Python输出的方法与格式详细总结(print与stdout)

相信许多刷编程题的小伙伴在刷题过程中已经感受到编程题的输出要求是多种多样的，有些输出格式看着就已经头皮发麻了，比如下面这个...

输出样例:

```
C...C CCCCC C.... C.... .CCC.
C...C C.... C.... C.... C...C
C...C C.... C.... C.... C...C
CCCCC CCCC. C.... C.... C...C
C...C C.... C.... C.... C...C
C...C C.... C.... C.... C...C
C...C CCCCC CCCCC CCCCC .CCC.

C...C .CCC. CCCC. C.... CCCC.
C...C C...C C...C C.... C...C
C...C C...C CCCC. C.... C...C
C.C.C C...C CC... C.... C...C
CC.CC C...C C.C.. C.... C...C
C...C C...C C..C. C.... C...C
C...C .CCC. C...C CCCCC CCCC.
```


面对这些较为复杂的输出情况，我们就需要掌握各种各样的输出方法与输出格式了，下面就对常用的输出方式与方法做一些总结。

1.print()普通用法与 格式化 输出

(1).直接输出(适用于无特殊输出要求的情况)

```
print(1)
#输出: 1
print('very good!')
#输出: very good!
print('A')
#输出: A
print([1,2,3,4,5])
#输出: [1,2,3,4,5]
print({1:'a',2:'b',3:'c'})
#输出: {1:'a',2:'b',3:'c'}
'
```

AI助手 运行

(2).加星号拆包(更加适用于一行打印出数组元素)

```
x = [1,2,3,4,5]
print(*x)
#输出: 1 2 3 4 5
y = {1:'a',2:'b',3:'c',4:'d'}
print(*y)
#输出: 1 2 3 4
'
```

AI助手 运行

(3).加入%进行格式化控制(适用于保留几位小数、进制、对齐、科学计数法等较为复杂的情况)

首先我们先了解一下格式符号

以下为标准格式化格式符

格式符	表示
%s	字符串
%d	十进制整数
%f	浮点型
%c	字符型
%u	无符号十进制整数
%o	八进制
%x	十六进制
%e	科学计数法

还有一些常用的格式符

格式符	表示

%.2f	保留两位小数
%04d	输出前面4个字符空间补零
%4d	右对齐前面留出4个字符空间
%-4d	左对齐并占用4个字符空间

介绍一些常见输出场景

```
x = 123
print('%d'%x)
#输出：123
print('这有一个整数%d'%666)
#输出：这有一个整数666
print('%.2f'%3.141592)
#输出：3.14
print('%e'%14332231433223)
#输出：1.433223e+13
print('%04d'%99)
#输出：000099
print('%9s'%'hahaha')
#输出：    hahaha(注意左侧有多个空格)
'
```

AI助手 运行

2.print()+format()格式化输出(格式化输出推荐用法)

format进制格式化字符

格式符	表示
{:s}	字符串
{:d}	十进制整数
{:f}	浮点型
{:c}	字符型
{:u}	无符号十进制整数
{:o}	八进制
{:x}	十六进制
{:e}	科学计数法

以下是format常用格式化字符

格式符	表示
{:.2f}	保留两位小数
{:+.2f}	保留两位小数并携带正负号
{:0>2d}	左侧补零，整体占两个字符
{:0<2d}	右侧补零，整体占两个字符
{:*<4d}	右侧补星号，整体占4个字符
{:5d}	右对齐，整体占5个字符
{:<5d}	左对齐，整体占5个字符
{:^5d}	居中对齐，整体占5个字符
{:.2e}	科学计数法并保留两位小数
{:,}	以逗号分割的数字

介绍一下format常见输出场景:

```
x = 123
print('{}'.format(x))
#输出：123
print('hello!{}'.format('World'))
#输出：hello!World
print('{}'.format(['1','2','3','4','5']))
#输出：['1','2','3','4','5']
```

AI助手 运行

```
print('{}->{}->{}'.format(1,2,3))
#输出：1->2->3
print('{2}->{0}->{1}'.format(1,2,3))
#输出：3->1->2
print('The Number is {}'.format(666))
#输出：The Number is 666
print('{:.2f}'.format(3.245))
#输出：3.25
print('{:0>4d}'.format(12))
#输出：0012
print('{:>8s}'.format('abc'))
#输出：      abc(注意abc前面有空格)
print('{:<8d}'.format(999))
#输出：999     (注意999后面有空格)
print('{:.2e}'.format(1234554321))
#输出：1.23e+09

'
```

另外，print()+format()的搭配还可以这样写

```
x = 123
print(f'The Number is {x:d}')
#输出：The Number is 123
y = 3.1415
print(f'PI is {y:.2f} ...')
#输出：PI is 3.14
z = [1,3,5,7,9]
print(f'{z[0]:d} {z}')
#输出：1 [1, 3, 5, 7, 9]
'
```

AI助手 运行

3.print()其他常用搭配以及复杂型输出示例(数组 、矩阵、不规则输出等)

(1).打印一行数组

```
x = [1,2,3,4,5,6,7]
print(' '.join(map(str,x)))
#输出：1 2 3 4 5 6 7
'
```

AI助手 运行

(2).打印一行以任意字符分割的数组

```
x = [1,2,3,4,5,6,7]
print('*'.join(map(str,x)))
#输出：1*2*3*4*5*6*7
'
```

AI助手 运行

(3).打印多行数组/二维数组-两种方法

```
x = [[1,2,3],[4,5,6],[7,8,9]]
for i in x:
    print(*i)
#输出：
1 2 3
4 5 6
7 8 9
```

AI助手

或者

```
x = [[1,2,3],[4,5,6],[7,8,9]]
for i in x:
    print(' '.join(map(str,i)))
#输出：
1 2 3
4 5 6
7 8 9
```

AI助手

(4).打印一行且不换行

```
print("no enter",end="")
'
```

AI助手 运行

(5).将两个不同行列的二维数组并排打印

```
'''
二维数组A:  二维数组B:
1 2 3 4    1 3 5
5 6 7 8    2 4 6
6 7 8 9    7 8 9
           3 2 1
'''
#定义两个二维数组
a = [[1,2,3,4],[5,6,7,8],[6,7,8,9]]
b = [[1,3,5],[2,4,6],[7,8,9],[3,2,1]]
```

AI助手 运行

像上面这种情况需要较多数据处理的过程的输出还有很多，比如打印乘法口诀表、打印沙漏等等

面对这些情况，解决问题的关键就在于输出数据的预处理了，而不是输出方法与方式的选择

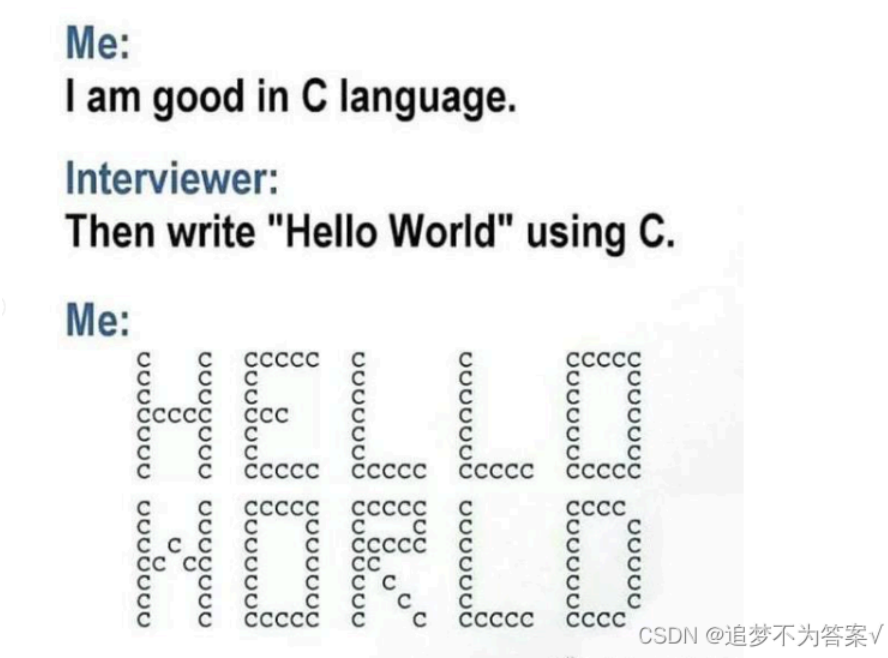
比如在文章开头提到的这个题，题目来自PAT乙级1109

题目链接: PAT乙级1109

1109 擅长C 分数 20

全屏浏览题目 切换布局

当你被面试官要求用 C 写一个“Hello World”时，有本事像下图显示的那样写一个出来吗？



Python题解:

```
import sys
alpha_dict = {}
out_list = []
start = 'abcdefghijklmnopqrstuvwxyz'
str_1 = ''
if_has_pt = False
def printLine(in_str,dict_in,num):  '''打印整行字母矩阵，矩阵的1行等于输出7行'''
    out_line = []
    for i in range(0,7):
        for j in in_str:
            try:
                out_line.append(dict_in[j][i])
```

AI助手

4.stdout.write()标准化输出

这个输出方法是python的标准化输出，需要我们导入sys模块才能使用

```
import sys
x = '123123'
sys.stdout.write(x)
#输出：123123
'
```

AI助手 运行

stdout.write与print的不同是，print打印时默认换行，而stdout.write不换行

还有一点就是stdout.write只支持字符串类型的输出，所以感觉还是print好用