Python 入门演示

简单的数学运算

整数相加，得到整数：

In [1]:2 + 2

Out[1]:4

字符串 String

字符串的生成，单引号与双引号是等价的：

In [5]:s = "hello world"

s

Out[5]:'hello world'

In [6]:s = 'hello world'

s

Out[6]:'hello world'

三引号用来输入包含多行文字的字符串：

In [7]:s = """hello

world"""

print s

Out[7] :hello

world

字符串的加法：

In [9]:s = "hello" + " world"

s

Out[9]:'hello world'

符串的分割：

In [13]:s = "hello world"

s.split()

Out[13]:['hello', 'world']

查看字符串的长度：

In [14]:len(s)

Out[14]:11

列表 List

Python用[]来生成列表

In [15]:a = [1, 2.0, 'hello', 5 + 1.0]

a

Out[15]:[1, 2.0, 'hello', 6.0]

列表加法：

In [16]:a + a

Out[16]:[1, 2.0, 'hello', 6.0, 1, 2.0, 'hello', 6.0]

列表索引：

In [17]:a[1]

Out[17]:2.0

列表长度：

In [18]:len(a)

Out[18]:4

向列表中添加元素：怎样添加到自己想要的位置？？？

In [19]:a.append("world")

a

Out[19]:[1, 2.0, 'hello', 6.0, 'world']

集合 Set

Python用{}来生成集合，集合中不含有相同元素。

In [20]:s = {2, 3, 4, 2}

s

Out[20]:{2, 3, 4}

集合的长度：

In [21]:len(s)

Out[21]:3

向集合中添加元素：

In [22]:s.add(1)

s

Out[22]:{1, 2, 3, 4}

集合的交：

In [23]:a = {1, 2, 3, 4}

b = {2, 3, 4, 5}

a & b

Out[23]:{2, 3, 4}

并：

In [24]:a | b

Out[24]:{1, 2, 3, 4, 5}

差：

In [25]:a - b

Out[25]:{1}

对称差：

In [26]:a ^ b

Out[26]:{1, 5}

字典 Dictionary

Python用{key:value}来生成Dictionary。

In [27]:d = {'dogs':5, 'cats':4}

d

Out[27]:{'cats': 4, 'dogs': 5}

字典的大小

In [28]:len(d)

Out[28]:2

查看字典某个键对应的值：

In [29]:d["dogs"]

Out[29]:5

修改键值：

In [30]:d["dogs"] = 2

d

Out[30]:{'cats': 4, 'dogs': 2}

插入键值：

In [31]:d["pigs"] = 7

d

Out[31]:{'cats': 4, 'dogs': 2, 'pigs': 7}

所有的键：

In [32]:d.keys()

Out[32]:['cats', 'dogs', 'pigs']

所有的值：

In [33]:d.values()

Out[33]:[4, 2, 7]

所有的键值对：

In [34]:d.items()

Out[34]:[('cats', 4), ('dogs', 2), ('pigs', 7)]

数组 Numpy Arrays

需要先导入需要的包，Numpy数组可以进行很多列表不能进行的运算。

In [35]:from numpy import array

a = array([1, 2, 3, 4])

a

Out[35]:array([1, 2, 3, 4])

加法：

In [36]:a + 2

Out[36]:array([3, 4, 5, 6])

In [37]:a + a

Out[37]:array([2, 4, 6, 8])

画图 Plot

In [38]:%matplotlib inline

from matplotlib.pyplot import plot

plot(a, a\*\*2)

Out[38]:[<matplotlib.lines.Line2D at 0x9fb6fd0>]

循环 Loop

In [39]:line = '1 2 3 4 5'

fields = line.split()

fields

Out[39]:['1', '2', '3', '4', '5']

In [40]:total = 0

for field in fields:

total += int(field)

total

Out[40]:15

Python中有一种叫做列表推导式(List comprehension)的用法：

In [41]:numbers = [int(field) for field in fields]

numbers

Out[41]:[1, 2, 3, 4, 5]

In [42]:sum(numbers)

Out[42]:15

写在一行：

In [43]:sum([int(field) for field in line.split()])

Out[43]:15

文件操作 File IO

cd ~

d:\Users\lijin

写文件：

In [45]:

f = open('data.txt', 'w')

f.write('1 2 3 4\n')

f.write('2 3 4 5\n')

f.close()

读文件：

In [46]:f = open('data.txt')

data = []

for line in f:

data.append([int(field) for field in line.split()])

f.close()

data

Out[46]:[[1, 2, 3, 4], [2, 3, 4, 5]]

In [47]:for row in data:

print row

[1, 2, 3, 4]

[2, 3, 4, 5]

删除文件：

In [48]:import os

os.remove('data.txt')

函数 Function

Python用关键词def来定义函数。

In [49]:def poly(x, a, b, c):

y = a \* x \*\* 2 + b \* x + c

return y

x = 1

poly(x, 1, 2, 3)

Out[49]:6

用Numpy数组做参数x：

In [50]:x = array([1, 2, 3])

poly(x, 1, 2, 3)

Out[50]:array([ 6, 11, 18])

可以在定义时指定参数的默认值：

In [51]:from numpy import arange

def poly(x, a = 1, b = 2, c = 3):

y = a\*x\*\*2 + b\*x + c

return y

x = arange(10)

x

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

Out[51]:array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [52]:poly(x)

Out[52]:array([ 3, 6, 11, 18, 27, 38, 51, 66, 83, 102])

In [53]:poly(x, b = 1)

Out[53]:array([ 3,5, 9, 15, 23, 33, 45, 59, 75, 93])

模块 Module

Python中使用import关键词来导入模块。

In [54]:import os

当前进程号：

In [55]:os.getpid()

Out[55]:4400

系统分隔符：

In [56]:os.sep

Out[56]:'\\'

- 类 Class

用class来定义一个类。 Person(object)表示继承自object类； \_\_init\_\_函数用来初始化对象； self表示对象自身，类似于C Java里面this。

In [57]:class Person(object):

def \_\_init\_\_(self, first, last, age):

self.first = first

self.last = last

self.age = age

def full\_name(self):

return self.first + ' ' + self.last

构建新对象：

In [58]:person = Person('Mertle', 'Sedgewick', 52)

调用对象的属性：

In [59]:person.first

Out[59]:'Mertle'

调用对象的方法：

In [60]:person.full\_name()

Out[60]:'Mertle Sedgewick'

修改对象的属性：

In [61]:person.last = 'Smith'

添加新属性，d是之前定义的字典：

In [62]:person.critters = d

person.critters

Out[62]:{'cats': 4, 'dogs': 2, 'pigs': 7}

Python 数据类型

常用数据类型 Common Data Types

类型 例子

整数 -100

浮点数 3.1416

字符串 'hello'

列表 [1, 1.2, 'hello']

字典 {'dogs': 5, 'pigs': 3}

Numpy数组 array([1, 2, 3])

其他类型 Others

类型 例子

长整型 1000000000000L

布尔型 True, False

元组 ('ring', 1000)

集合 {1, 2, 3}

Pandas类型 DataFrame, Series

自定义 Object Oriented Classes

数字

整型 Integers

整型运算，加减乘：

In [1]:2 + 2

Out[1]:4

In [2]:3 - 4

Out[2]:-1

In [3]:4 \* 5

Out[3]:20

在Python 2.7中，整型的运算结果只能返回整型，除法的结果也不例外。

例如12 / 5返回的结果并不是2.4，而是2：

In [4]:12 / 5

Out[4]:2

幂指数：

In [5]:2 \*\* 5

Out[5]:32

取余：

In [6]:32 % 5

Out[6]:2

赋值给变量：

In [7]:a = 1

a

Out[7]:1

使用type()函数来查看变量类型：

In [8]:type(a)

Out[8]:int

整型数字的最大最小值：

在 32 位系统中，一个整型 4 个字节，最小值 -2,147,483,648，最大值 2,147,483,647。

在 64 位系统中，一个整型 8 个字节，最小值 -9,223,372,036,854,775,808，最大值 9,223,372,036,854,775,807。

In [9]:import sys

sys.maxint（最大整数值，一般是2的32次方）

Out[9]:2147483647

长整型 Long Integers

当整型超出范围时，Python会自动将整型转化为长整型，不过长整型计算速度会比整型慢。

In [10]:a = sys.maxint + 1

print type(a)

<type 'long'>

长整型的一个标志是后面以字母L结尾：

In [11]:a

Out[11]:2147483648L

可以在赋值时强制让类型为长整型：

In [12]:b = 1234L

type(b)

Out[12]:long

长整型可以与整型在一起进行计算，返回的类型还是长整型：

In [13]:a - 4

Out[13]:2147483644L

浮点数 Floating Point Numbers

In [14]:a = 1.4

type(a)

Out[14]:float

在之前的除法例子12 / 5中，假如想要使返回的结果为2.4，可以将它们写成浮点数的形式：

In [15]:12.0 / 5.0 怎样保留两位小数？？？

Out[15]:2.4

In [16]:12 / 5.0

Out[16]:2.4

In [17]:12.0 / 5

Out[17]:2.4

上面的例子说明，浮点数与整数进行运算时，返回的仍然是浮点数：

In [18]:5 + 2.4

Out[18]:7.4

浮点数也可以进行与整数相似的运算，甚至可以取余：

In [19]:3.4 - 3.2

Out[19]:0.19999999999999973

In [20]:12.3 + 32.4

Out[20]:44.7

In [21]:2.5 \*\* 2

Out[21]:6.25

In [22]:3.4 % 2.1

Out[22]:1.2999999999999998

注意看 3.4 - 3.2 的结果并不是我们预期的0.2，这是因为浮点数本身储存方式引起的，浮点数本身会存在一点误差。

事实上，Python 中储存的值为'0.199999999999999733546474089962430298328399658203125'，因为这是最接近0.2的浮点数。|

In [23]:'{:.52}'.format(3.4 - 3.2)

Out[23]:'0.199999999999999733546474089962430298328399658203125'

当我们使用print显示时，Python会自动校正这个结果

In [24]:print 3.4 - 3.2

Out[25]:0.2

可以用sys.float\_info来查看浮点数的信息：

In [25]:import sys

sys.float\_info

Out[25]:sys.float\_info(max=1.7976931348623157e+308, max\_exp=1024, max\_10\_exp=308, min=2.2250738585072014e-308, min\_exp=-1021, min\_10\_exp=-307, dig=15, mant\_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)

例如浮点数能表示的最大值：

In [26]:sys.float\_info.max

Out[26]:1.7976931348623157e+308

浮点数能表示的最接近0的值：

In [27]:sys.float\_info.min

Out[27]:2.2250738585072014e-308

浮点数的精度：

In [28]:sys.float\_info.epsilon

Out[28]:2.220446049250313e-16

复数 Complex Numbers

Python 使用 j 来表示复数的虚部：

In [29]:a = 1 + 2j

type(a)

Out[29]:complex

可以查看它的实部，虚部以及共轭：

In [30]:a.real

Out[30]:1.0

In [31]:a.imag

Out[31]:2.0

In [32]:a.conjugate()

Out[32]:(1-2j)

交互计算

在Python中运算是有优先级的，优先级即算术的先后顺序，比如“先乘除后加减”和“先算括号里面的”都是两种优先级的规则，优先级从高到低排列如下：

( ) 括号

\*\* 幂指数运算

\* / // % 乘，除，整数除法（整数除法，返回的是比结果小的最大整数值），取余运算

'+ -' 加减

简单的数学函数

绝对值：abs()

取整：round()

最大最小值：print min(2, 3, 4, 5)，print max(2, 4, 3)

变量名覆盖（不要用内置的函数来命名变量，否则会出现意想不到的结果）

类型转换

浮点数转整型，只保留整数部分：

In [42]:print int(12.324)

print int(-3.32)

12

-3

整型转浮点型：

In [43]:print float(1.2)

1.2

其他表示

除了10进制外，整数还有其他类型的表示方法。

科学计数法：

In [44]:1e-6

Out[44]:1e-06

16进制，前面加0x修饰，后面使用数字0-9A-F：

In [45]:0xFF

Out[45]:255

8进制，前面加0或者0o修饰，后面使用数字0-7：

In [46]:067

Out[46]:55

2进制，前面加0b修饰，后面使用数字0或1：

In [47]:0b101010

Out[47]:42

原地计算 In-place

Python可以使用下面的形式进行原地计算：

In [48]:b = 2.5

b += 2

print b

b \*= 2

print b

b -= 3

print b

4.5

9.0

6.0

布尔型 Boolean Data Type

布尔型可以看成特殊的二值变量，其取值为True和False：

In [49]:q = True

type(q)

Out[49]:bool

可以用表达式构建布尔型变量：

In [50]:q = 1 > 2

print q

False

常用的比较符号包括：

<, >, <=, >=, ==, !=

Python支持链式比较：

In [51]:x = 2

1 < x <= 3

Out[51]:True

生成字符串

Python中可以使用一对单引号''或者双引号""生成字符串。

分割

s.split()将s按照空格（包括多个空格，制表符\t，换行符\n等）分割，并返回所有分割得到的字符串。

In [52]:line = "1 2 3 4 5"

numbers = line.split()

print numbers

Out[52]: ['1', '2', '3', '4', '5']

s.split(sep)以给定的sep为分隔符对s进行分割。

In [53]:In [7]:line = "1,2,3,4,5"

numbers = line.split(',')

print numbers

Out[53]: ['1', '2', '3', '4', '5']

连接

与分割相反，s.join(str\_sequence)的作用是以s为连接符将字符串序列str\_sequence中的元素连接起来，并返回连接后得到的新字符串：

In [8]:s = ' '

s.join(numbers)

Out[8]:'1 2 3 4 5'

In [9]:s = ','

s.join(numbers)

Out[9]:'1,2,3,4,5'