3.Python核心数据类型概览

在Python中，我们使用一些东西在做事情,东西就是我们要操作的对象,事情就是这个对象能做什么, 这是我们本节所要讨论的内容。在Python中数据以对象的形式出现, 它包含了数值以及相关操作的集合，Python的程序可以分解成 模块 语句 表达式 对象.

1.程序由模块组成

2.模块包含语句

3.语句包含表达式

4.表达式建立并且处理对象

为什么使用内置类型 写过C C++这样底层语言的程序员, 在编码过程中要考虑内存分配, 实现搜索(二分查找)等工作, 这些工作非常乏味 难学 容易出错; 而且往往背离你要开发的程序的真正目标, 但是在Python程序中，这些乏味的工作大部分都消失了, 因为Python提供了强大的对象类型作为语言的组成部分, 我们可以在编码的过程中更加专注的去解决怎么去实现程序的最终目标.

核心数据类型

布尔 数字 字符串 列表 元组 集合 字典 文件

In [3]: x = 3 ##数字型 int() float()

In [4]: x = True ##布尔型

In [5]: x = 'apple' In [6]: x = "apple"

In [7]: x = [1,2,3] ##定义列表

In [8]: x = (1, 2,3) ##定义元组

In [9]: x = set([1,2,3]) ##定义集合

In [10]: x = {'x': 1} ##定义字典

In [11]: f = open('123.log', 'w+') ##创建一个文件对象

一旦创建了一个对象，那么这个对象就和他的相应的可操作集合绑定了, 也就是字符串只能做字符串相关的操作, 列表只能做列表相关的操作等

一.字符串

定义: 用来记录文本信息, 是一个序列对象, 序列中的元素包含了一个从左到右的顺序, 序列中的元素可以根据他们的相对位置来进行存储和读取, 元组 列表 也是序列对象, 作为序列, 它支持迭代 也支持位置索引序列操作, 从左到右位置索引从0开始, 右到左位置索引从-1开始

In [4]:
```
x = 'apple' #或 x = "apple" 来定义字符串
##索引操作
x[0]
```

Out[4]: 'a'

In [5]:
```
x[1] ##索引操作
```

Out[5]: 'p'

In [6]:
```
x[-1]##索引操作
```

Out[6]: 'e'

In [7]:
```
##分片操作
x[1:] ##从索引1开始一直到最后
```

Out[7]: 'pple'

In [8]:
```
x[0:3]##从索引0开始到索引3
```

Out[8]: 'app'

In [9]:
```
x[:3]##从索引0开始到索引3
```

Out[9]: 'app'

In [10]:
```
x[:-1]
```

Out[10]: 'appl'

In [11]:
```
y = x[:]##对字符串做了一次拷贝
print y
```

```
apple
```

In [13]:
```
x[::-1]##翻转字符串
```

Out[13]: 'elppa'

In [15]:
```
#特殊的分片: 分片表达式增加了一个可选的第三个索引， 用作步进
x[::2]
```

Out[15]: 'ape'

In [16]:
```
x[::3]
```

Out[16]: 'al'

不可变性

字符串不可以在原处被修改,但是可以通过建立一个新的字符串并以同一个变量名字对其进行赋值来修改

In [18]:
```
x = "apple"
x[0] = 'ccc'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-18-e54100cddcde> in <module>()
      1 x = "apple"
----> 2 x[0] = 'ccc'

TypeError: 'str' object does not support item assignment
```

字符串操作:

字符串可以通过 + 进行拼接， 也可以通过 * 进行重复

In [1]:
```
x = 'apple'
y = "banana"

x + " " + y ##通过 + 对字符串做拼接
```

Out[1]: 'apple banana'

In [2]:
```
x * 4 ##通过 * 运算符对字符串进行重复
```

Out[2]:
```
'appleappleappleapple'
```

In [3]:
```
len(x) ##返回字符串的长度
```

Out[3]:
```
5
```

二.列表

列表也是序列的一种, 支持所有刚刚我们在字符串中讨论的序列操作, 列表支持在原处修改

In [4]:
```
x = [1, 2, 3, 4, 5]

len(x) #返回列表长度
```

Out[4]:
```
5
```

In [5]:
```
x[0] #索引运算
```

Out[5]:
```
1
```

In [6]:
```
x[0:-1] #分片运算
```

Out[6]:
```
[1, 2, 3, 4]
```

In [8]:
```
x[0] = 'banana'  ##列表支持原处修改
print x
```

```
['banana', 2, 3, 4, 5]
```

边界检查

不允许引用不存在的元素, 超出列表末尾元素之外的元素索引, 会导致错误

In [9]:
```
print x[10]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-9-c3e4c370bd90> in <module>()
----> 1 print x[10]

IndexError: list index out of range
```

嵌套:

列表的一个优秀特性就是可以支持任意嵌套， 能够以任意的组合对其进行嵌套， 并且可以多层嵌套

In [10]:
```
#定义一个二维列表
x = [[1,2,3,4], [2,3,4,5]]

print x[1][0] ##二维列表的索引
```

```
2
```

三.元组:

定义元组 x = (1,2,3,4)

In [11]:
```
x = (1,2,3,4)
print x[0] ##索引运算
```

```
1
```

In [13]:
```
len(x) ##返回元组的长度
```

Out[13]:
```
4
```

In [14]:
```
print x[0:2] #分片运算
```

```
(1, 2)
```

In [15]:
```
x[0] = 'apple' #改变元组内的元素，代码会报错， 元组是不可变的
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-576bdb52028a> in <module>()
----> 1 x[0] = 'apple'

TypeError: 'tuple' object does not support item assignment
```

四.集合

集合的特点: 确定性 互异性 无序性 集合的运算: 交集 并集 补集等

集合的定义:

In [19]:
```
x = set([1,2,3,4,5]) ##定义集合的方式1
y = {"x", "y", "z"}

print x
print y
```

```
set([1, 2, 3, 4, 5])
set(['y', 'x', 'z'])
```

集合不是序列类型，所以对集合不可以进行序列的运算 如: 索引 分片

In [17]:
```
print x[0] ##代码报错
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-17-2585ca5362ae> in <module>()
----> 1 print x[0] ##代码报错

TypeError: 'set' object does not support indexing
```

In [18]:
```
len(x) ##通过内置函数len可以获取集合的长度
```

Out[18]:
```
5
```

五.字典

字典不是序列而是一种映射关系, 它通过键来进行值的索引而不是通过位置, 由于字典是映射, 所以它没有任何可靠的从左到右的顺序, 字典是核心数据类型中唯一的一种映射类型, 它也具有可变性

In [21]:
```
x = {'a': 1, 'b': 2, 'c': 3}  ##字典的定义

print x['b']  ##通过索引key来获取相应的value
```

```
2
```

In [22]:
```
x["c"] = "apple" ##字典中的元素具有可变性
print x
```

```
{'a': 1, 'c': 'apple', 'b': 2}
```

六. 布尔型 True or false

Python中真和假的含义 与大多数的程序设计语言一样整数0代表假;整数1代表真. 不过除此之外Python把任意的空数据也认为是假;把任何的非空数据结构视为真; 数字如果非0 则为真; 其他对象如果非空则为真

In [24]:
```
print bool("apple") ##非空字符串为真
print bool(0) #数字0为假
print bool(1) #所有非0数字为真
print bool("")#空字符串为假
print bool([])#空列表为假
print bool(())#空元组为假
print bool(None)#特殊类型None为假
```

```
True
False
True
False
False
False
```

七.数字与运算符 整型: 1,2,3,4,5 浮点型: 3.14, 1.1, 2.2 运算符:

```
+ - * / % ** //
```

比较运算符: > < = <= >= != 逻辑运算符: 布尔与 and
布尔或 or 布尔非 not

八.文件

Python代码对电脑磁盘中的文件操作的主要接口, 如果要创建文件对象, 需要使用open()函数, 对磁盘中的文件进行打开

f = open('123.log', 'w+') ##创建文件对象f

f.write('123') ##写入一行数据到文件对象中

f.close() ##关闭文件对象