学习Python第一天

张朝龙

目录

1 Python简介 **2**

2 人生苦短，Python必选。 **2**

3 上手Python解释器 **2**

4 把Python当计算器用 **2**

4.1 数值计算 2

4.2 字符串 4

4.3 列表（list） 7

5 跨出编程的第一步 **9**

今天是学习Python第一天，我从其官方文档入手，英文版地址 中文版地址 。在学习的过程中，我以英文版为主，参照了中文版，这一系列博文是我的学习笔记，不是严格的文档翻译，很多地方按照我的理解进行了转述，我认为每个人的第一份Python学习资料都应该是其官方文档。尽管从这个文档不可能学到所有的Python特性（要想学习到所有的Python特性需要阅读其官方手册和大量的标准库文档。），但是你学到的是最正统的Python。

我的学习环境是Windows + Anaconda + Python3.5 + Emacs. 我在Windows 上使用Emacs作为Python3.5的前端。在Emacs里使用IPython这样的REPL环境，大大提高了我的学习效率。另外为了使得报错提示显示为英文，我把Windows的语言环境改成了英文。

嗯，开始吧！

1 Python简介

Python简单易用，功能强大。相对于其他语言，它有更高级别的语言抽象和更有效的面向对象编程范式。Python的语法造就了Python程序格式上的优美，对于强迫症患者来说是福音。Python的解释特性使其成为脚本语言和快速开发部署的绝佳选择。

从Python的官网上，可以获得Python解释器和更多的第三方扩展库。当然，Python自带功能强大的标准库 。Python的解释器使其可以方便的同其他语言(C/C++)进行扩展 ，从而强化了Python的可扩展性。

2 人生苦短，Python必选。

每个人都有自动化大量重复性劳动的冲动。因为重复性的劳动让人大脑迟钝，生活变得索然无味。在不停的重复过程中，生命就这么浪费了。Python会以出人意料的速度帮你完成重复性劳动，让你有时间去爬山。如果你是一个程序员，肯定发现C/C++这类语言在编写代码编译代码重新编译代码的循环中浪费了好多青春。即便程序最终可以运行，为这个程序写大量的测试用例也要占用不少时间。

人生苦短，Python必选。

Python提供了不但Unix shell一样的批处理，还提供了可移植性。Python允许你的代码高度解耦，模块的重用易如反掌。Python本身携带了大量的标准模块，任君选择。作为一门解释性语言，Python不需要编译和链接，解释器也可以交互式的实用，随时查看程序结果。

3 上手Python解释器

这里我就不废话怎么安装Python环境了。我的IPyton启动后画面是：

```
Python 3.5.2 |Anaconda 4.2.0 (64-bit)| (default, Jul  5 2016, 11:41:13) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

图 1: 我的Python启动画面

Python文件默认编码方式是 `UTF-8` 。可以在文件头部添加:

```
# -*- coding: UTF-8 -*-
```

指定源文件采用 `UTF-8` 编码。在Python中使用 `#` 作为注释开始标志。

4 把Python当计算器用

4.1 数值计算

```
In [28]: 2/7
Out[42]:
0.2857142857142857

In [43]: 50 - 9/6
Out[59]:
48.5
In [60]:
```

整数 2,4,9 类型是 int ； 有小数点的数 48.5 类型是 float ; 除法计算 / 总是返回浮点数。为舍弃小数点部分，需要使用 // , 示例：

```
In [60]: 17/3
Out[67]:
5.666666666666667

In [68]: 17//3
Out[78]:
5
```

可以看出 // 进行的不是四舍五入，是直接舍弃小数部分，只保留整数部分。

Python使用 ** 完成幂计算，示例：

```
In [82]: 2**10
Out[92]: 1024

In [93]: 3**8
Out[100]:6561
```

= 号的一个作用是变量赋值，示例：

```
In [101]: width = 20

In [109]: height = 5*9

In [117]: width * height
Out[128]:
900
```

如果变量没有被定义，会报错，示例：

```
In [129]: n
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-133-fe13119fb084> in <module>()
----> 1 n

NameError: name 'n' is not defined
```

当整型和浮点类型一起混合使用时，结果为浮点类型：

```
In [134]: 3 * 3.75/1.5
Out[153]:
7.5
```

在交互式环境中，最后一个打印的表达式可以用 _ 来表示：

```
In [154]: tax = 12.5/100

In [176]: price = 100.50

In [187]: price*tax
Out[194]:
12.5625

In [195]: price + _
```

```
Out[205]:
113.0625

In [206]: round(_,2)
Out[212]:
113.06
```

除了 int 和 float , Python 支持也其他类型的数据，比如 Decimal 和 Fraction 。另外Python内置复数类型比如 3+5j

```
In [213]: n= 3+5i
  File "<ipython-input-220-1c979fa6598a>", line 1
    n= 3+5i
       ^
SyntaxError: invalid syntax


In [221]: n = 3+5j

In [233]: abs(n)
Out[233]:
5.830951894845301
```

注意是 3+5j 而不是 3+5i .

4.2 字符串

Python的字符串可以使用单引号 ='...'= 也可以使用双引号 ="..."= 。使用 \ 做转义字符。

```
In [234]: 'span eggs'
Out[249]:
'span eggs'

In [250]: 'doesn\'t'
Out[278]:
"doesn't"

In [279]: "doesn't"
Out[291]:
"doesn't"

In [292]: '"Yes," he said.'
Out[312]:
'"Yes," he said.'

In [313]: "\"Yes,\" he said."
Out[333]:
'"Yes," he said.'

In [334]: '"Isn\'t," she said.'
Out[368]:
'"Isn\'t," she said.'
```

在交互式环境中，字符串的输出使用双引号；特殊字符使用反斜杠保留。 print() 函数打印的结果更具可读性：

```
In [334]: '"Isn\'t," she said.'
Out[368]:
'"Isn\'t," she said.'

In [369]: print('"Isn\'t," she said')
"Isn't," she said

In [402]: s = 'First line.\nSecond line.'

In [441]: print(s)
```

```
First line.
Second line.
```

如果你不希望 print() 中 \ 转义，可以使用 r 选项：

```
In [446]: print("C:\some\name")
C:\some
ame

In [466]: print(r"C:\some\name")
C:\some\name
```

Python 使用 =”””...”””= 支持多行字符：

```
print("""
This␣is␣a
string
occupying␣multiple␣lines
""")
```

输出：

```
In [487]:
This is a
string
occupying multiple lines
```

也可以使用 + 级联多个字符串。

```
In [518]: 4*"love␣" + "u"
Out[518]:
'love␣love␣love␣love␣u'
```

多个字符串放在一起也可以实现自动级联：

```
In [519]: "python" "is" 'good'
Out[547]:
'pythonisgood'
```

但是这样的方式只支持字符串级联，不支持变量和字符串级联：

```
In [548]: prefix = 'py'

In [561]: prefix 'thon'
  File "<ipython-input-570-37b5e5a6971f>", line 1
    prefix 'thon'
           ^
SyntaxError: invalid syntax
```

如果要变量和字符串级联，需要使用 + :

```
In [571]: prefix + 'thon'
Out[587]:
'python'
```

可以使用下表来索引字符串中的字符：

```
In [626]: word = 'python'

In [643]: word[0]
Out[654]:
'p'

In [655]: word[6]
---------------------------------------------------------------------------
```

```
IndexError                                Traceback (most recent call last)
<ipython-input-658-9f3bdf74c108> in <module>()
----> 1 word[6]

IndexError: string index out of range

In [659]: word[5]
Out[662]:
'n'
```

注意下标索引和 C/C++ 一样从 0 开始，并且不能越界。除了这种索引方式, Python 提供了从后面倒着索引的方法：

```
In [663]: word[-1]
Out[674]:
'n'

In [675]: word[-5]
Out[684]:
'y'

In [685]: word[-6]
Out[688]:
'p'

In [689]: word[-7]
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-699-03d2d1c3cbaf> in <module>()
----> 1 word[-7]

IndexError: string index out of range
```

注意倒着索引也有越界的问题。

除了逐个索引单字符，索引字符串中的某几个部分也是可以的：

```
In [700]: word[0:2]
Out[703]:
'py'

In [704]: word[3:5]
Out[714]:
'ho'
```

注意部分索引时，冒号前面的索引号是包含的，冒号后面的索引号是不包含的， n:N 的含义是：从第 n (包括) 到第 N (不包括)。

```
In [715]: word[:2] + word[2:]
Out[738]:
'python'

In [739]: word[:5] + word[5:]
Out[751]:
'python'
```

Python 不允许修改字符串，如果你需要一个新的字符，重新创建一个:

```
In [752]: word[0] = 'J'
---------------------------------------------------------------
TypeError                         Traceback (most recent call last)
<ipython-input-771-197b67ffdd83> in <module>()
----> 1 word[0] = 'J'

TypeError: 'str' object does not support item assignment


In [772]: 'J' + word[1:]
```

```
Out[778]:
'Jython'
```

内置的 len() 函数返回字符串的长度：

```
In [779]: s = 'askkjljlkjioejkjlkjlskjlj'

In [793]: len(s)
Out[801]:
25
```

4.3 列表（list）

Python支持很多复杂的数据类型，列表（list）是比较常用的一个:

```
In [802]: squares = [1,4,9,16,25]

In [833]: squares
Out[837]:
[1, 4, 9, 16, 25]
```

就像字符串一样，列表也可以用下表正向索引或者逆向索引

```
In [890]: squares[-2:1]
Out[900]:
[]

In [901]: squares[-2:-4]
Out[901]:
[]

In [902]: squares[2:4]
Out[902]:
[9, 16]

In [903]: squares[-4:-2]
Out[909]:
[4, 9]
```

list支持级联：

```
In [910]: squares + [36,49,64,81,100]
Out[952]:
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

不像字符串不能被修改，list是可以修改的:

```
In [953]: cubes = [1, 8, 27,65,125]

In [988]: cubes[3] = 64

In [1000]: cubes
Out[1004]:
[1, 8, 27, 64, 125]
```

可以使用 append() 方法向list的尾巴上追加数据：

```
In [1000]: cubes
Out[1004]:
[1, 8, 27, 64, 125]

In [1005]: cubes.append(216)

In [1020]: cubes.append(7**3)
```

```
In [1038]: cubes
Out[1038]:
[1, 8, 27, 64, 125, 216, 343]
```

list也可以部分索引：

```
In [1039]: letters = ['a','b','c','d','e','f','g','h']

In [1075]: letters
Out[1075]:
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']

In [1076]: #replace some values

In [1081]: letters[2:5] = ['C','D','E']

In [1095]: letters
Out[1095]:
['a', 'b', 'C', 'D', 'E', 'f', 'g', 'h']

In [1096]: #now remove them

In [1108]: letters[2:5] = []

In [1115]: letters
Out[1123]:
['a', 'b', 'f', 'g', 'h']

In [1124]: # clear all items

In [1136]: letters[:] = []

In [1147]: letters
Out[1147]:
[]
```

内置的 `len()` 函数同样返回list的长度。

可以嵌套list:

```
In [1148]: a = ['a','b','c']

In [1157]: n = [1,2,3]

In [1168]: x = [a,n]

In [1177]: x
Out[1181]:
[['a', 'b', 'c'], [1, 2, 3]]

In [1182]: x[0]
Out[1189]:
['a', 'b', 'c']

In [1190]: x[1]
Out[1190]:
[1, 2, 3]

In [1191]: len(x)
Out[1195]:
2

In [1196]: len(x[0])
Out[1207]:
3
```

5 跨出编程的第一步

Python能做的事情可不止一个计算器那么简单。看下面的小例子：

```
1 # Fibonacci series:
2 # the sum of two elements defines the next
3 a,b = 0,1
4 while b<10:
5     print(b)
6     a,b = b,a+b
```

输出为：

```
1
1
2
3
5
8
```

解读一下上面的程序：

1. 第一行和第二行是注释，第三行执行了多变量赋值操作。变量 a 和 b 同时被赋值为 0 和 1 。第6行也用了多变量赋值操作。
2. while 循环会一直循环下去, 只要 (b<10) 为真。在Python中，任何非零的值都会被当做true来处理，但是在Java中 true和false是boolean类型。所以Python语言是一种比Java对类型约束比较弱的语言，可以说是弱类型语言。
3. while 的内容通过缩进表示。
4. 可以使用 end 来避免 print() 函数生成新行

```
#+begin_src python -n
# Fibonacci series:
# the sum of two elements defines the next
a,b = 0,1
while b<10:
    print(b,end = ',')
    a,b = b,a+b
```

输出为：

```
1,2,3,5,8
```