开始编程吧

第二章

目标

- 掌握Python数据类型
- 理解Python运算符
- 掌握Python流程控制
- 掌握Python内置数据结构

案例

- 四则运算；
- 猜数字小程序；

Python数据类型
数字和字符串

第四节

邹琪鲜

案例

- 123和“123”一样吗
- 四则运算器

Python数据类型

- 计算机是用来辅助人们的，在程序设计中也映射了现实世界的分类，以便于抽象的分析。
- 数字
- 字符串
- 列表
- 元组
- 字典

Python数据类型

- 数字类型
  - 整型
  - 长整型
  - 浮点型
  - 复数型

数字类型-整数int

- 整数int表示的范围-2,147,483,648到2,147,483,647
  - 例如：0, 100, -100
- Int的范围示例：
  - >>> num=2147483647
  - >>> type(num)
  - <type 'int'>

数字类型-长整型long

- long的范围很大很大，几乎可以说任意大的整数均可以存储。
- 为了区分普通整数和长整数，需要在整数后加L或小写l。
  - 例如：51856678L，-0x22345L
- 示例：
  - >>> num=11　　（注意后面不是1是小写的L）
  - >>> type(num)
  - <type 'long'>
  - >>> num=99999999999999999999
  - >>> type(num)
  - <type 'long '>

数字类型-浮点float

- 例如： 0.0， 12.0， -18.8， 3e+7等
- 示例：
  - >>> num=0.0
  - >>> type(num)
  - <type 'float'>
  - >>> num=12
  - >>> type(num)
  - <type 'int'>
  - >>> num=12.0
  - >>> type(num)
  - <type 'float'>

数字类型-复数型complex

- Python对复数提供内嵌支持，这是其他大部分软件所没有的；
- 复数举例：3.14j, 8.32e-36j
- 示例：
  - >>> num=3.14j
  - >>> type(num)
  - <type 'complex'>
  - >>> num
  - 3.1400000000000001j
  - >>> print num
  - 3.14j

字符串String

- 使用引号定义的一组可以包含数字，字母，符号（非特殊系统符号）的集合。
  - Strval =‘This is a test!’
  - Strval =“This is a test!”
  - Strval =“””This is a test!”””
    - 三重引号（docstring）通常用来制作字符串，在面向对象时详解
- raw_imput()

小结

- 掌握四种数字类型；
- 掌握字符串类型的三种定义方式；

Python数据类型
序列

第五节

邹琪鲜

案例

- 123和“123”一样吗
- ()[]{}

Python数据类型

- 计算机是用来辅助人们的，在程序设计中也映射了现实世界的分类，以便于抽象的分析。
- 数字
- 字符串
- 列表
- 元组
- 字典

序列

- 列表、元组和字符串都是序列
- 序列的两个主要特点是索引操作符和切片操作符。
  - 索引操作符让我们可以从序列中抓取一个特定项目。
  - 切片操作符让我们能够获取序列的一个切片，即一部分序列。

序列

- 切片操作符（[][:][::]）

- 序列[索引]

[索引]				
倒索引	-5	-4	-3	-2
数值	5	6	7	8
正索引	0	1	2	3

序列

- 索引[index]用来取得序列中的单个项目。也称作是下标操作。
  - 每当你用方括号中的一个数来指定一个序列的时候，Python会为你抓取序列中对应位置的项目。Python的索引从0开始计数。
  - 索引同样可以是负数，位置是从序列尾开始计算的
- 切片操作符是序列名后跟一个方括号，方括号中有一对可选的数字，并用冒号分割。
  - 注意这与你使用的索引操作符十分相似。数是可选的，而冒号是必须的。
  - 切片操作符中的第一个数（冒号之前）表示切片开始的位置，第二个数（冒号之后）表示切片到哪里结束。如果不指定第一个数，Python就从序列首开始。如果没有指定第二个数，则Python会停止在序列尾。注意，返回的序列从开始位置 开始，刚好在 结束 位置之前结束。即开始位置是包含序列切片中的，而结束位置被排斥在切片外。

序列

- 序列类的操作符
  - 成员关系操作符（in not in）
  - 成员关系符就是判断一个字符是否属于这个字符串，再就是这个字符串是否属于这个元组，或者列表。返回值也是布尔值（True，Flase）。
- 连接操作符（“+”）
  - 序列 + 序列
  - 以把2个序列组合到一个新的序列中去。
- 重复操作符（“*”）
  - 序列 * 整数
- 比较
  - < > ==

序列

- 序列的内建 函数
  - str（）
  - list（） 把一个元组，或者字符串转换为一个列表
  - tuple（） 把一个列表，或者字符串转换为一个元组
  - max（） 最大
  - min(（） 最小
  - len （） 统计长度的

元组

- 元组是处理一组有序项目的数据结构，即你可以在一个元组中存储一个序列的项目
  - 元组通常用在使语句或用户定义的函数能够安全地采用一组值的时候，即被使用的元组的值不会改变。

元组

- 创建
  - 元组通过圆括号中用逗号分割的项目定义
  - testtuple=（）
  - testtuple=（“v1”，）
  - testtuple=（“v1”，“v2”，“vn”）
- 访问
  - 遵循序列操作方法，即索引和切片等

列表

- 列表和元组类似，是一个数据的集合，是序列的一种。
  - 字符串只能由字符组成且不能改变
  - 元组是一个数据的集合，不可改变
  - 列表是一个容器，能保留任何python对象。
  - 可以排序，以及对单独元素 插入 或者删除。
  - 由于可以增加或删除项目，我们说列表是可变的数据类型，即这种类型的数据是可以被改变的。

列表

- 创建
  - 列表由[ ] 来定义，用“，”分割元素。

```
a= 567
l=[]
l=[a]
alist = [123,456,"hou",[ 'hou','zai','cun'], a,('uu','jj'')]
```

  - 列表的元素可以是任何元素 上面是例子：数字 变量 子元组 字符串 子列表 。

列表

- 访问列表
  - 遵循序列操作方法，即索引和切片。
- 更新列表
  - L[index] = "value"
  - L.append("value")
- 删除
  - L.remove("value")
  - del(L[index])

对象与类的快速入门

- list_test=[1,2,3]
- 列表是个类别，抽象的描述了列表的特征
- list_test 是列表这个类别当中的一个具体的实例，是可操作的一个对象
- 每个类别都有对应的操作方法，通过对象来操作，比如list_test.append[6]。append就是列表对象的方法
- help（）

字典

- 字典(dictionary)是除列表以外python之中最灵活的内置数据结构类型。
- 列表是有序的对象结合，字典是无序的对象集合。
- 区别在于，字典当中的元素是通过键来存取，而不是通过偏移存取。

字典属性

- 通过键而不是偏移量来读取
- 任意对象的无序集合
- 可变，异构，任意嵌套
- 属于可变映射类型
- 对象引用表（哈希表）

字典创建

- 创建方法①:
- dict2 = {'key': 'value', 'key': value}

```
>>> dict1 = {}
>>> dict2 = {'name': 'earth', 'port': 80}
>>> dict1, dict2
({}, {'port': 80, 'name': 'earth'})
```

字典访问

- 通过Key访问Value
- 单个访问
  - dict['key']
- 遍历

```
for key in dict:
... print 'key=%s, value=%s' % (key, dict[key])
```

- 方法

```
'name' in dict  或 dict.has_key('name')
```

字典更新

- 采取覆盖更新
  - dict['key']='earth';
  - 更新 dict['key']='abc';

删除字典和字典元素

- del dict2['name'] # 删除键为“name”的条目
- dict2.clear()　　# 删除dict2 中所有的条目
- del dict2　　# 删除整个dict2 字典
- dict2.pop('name') # 删除并返回键为“name”的条目

字典内建方法

- 字典key值:dict9.keys()
- 字典值: dict9.values()
- 字典所有项:dict9.items()
- 返回字典值： dict9.get('y')

小结

- 掌握四种数字类型；
- 掌握字符串类型的三种定义方式；