

CS001编程零基础 Python语言入门

第六讲

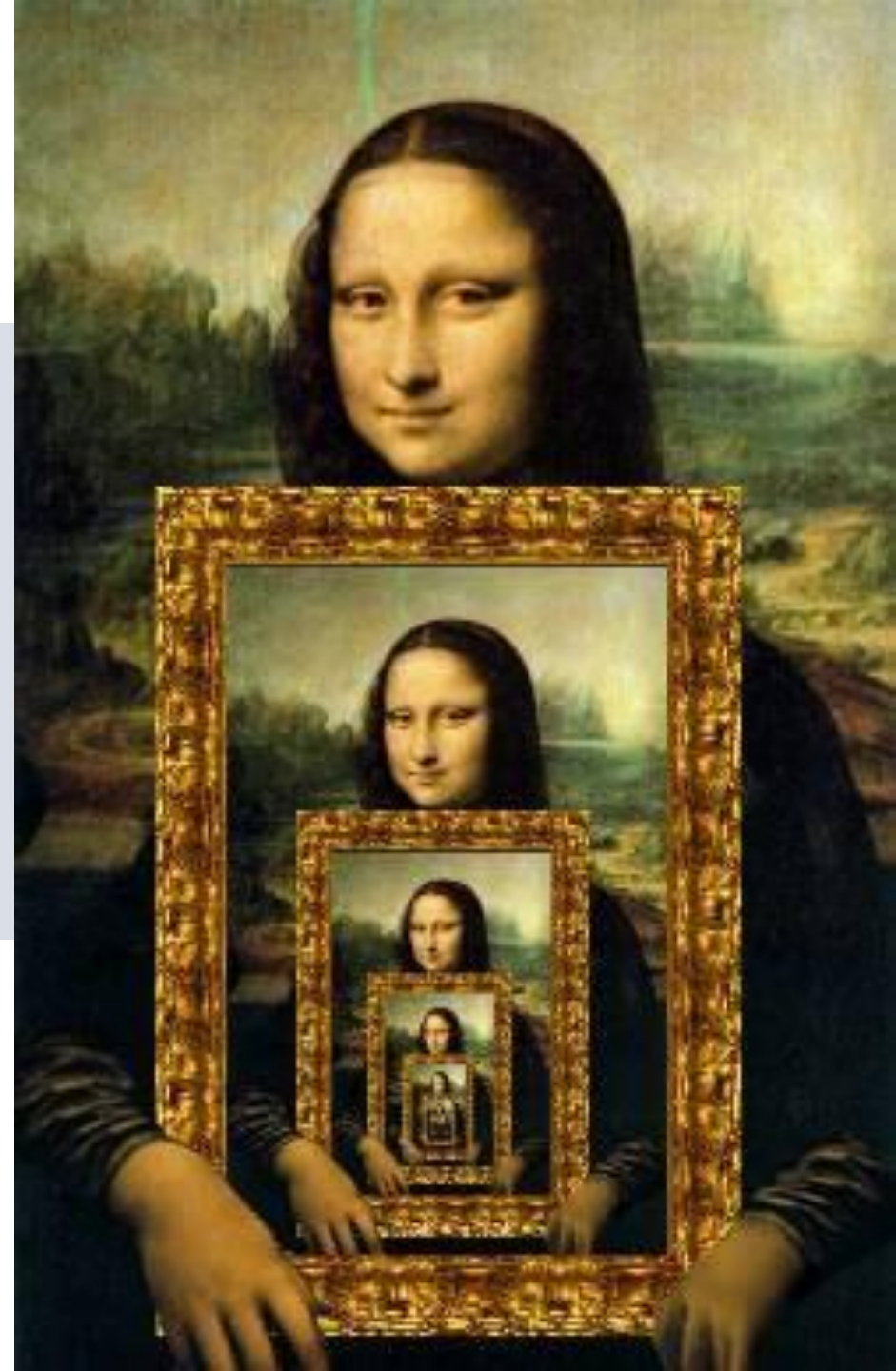
清明节调课通知：

A班周六班，4/1的课换到4/3上课10:10-12:10

B班周六班，4/1的课换到4/3上课13:00-15:00

C班周六班，4/1的课换到4/3上课8:00-10:00

D班周五班不变，3/31正常上课19:00-21:00





作业回顾



第一题：死循环

答案文件名week05solution01.py

```
while True:  
    print("I will never stop")
```

第二题： for语句改成while语句

答案文件名week05solution02.py

```
for n in range(2,100):
```

```
    PRIME = True
```

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

```
        if n % i == 0:
```

```
            PRIME=False
```

```
            print(n,"=",i,"*",n//i)
```

```
            break
```

```
if PRIME:
```

```
    print(n,"is a prime number.")
```

注意
缩进

```
n=2
```

```
while n<100:
```

```
    PRIME = True
```

```
    i=2
```

```
    while i<n:
```

```
        if n % i == 0:
```

```
            PRIME=False
```

```
            print(n,"=",i,"*",n//i)
```

```
            break
```

```
        i=i+1
```

```
if PRIME:
```

```
    print(n,"is a prime number.")
```

```
n=n+1
```


附加题：猜数字游戏升级

附加题（不要求提交解答，只作为拓展）

第一题 猜数字游戏

在第四周介绍的猜数字游戏程序 `guess-game.py` 里，或者是第五周介绍的程序 `guess-game2.py` 里，我们都是用 4 位数作为密码。

1. 请改动原程序，使这个游戏升级成 5 位数密码
2. 请改动原程序，使这个游戏升级，让用户可以通过输入来选择密码是几位数

附加题：猜数字游戏升级

```
import random
def randomanswer(num):
    seeds=list(range(10))
    answer=""
    for i in range(num):
        x=random.randrange(0,10-i)
        answer=answer+str(seeds[x])
        seeds.remove(seeds[x])
    return answer
```

答案文件名guess-game3.py

生成num个随机的不
重复数字

```
def countAB(x,y,num):
    countA,countB=0,0
    for j in range(num):
        if x[j]==y[j]:
            countA=countA+1
        elif x[j] in y:
            countB=countB+1
    print(countA,"A",countB,"B")
    return countA,countB
```

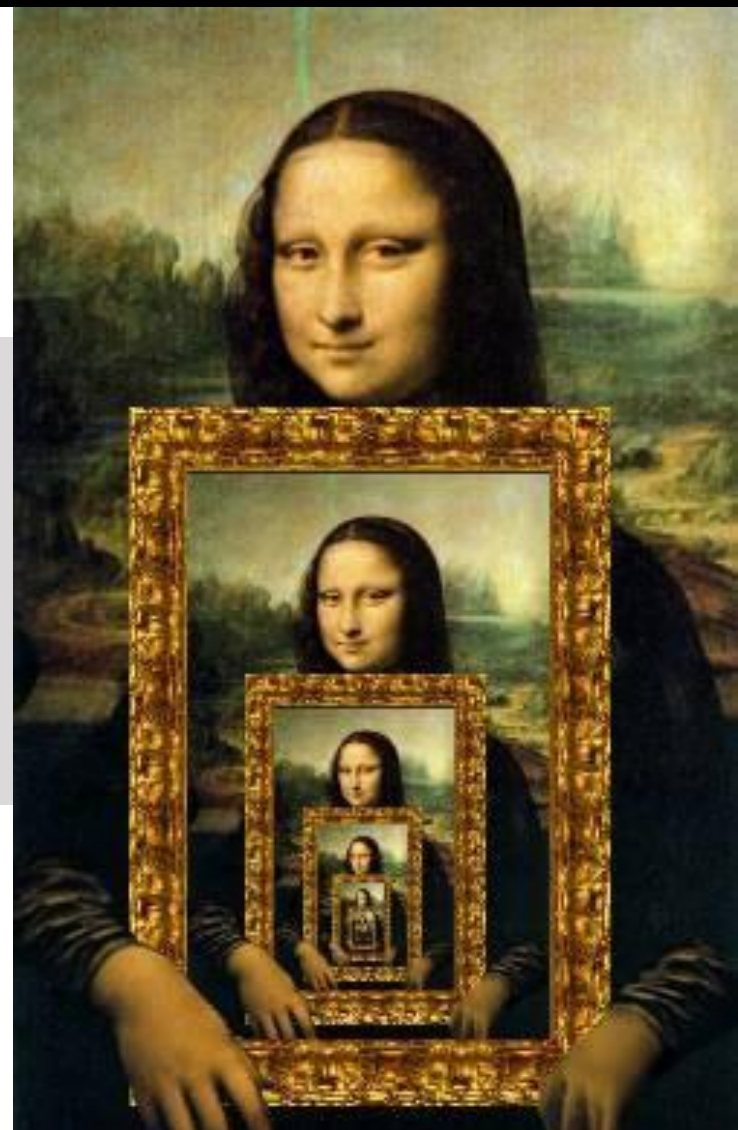
统计答对情况

#主程序

```
level=int(input("How many digits? "))
y=randomanswer(level)
print("Hi! We have generated the answer. You
have 10 chances to guess.")
for i in range(10):
    x=input("What is your guess #"+str(i+1)+"?
")
    A,B=countAB(x,y,level)
    if A==level:
        print("Bingo! You are a genius!")
        break
    if A!=level:
        print("Sorry, the answer is "+str(y))
```



递归 (recursion)



吓得我抱起了

抱着抱着抱着我的小鲤鱼的我的我的我









Hyphaene Compressa - Doum Palm

© Shlomit Pinter



https://www.google.com/#newwindo

Google

recursion

Web

Images


Books

Videos

About 6,570,000 results (0.32 seconds)

Did you mean: *recursion*

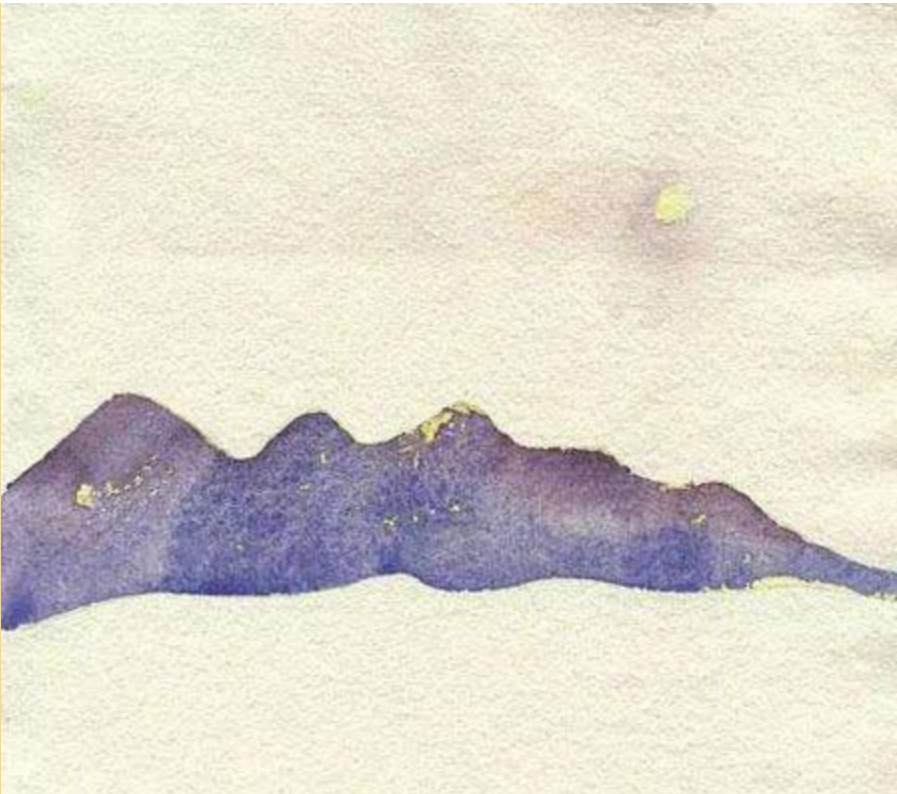
生活中的递归现象



从前有座山，山里有个庙，庙里有个老和尚，给小和尚讲故事。

故事讲的是：从前有座山，山里有个庙，庙里有个老和尚，给小和尚讲故事。

故事讲的是：从前有座山，山里有个庙。。。。。



递归问题举例

有一对兔子,从出生后第3个月起每个月都生一对兔子,小兔子长到第三个月后每个月又生一对兔子,假如兔子都不死,问每个月的兔子对数为多少? (提示: 兔子对数为数列1,1,2,3,5,8,13,21....)

递归问题举例

有5个人坐在一起，问第五个人多少岁？他说比第4个人大2岁。问第4个人岁数，他说比第3个人大2岁。问第三个人，又说比第2人大两岁。问第2个人，说比第一个人大两岁。最后问第一个人，他说是10岁。请问第五个人多大？

递归问题举例

海滩上有一堆桃子，五只猴子来分。第一只猴子把这堆桃子凭据分为五份，多了一个，这只猴子把多的一个扔入海中，拿走了一份。第二只猴子把剩下的桃子又平均分成五份，又多了一个，它同样把多的一个扔入海中，拿走了一份，第三、第四、第五只猴子都是这样做的，问海滩上原来最少有多少个桃子？

递归(recursion)

递归(recursion)是一个函数/功能在其定义中调用自身的一种方法。

递归通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解。

递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。

递归的能力在于用有限的语句来定义对象的无限集合。用递归思想写出的程序往往十分简洁易懂。

吓得我抱起了

抱着抱着抱着我的小鲤鱼的我的我的我



递归练习:求和

递归求和 mysum.py

```
def mysum(n):  
    if n==1:  
        return 1  
    else:  
        return n + mysum(n-1)
```

```
print(mysum(1))  
print(mysum(3))  
print(mysum(10))  
print(mysum(100))
```

申明mysum函数，它的输入为n

如果n==1，返回1，终止递归

否则，继续递归计算mysum(n-1)，将递归结果加上n，返回该数值

递归深度限制如何解除

递归求和 mysum.py

```
def mysum(n):  
    if n==1:  
        return 1  
    else:  
        return n + mysum(n-1)
```

```
print(mysum(1))  
print(mysum(3))  
print(mysum(10))  
print(mysum(100))  
print(mysum(1000))
```

最后一句报错了

RecursionError: maximum recursion depth exceeded in comparison

```
Traceback (most recent call last):  
  File "C:\Users\lenovon\AppData\Local\Programs\Python\Python36-32\mysum.py", line 11, in <module>  
    print(mysum(1000))  
  File "C:\Users\lenovon\AppData\Local\Programs\Python\Python36-32\mysum.py", line 5, in mysum  
    return n+mysum(n-1)  
  File "C:\Users\lenovon\AppData\Local\Programs\Python\Python36-32\mysum.py", line 5, in mysum  
    return n+mysum(n-1)  
  File "C:\Users\lenovon\AppData\Local\Programs\Python\Python36-32\mysum.py", line 5, in mysum  
    return n+mysum(n-1)  
  [Previous line repeated 989 more times]  
  File "C:\Users\lenovon\AppData\Local\Programs\Python\Python36-32\mysum.py", line 2, in mysum  
    if n==1:  
RecursionError: maximum recursion depth exceeded in comparison  
>>>
```

递归深度限制如何解除

递归求和 mysum2.py

```
import sys
```

```
def mysum(n):
```

```
    if n==1:
```

```
        return 1
```

```
    else:
```

```
        return n + mysum(n-1)
```

```
sys.setrecursionlimit(2000)
```

```
print(mysum(1))
```

```
print(mysum(3))
```

```
print(mysum(10))
```

```
print(mysum(100))
```

```
print(mysum(1000))
```

导入sys系统工具模块

通过sys模块里的setrecursionlimit()工具
修改递归的最大层数限制

递归练习：阶乘

```
def myfactorial(n):
```

```
    if n==1 or n==0:
```

```
        return 1
```

```
    else:
```

```
        return n * myfactorial(n-1)
```

```
print(mymfactorial(0))
```

```
print(mymfactorial(1))
```

```
print(mymfactorial(3))
```

```
print(mymfactorial(5))
```

```
print(mymfactorial(10))
```

```
print(mymfactorial(100))
```

递归求阶乘 myfactorial.py

申明myfactorial函数，它的输入为n

如果n==1或者0，返回1，终止递归

否则，继续递归计算myfactorial(n-1)，将递归结果乘上n，返回该数值

回顾： def函数定义举例： 数有几个奇数

文件名countodd.py

```
def countodd(a):  
    count=0  
    for x in a:  
        if x % 2==1:  
            count=count+1  
    return count
```

```
t=[1,2,3,4,5,11,12,13,14,15]  
print(countodd(t))  
t.remove(5)  
print(countodd(t))  
t.remove(3)  
print(countodd(t))  
t=t+[9,9,9]  
print(countodd(t))
```

如何用递归改
写这个函数？

递归练习：数有几个奇数

文件名countodd.py

```
def countodd(a):  
    count=0  
    for x in a:  
        if x % 2==1:  
            count=count+1  
    return count
```

非递归版本

```
t=[1,2,3,4,5,11,12,13,14,15]  
print(countodd(t))  
t.remove(5)  
print(countodd(t))  
t.remove(3)  
print(countodd(t))  
t=t+[9,9,9]  
print(countodd(t))
```

文件名countodd2.py

```
def countodd2(a):  
    if a==[]:  
        return 0  
    else:  
        return a[0] % 2 + countodd2(a[1:])
```

递归版本

```
t=[1,2,3,4,5,11,12,13,14,15]  
print(countodd2(t))  
t.remove(5)  
print(countodd2(t))  
t.remove(3)  
print(countodd2(t))  
t=t+[9,9,9]  
print(countodd2(t))
```

递归方法总结：三要素

递归有三个要素：

1. 结束条件
2. 递归前进段
3. 递归返回段

当结束条件不满足时，递归前进；
当结束条件满足时，递归返回。

在使用递归时，必须有一个明确的递归结束条件，称为递归出口，否则将无限递归下去！

递归的三个要素

请在右侧代码中识别出递归的三个要素：

1. 结束条件
2. 递归前进段
3. 递归返回段

```
def countodd2(a):
```

```
    if a==[]:
```

```
        return 0
```

```
    else:
```

```
        return a[0] % 2 + countodd2(a[1:])
```

递归版本

```
def mysum(n):
```

```
    if n==1:
```

```
        return 1
```

```
    else:
```

```
        return n + mysum(n-1)
```

递归版本

```
def myfactorial(n):
```

```
    if n==1 or n==0:
```

```
        return 1
```

```
    else:
```

```
        return n * myfactorial(n-1)
```

递归版本

递归思维 vs 循环思维

```
def countodd(a):  
    count=0  
    for x in a:  
        if x % 2==1:  
            count=count+1  
    return count
```

非递归版本

```
def mysum(n):  
    tot=0  
    for i in range(1,n+1):  
        tot=tot+i  
    return tot
```

非递归版本

```
def myfactorial(n):  
    tot=1  
    for i in range(1,n+1):  
        tot=tot*i  
    return tot
```

非递归版本

```
def countodd2(a):  
    if a==[]:  
        return 0  
    else:  
        return a[0] % 2 + countodd2(a[1:])
```

递归版本

```
def mysum(n):  
    if n==1:  
        return 1  
    else:  
        return n + mysum(n-1)
```

递归版本

```
def myfactorial(n):  
    if n==1 or n==0:  
        return 1  
    else:  
        return n * myfactorial(n-1)
```

递归版本

递归方法解汉诺塔游戏

有三根杆子A，B，C。A杆上有N个($N > 1$)穿孔圆盘，盘的尺寸由下到上依次变小。要求按下列规则将所有圆盘移至C杆：

- 1、每次只能移动一个圆盘；
- 2、大盘不能叠在小盘上面。

提示：可将圆盘临时置于B杆，也可将从A杆移出的圆盘重新移回A杆，但都必须遵循上述两条规则。

问：如何移？最少要移动多少次？



递归方法解汉诺塔游戏

有三根杆子A，B，C。A杆上有N个($N > 1$)穿孔圆盘，盘的尺寸由下到上依次变小。要求按下列规则将所有圆盘移至C杆：

- 1、每次只能移动一个圆盘；
- 2、大盘不能叠在小盘上面。

提示：可将圆盘临时置于B杆，也可将从A杆移出的圆盘重新移回A杆，但都必须遵循上述两条规则。

问：如何移？最少要移动多少次？



$N=1$ 时，1个圆盘的移动总次数=1

$N > 1$ 时， N 个圆盘移动总次数= $(N-1)$ 个圆盘移动总次数 * 2 + 1

递归方法解汉诺塔游戏

文件名hanoi.py

```
def hanoi(n,x,y,z):  
    if n==1:  
        print("Move disk",1,"from",x,"to",z)  
    else:  
        hanoi(n-1,x,z,y)  
        print("Move disk",n,"from",x,"to",z)  
        hanoi(n-1,y,x,z)  
  
hanoi(3,"A","B","C")
```

请识别出递归的三个要素：

1. 结束条件
2. 递归前进段
3. 递归返回段

```
Move disk 1 from A to C  
Move disk 2 from A to B  
Move disk 1 from C to B  
Move disk 3 from A to C  
Move disk 1 from B to A  
Move disk 2 from B to C  
Move disk 1 from A to C  
>>>
```

递归方法解汉诺塔游戏

文件名hanoi2.py

```
def hanoi(n,x,y,z):  
    if n>0:  
        hanoi(n-1,x,z,y)  
        print("Move disk",n,"from",x,"to",z)  
        hanoi(n-1,y,x,z)  
  
hanoi(3,"A","B","C")
```

请识别出递归的三个要素：

1. 结束条件
2. 递归前进段
3. 递归返回段

```
Move disk 1 from A to C  
Move disk 2 from A to B  
Move disk 1 from C to B  
Move disk 3 from A to C  
Move disk 1 from B to A  
Move disk 2 from B to C  
Move disk 1 from A to C  
>>>
```


递归方法解汉诺塔游戏

打开文件名为minimal-hanoi.py的程序，尝试理解每段代码含义

