20
>>> student['age'] = 19
>>> student
{'name': 'Tom', 'age': 19, 'course': ['math', 'computer'], '
gender': 'Male'}
>>> student['course'].append('chemistry')
>>> student
{'name': 'Tom', 'age': 19, 'course': ['math', 'computer', 'c
hemistry'], 'gender': 'Male'}
>>> 'gender' in student
True
>>> student.keys()
dict_keys(['name', 'age', 'course', 'gender'])
>>> student.values()
dict_values(['Tom', 19, ['math', 'computer', 'chemistry'], '
Male'])
>>> student.items()
dict_items([('name', 'Tom'), ('age', 19), ('course', ['math'
, 'computer', 'chemistry']), ('gender', 'Male')])
```

建立大型数据结构

- 嵌套列表
  - 列表的元素是一些列表
  - alist[i][j]
- 字典的元素可以是任意类型，甚至也可以是字典
  - bands={'Marxes':['Moe','Curly']}
- 字典的键值可以是任意不可变类型，例如用元组来作为坐标，索引元素
  - poi={(100,100):'bus stop'}

```
>>> alist=[ [23, 34, 45], [True, 'ab']]
>>> alist[0][2]
45
>>> bands={'Marxes':['Moe','Curly'], 'KK':[True, 'moon']}
>>> bands['KK'][0]
True
>>> poi={(100,100):'Zhongguancun', (123,23):'Pizza'}
>>> poi[(100,100)]
'Zhongguancun'
```

上机练习

- 列表、元组基本操作
  - +, *, len(), [], in
- 列表、元组高级操作
  - mylist=[1,2,3,4,5]
  - 切片：获得[2,3,4]，获得[3,4,5]，获得[3,2,1]， 获得[1,3,5]
  - mytpl=(1,2,3,4,5)同上操作
  - t='Mike and Tom'
  - split拆分、join合成为'Mike/and/Tom'
- 集合基本操作
  - a=set([1,2,3,4,5])
  - b=set([2,4,6,8,10])
  - 并、交、差、异或、子集
  - 添加、删除、是否空集
- 字典基本操作
  - mydict = { 1:'Mon', 'line1':3332 }
  - 添加、删除、是否空字典
  - 取字典所有的key／value
  - 判断key是否存在

获取输入：input函数

- 用户给程序的数据在他脑子里，怎么告诉计算机呢?
- input函数通过键盘获取用户输入的字符串
  - 以回车符作为输入结束，一行
- 可以加一个提示符
- 可以把得到的字符串直接转换成其他数据类型

```
>>> x = input("x :")
x :7
>>> y = input("y :")
y :8
>>> x + y
'78'
>>> type(x)
<class 'str'>
>>>
>>> x = int(input("x :"))
x :7
>>> y = int(input("y :"))
y :8
>>> x + y
15
>>> type(x)
<class 'int'>
```

打印输出：print函数

- 计算机把处理结果反馈给用户
- 用print在屏幕上显示数据对象或者变量的值
- print(v1, v2, v3, ...)
- 格式化字符串f-strings
  - f"Hello, {name}!"
  - f"{name}, you have tried {n} times."
- 可选的参数
  - sep=" ", end="\n"

```
>>> name = "Tim"
>>> n = 7
>>> print("Hello", name, "!")
Hello Tim !
>>> print(f"Hello {name}!")
Hello Tim!
>>> print(f"{name}, you have tried {n} times.")
Tim, you have tried 7 times.
>>> print(name, n, sep="#")
Tim#7
```

写一个完整的Python程序

① 导入模块 import
- 用import导入需要用到的模块；

② 定义函数 def
- 根据需要定义一批函数；

③ 获取数据 input
- 从键盘输入或者文件读入需要处理的数据；

④ 计算处理
- 按照设计好的算法来进行计算或者处理数据；

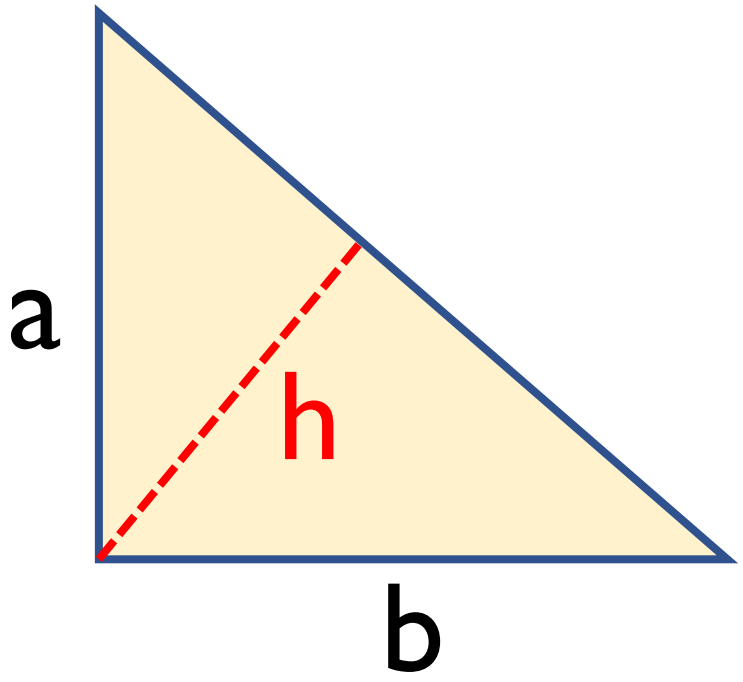
⑤ 输出结果 print
- 将结果输出到屏幕或者写入文件中。

```
# 程序功能：
# 找到不小于用户输入数的最小质数

# 1，导入需要的模块
import math

# 2，定义函数
def isprime(n):
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    else:
        return True

# 3，获取用户输入的数据
n = int(input("Please input an integer:"))

# 4，开始计算搜寻
temp = n
while not isprime(temp):
    temp = temp + 1

# 5，输出结果
print("Next prime number is:", temp)
```

随堂作业：计算直角三角形斜边的高

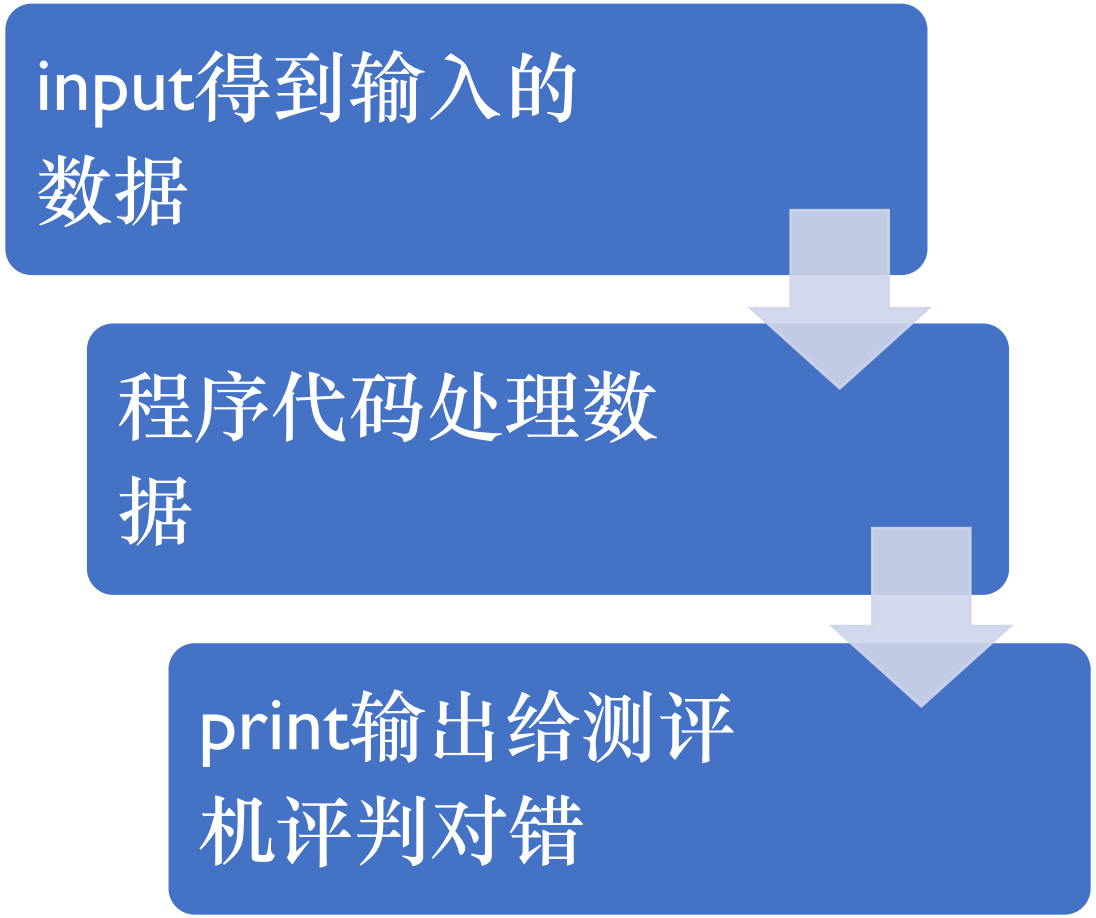
- 写一个完整的程序tc.py
- 要求输入两个直角边长度
  - a, b
- 打印输出斜边上的高h，保留小数点后2位
  - 打印输出如何保留小数点后位数?

```
>>> import math
>>> c = math.sqrt(7)
>>> print(f"c = {c:.5f}")
c = 2.64575
>>> c
2.6457513110645907
```



什么是OJ?

- Online Judge，在线测评
- 会有若干组输入输出来判断程序代码是否正确
- 每组输入输出
  - 输入：由input读入
  - 输出：print输出
- 测评机只判断输出的字符串跟标准答案是否相等
  - 输出分行、分隔符号、小数点位数、大小写等等都要完全一致



关于Online Judge中Python代码的技巧

- 提示：不要在input里加任何提示符的参数！
- 读入数据：一行就一个值
  - `astr = input()`
  - `n = int(input())`
  - `f = float(input())`
- 读入数据：一行多个整数值
  - `alist = list(map(int, input().split()))`

关于Online Judge中Python代码的技巧

- 输出数据：一行就一个值
  - `print(n)`
  - `print("%.2f", f)  # 小数点后两位`
- 输出数据：一行多个值
  - `print(m, n, i)  # 三个整数`
  - `print(" ".join(map(str, alist)))`

【H4】完成OJ题：基本表达式

- 请到如下网址完成练习题：
  - https://vijos.org/d/pkuchenbin/training/5c8e53d5f413620934d099a8
- 做完后将带有Accepted字样的截图上传提交表单

章节 1. 基本的表达式　　进行中

收缩

状态	题目	递交	% AC	难度
✓ Accepted	☆ 1001 A+B问题	180	32	5
✓ Accepted	☆ 1002 反向输出一个三位数	141	33	5
✓ Accepted	☆ 1003 圆面积周长	232	20	7
	☆ 1004 球的体积 RP+70	196	21	7