【白话python连载（3）】巧妙使用输入输出--与python尽情畅聊

* 从小入手-简单的python小应用

例1、问题描述：在有序的数组中新插入一个不存在的数，并且保持该数组仍然有序

|  |
| --- |
| *#!/usr/bin/python*  *# -\*- coding: utf-8 -\*-*  *#问题描述：生成一组不重复的数组，并进行排序*  **import** random  **def ori\_data**(*start*, *stop*, *length*)**:** *#准确指定起始范围由小到大* start,stop **=** (int(*start*),int(*stop*)) **if** *start* **<=** *stop* **else** (int(*stop*),int(*start*))  length **=** int(abs(*length*)) **if** *length* **else** 0  **if** int(abs(*start***-***stop*)) **<** *length***:** print(**'所选范围不符合要求，请重新指定'**)  **return None  else:** *#利用随机函数生成随机数组，并且保证数值唯一* random\_list **=** []  **for** i **in** range(*length*)**:** one\_number **=** random.**randint**(*start*, *stop*)  **while** one\_number **in** random\_list**:** one\_number **=** random.**randint**(*start*,*stop*)  print(**'有重复，已经重新生成'**)  random\_list.**append**(one\_number)  random\_list **=** sorted(random\_list)  **return** random\_list  **if** \_\_name\_\_ **== '\_\_main\_\_':** ori\_data **= ori\_data**(38,22,10)  print(**'新生成的数组为：{}'**.**format**(ori\_data)) |
| *"D:\anaconda python\python3.6.exe" D:/vernacular/coding\_ziyi/InsertAndSort.py*  *有重复，已经重新生成*  *有重复，已经重新生成*  *有重复，已经重新生成*  *新生成的数组为：[23, 25, 26, 31, 32, 34, 35, 36, 37, 38]* |

例2、问题描述：按格式输出九九乘法表

|  |
| --- |
| *#!/usr/bin/python # -\*- coding: utf-8 -\*-* **for** i **in** range(1,10)**:  for** j **in** range(1,i**+**1)**:** multip **=** i **\*** j  print(**'{}\*{}={}'**.**format**(j,i,multip),end**=' '**)  print(**''**) |
| *"D:\anaconda python\python3.6.exe" D:/vernacular/coding\_ziyi/multiplication\_table.py*  *1\*1=1*  *1\*2=2 2\*2=4*  *1\*3=3 2\*3=6 3\*3=9*  *1\*4=4 2\*4=8 3\*4=12 4\*4=16*  *1\*5=5 2\*5=10 3\*5=15 4\*5=20 5\*5=25*  *1\*6=6 2\*6=12 3\*6=18 4\*6=24 5\*6=30 6\*6=36*  *1\*7=7 2\*7=14 3\*7=21 4\*7=28 5\*7=35 6\*7=42 7\*7=49*  *1\*8=8 2\*8=16 3\*8=24 4\*8=32 5\*8=40 6\*8=48 7\*8=56 8\*8=64*  *1\*9=9 2\*9=18 3\*9=27 4\*9=36 5\*9=45 6\*9=54 7\*9=63 8\*9=72 9\*9=81* |

例3、问题描述：使用递归调用的方法计算某个数的阶乘

|  |
| --- |
| *#!/usr/bin/python # -\*- coding: utf-8 -\*-* **import** sys **def Factorial**(*num*)**:  if** *num* **==** 1 **:  return** 1  **else:** result **=** *num* **\* Factorial**(*num***-**1)  **return** result **if** \_\_name\_\_ **== '\_\_main\_\_' :** print(**'请输出你要计算的数: '**)  first\_num **=** int(sys.stdin.**readline**( ))  *# first\_num = int(input('请输入你要计算的数: '))* result **= Factorial**(first\_num)  sys.stdin.write(**'{}的阶乘是：{}'**.**format**(first\_num, result)) |
| *"D:\anaconda python\python3.6.exe" D:/vernacular/coding\_ziyi/recursive.py*  *请输入你要计算的数: 5*  *5的阶乘是：120* |

* python的输入输出介绍

input&&print

python的内置函数input()可以接受用户输入的相关内容，提供给计算机程序所需的信息，更好的完成特定的任务。在python3.x中raw\_input( )和input( )进行了整合，去除了raw\_input( )，仅保留了input( )函数，其接收任意任性输入，将所有输入默认为字符串处理，并返回字符串类型。

**根据实际需求指定输入内容的类型：**

以第三个案例为例，first\_num = int(input('请输入你要计算的数: '))就是为了获得用户想要计算阶乘的一个数字，但是input函数是以字符串的格式存储用户输入的内容的，因此需要在只输入之前强制转化一下数据的类型。而常用的类型转换函数还有str、float、list以及一些第三方包numpy和pandas中特有的类型array、DataFrame、series等等

有了输入，计算机程序可以更好的按照需求完成相关任务。但是程序也需要将运行结果输出，让用户更好地明确程序的准确性。同样python的内置函数print()就可以很好的完成这一要求。按照常规套路，在安装教程中通常都会执行print(‘hello world’)来验证开发环境是否安装成功。因此输出函数是辅助调试程序的很好地助手。养成良好的格式化输出习惯，对于调整复杂程序或许能起到事半功倍的效果。

**建立良好的输出习惯：**

print(‘hello world’)是输出单个字符串，如果要输出多个字符串，可以以逗号分隔每个字符串并且在同一行完成一条print操作，执行完一条print操作会默认执行换行操作，因此如果想要多语句同时输出到同一行，则可以指定参数end=‘，’完成连接多行输出。而且也可以直接在输出中完成数值运算操作（字符串格式的内容必须写入引号中，数值型不需要）。

|  |  |
| --- | --- |
| print(‘hello world’) | hello world |
| print(‘hello’, 'world’) | hello world |
| print(‘hello’)  print(‘world’) | hello  world |
| print(‘hello’, end=’’)  print(‘world’) | hello world |
| Print（"{1} {0} {1}".format("hello", "world")） | world hello world |
| print（100+200） | 300 |

在上面的小应用中都使用的是格式化动态输出变量的方法，这是python2.6之后推出新的格式化方法。按照指定位置（或设置参数）完成变量的格式化输出。format中有丰富的格式限定符，有很多格式限定方法，使用更加灵活

print('新生成的数组为：{name}'.format(name=ori\_data))

print('{0}\*{1}={2}'.format(j,i,multip),end=' ')，

传统的格式化方法【print（‘%s\*%s=%s’%j，%i，%multip）】也依旧是可以应用的，但是更多的是推荐使用新的格式化方法。

sudin&&stdout

除了python内置的输入输出函数input和print，sys.stdin与sys.stdout（python的标准输入输出）也可以等价的完成类似的功能。因为input和print在原则就是通过调用sys.stdin的readline()方法和sys.stdout的write()方法上述两种方法而完成输入输出的，python内置的input和print操作也更便捷，input可以直接内置提示语，不需要另外输出提示语。但是sys.stdin与sys.stdout也有其他不同的功能，要根据实际需求看是否要应用标准的文件格式的输入输出操作。

总之，一个良好的python应用的形成不会是一蹴而就的，因此每次在编写应用程序时，需要不断的进行代码的调试。这时为了很好的找到问题的症结所在，就需要很好地借助python的输入输出函数来完善python程序。

* python的运算详解

近几次列举的python小应用大多都和数学运算或者逻辑运算相关。计算机实则因计算而生，因此实现运算操作是必不可少的重要功能。

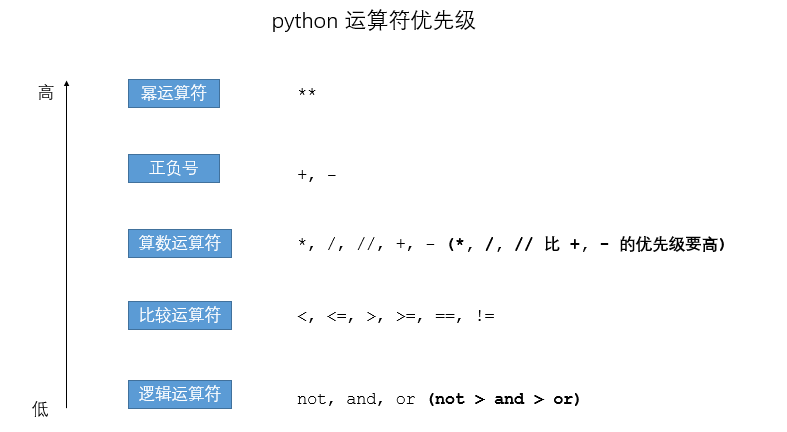
Python本身可以完成常规的数值运算操作、比较运算操作、逻辑运算操作、成员运算操作以及身份运算操作。如果借助numpy、pandas、scipy等第三方库是大数据分析运算的主要利器。接下来详细了解一下运算在python中的具体操作规则

**数值运算&比较运算：**

|  |  |
| --- | --- |
| +，-，\*(print(2-3)→-1) | 加减乘（与常规认识一致） |
| /,(print(1/2,1.0/2)→0.5，0.5)  (print 1/2→0，print 1.0/2.0→0.5) | python3中则都是精确除法。python2中传统除法：整数执行地板除，浮点数执行精确除。 |
| //,(print(1//2,1.0//2.0)→0，0) | 地板除法（整除）：总是会舍去小数部分，返回数字序列中比真正的商小的最接近的数字 |
| %，(print(1%2)→1) | 除法取余：最后得到的结果是除法的余数 |
| divmod(1,2),→（0,1） | 内建函数divmod (a,b)，返回(a//b,a%b) |
| \*\*，(print(2\*\*2）)→ | 乘方运算，也叫 |
| <,<=,>,>=,==,!= | 小于（等于），大于（等于），恒等于，不等于 |

**逻辑运算操作：**

|  |  |
| --- | --- |
| and 与（表达式1 and 表达式2） | 当表达式1为真时才会执行表达式2 |
| or 或（表达式1 or 表达式2） | 当表达式1为假时才会执行表达式2 |
| not 非（！表达式） | 对表达式的结果取反 |



**成员运算&身份运算：**

|  |  |
| --- | --- |
| in，one\_number in random\_list | 随机生成的数字是否已经加入了随机数组列表中 |
| not in ,(>>>a not in [d,b]→Ture) | 成员a不在列表[d,b]，所以表达式结果是True |
| is | 判断左侧对象的引用是否相同于右侧对象的应用 |
| not is | e.g. name=’ad’ user=‘ad' name is user |

通常情况比较运算、逻辑运算、成员和身份运算都会与循环和判断语句同时使用，来更好的完成不容任务情景的处理。在生产随机数组的应用中，就结合应用了多种运算

