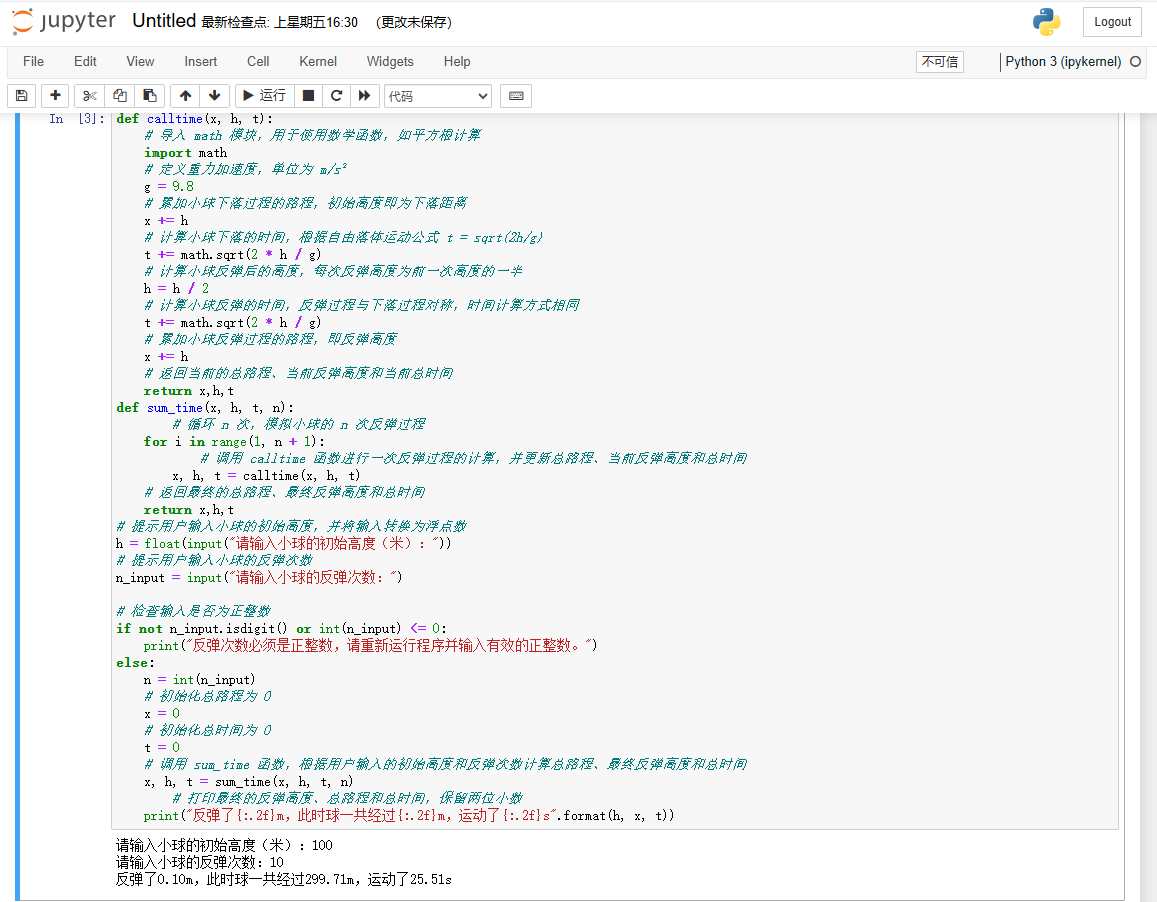
第10次掉下并反弹到最高，反弹了0.10m，此时球一共经过299.71m，运动了25.51s

截图：



反弹高度是下落高度的一半，第n次掉下（n为正整数）并反弹到最高，反弹了米，则第n次下落上弹过程运动了米，则一共运动了米，反弹高度是下落高度的一半，则下落所用的时间是反弹所用的时间的倍，第n次下落上弹过程所用的时间为秒。则一共运动了秒。

先求出一个通用的每一次下落上弹求高度，路程，时间的函数，然后定义另一个函数，调用这个函数根据循环去求第n步的高度，路程，时间，然后把路程、时间叠加起来，即可。如下：

def calltime(x, h, t):

# 导入 math 模块，用于使用数学函数，如平方根计算

import math

# 定义重力加速度，单位为 m/s²

g = 9.8

# 累加小球下落过程的路程，初始高度即为下落距离

x += h

# 计算小球下落的时间，根据自由落体运动公式 t = sqrt(2h/g)

t += math.sqrt(2 \* h / g)

# 计算小球反弹后的高度，每次反弹高度为前一次高度的一半

h = h / 2

# 计算小球反弹的时间，反弹过程与下落过程对称，时间计算方式相同

t += math.sqrt(2 \* h / g)

# 累加小球反弹过程的路程，即反弹高度

x += h

# 返回当前的总路程、当前反弹高度和当前总时间

return x,h,t

#定义总的小球下降到上升的函数

def sum\_time(x, h, t, n):

# 循环 n 次，模拟小球的 n 次反弹过程

for i in range(1, n + 1):

# 调用 calltime 函数进行一次反弹过程的计算，并更新总路程、当前反弹高度和总时间

x, h, t = calltime(x, h, t)

# 返回最终的总路程、最终反弹高度和总时间

return x,h,t

# 提示用户输入小球的初始高度，并将输入转换为浮点数

h = float(input("请输入小球的初始高度（米）："))

# 提示用户输入小球的反弹次数

n\_input = input("请输入小球的反弹次数：")

# 检查输入是否为正整数，不然直接报错。

if not n\_input.isdigit() or int(n\_input) <= 0:

print("反弹次数必须是正整数，请重新运行程序并输入有效的正整数。")

else:

n = int(n\_input)

# 初始化总路程为 0

x = 0

# 初始化总时间为 0

t = 0

# 调用 sum\_time 函数，根据用户输入的初始高度和反弹次数计算总路程、最终反弹高度和总时间

x, h, t = sum\_time(x, h, t, n)

# 打印最终的反弹高度、总路程和总时间，保留两位小数

print("反弹了{:.2f}m，此时球一共经过{:.2f}m，运动了{:.2f}s".format(h, x, t))

github地址：https://github.com/Yiyun1258/Data-driven-research-in-matter-science-hw2

截图：