## 1.

S1 == 0; S2 == 1; S3 == 0

## 2.

D = ABC + C + B +A

E = C + B + BC

## 3.

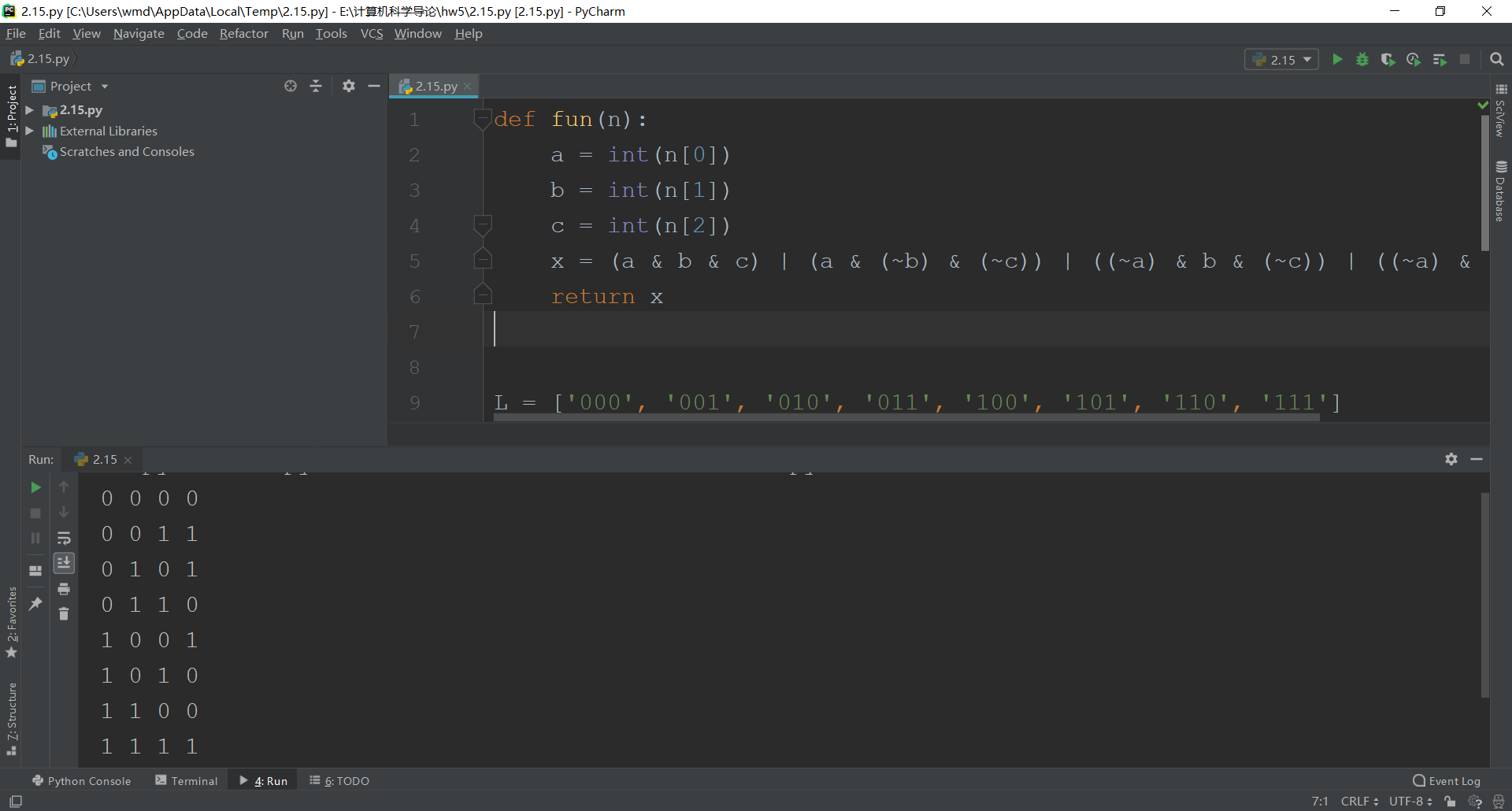
(1) BC

(2) AE+C+C

(3) AB++B

## 4.

def fun(n):  
 a = int(n[0])  
 b = int(n[1])  
 c = int(n[2])  
 x = (a & b & c) | (a & (~b) & (~c)) | ((~a) & b & (~c)) | ((~a) & (~b) & c)  
 return x  
  
  
L = ['000', '001', '010', '011', '100', '101', '110', '111']  
for each in L:  
 print(each[0], end=' ')  
 print(each[1], end=' ')  
 print(each[2], end=' ')  
 print(fun(each))



## 5.﷒

当其中一个列表长度为0时直接返回了另外一个， 没有将carry与其进一步计算

## 6.

-1或-6

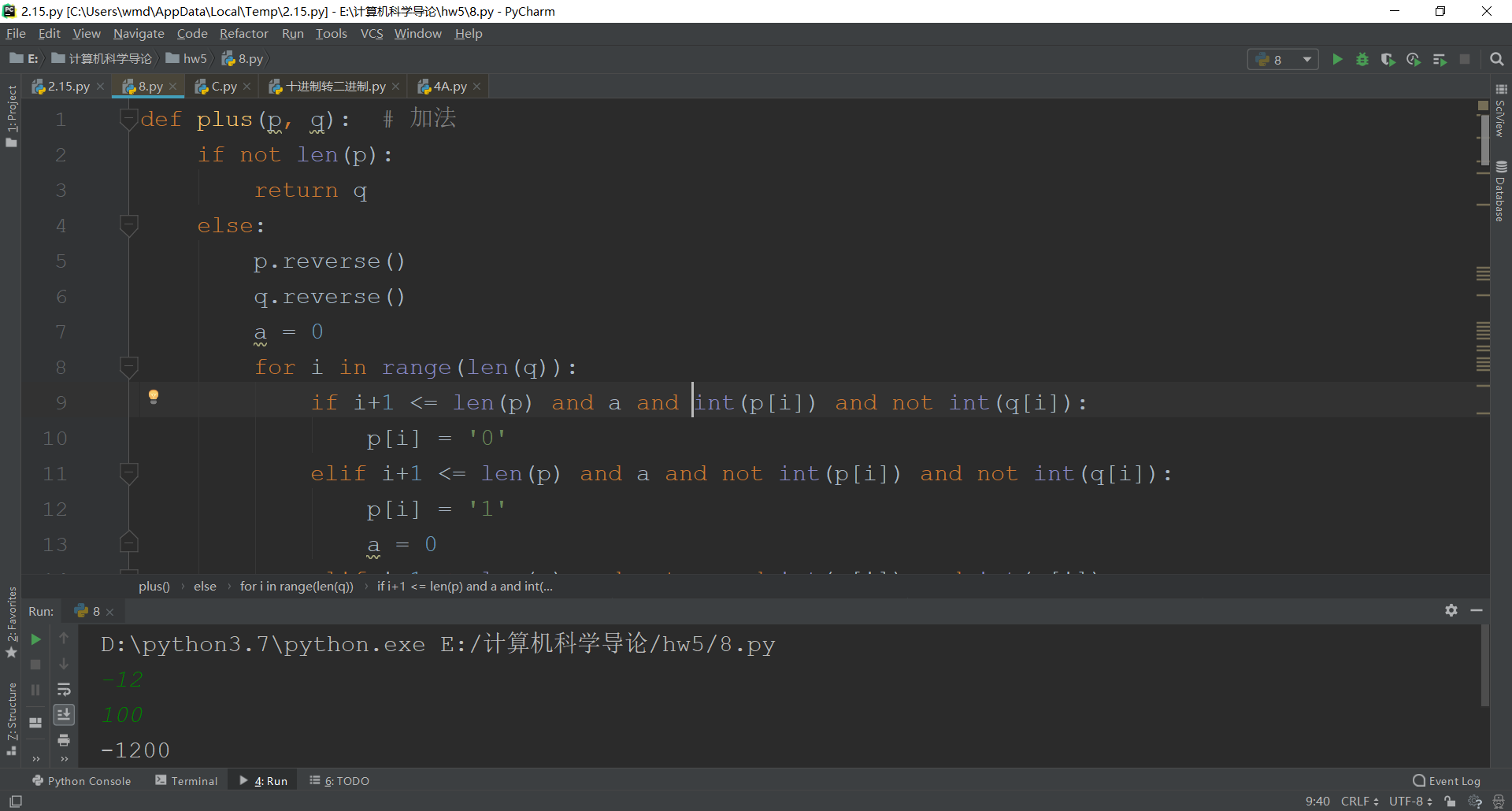
## 7.

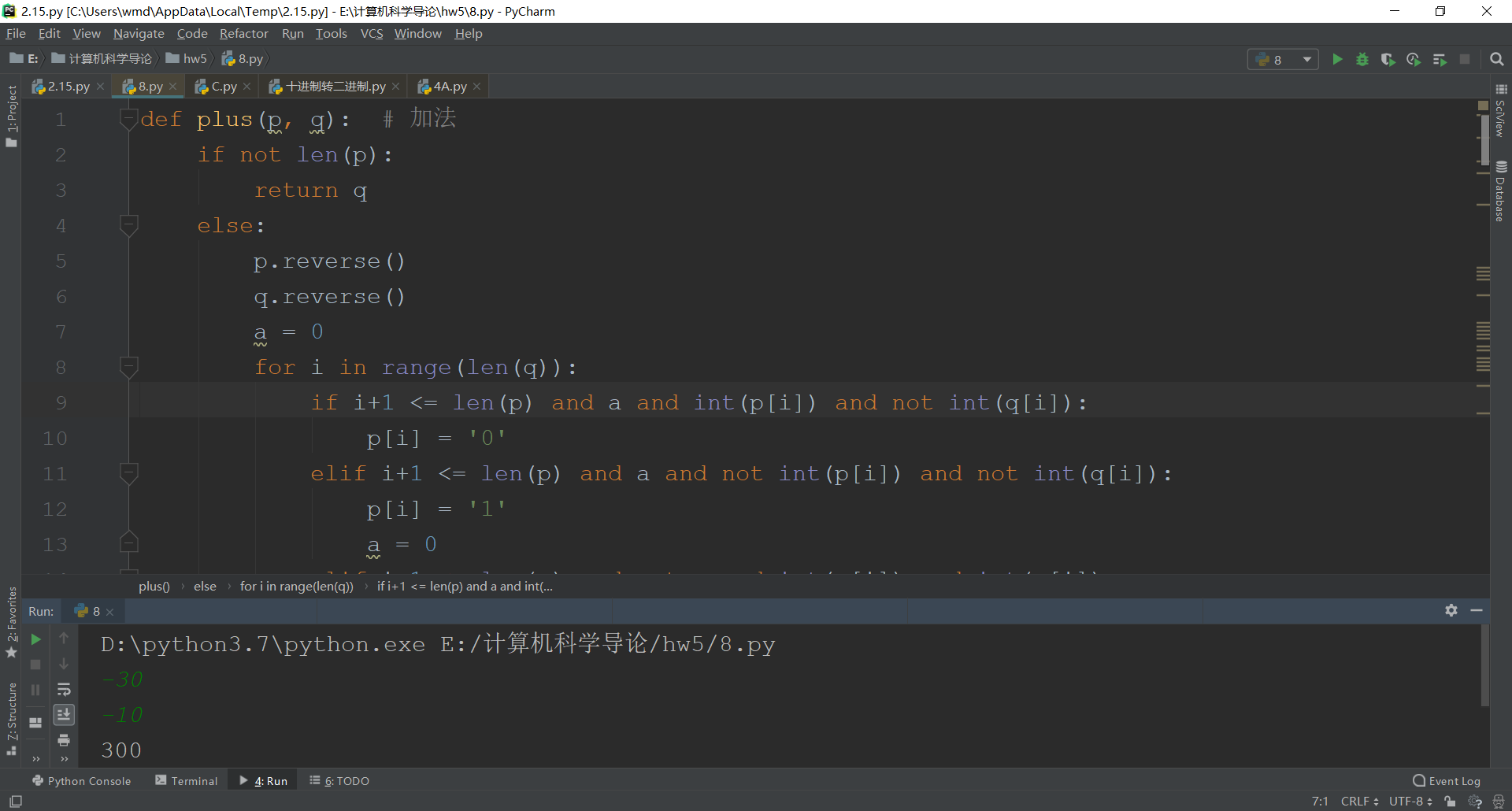
A: 10111110000000000000000000000000

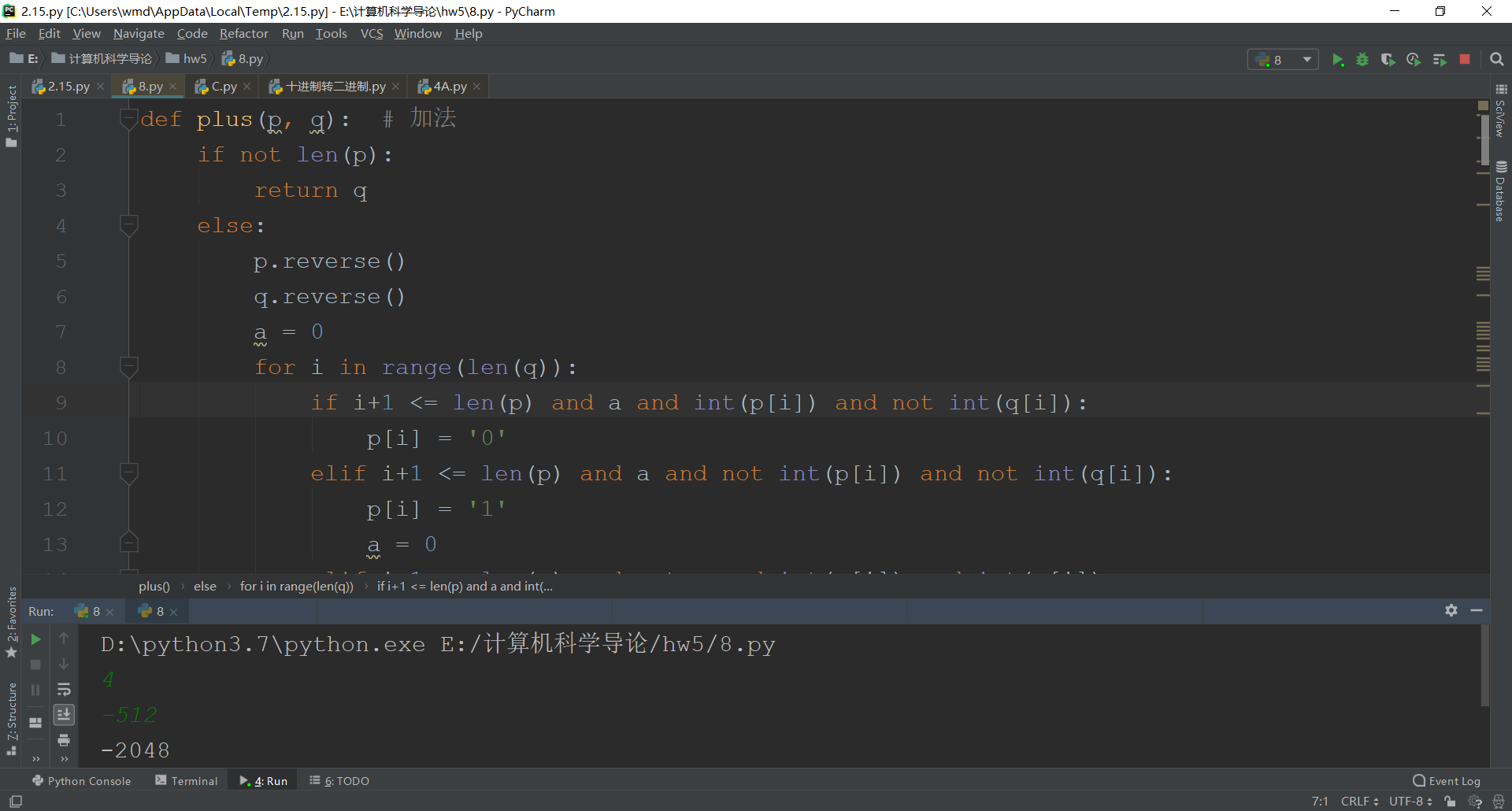
B: 01000100100000000001110000000000

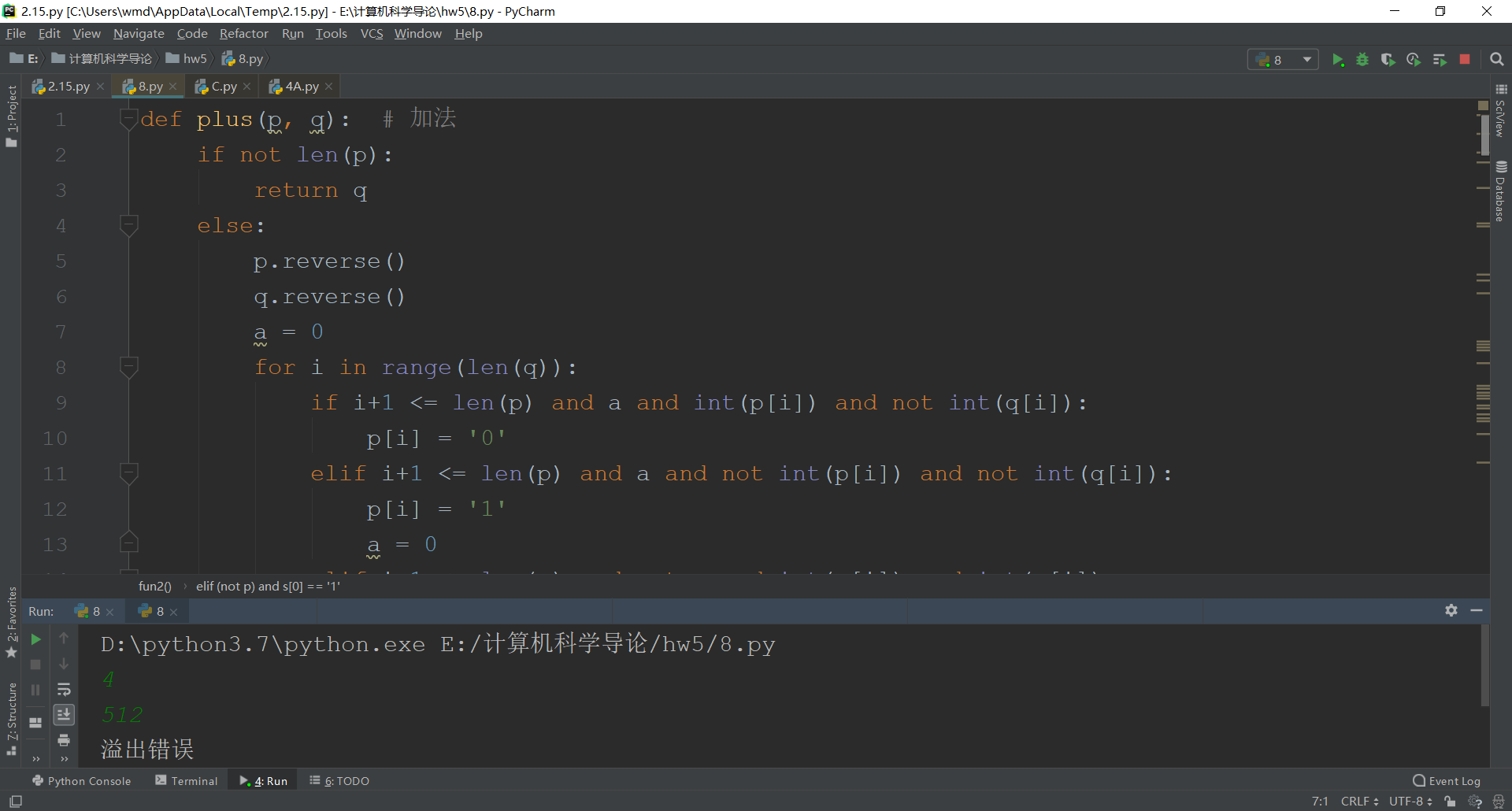
## 8.

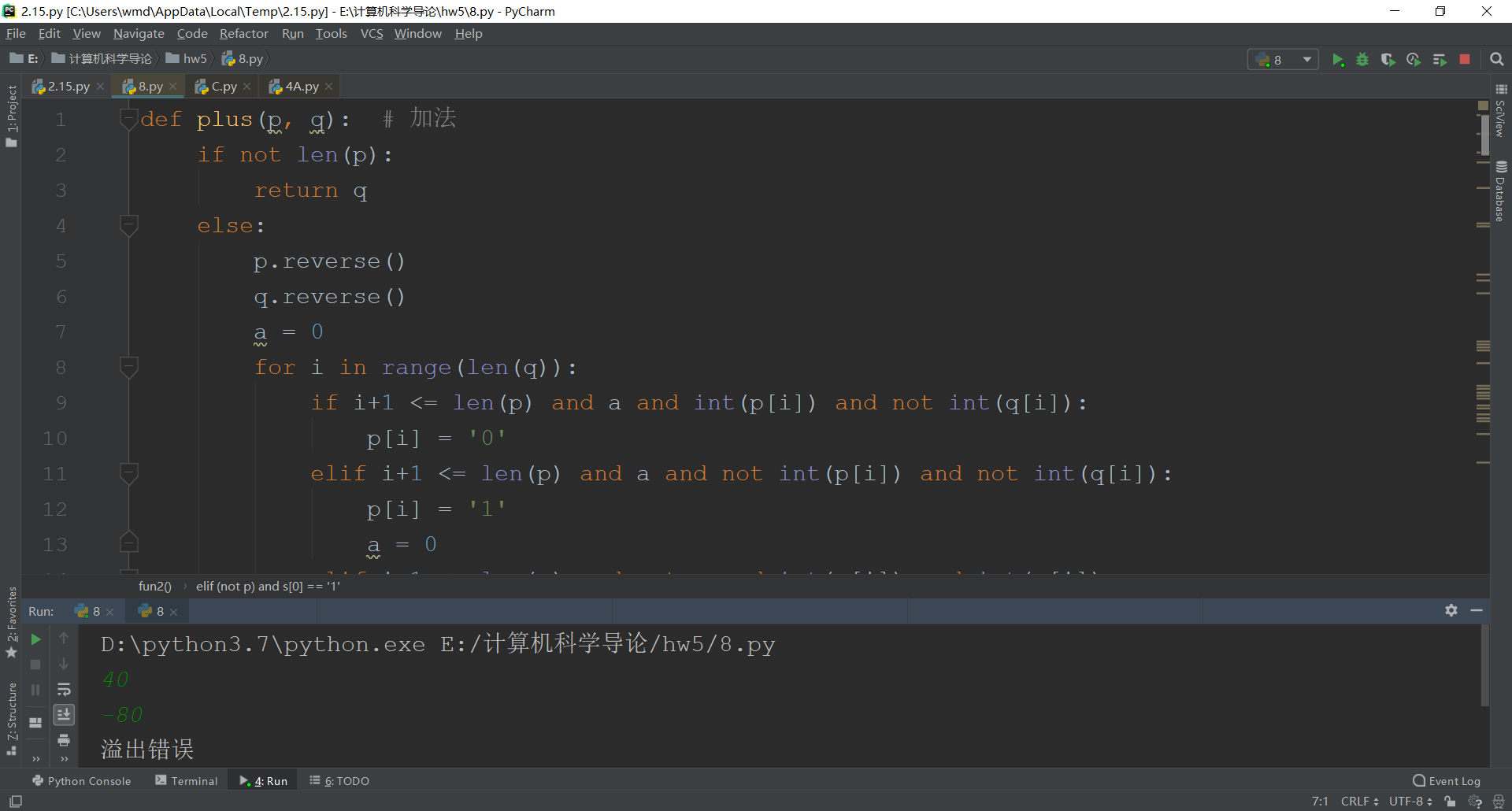
def plus(p, q): # 加法  
 if not len(p):  
 return q  
 else:  
 p.reverse()  
 q.reverse()  
 a = 0  
 for i in range(len(q)):  
 if i+1 <= len(p) and a and int(p[i]) and not int(q[i]):  
 p[i] = '0'  
 elif i+1 <= len(p) and a and not int(p[i]) and not int(q[i]):  
 p[i] = '1'  
 a = 0  
 elif i+1 <= len(p) and not a and int(p[i]) and int(q[i]):  
 p[i] = '0'  
 a = 1  
 elif i+1 <= len(p) and not a and not int(p[i]) and int(q[i]):  
 p[i] = '1'  
 elif i+1 > len(p):  
 if a and int(q[i]):  
 p.append('0')  
 elif a:  
 p.append('1')  
 a = 0  
 elif int(q[i]):  
 p.append('1')  
 else:  
 p.append('0')  
 if a:  
 p.append('1')  
 p.reverse()  
 return p  
  
  
def fun(a): # 转码  
 L = []  
 for each in a:  
 L.append(each)  
 for i in range(len(L)):  
 if L[i] == '1':  
 L[i] = '0'  
 else:  
 L[i] = '1'  
 k = 1  
 for i in range(len(L)-1, -1, -1):  
 if k and L[i] == '0':  
 L[i] = '1'  
 k = 0  
 elif k and L[i] == '1':  
 L[i] = '0'  
 p = ''  
 for each in L:  
 p = p + each  
 return p  
  
  
def fun2(s, p): # 转十进制  
 if len(s) > 12:  
 return 9999  
 elif p and s[0] == '0':  
 return 9999  
 elif (not p) and s[0] == '1':  
 return 9999  
 else:  
 r = int(s, 2)  
 if r >= 2048:  
 return r - 4096  
 else:  
 return r  
  
  
def fun1(a): # 十进制转二进制  
 x = int(a)  
 r = 0  
 Rs = []  
 while x:  
 r = x % 2  
 x //= 2  
 Rs = [r] + Rs  
 ans = ''  
 for i in range(len(Rs)):  
 ans = ans + str(Rs[i])  
 return ans  
  
  
def fun3(x, y): # 乘法  
 L1 = [i for i in x]  
 L2 = [i for i in y]  
 a = len(L1)  
 b = len(L2)  
 if a > b:  
 a = L1  
 b = L2  
 else:  
 a = L2  
 b = L1 # a是大的，b是小的  
 L = []  
 for i in range(len(b)):  
 if b[len(b) - i - 1] == '1':  
 lll = a + ['0'] \* i  
 L = plus(L, lll)  
 ans = ''  
 for each in L:  
 ans = ans + each  
 while len(ans) <= 11:  
 ans = '0' + ans  
 return ans  
  
  
def mult(x, y, p):  
 ans = fun3(x, y)  
 if p:  
 ans = fun(ans)  
 return fun2(ans, p)  
  
  
a = int(input())  
b = int(input())  
if -2048 <= a <= 2047 and -2048 <= b <= 2047:  
 p = 0  
 if a < 0 and b < 0:  
 a = -a  
 b = -b  
 elif b < 0 < a:  
 p = 1  
 b = -b  
 elif a < 0 < b:  
 p = 1  
 a = -a  
 x = fun1(a)  
 y = fun1(b)  
 q = mult(x, y, p)  
 if q == 9999:  
 print('溢出错误')  
 else:  
 print(q)  
else:  
 print('溢出错误')





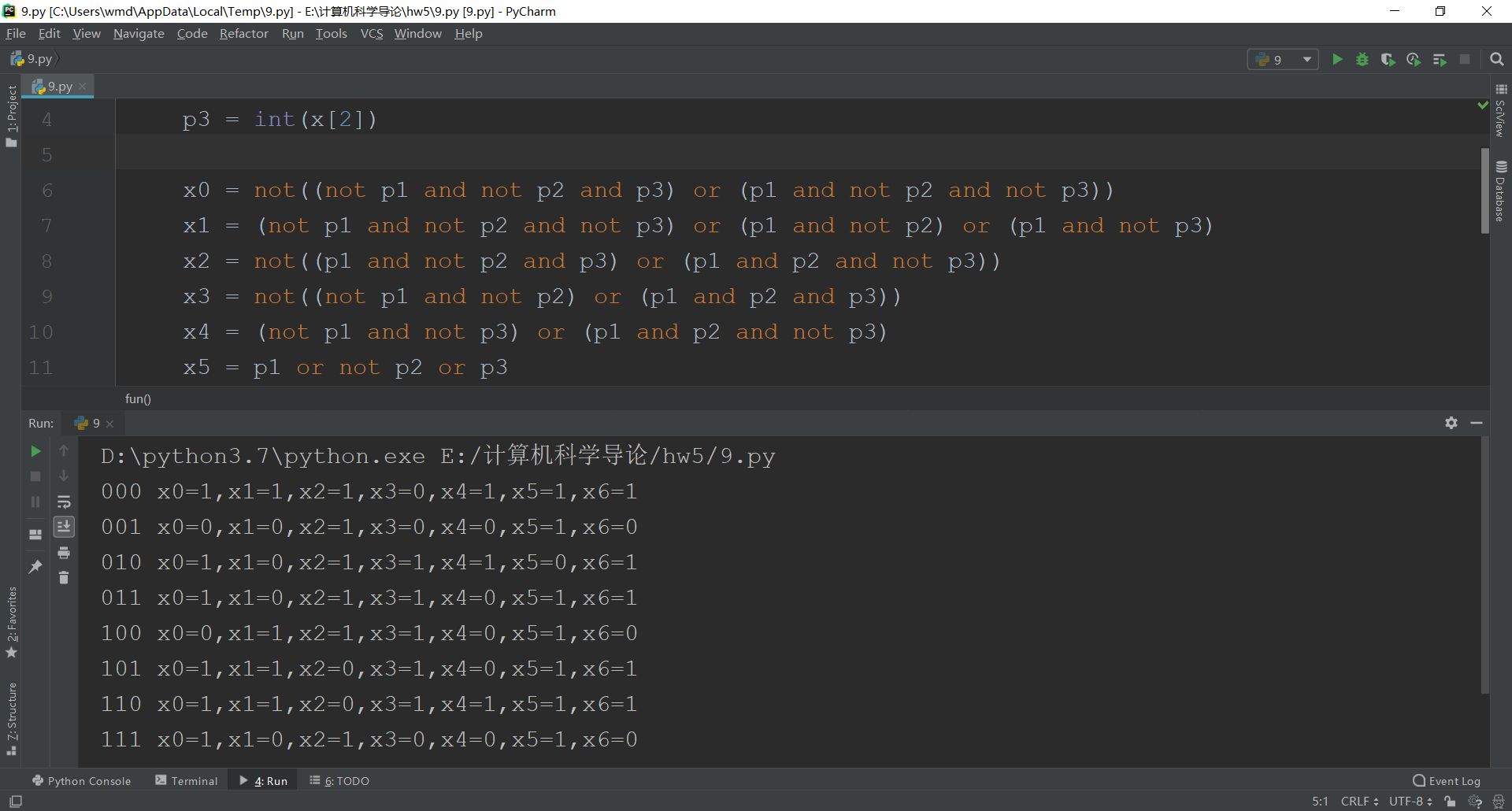
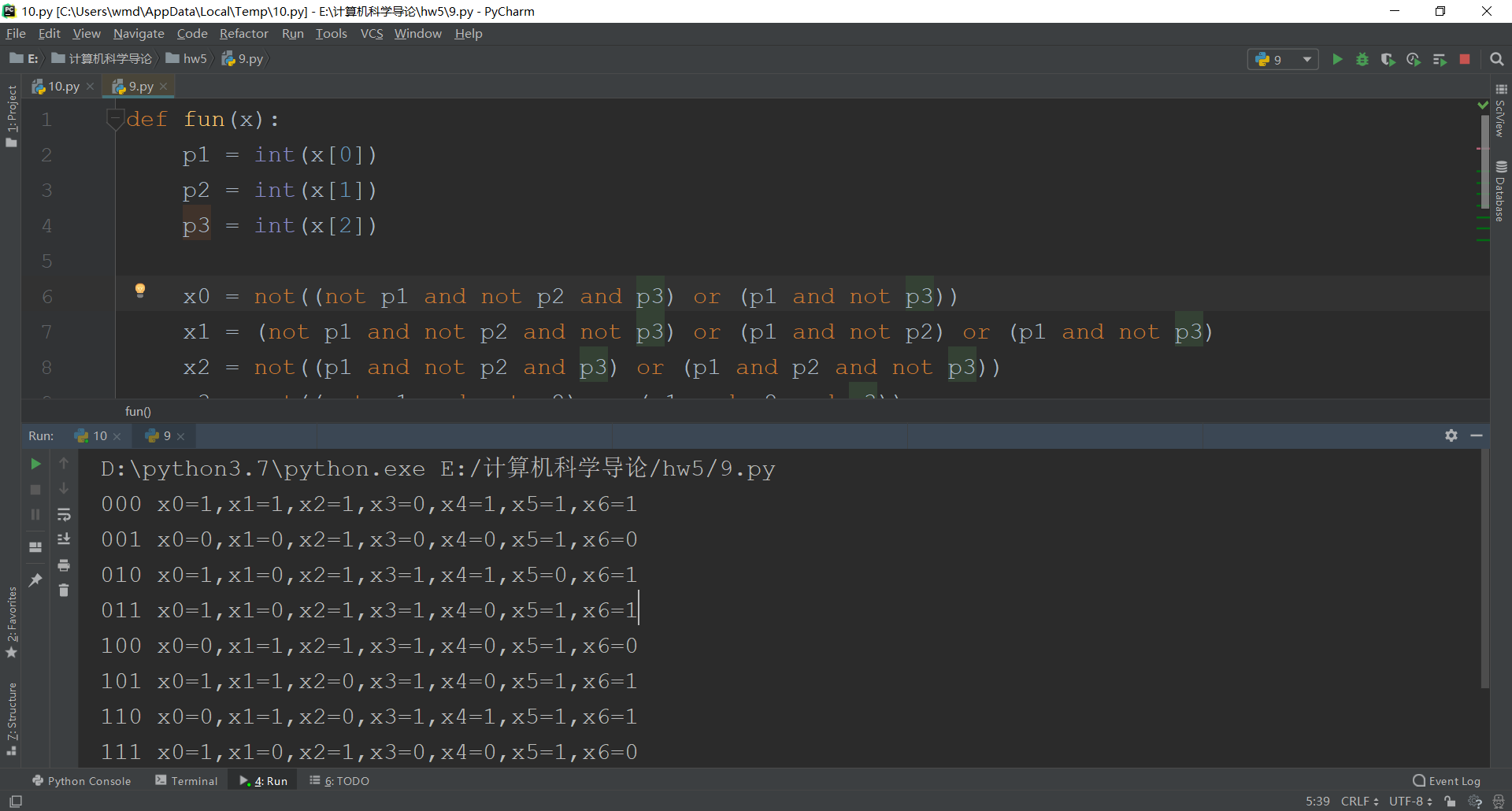






## 9.

def fun(x):  
 p1 = int(x[0])  
 p2 = int(x[1])  
 p3 = int(x[2])  
  
 x0 = not((not p1 and not p2 and p3) or (p1 and not p3))  
 x1 = (not p1 and not p2 and not p3) or (p1 and not p2) or (p1 and not p3)  
 x2 = not((p1 and not p2 and p3) or (p1 and p2 and not p3))  
 x3 = not((not p1 and not p2) or (p1 and p2 and p3))  
 x4 = (not p1 and not p3) or (p1 and p2 and not p3)  
 x5 = p1 or not p2 or p3  
 x6 = not((not p1 and not p2 and p3) or (p1 and not p2 and not p3) or (p1 and p2 and p3))  
 print(x, end=' ')  
 print('x0=%d,x1=%d,x2=%d,x3=%d,x4=%d,x5=%d,x6=%d' % (x0, x1, x2, x3, x4, x5, x6))  
  
  
L = ('000', '001', '010', '011', '100', '101', '110', '111')  
for i in L:  
 fun(i)

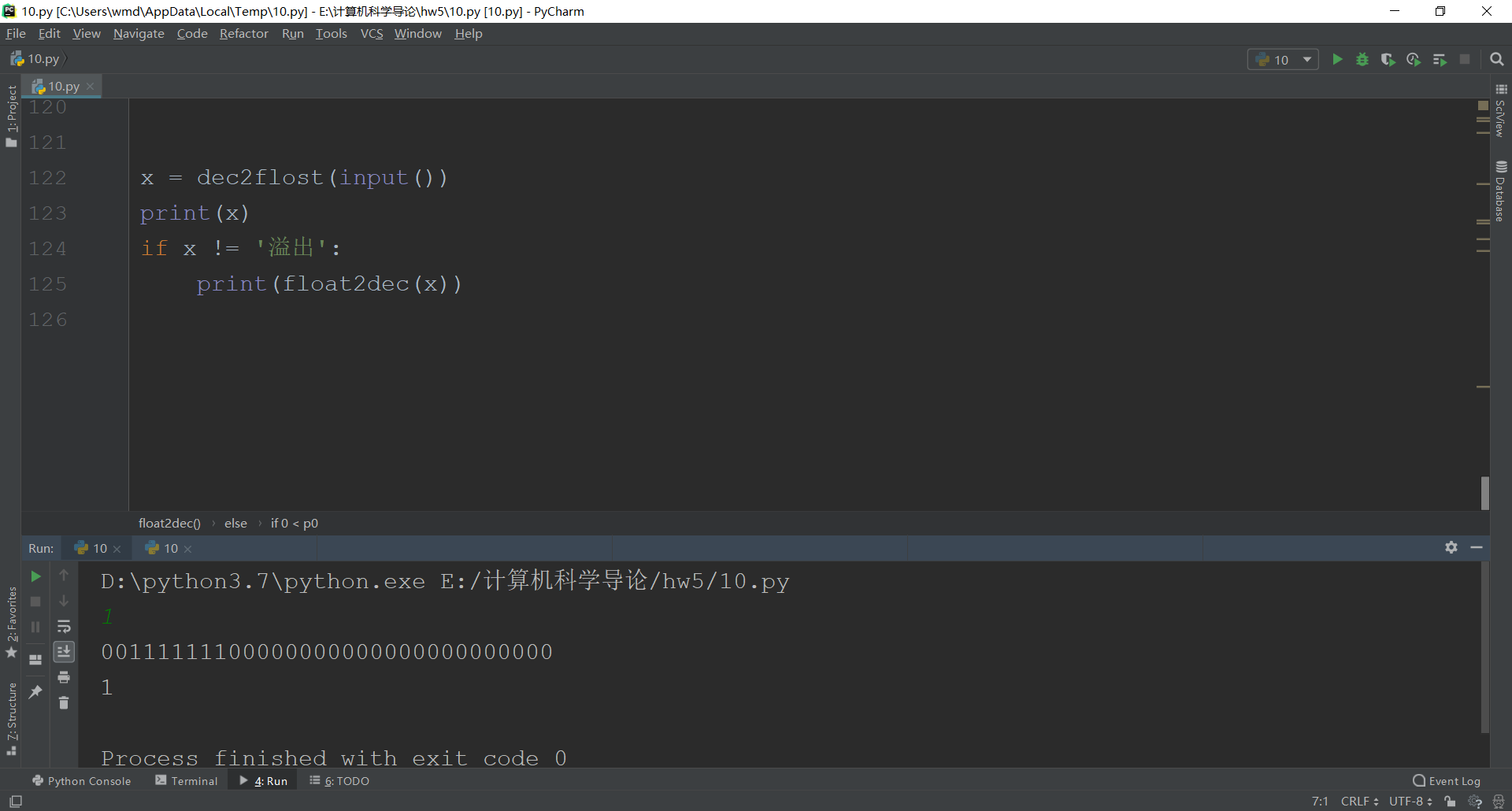


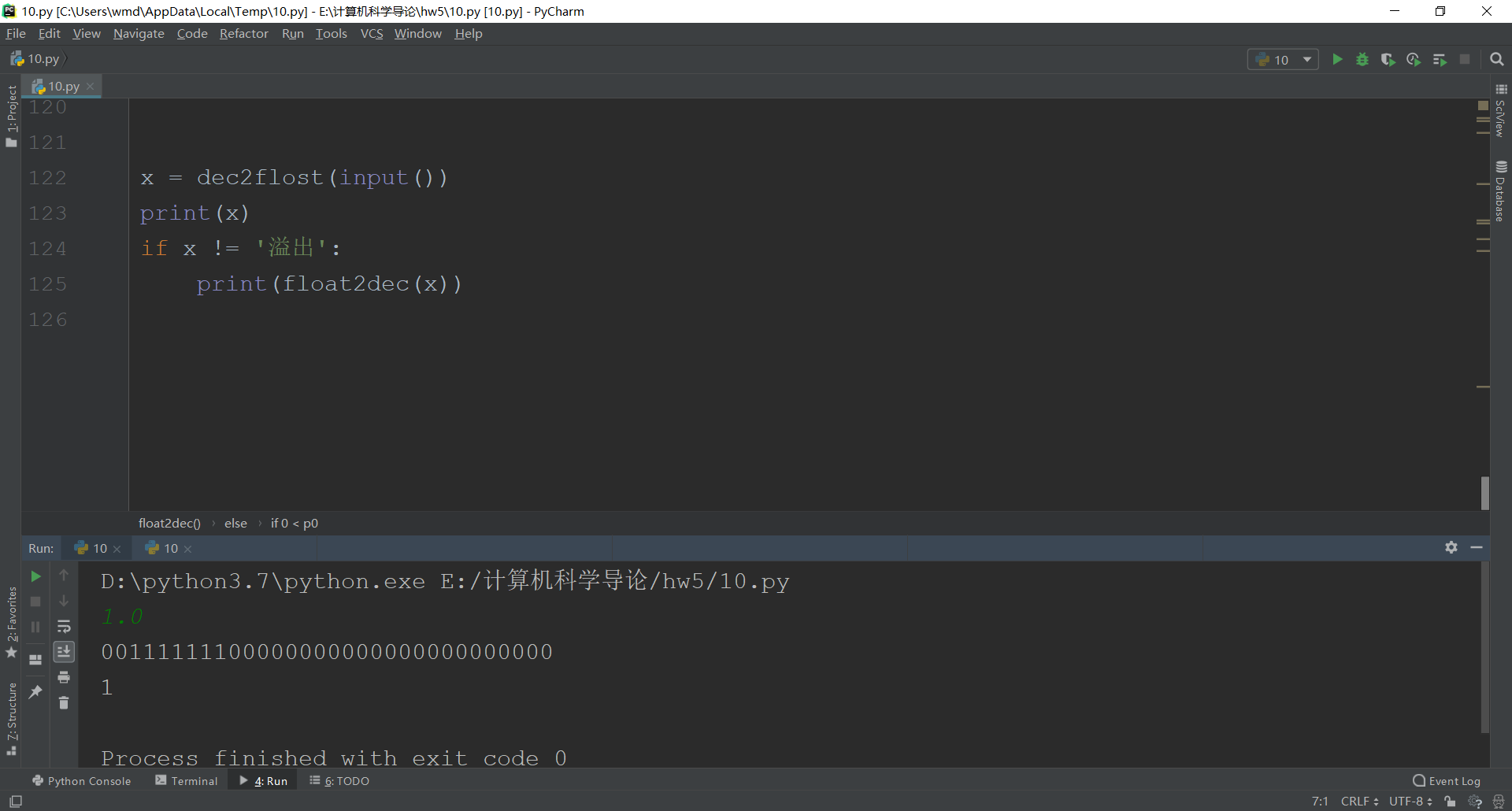
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | X0 | X1 | X2 | X3 | X4 | X5 | X6 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

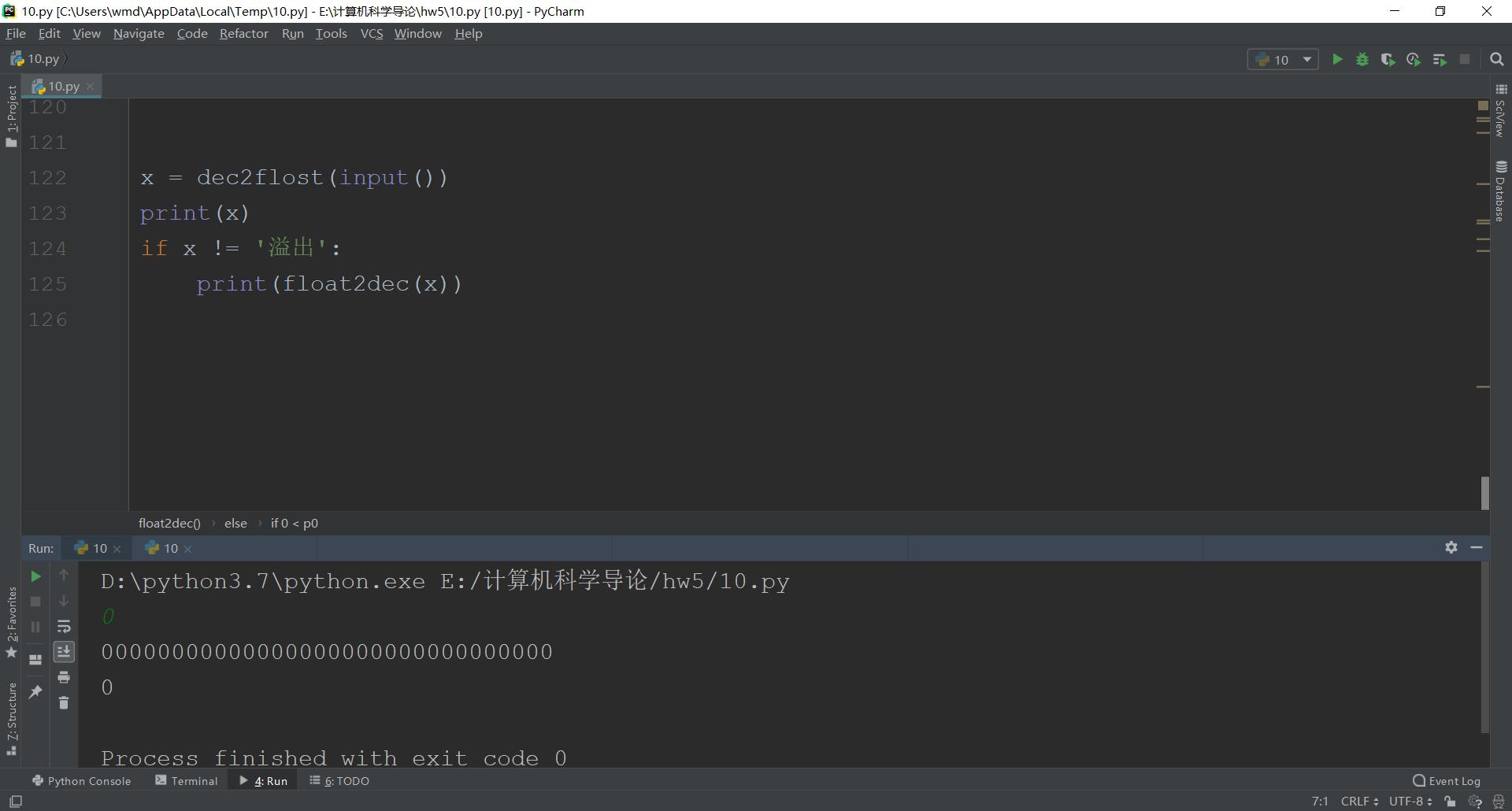
x0 = not((not a and not b and c) or (a and not c))  
x1 = (not a and not b and not c) or (a and not b) or (a and not c)  
x2 = not((a and not b and c) or (a and b and not c))  
x3 = not((not a and not b) or (a and b and c))  
x4 = (not a and not c) or (a and b and not c)  
x5 = a or not b or c  
x6 = not((not a and not b and c) or (a and not b and not c) or (a and b and c))

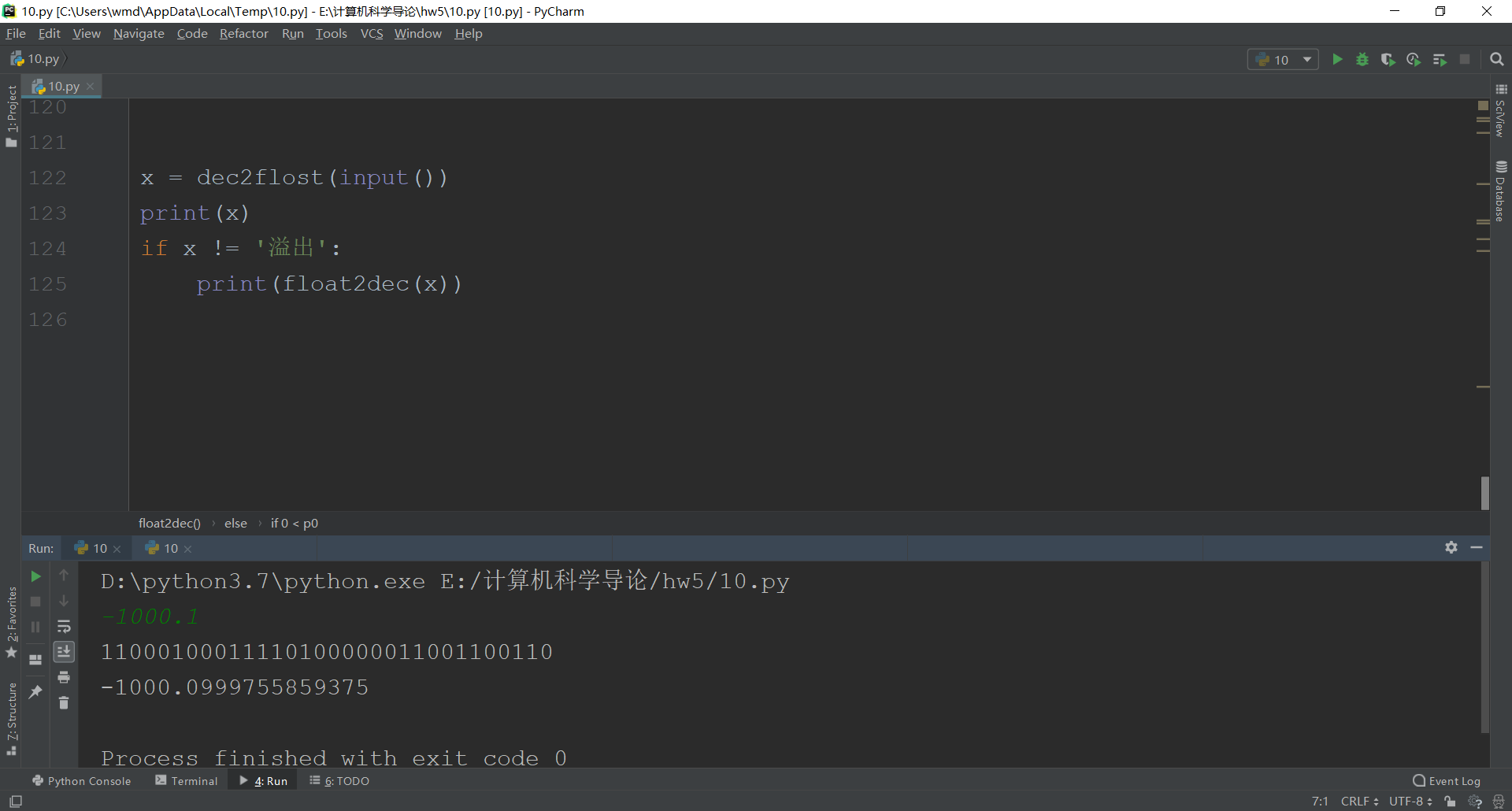
## 10.

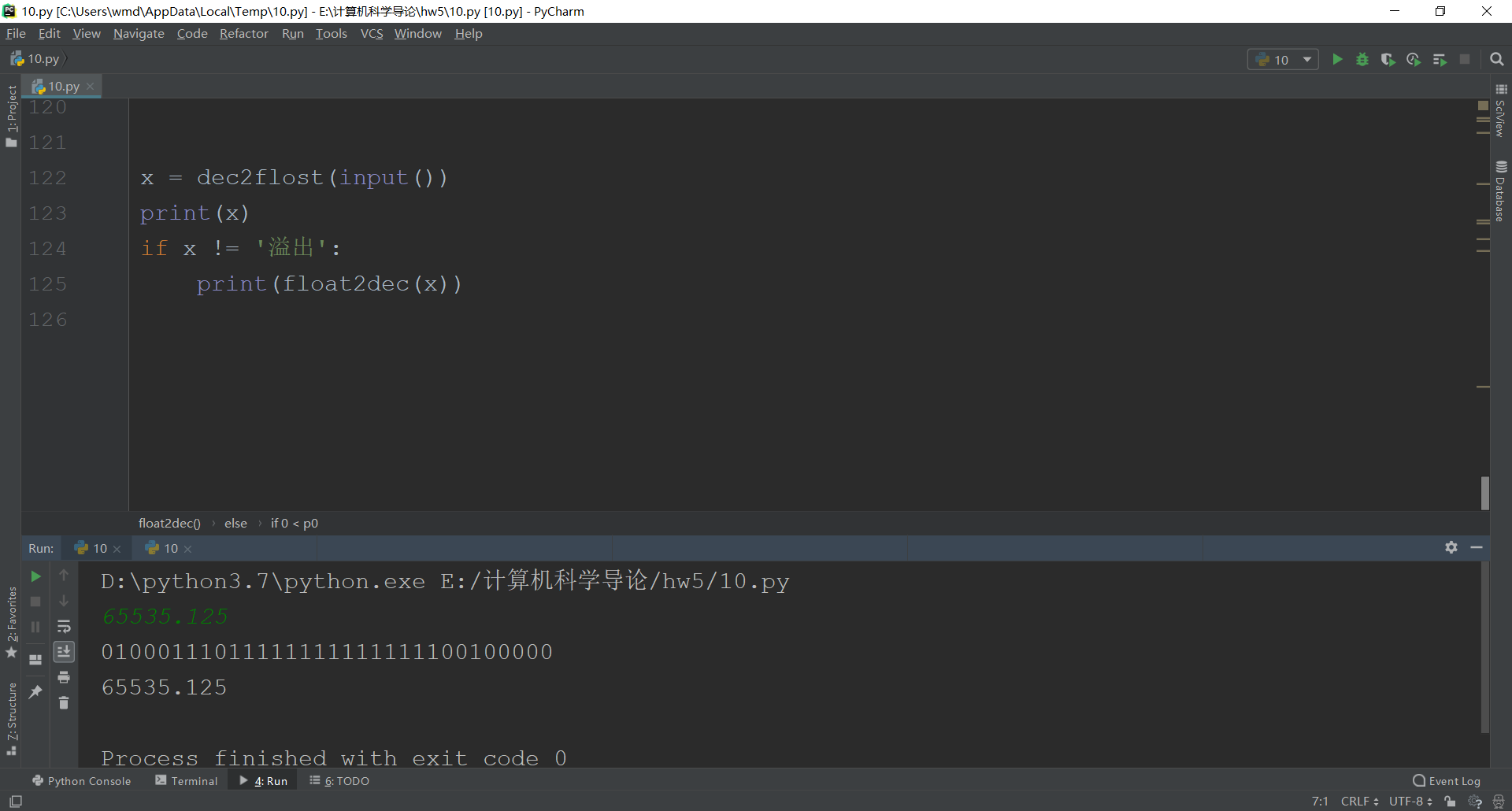
def fun1(a): # 十进制转二进制  
 x = int(a)  
 Rs = []  
 while x:  
 r = x % 2  
 x //= 2  
 Rs = [r] + Rs  
 return Rs  
  
  
def bin2dec(b): # 二进制小数部分转十进制  
 d = 0  
 k = 0.5  
 for i in range(len(b)):  
 if b[i] == '1':  
 d += k  
 k /= 2  
 return d  
  
  
def fun2(a):  
 a = float('0.' + a)  
 k = []  
 x = 0.5  
 while a != 0:  
 if a >= x:  
 k = k + [1]  
 a = a - x  
 else:  
 k = k + [0]  
 x /= 2  
 return k  
  
  
def fun3(b):  
 x, y = b.split('.')  
 x = fun1(x)  
 y = fun2(y)  
 return x, y  
  
  
def dec2flost(x):  
 ans = [0]\*32  
 if float(x) < 0:  
 ans[0] = 1  
 x = x[1:]  
 if float(x) == 0:  
 return '0'\*32  
 if '.' in x:  
 spp = x  
 else:  
 spp = str(float(x))  
 q0, q1 = fun3(spp)  
 if q0:  
 q2 = len(q0) - 1  
 if q2 > 127:  
 return '溢出'  
 else:  
 p1 = bin(q2+127)[2:][::-1]  
 for i in range(len(p1)):  
 ans[8 - i] = int(p1[i])  
 q3 = q0[1:][::-1]  
 for each in q3:  
 q1 = [each] + q1  
 else:  
 k = 1  
 while q1[0] == 0:  
 k += 1  
 q1 = q1[1:]  
 q1 = q1[1:]  
 if k > 126:  
 return '0'\*32  
 jkl = fun1(-k+127)  
 while len(jkl) < 8:  
 jkl = [0] + jkl  
 t = jkl[::-1]  
 for i in range(len(t)):  
 ans[8 - i] = int(t[i])  
 if len(q1) < 23:  
 omg = len(q1)  
 else:  
 omg = 23  
 for i in range(omg):  
 ans[9+i] = q1[i]  
 fnc = ''  
 for i in ans:  
 fnc = fnc + str(i)  
 return fnc  
  
  
def float2dec(x):  
 if x == '0'\*32:  
 return 0  
 else:  
 k = 1  
 if x[0] == '1':  
 k = -1  
 p0 = x[1:9]  
 p0 = int(p0, 2) - 127 # 指数  
 p1 = x[9:]  
 p2 = '1'  
 if 0 < p0:  
 while p0 != 0 and p1 != '':  
 p2 = p2 + p1[0]  
 p1 = p1[1:]  
 p0 -= 1  
 p1 = bin2dec(p1)  
 t = (int(p2, 2) + p1)\*2\*\*p0  
 elif p0 < 0:  
 while p0 != -1:  
 p2 = '0' + p2  
 p0 += 1  
 p2 = p2 + p1  
 p2 = bin2dec(p2)  
 t = p2  
 else:  
 p3 = bin2dec(p1)  
 t = 1 + p3  
 return t \* k  
  
  
x = dec2flost(input())  
print(x)  
if x != '溢出':  
 print(float2dec(x))

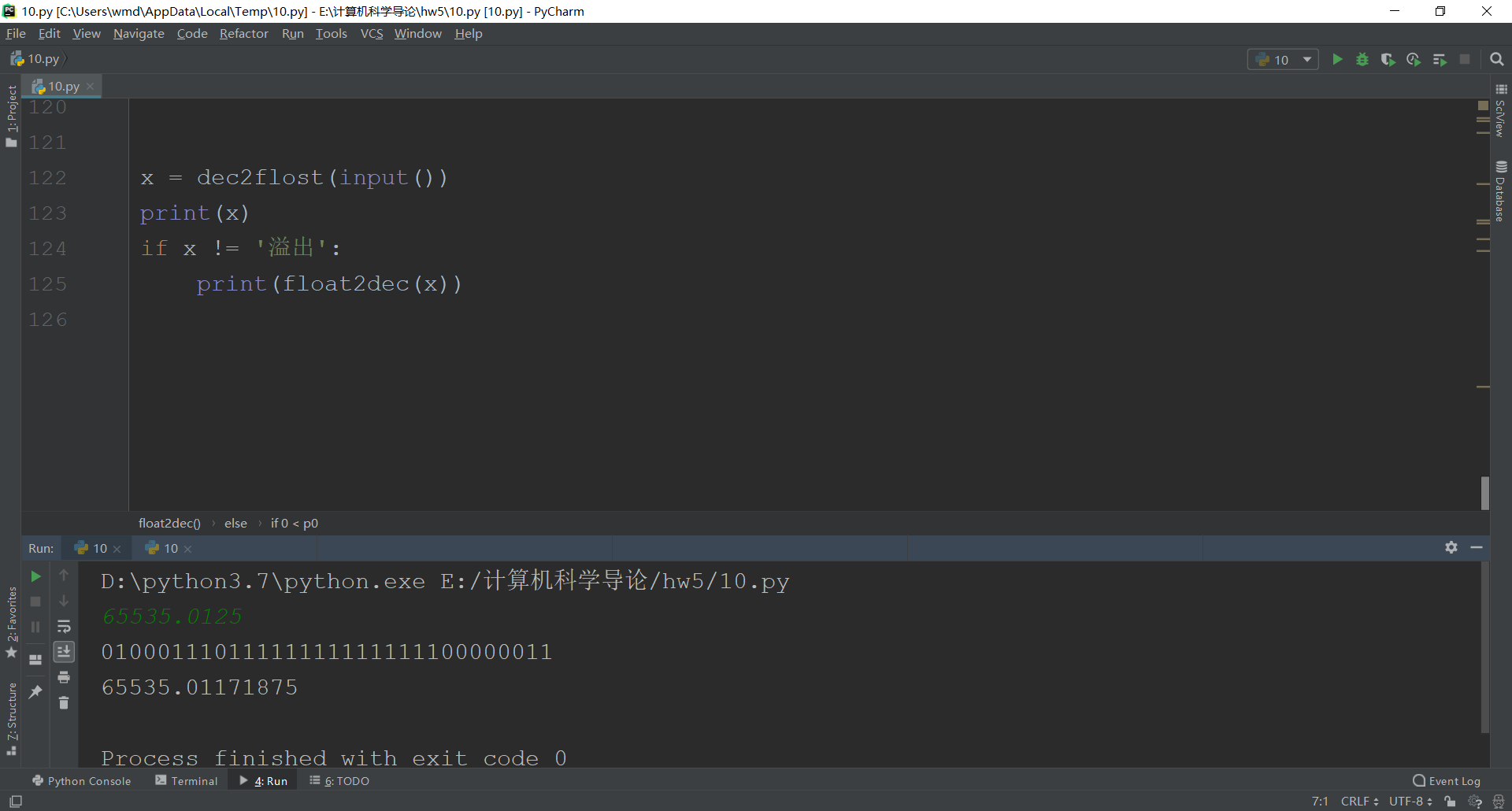


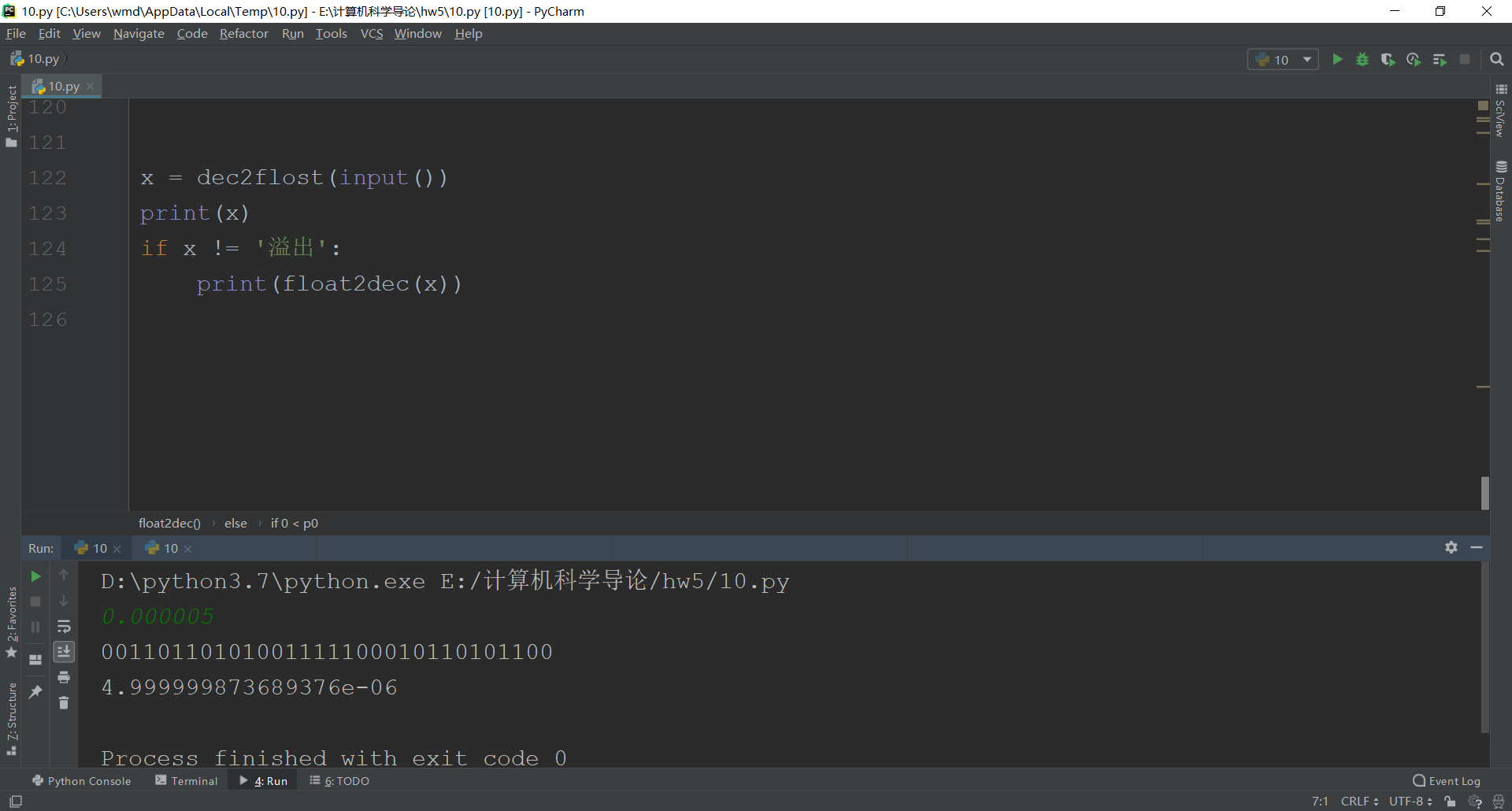


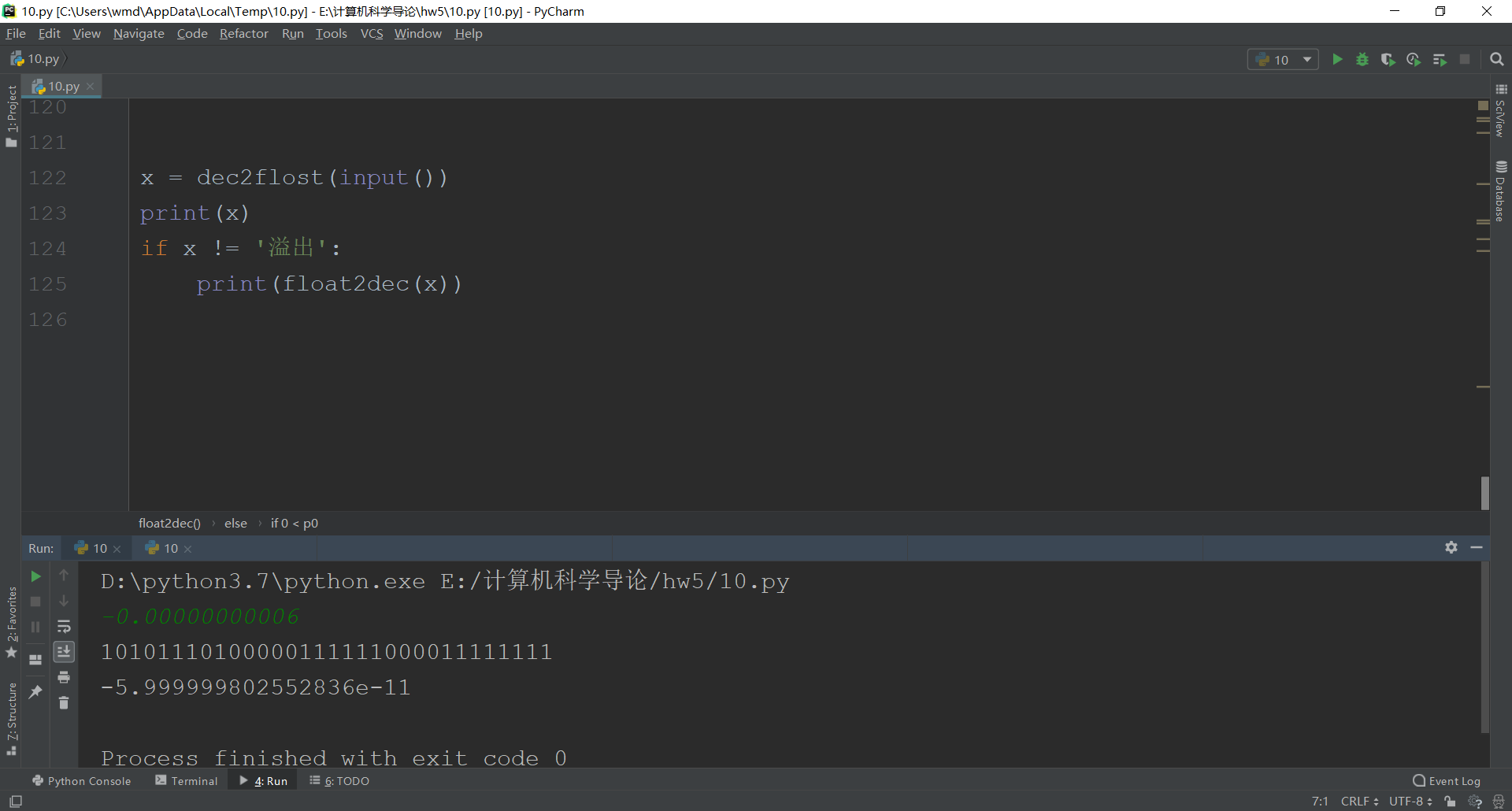




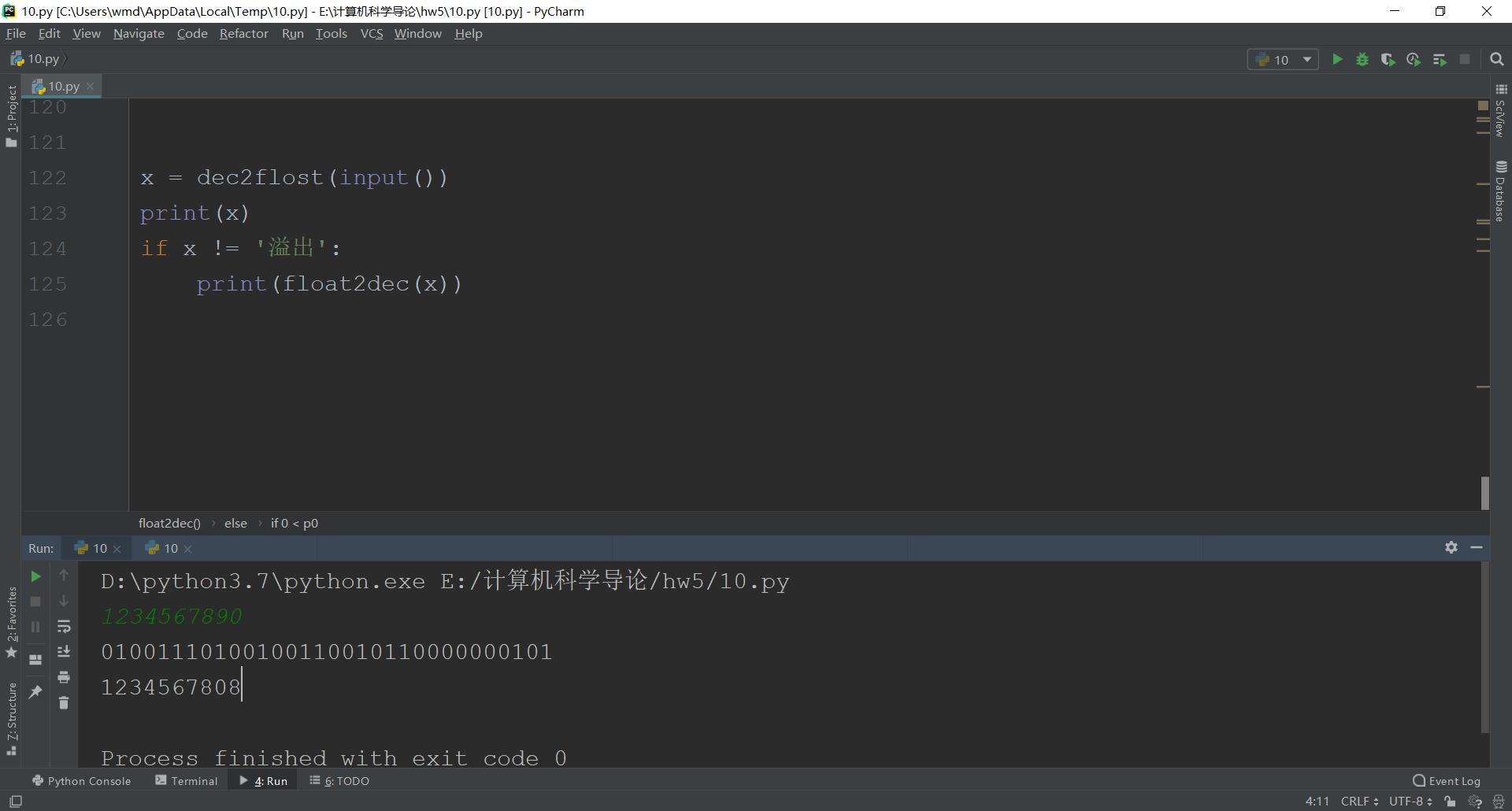


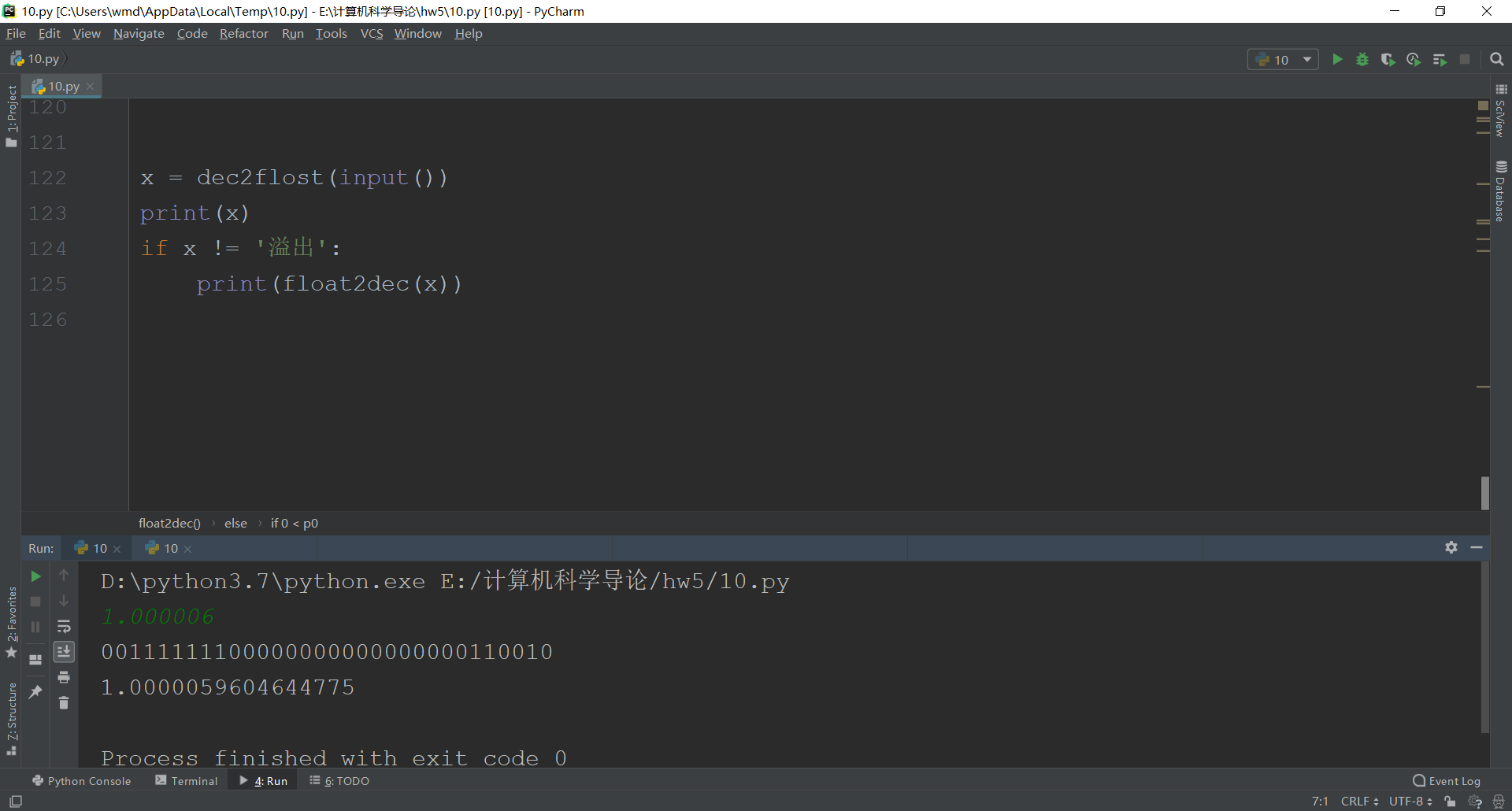


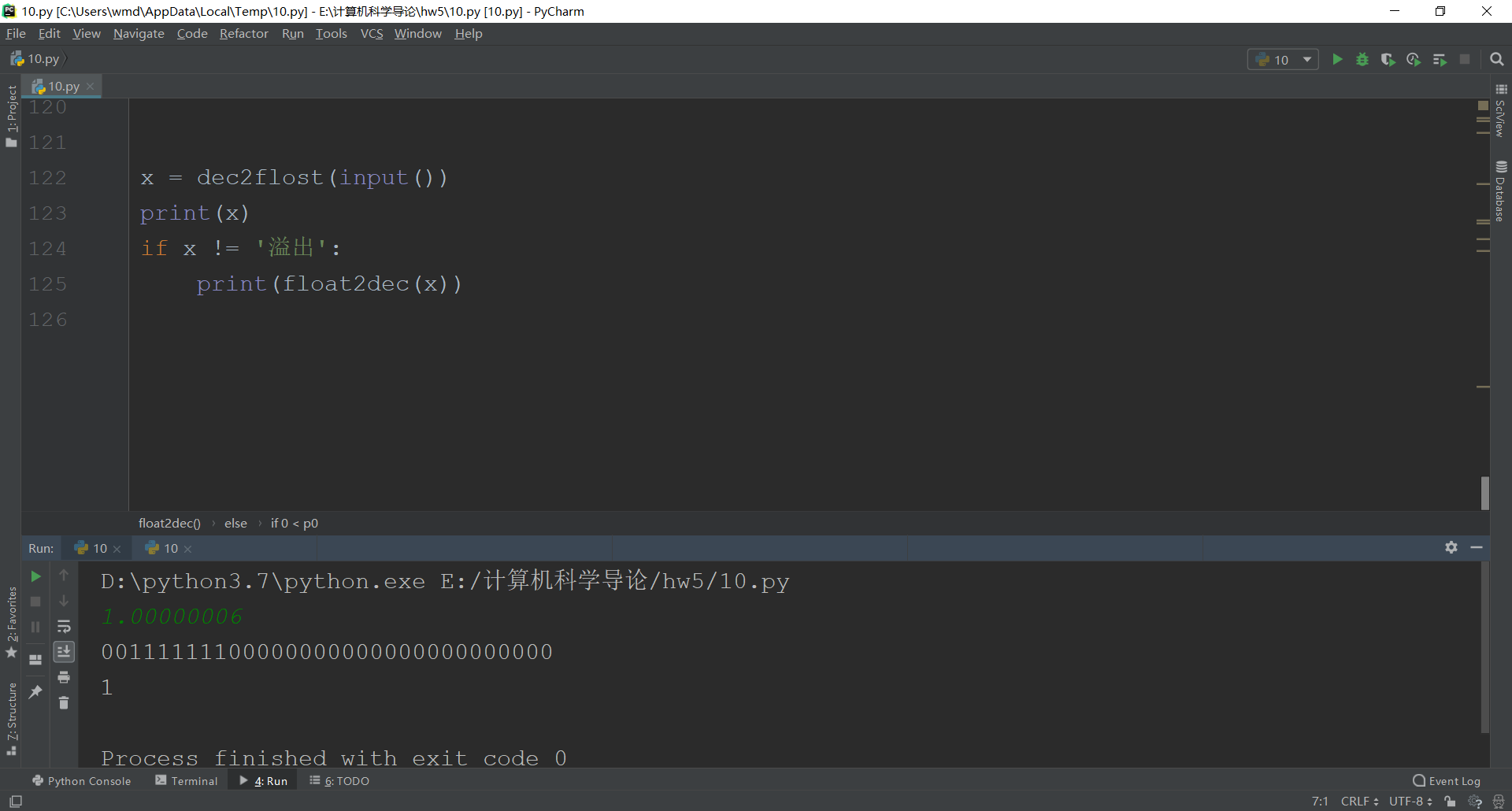












b) 64位

def fun1(a): # 十进制转二进制  
 x = int(a)  
 Rs = []  
 while x:  
 r = x % 2  
 x //= 2  
 Rs = [r] + Rs  
 return Rs  
  
  
def bin2dec(b): # 二进制小数部分转十进制  
 d = 0  
 k = 0.5  
 for i in range(len(b)):  
 if b[i] == '1':  
 d += k  
 k /= 2  
 return d  
  
  
def fun2(a):  
 a = float('0.' + a)  
 k = []  
 x = 0.5  
 while a != 0:  
 if a >= x:  
 k = k + [1]  
 a = a - x  
 else:  
 k = k + [0]  
 x /= 2  
 return k  
  
  
def fun3(b):  
 x, y = b.split('.')  
 x = fun1(x)  
 y = fun2(y)  
 return x, y  
  
  
def dec2double(x):  
 ans = [0]\*64  
 if float(x) < 0:  
 ans[0] = 1  
 x = x[1:]  
 if float(x) == 0:  
 return '0'\*64  
 if '.' in x:  
 spp = x  
 else:  
 spp = str(float(x))  
 q0, q1 = fun3(spp)  
 if q0:  
 q2 = len(q0) - 1  
 if q2 > 1023:  
 return '溢出'  
 else:  
 p1 = bin(q2+1023)[2:][::-1]  
 for i in range(len(p1)):  
 ans[11 - i] = int(p1[i])  
 q3 = q0[1:][::-1]  
 for each in q3:  
 q1 = [each] + q1  
 else:  
 k = 1  
 while q1[0] == 0:  
 k += 1  
 q1 = q1[1:]  
 q1 = q1[1:]  
 if k > 1022:  
 return '0'\*64  
 jkl = fun1(-k+1023)  
 while len(jkl) < 11:  
 jkl = [0] + jkl  
 t = jkl[::-1]  
 for i in range(len(t)):  
 ans[11 - i] = int(t[i])  
 if len(q1) < 52:  
 omg = len(q1)  
 else:  
 omg = 52  
 for i in range(omg):  
 ans[12+i] = q1[i]  
 fnc = ''  
 for i in ans:  
 fnc = fnc + str(i)  
 return fnc  
  
  
def double2dec(x):  
 if x == '0'\*64:  
 return 0  
 else:  
 k = 1  
 if x[0] == '1':  
 k = -1  
 p0 = x[1:12]  
 p0 = int(p0, 2) - 1023 # 指数  
 p1 = x[12:]  
 p2 = '1'  
 if 0 < p0:  
 while p0 != 0 and p1 != '':  
 p2 = p2 + p1[0]  
 p1 = p1[1:]  
 p0 -= 1  
 p1 = bin2dec(p1)  
 t = (int(p2, 2) + p1)\*2\*\*p0  
 elif p0 < 0:  
 while p0 != -1:  
 p2 = '0' + p2  
 p0 += 1  
 p2 = p2 + p1  
 p2 = bin2dec(p2)  
 t = p2  
 else:  
 p3 = bin2dec(p1)  
 t = 1 + p3  
 return t \* k  
  
  
x = dec2double(input())  
print(x)  
if x != '溢出':  
 print(double2dec(x))

