

Python HW Review

Chenyuan

2020/06/09

Python和C语言的区别

- z已赋值, $x=(y=z+1)$ 语句是错误语句
True Python里面赋值语句本身不是表达式 不能放在等号右边
但你可以 $x=y=z+1$
- $3>2>=2$ 的值为True
True 等价于 $3>2$ and $2>=2$ 两个等式都是成立的 和C语言不同
- Python字符串以'\0'为结束符。
Python的字符串没有\0
- 变量不需事先声明就可使用。
对 Python动态类型 可以直接用全局变量

把自己当成Python解释器

2-2 (样卷) 输入10, 下面程序行号为2的语句输出是多少? (2分)

```
the_max = int(input("Enter the upper limit:"))
the_sum = 0
extra = 0
for number in range(1,the_max):
    if number%2 and not number%3:
        the_sum = the_sum + number
    else:
        extra = extra + 1
print(the_sum) # 行号为 2
print(extra) # 行号为 3
```

```
the_max = 10
the_sum = 0
extra = 0
for 循环从1到10 不包含10
if number%2 !=0 and
number%3==0:
    不能被2整除 同时 能被3整除:
        累加到the_sum里面
```

从1到9 满足条件的有3和9
the_sum就是3+9=12

写好草稿

2-11 (样卷) 下列程序运行输出结果为_。(4分)

```
import math
def factors(x):
    y=int(math.sqrt(x))
    for i in range(2,y+1):
        if (x%i ==0):
            print(i,end=' ');
            factors(x//i)
            break
        else:
            print(x,end=' ')
    return
factors(18)
```

factors(18)
x=18
y=18开根号向下取整=4
for i从2到4 包括4
 i=2
 if 18能被2整除:
 print(2) ←
 factors(9)
 break return不用回来

factors(9)
x=9
y=3
i从2到3
 if 9%2==0 不成立
 print(9) ←
 继续循环i=3
 if 9%3==0 成立
 print(3) ←
 factors(3)
 break return

答案: 2 9 3

注意空格

局部变量 全局变量

2-15 (样卷) 下列程序运行输出结果为_。

```
b,c=2,4
def g_func():
    b=1
    b=b*c
    d=b
    print(b,d,end=' ')
g_func()
print(b,c)
```

全局变量:

b=2 c=4

g_func局部变量:

函数里面被赋值了的就是局部变量

b, d是局部变量

执行g_func:

局部b=1 不改变全局的b

b=1*4=4 读取了全局的c

d=4

print(4,4)

返回后print(b,c) 全局的b还是2

print(2,4)

答案4 4 2 4

对齐

2-10 (样卷)下面代码的输出结果是_。(2分)

```
a=666666  
b="T"  
print("{0:{2}^{1}}\n{0:{2}>{1}}\n{0:{2}<{1}}".format(a,20,b))
```

- A. TTTTTTTT666666TTTTTTTT
TTTTTTTTTTTTTTTT666666
666666TTTTTTTTTTTTTTTT

把\n拆开看成三行:

{0:{2}^{1}}

{0:{2}>{1}}

{0:{2}<{1}}

把值填进去:

{0:T^20} 居中 主体还是666666, 用T补齐20个

>把6放在右边 默认的 '{0:20}'.format(666666)

<左边

编程题考试技巧

- 一步步写，写一点就测一点
- 能用内置的函数就用上 比如sum
- 注意空格
- 仔细读题，关注上界
- 题目有点复杂就写函数，降低思维负担
- 递归函数想清楚参数和返回值的含义
- 实在不行就骗分

能用上自带函数的就用上

计算 $21+22+23+\dots+m$

```
print("sum =", sum(range(21,int(input())+1)))
```

- 注意空格 等号后面
- 用range创建了数组
- sum只能对数字数组

```
>>> range(21, 23)
range(21, 23)
>>> list(range(21, 23))
[21, 22]
>>> sum(list(range(21, 23)))
43
>>> print("sum = ", sum(list(range(21, 23))))
sum = 43
```


能用上自带函数的就用上

统计单词的个数

- **仔细读题：**单词间空格数可以是多个。
不能用.count(" ")

```
>>> "aaab".count("a")
3
>>> "a b c".count(" ")
2
>>> "a b  c".count(" ")
3
>>> "a b  c".split()
['a', 'b', 'c']
>>> "a b  c".split(" ")
['a', 'b', '', 'c']
```

```
print("count =", len(input().split()))
```

直接.split()就可以按空格拆成数组 用len就知道多少个了

写个函数呗

输出一个 n 阶的方阵 输入 n 和 d

```
2 2 2 2
2 1 1 2
2 1 1 2
2 2 2 2
```

$n=4$ (4行4列)

$d=2$ (中间是1, 周围是2)

主体: 输出 n 行数字

每一行数字要么全部是 $digit$, 要么是左右两边各一个 $digit$, 中间 $n-2$ 个 $digit-1$

把显示一行数字提取成一个函数:

```
def printlist(l):
    print(" ".join([str(i) for i in l])+" ")
```

其实你可以 `print(*l, end="\n")`

写个函数呗

输出一个 n 阶的方阵

```
2 2 2 2
2 1 1 2
2 1 1 2
2 2 2 2
```

n=4 (4行4列)

d=2 (中间是1, 周围是2)

如何生成5个0的数组?

[0]*5

列表怎么合并?

[1]+[2,3]

```
>>> [0]*5
[0, 0, 0, 0, 0]
>>> [1]+[2,3]
[1, 2, 3]
>>>
```

```
n, d = map(int, input().split())
```

```
printlist([d]*n)
```

```
for i in range(n-2):
```

```
    printlist([d] + [d-1]*(n-2) + [d] )
```

```
printlist([d]*n)
```

递归函数：抓住输入参数和返回值

列表元素的个数加权和(2)

第1层每个算10个，第二层每个算9个，……

如[1,2,[3,4,[5,6],7],8] 就是

$$10+10 + 9+9 + 8+8 + 9 + 10=73$$

calc 输入(data, depth) 返回计算的结果r

比如说一开始调用的就是calc([1,2,[3,4,[5,6],7],8], 0) 这里为啥depth设置成0?

最里面的调用就是calc([5,6], 2) 返回 $2*(10-2)=16$

->最简单的情况 数组里面全部是数字

然后再考虑还有列表怎么办？递归调用自身咯

递归函数：抓住输入参数和返回值

列表元素的个数加权和(2)

```
def calc(data, depth):  
    r = 0  
    for i in data:  
        if isinstance(i, list):  
            r += calc(i, depth+1)  
        else:  
            r += 1 * (10-depth)  
    return r  
  
print(calc(eval(input()), 0))
```

做不出题目 不要放弃 努力骗分

列表元素的个数加权和(2)

```
1 data=eval(input())
2 ans=0
3 for i in data:
4     if isinstance(i, int):
5         ans+=10
6     else:
7         for j in i:
8             if isinstance(j, int):
9                 ans+=9
10            else:
11                for k in j:
12                    if isinstance(k, int):
13                        ans+=8
14                else:
15                    pass
16 print(ans)
```

提交结果

提交时间	状态	分数	题目	编译器	耗时
2020/06/08 22:08:27	部分正确 	50	7-4	Python (python 3)	42 ms
测试点	结果	耗时	内存		
0	答案正确	42 ms	2976 KB		
1	答案正确	33 ms	2944 KB		
2	答案错误 	24 ms	2984 KB		
3	答案错误 	25 ms	2984 KB		

做不出题目 不要放弃 努力骗分

[< 返回](#)

7-7 阶乘的非零尾数 (20分)

“求 N 阶乘末尾的第一个非零数字”是一道常见的企业笔试题。这里我们略微做个变化，求 N 阶乘末尾的第一个非零 K 位数，同时输出末尾有多少个零。

输入格式:

输入给出一个不超过 10^7 的正整数 N 和要求输出的位数 $0 < K < 10$ 。

输出格式:

在一行中输出 N 阶乘末尾的第一个非零 K 位数（注意前导零也要输出）、以及末尾 0 的个数，其间以 1 个空格分隔。

输入样例:

```
18 5
```

输出样例:

```
05728 3
```

作者	陈越
单位	浙江大学
代码长度限制	16 KB
时间限制	800 ms
内存限制	64 MB

[题目详情](#)

看上界 10^7 的 7 次方算阶乘?

不要放弃!

直接暴力算阶乘，转成字符串

右边去掉 0——用 `rstrip("0")`
输出最右边 K 位数 `[-K:]`

最右边有多少个 0? 长度减一减

做不出题目 不要放弃 努力骗分

[< 返回](#)

7-7 阶乘的非零

“求 N 阶乘末尾的第一个非零数字”是一道常见的企业笔试题。现在题目有了几个变化，求 N 阶乘末尾的第一个非零 K 位数，同时输出未

输入格式：

输入给出一个不超过 10^7 的正整数 N 和要求输出的位数 0

输出格式：

在一行中输出 N 阶乘末尾的第一个非零 K 位数（注意前导末尾 0 的个数，其间以 1 个空格分隔。

输入样例：

```
18 5
```

输出样例：

```
05728 3
```

提交时间	状态	分数	题目	编译器	耗时
2020/05/20 11:26:55	部分正确 ⓘ	14	7-7	Python (python 3)	22 ms
测试点	结果	耗时	内存		
0	答案正确	21 ms	2932 KB		
1	答案正确	22 ms	3036 KB		
2	答案正确	20 ms	3048 KB		
3	答案错误 ⓘ	21 ms	3016 KB		
4	答案正确	20 ms	2984 KB		
5	运行超时 ⓘ	--	0 KB		
6	运行超时 ⓘ	--	0 KB		
7	运行超时 ⓘ	--	0 KB		

答案全部已知? 直接骗分

下面是一个完整的下三角九九口诀表:

```
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
1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63
```

本题要求对任意给定的一位正整数 N , 输出从 $1*1$ 到 $N*N$ 的

```
1 data=""1*1=1...
2 1*2=2...2*2=4...
3 1*3=3...2*3=6...3*3=9...
4 1*4=4...2*4=8...3*4=12...4*4=16...
5 1*5=5...2*5=10...3*5=15...4*5=20...5*5=25...
6 1*6=6...2*6=12...3*6=18...4*6=24...5*6=30...6*6=36...
7 1*7=7...2*7=14...3*7=21...4*7=28...5*7=35...6*7=42...7*7=49...
8 1*8=8...2*8=16...3*8=24...4*8=32...5*8=40...6*8=48...7*8=56...8*8=64...
9 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...""
10 print("\n".join(data.split("\n")[:int(input())]))
11 #偷懒的做法, 当你知道题目可能的最多的输出的时候就可以这样
12 #输入是4 实际上就是全部文本的前4行
```

输出全排列

输入整数n ($3 \leq n \leq 7$) ,编写程序输出 `1,2,...,n` 整数的全排列, 按字典序输出。

输入格式:

一行输入正整数n。

输出格式:

按字典序输出1到n的全排列。每种排列占一行, 数字间无空格。

输入样例:

在这里给出一组输入。例如:

```
3
```

输出样例:

在这里给出相应的输出。例如:

```
123
132
213
231
312
321
```

设计函数原型: 输入n和prefix, 不返回东西

第一行的123看成是func(3, "12")
prefix已经有了1和2, 剩余的就只有3了
函数里面print(prefix+3)

func(3, "1") 这时候有两个数字可以选: 2,3
func(3, "12")
func(3, "13")
需要循环可选数字

输出全排列

输入整数n ($3 \leq n \leq 7$) ,编写程序输出 `1,2,...,n` 整数的全排列, 按字典序输出。

输入格式:

一行输入正整数n。

输出格式:

按字典序输出1到n的全排列。每种排列占一行, 数字间无空格。

输入样例:

在这里给出一组输入。例如:

```
3
```

输出样例:

在这里给出相应的输出。例如:

```
123
132
213
231
312
321
```

```
def func(n, prefix):
    if len(prefix) == n:
        print(prefix)
        return
    allowedchars = [str(i) for i in range(1,n+1) if str(i) not in prefix]
    for i in allowedchars:
        func(n, prefix+i)

func(int(input()), "")
```

输出全排列

输入整数n ($3 \leq n \leq 7$) ,编写程序输出 `1,2,...,n` 整数的全排列, 按字典序输出。

输入格式:

一行输入正整数n。

输出格式:

按字典序输出1到n的全排列。每种排列占一行, 数字间无空格。

输入样例:

在这里给出一组输入。例如:

```
3
```

输出样例:

在这里给出相应的输出。例如:

```
123
132
213
231
312
321
```

了解一下itertools

```
>>> import itertools
>>> itertools
<module 'itertools' (built-in)>
>>> dir(itertools)
['_doc__', '__loader__', '__name__', '__package__', '__spec__', '_grouper', '_tee', '_tee',
'chain', 'combinations', 'combinations_with_replacement', 'compress', 'count', 'cycle', 'drop
by', 'islice', 'permutations', 'product', 'repeat', 'starmap', 'takewhile', 'tee', 'zip_lo
>>> help(itertools.permutations)
Help on class permutations in module itertools:

class permutations(builtins.object)
    permutations(iterable[, r]) --> permutations object

    Return successive r-length permutations of elements in the iterable.

    permutations(range(3), 2) --> (0, 1), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1)

    Methods defined here:

    __getattr__(self, name, /)
        Return getattr(self, name).

    __iter__(self, /)
        Implement iter(self).

    __new__(*args, **kwargs) from builtins.type
        Create and return a new object.  See help(type) for accurate signature.

    __next__(self, /)
        Implement next(self).
```

输出全排列

输入整数n ($3 \leq n \leq 7$) ,编写程序输出 `1,2,...,n` 整数的全排列, 按字典序输出。

输入格式:

一行输入正整数n。

输出格式:

按字典序输出1到n的全排列。每种排列占一行, 数字间无空格。

输入样例:

在这里给出一组输入。例如:

```
3
```

输出样例:

在这里给出相应的输出。例如:

```
123
132
213
231
312
321
```

了解一下itertools

```
n=int(input())
import itertools
str_1toN = "".join([str(i) for i in range(1,n+1)])
for i in itertools.permutations(str_1toN, n):
    print(*i, sep="")
```

利用标准库已有的函数解放生产力
print的sep参数 每个参数之间的分隔
默认是空格

学会sorted和lambda排序

找出总分最高的学生

```
n=int(input())
students = [input() for _ in range(n)] #students每个元素是一行
thebest = sorted(students, key=lambda i: sum([int(x) for x in i.split()[-3:]]), reverse=True)[0]

#排序用的key是一个函数 输入一行i 返回这一行最后三个数求和; reverse=True逆序,
#结果第0个就是最大的
id, name, f1, f2, f3 = thebest.split()
print(name,id,sum(map(int, [f1,f2,f3])))
```

输入样例:

```
5
00001 huanglan 78 83 75
00002 wanghai 76 80 77
00003 shenqiang 87 83 76
10001 zhangfeng 92 88 78
21987 zhangmeng 80 82 75
```

输出样例:

```
zhangfeng 10001 258
```

二维数组的处理

输入样例1:

```
4
1 7 4 1
4 8 3 6
1 6 1 2
0 7 8 9
```

怎么读取?

```
n=int(input())
data=[]
for _ in range(n):
    data.append([int(i) for i in input().split()])
```

```
>>> data
[[1, 7, 4, 1], [4, 8, 3, 6], [1, 6, 1, 2], [0, 7, 8, 9]]
```

怎么得到第i行?

```
data[i]
```

怎么得到第j列?

```
[x[j] for x in data]
```

```
>>> j=2
>>> [x[j] for x in data]
[4, 3, 1, 8]
```

二维数组的处理

求矩阵鞍点的个数

一个矩阵元素的“鞍点”是指该位置上的元素值在该行上最大、在该列上最小。

找出一行中最大的那个数->判断是不是所在列最小的

错在哪？一行最大的那个数不唯一！

二维数组的处理

求矩阵鞍点的个数

```
1  n=int(input())
2  data=[]
3  for _ in range(n):
4      data.append(list(map(int, input().split())))
5  count=0
6  for i in range(n):
7      for j in range(n):
8          val = data[i][j]
9          if max(data[i])==val and min([data[k][j] for k in range(n)])==val:
10             count+=1
11  print(count)
```

正确做法：直接对每个点判断是否满足鞍点条件

循环初始化问题

判断素数

```
1 n=int(input())
2 data=[int(input()) for i in range(n)]
3 a=2
4 for i in data:
5     while a<i:
6         if i%a==0:
7             print('No')
8             break
9         a=a+1
10    else:
11        print('Yes')
```

判断一个给定的正整数是否素数

输入格式:

输入在第一行给出一个正整数N (≤ 10)，随后N行，每行给出一个小于1000000的需要判断的正整数

输出格式:

对每个需要判断的正整数，如果它是素数，则在一行中输出Yes，否则输出No

输入样例:

在这里给出一组输入。例如:

```
2
11
111
```

Submit Time	Status	Score	Problem	Compiler	Run Time
06/09/2020 12:15:47 AM	Partially Accepted ⓘ	10	7-49	Python (python 3)	31 ms
Case	Result	Run Time	Memory		
0	Accepted	19 ms	2952 KB		
1	Wrong Answer ⓘ	31 ms	2912 KB		

循环初始化问题

发挥主观能动性 多尝试 不要满足于题目给的测试点

另外， 不要从测试样例找规律！ 题目没说就不要自己加戏

```
>>> %Run test.py
```

```
4
```

```
1
```

```
2
```

```
3
```

```
4
```

```
Yes
```

```
Yes
```

```
Yes
```

```
Yes
```

循环初始化问题

哪个循环负责这个变量，就在这个循环之前做初始化，保证外层循环互不干扰

```
1 n=int(input())
2 data=[int(input()) for i in range(n)]
3 for i in data:
4     a=2 ←
5     while a<i:
6         if i%a==0:
7             print('No')
8             break
9         a=a+1
10    else:
11        print('Yes')
```

需要记住点啥

- 类型: `input()`返回的是字符串, 用`int`, `float`转换; 用`isinstance`判断类型; 哪些是`False`
- 数字: 除以0的异常, 进制转换, 判断素数, 格式化输出, 优先级
- 字符串: `len`, `upper`, `lower`, `strip`, `split`, `join`
- 列表: `append`, `index`, `sorted(key=lambda i:-i)`, 数组的数组, 分片, `[::-1]`, 列表推导式
- 字典: `keys`, `items`, `update`
- 循环: `range`, `for i in ...`, `while ...`, 变量初始化
- 函数: 传递一个列表? (引用传递) 局部变量, 全局变量
- `tuple`; `set`; 文件; 类

Thanks

- <https://blog.chenyuan.me/PythonCourse/>
- 祝大家考出好成绩，玩会Python

Table of contents

先本地测试能通过再提交

全角的符号

input只读取输入的一行

扩展一下：你可以使用map函数来批量调用函数

不要输出任何多余的提示信息

input函数返回的是字符串

注意类型转换的int是要调用的 和C语言不一样

注意括号的匹配

方法是需要调用的

注意缩进 不能有多余空格

print的用法

你还可以使用f-string

注意引号括号的匹配，尤其是变得复杂的时候

for,if,else右边都要有分号，每个分号的下一行增加缩进

继续强调 不要出现中文括号

不要出现return 0

看清题目

输入实数别用int()

看清题目输入的格式

看清题目的允许范围

复制粘贴题目的文本就行 不要自己敲

不要追求一行写完

把自己当成Python解释器 要有耐心慢慢做

4-7 (样卷) 下面语句的输出是 (2分)

```
m=[2,1,0,5,3,5,7,4]
def naive(M,A=None):
    if A is None:
        A=set(range(len(M)))
    if len(A)==1:return(A)
    B=set(M[i] for i in A)
    C=A-B
    if C:
        A.remove(C.pop())
        return naive(M,A)
    return A
print(sum(list(naive(m))))
```

```
m=[2,1,0,5,3,5,7,4]
naive(m)
M=m=[2,1,0,5,3,5,7,4]
A=None
if A is None: 成立
    A={0,1,2,3,4,5,6,7}
if len(A)==1: 不成立
B={2,1,0,5,3,7,4}={0,1,2,3,4,5,7}
C=A-B={6}
if C: 成立
    A.remove(6)
    A={0,1,2,3,4,5,7}
    return naive(M, {0,1,2,3,4,5,7})
```

把自己当成Python解释器 要有耐心慢慢做

4-7 (样卷) 下面语句的输出是 (2分)

```
m=[2,1,0,5,3,5,7,4]
def naive(M,A=None):
    if A is None:
        A=set(range(len(M)))
    if len(A)==1:return(A)
    B=set(M[i] for i in A)
    C=A-B
    if C:
        A.remove(C.pop())
        return naive(M,A)
    return A
print(sum(list(naive(m))))
```

```
naive(M, {0,1,2,3,4,5,7})
A= {0,1,2,3,4,5,7}
M=[2,1,0,5,3,5,7,4]
if A is None不成立
if len(A)==1 不成立
B=set([2,1,0,5,3,5,4])={0,1,2,3,4,5}
C={7}
if C成立
    A={0,1,2,3,4,5}
    return naive(M, {0,1,2,3,4,5})
```

```
naive(M, {0,1,2,3,4,5})
B={2,1,0,5,3,5}={0,1,2,3,5}
C={4}
A={0,1,2,3,5}
return naive(M, {0,1,2,3,5})
```


把自己当成Python解释器 要有耐心慢慢做

4-7 (样卷) 下面语句的输出是 (2分)

```
m=[2,1,0,5,3,5,7,4]
def naive(M,A=None):
    if A is None:
        A=set(range(len(M)))
    if len(A)==1:return(A)
    B=set(M[i] for i in A)
    C=A-B
    if C:
        A.remove(C.pop())
        return naive(M,A)
    return A
print(sum(list(naive(m))))
```

```
M=[2,1,0,5,3,5,7,4]
naive(M, {0,1,2,3,5})
B={2,1,0,5,5}={0,1,2,5}
C={3}
A={0,1,2,5}
return naive(M, {0,1,2,5})
```

```
naive(M, {0,1,2,5})
B={2,1,0,5}
C={}
return A={0,1,2,5}
```

答案=0+1+2+5=8