模拟题讲解

Python模拟题讲解ppt

江苏省信息技术学业水平考试

2023.12.01

19. 防偷窥密码锁

该密码锁设计比较人性化，输入的密码中只要包含正确密码(需要连续，比如，密码设置为“202312”，当输入“4568348720231224564”，也可以认为输入正确)，就可打开，但是如果连续3次输入都错误就会被锁定。

6位数字连续且正确

```
password="202312"  #设初始密码为202312
n=0 输入次数赋值给变量n，n的初始值赋值为
while n<3:
    yours=input("请输入密码：")
    n+=1
    if password ① yours:
        print("欢迎回家")
        break )  #跳出循环
    else ):
        print("密码错误")
if n == ) 3 :
    print("错误3次，请稍后再试")
```

输入密码（可输3次）

while循环，只要n<3 就循环。

判断分支，如果密码正确，就打印“欢迎回家”并跳出循环；如果密码错误，打印“密码错误”并循环（3次以内）。

判断分支，如果错误3次，就打印“错误3次，请稍后再试”并结束程序。

读懂题目，读懂代码，先将懂的代码填写好。

19. 防偷窥密码锁

该密码锁设计比较人性化，输入的密码中只要包含正确密码(需要连续，比如，密码设置为“202312”，当输入“456834872023122456 4”，也可以认为输入正确)，就可打开，但是如果连续3次输入都错误就会被锁定。

6位数字连续且正确

```
password="202312"  #设初始密码为202312
n=0
while n<3:
    yours=input("请输入密码：")
    n+=1
    if passwo in ① yours:
        print("欢迎回家")
        break ) #跳出循环
    else ):
        print("密码错误")
if n == ④ 3 :
    print("错误3次，请稍后再试")
```

什么数据类型？ **字符串(str)**

字符串的 in 运算

in 包含　　not in 不包含

```
i="hello"
j="he"
print(j in i)
```
运行结果：True

```
i="hello"
j="eo"
print(j in i)
```
运行结果：False

```
i="1234"
j="23"
print(j in i)
```
运行结果：True

```
i="1234"
j="32"
print(j in i)
```
运行结果：False

in 运算还可以用于列表等

20. 生成连续数

输入任意一个正整数n，编程实现：输出由1到n（包括n）按顺序组合成的新数123……n，例如输入n为9，则输出新数为123456789。

```
n= int(input('请输入自然数n:'))
new=''
for i in range(1,n+1):
    new=new+str(i)      #生成新的数
print(new)
```

输入正整数赋值给变量n

空字符串赋值给变量new

for循环，通过循环不断增加数字以生成新的连续数。

循环结束，输出最终结果。

读懂题目，读懂代码，先将懂的代码填写好。

new = new + i

字符串数据 +? 整数型数据

字符串数据能和整数型数据相加吗？

字符串类型数据不可以和整数型数据相加！

因为最终输出变量new是字符串类型，所以必须改变 变量 i 的数据类型

i ⟶ str(i)

21. 回文数

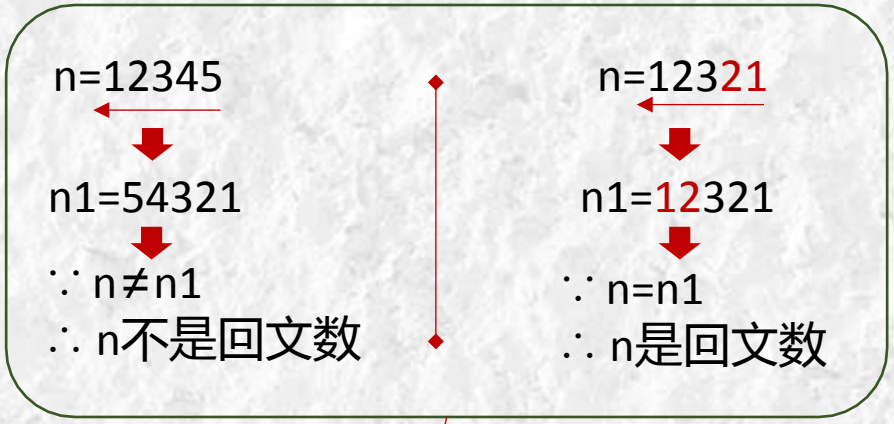
如果一个正整数n，若它的反向排列所得的自然数n1与它本身相等，则n为回文数。
例如12321就是回文数。编程实现：输入任意一个正整数n，判断它是否为回文数。

```
def rev(x):
    m=x
    n1=0
    while    ①    :
        n1=n1*10+
        x=x//10
    if m ==   )    n1:
        return "是回文数。"
    else:
        return "不是回文数。"
n=int(input("请输入任意正整数："))
print( n    rev(n      )
                   )
```

② 通过循环将输入的数反向排列，赋值给n1。

根据是否相等判断是否是回文数。

自定义函数



21. 回文数

如果一个正整数n，若它的反向排列所得的自然数n1与它本身相等，则n为回文数。
例如12321就是回文数。编程实现：输入任意一个正整数n，判断它是否为回文数。

```
def rev(x):
    m=x
    n1=0
    while ①  :
        n1=n1*10+
        x=x//10
    if m ==  )  n1:
        return "是回文数。"
    else:
        return "不是回文数。"
n=int(input("请输入任意正整数："))
print( n rev(n
)
```

②

通过循环将输入的数反向排列，赋值给n1。

根据是否相等判断是否是回文数。

自定义函数

21. 回文数

如果一个正整数n，若它的反向排列所得的自然数n1与它本身相等，则n为回文数。
例如12321就是回文数。编程实现：输入任意一个正整数n，判断它是否为回文数。

```
def rev(x):
    m=x
    n1=0
    while ①  :
        n1=n1*10+
        x=x//10
    if m ==  )  n1:
        return "是回文数。"
    else:
        return "不是回文数。"
n=int(input("请输入任意正整数："))
print( n rev(n
)
```

②

通过循环将输入的数反向排列，赋值给n1。

根据是否相等判断是否是回文数。

自定义函数

21. 回文数

如果一个正整数n，若它的反向排列所得的自然数n1与它本身相等，则n为回文数。
例如12321就是回文数。编程实现：输入任意一个正整数n，判断它是否为回文数。

```
def rev(x):
    m=x
    n1=0
    while x>0 :
        n1=n1*1 x%10 )
        x=x//10
    if m ==  )  n1:
        return "是回文数。"
    else:
        return "不是回文数。"
n=int(input("请输入任意正整数："))
print( n rev(n
    )
```

为什么要设置一个m变量呢？

通过循环将输入的数反向排列，赋值给n1。

根据是否相等判断是否是回文数。

自定义函数

x	→	ge	→	n1
12345	12345%10	5	0*10+5	5
1234	1234%10	4	5*10+4	54
123	123%10	3	54*10+3	543
12	12%10	2	543*10+2	5432
1	1%10	1	5432*10+1	54321
x=x//10		ge=x%10	n1=n1*10+ge	

n1=n1*10+x%10

22. 回文素数

寻找出100-1000范围内是素数（又称质数）的回文数。

```
import math
def prime(n):
    for i in range(2,int(math.sqrt(n))+1):
        if ____①____:
            return False
    _____②_____

def rev(n):
    t=0
    while ____③____:
        t=t*10+n%10
        n=n//10
    return t

for i in range(100,1001):
    if rev(i)==i:
        if _____④_____:
            print(i,end=" ")
```

22. 回文素数

素数（质数）是指除了1和它本身外，不能被其它自然数整除的自然数。

寻找出100-1000范围内是素数（又称质数）的回文数。

见上一讲

整除→取余　n%i==0

22. 回文素数

素数（质数）是指除了1和它本身外，不能被其它自然数整除的自然数。

寻找出100-1000范围内是素数（又称质数）的回文数。

编程解决：输入一个自然数，判断这个自然数是不是素数。

整除→取余　n%i==0

包含2到n-1的数

```
n=int(input("输入一个自然数："))
for i in range(2,n):
    if n%i==0:
        print(n,"不是素数")
        break
else:
    print(n,"是素数")
```

比如我们输入10

我们可以用比10小的所有数（除了1和10本身以外）依次作为除数，循环去除输入的数。只要能被其中一个数整除，那么这个数就不是素数。

只要能被其它数整除，就不是素数，用break终止循环。

否则，继续循环。所有数都不能整除，则该数为素数。

但是，这个程序有个小问题！

这个else代码部分一直在循环当中。？

22. 回文素数

素数（质数）是指除了1和它本身外，不能被其它自然数整除的自然数。

寻找出100-1000范围内是素数（又称质数）的回文数。

编程解决：输入一个自然数，判断这个自然数是不是素数。

整除→取余　n%i==0

```
n=int(input("输入一个自然数："))
for i in range(2,n):
    if n%i==0:
        print(n,"不是素数")
        break
else:
    print(n,"是素数")
```

比如我们输入10

我们可以用比10小的所有数（除了1和10本身以外）依次作为除数，只要能被其中一个数整除，那么这个数就不是素数。

只要能被其它数整除，就不是素数，用break终止循环。

否则，继续循环。所有数都不能整除，则该数为素数。

将else代码整体移出循环。

22. 回文素数

素数（质数）是指除了1和它本身外，不能被其它自然数整除的自然数。

寻找出100-1000范围内是素数（又称质数）的回文数。

编程解决：输入一个自然数，判断这个自然数是不是素数。

整除→取余　n%i==0

```
import math
n=int(input("输入一个自然数："))
for i in range(2,n):
    if n%i==0:
        print(n,"不是素数")
        break
else:
    print(n,"是素数")
```

数学老师说：判断一个数是否是素数不需要用（2……n-1）这么多的数来一个个整除。如果n是100，难道还得从2……99循环98次吗？不需要！只需要看从2到n的平方根之间是否有整除的自然数就OK

```
int(math.sqrt(n))+1
```

22. 回文素数

素数（质数）是指除了1和它本身外，不能被其它自然数整除的自然数。

寻找出100-1000范围内是素数（又称质数）的回文数。

编程解决：输入一个自然数，判断这个自然数是不是素数。

整除→取余　n%i==0

```
import math
n=int(input("输入一个自然数："))
for i in range(2,int(math.sqrt(n))+1):
        if n%i==0:
            print(n,"不是素数")
            break
else:
    print(n,"是素数")
```

数学老师说：判断一个数是否是素数不需要用（2……n-1）这么多的数来一个个整除。如果n是100，难道还得从2……99循环98次吗？不需要！只需要看从2到n的平方根之间是否有整除的自然数就OK

int(math.sqrt(n))+1

22. 回文素数

素数（质数）是指除了1和它本身外，不能被其它自然数整除的自然数。

寻找出100-1000范围内是素数（又称质数）的回文数。

编程解决：输入一个自然数，判断这个自然数是不是素数。

整除→取余　n%i==0

还有其它的程序写法：

```
import math
n=int(input())
s=1
for i in range(2, int(math.sqrt(n))+1):
    if n%i==0:
        s=0
        break
    else:
        s=1
if s==1:
    print(n, “是素数")
else:
    print(n, “不是素数")
```

22. 回文素数

寻找出100-1000范围内是素数（又称质数）的回文数。

```
import math
def prime(n):
    for i in range(2,int(math.sqrt(n))+1):
        if n%i==0 :
            return False
    return n
```

自定义函数确定输入的数是否是素数。

```
def rev(n):
    t=0
    while n>0 :
        t=t*10+n%10
        n=n//10
    return t
```

自定义函数反向排列输入的数。
反向排列的数赋值给变量 t

```
for i in range(100,1001):
    if rev(i)==i:
        if prime(i)==i :
            print(i,end=" ")
```

主程序调用两个自定义函数

反向排列的数和原数如果相等，即回文数。

如果输入数和返回值相等，即素数。

23. 杨辉三角

递归法打印杨辉三角。杨辉三角的特点是每个数字等于上一行的左右两个数字之和。即第 n+1 行的第 i 个数等于第 n 行的第 i-1 个数和第 i 个数之和。

```
def yanghui(x, y):
    if y ==   ①   or y == x:
        return 1
    else:
        z = yanghui(   ②   , y-1)+yanghui(x-1,   ③   )
# 第x行第y个数字等于第x-1行的第y-1个数和第y个数之和。
        return z

n = int(input("请输入杨辉三角形的行数："))
for i in range(1,   ④   ): # 根据输入的行数打印杨辉三角形
    for j in range(1, i+1):
        print(yanghui(i, j), end=" ") #调用函数，打印第i行的数
    print()
```

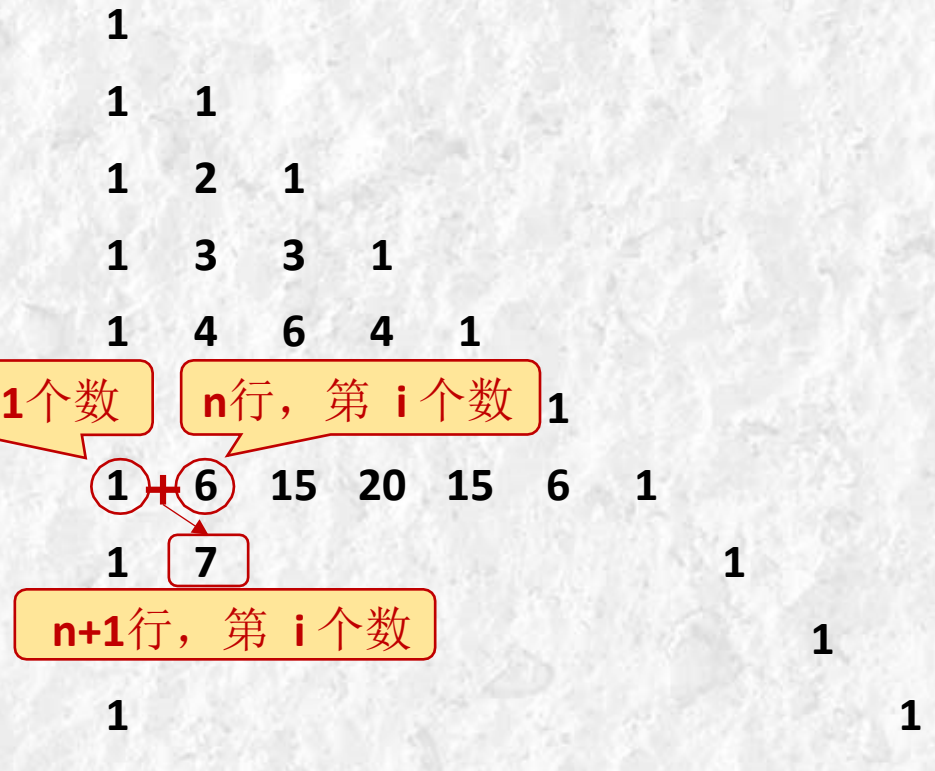
```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
1   6   15  20  15  6   1
1
1
1
```

23. 杨辉三角

递归法打印杨辉三角。杨辉三角的特点是每个数字等于上一行的左右两个数字之和。即第 n+1 行的第 i 个数等于第 n 行的第 i-1 个数和第 i 个数之和。

```
def yanghui(x, y):
    if y ==    ①    or y == x:
        return 1
    else:
        z = yanghui(    ②    , y-1)+yanghui(x-
# 第x行第y个数字等于第x-1行的第y-1个数和第y个数之和。
        return z

n = int(input("请输入杨辉三角形的行数："))
for i in range(1,    ④    ): # 根据输入的行数打印杨辉三角形
    for j in range(1, i+1):
        print(yanghui(i, j), end=" ") #调用函数，打印第i行的数
    print()
```



23. 杨辉三角

递归法打印杨辉三角。杨辉三角的特点是每个数字等于上一行的左右两个数字之和。即第 n+1 行的第 i 个数等于第 n 行的第 i-1 个数和第 i 个数之和。

```
def yanghui(x, y):
    if y ==    ①    or y == x:
        return 1
    else:
        z = yanghui(    ②    , y-1)+yanghui(x-1,    ③    )
# 第x行第y个数字等于第x-1行的第y-1个数和第y个数之和。
        return z

n = int(input("请输入杨辉三角形的行数："))
for i in range(1,    ④    ): # 根据输入的行数打印杨辉三角形
    for j in range(1, i+1):
        print(yanghui(i, j), end=" ") #调用函数，打印第i行的数
    print()
```

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
1   6   15  20  15  6   1
1   7                   1
1                       1
1                           1
```

23. 杨辉三角

递归法打印杨辉三角。杨辉三角的特点是每个数字等于上一行的左右两个数字之和。即第 n+1 行的第 i 个数等于第 n 行的第 i-1 个数和第 i 个数之和。

```
def yanghui(x, y):
    if y ==    ①    or y == x:
        return 1
    else:
        z = yanghui(    ②    , y-1)+yanghui(x-1,    ③    )
# 第x行第y个数字等于第x-1行的第y-1个数和第y个数之和。
        return z

n = int(input("请输入杨辉三角形的行数："))
for i in range(1,    ④    ): # 根据输入的行数打印杨辉三角形
    for j in range(1, i+1):
        print(yanghui(i, j), end=" ") #调用函数，打印第i行的数
    print()
```

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
1   6   15  20  15  6   1
1   7   21              1
                        1
1                           1
```

n行，第 i -1个数

n行，第 i 个数

n+1行，第 i 个数