# Python中的变量与运算

## 写在最前面

python是脚本语言，它和我们认知中的C，C++，java之间存在很大的差异；python是不需要编译的，如果你曾经接触过matlab，那么你可以认为python和matlab之间有更多的相同特点

## 注意

1. python中"#"为注释符号
2. python不同于C语言，可以先声明变量在赋值，python中的变量必须在声明的**同时**予以赋值

```
a = 1 #这行代码是合法的

########################################

b
b = 1 #这两行代码是非法的，运行到b时，就会报错
```

3. 在python中，我们无需声明变量的类型（int float...），解释器会**自动**根据变量被赋予的值 去设置变量的类型

## 基础输入输出

### 输入函数：input()

```
name = input("请输入名字：")
```

在运行上述函数之后，命令行会提示"请输入名字："

我们输入之后按下回车，输入的结果会被保存在变量name中

请注意：“回车”这个输入本身 不会被保存在变量name中；

### 输出函数：print()

```
print("hello,world")
```

注：python中的print也可以使用格式控制，如：

```
a = 1
print("a = %d", a)

# 将输出“a = 1”
```

格式控制的用法和C语言基本相同；

# 变量

## 数字类型

```
a = 1 # 整形
b = 1.0 # 浮点型
b = 1 + 2j # 复数，使用j后缀表示虚部
```

令人惊讶的是python居然提供了"**复数**"类型；怪不得大家都喜欢用python进行数据处理，从变量类型来看python确实为我们提供了很多意想不到的支持；

## 字符串

```
name = "lifugui" # name是一个string类型
name = 'lifugui' # name也是一个string类型
```

python中，用 **单引号** 或者 **双引号** 都可以声明字符串类型的变量

## list/tuple/range

```
# list 序列：作用类似于C语言中的数组，使用中括号定义
my_list = ["hi","my","friend"]
# ！！！值得指出的是，list中的内容物可以不是一个类型，例如：
my_list_2 = ["hi",2] # 在这个例子中，my_list_2中的第0个元素是string型，第1个元素是int型

# 使用index调用list中的元素，如：
print(my_list[0]) # 输出为"hi"

```

```
# tuple 元组：元组一旦定义就无法再修改，使用圆括号定义
my_tuple = ("apple","cherry","banana")
# 和list一样，tuple也可以同时容纳不同类型的元素
# 同样使用index调用元素
```

```
# range 迭代器：创建一个整数数列
# 语法：range(start,end,step)
# 其中：start可以不写，缺省为0；step可以不写，缺省为1；（这个用法看起来像极了matlab...）
# 注意：range实例包含start，但是不包含end
my_range = range(6) # 这个range是0，1，2，3，4，5；不包含6

# range提供了reverse()方法，可以将range实体倒置，例如：
my_range_reversed = reversed(my_range) # 此时，第一个元素是5，而非0
```

这三个家伙里唯一需要解释的是range，它最大的作用是用在循环中，例如：

```
for x in range(10):  #表示从0~9循环
    print(x)
```

**python中的for循环也和C语言中的for循环有较大的差异，暂时按下不表**

## dict 字典

字典的用法是比较多的...字典的精髓是“键-值对”，所有的数据都以"key":"value"的形式一对对的存储

```
# dict 字典：使用"key":"value"的模式存储数据，可以使用"key"调用对应的"value"，使用花括号定义
my_dict={
    "name":"lifugui",
    "age":26
}

# 可以通过key获取value，例如：
my_name = my_dict["name"]
print(my_name)
# 将输出lifugui


# dict类型提供了很多方法，比较常用的有下面几个
# 可以通过values()函数获取字典中的全部"value"
for value in my_dict.values():
    print(value)

# 可以通过items()方法读取"key":"value"对
for x, y in my_dict.items():
    print(x, y)

# 可以通过in判断某个key在不在字典中：
if "name" in my_dict:
    print(my_dict["name"])
else:
    print("my_dict中没有name字段")
```

## set 集合

注意区分set和dict

```
# set集合：集合是一种无序的存储方式，我们无法使用index访问具体的set
# 一旦某元素被add到一个set中，我们就只能删除这个元素，而不能修改；
# set的特点是，查询一个元素是否在set中非常快

my_set = {"1","2","3"} # 实际上我们无从得知“1”，“2”，“3”到底以什么样的顺序在my_set中
# 添加一个元素到my_set
my_set.add("4")
# 从my_set中删除一个元素
my_set.remove("4")
# 判断my_set的长度
length = len(my_set)
# 判断元素“1”是否在my_set中
if "1" in my_set:
    print("在")
else:
    print("不在")
# 我们还可以计算两个set的交集，并集，合集；这里不一一演示
```

# 基本语法

## ！！！注意：python的缩进；非常重要！！！

在学习语法之前，必须要注意！**python使用缩进表示对代码块的划分**...

```
# 例如:
def my_func():
    a = 1
    b = 2
c = 3
# 在这个例子中，我定义了一个叫做my_func的函数，只有a=1和b=2是函数代码块中的内容
# c=3不是函数代码块中的内容，因为c=3没有进行缩进，所以它不属于my_func代码块
```

**虽然python推荐使用四个空格作为缩进手段，不过你也可以选择使用tab缩进！！！但是万分注意！！！一定不能混用tab和四个空格！！！一旦混用将造成灾难性的后果！！！切记！！！**

## if...else...

```
a = 1
if a == 1:  # 需要注意的是在python中，if最后要加分号
    print("a=1")
else:       # else也要加分号
    print("a!=1")
```

```python
# python提供了 elif，即else if，用于二次判断;例如：
if a == 1:
    print("a=1")
elif a == 2:
    print("a=2")
else:
    print("a!=1 并且 a!=2")
```

## for循环

python中的for循环，需要一个**list**或**tuple**或**range**或**dict**实体；python中的for循环，本质上是个遍历，它会挨个将实体中的元素取出来；

```python
# 对一个list进行循环
my_list = ["a","b","c"]
for element in my_list:
    print(element)
# 输出：
# a
# b
# c

# 对tuple进行循环跟list差不多
```

可以看出python中的for循环不像c语言中一样，循环的标志是一个int型数字；python的for循环会直接将循环对象中的元素帮你取出来...

```python
# 对一个range实体循环
my_range = range(3)
for i in my_range:
    print(i)
# 输出：
# 0
# 1
# 2

# 这种循环方式使用起来更像c语言中的那种for循环...
```

## while循环

```python
i = 1
while i<7:
    i += 1
```

用起来和c语言的while几乎一样...没什么好说的，python的while也支持break跳出循环和continue越过本次循环；（为什么要强调python的while支持continue呢...因为确实有一些语言不支持...说的就是你！lua！）