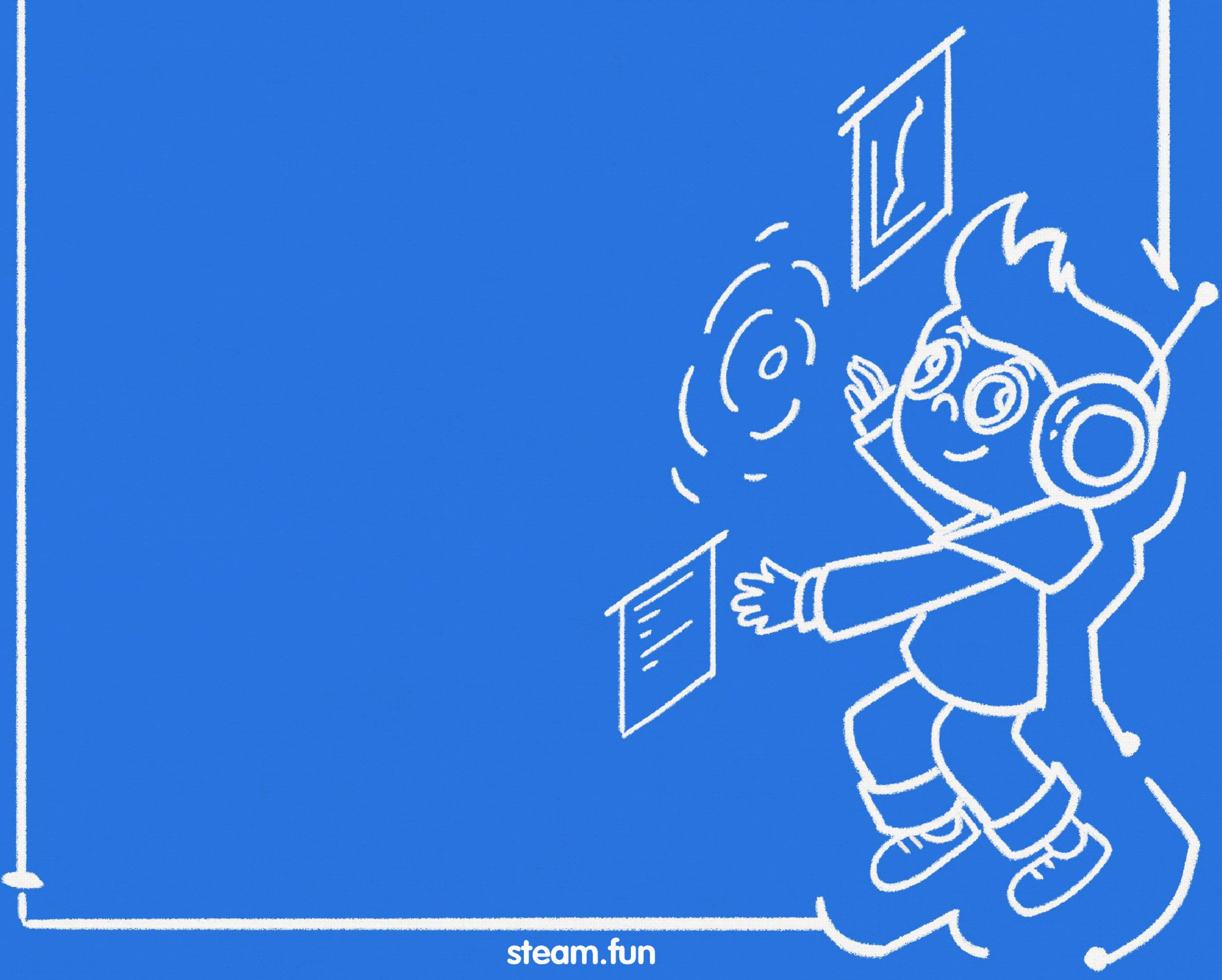
谁的恶作剧



1.探索新知

1.1 单个弹窗

1. 导入tkinter库

```
import tkinter as tk
```

2. 创建主窗口对象，并起名为root

```
root = tk.Tk()
```

3. 配置窗口标题

```
root.title("恶作剧")
```

4. 设置窗口大小

```
root.geometry('400x100')
```

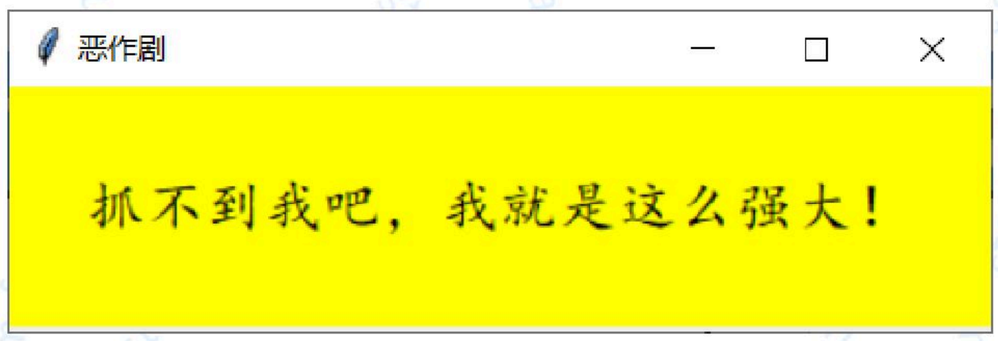
5. 设置嚣张的标签

```
tk.Label(root,
         text='抓不到我吧，我就是这么强大！',  # 显示文字
         bg='yellow',  # 背景颜色
         font=('楷体', 17),  # 字体和字体大小
         width=40, height=4  # 标签长宽
         ).pack()  # 配置标签位置
```

6. 让窗口活起来

```
root.mainloop()
```

效果如下：



任务一新增代码如下：

```
import tkinter as tk     # 导入tkinter库

root = tk.Tk()     # 创建窗口对象
root.title('恶作剧')    # 定义窗口的标题
root.geometry("400x100")  # 设置窗口的尺寸、位置
tk.Label(root,
         text='抓不到我吧，我就是这么强大！',  # 显示文字
         bg='yellow',  # 背景颜色
         font=('楷体', 17),  # 字体和字体大小
         width=40, height=4  # 标签长宽
         ).pack()  # 配置标签位置
root.mainloop()
```

1.2 弹窗轰炸

想要实现弹窗轰炸的效果，需要在短时间内接连出现大量的弹窗，这里我们以100个弹窗为例

为了方便使用，我们可以把单个弹窗的代码封装到一个函数中

```
def tanchuang():
    root = tk.Tk()
    root.title('恶作剧')
    root.geometry("400x100")
    tk.Label(root,
             text='抓不到我吧，我就是这么强大！',
             bg='yellow',
             font=('楷体', 17),
             width=40, height=4
             ).pack()
root.mainloop()
```

如何生成100个弹窗呢？

编程试试效果吧！

```
for i in range(100):
    tanchuang()
```

1.2 弹窗轰炸

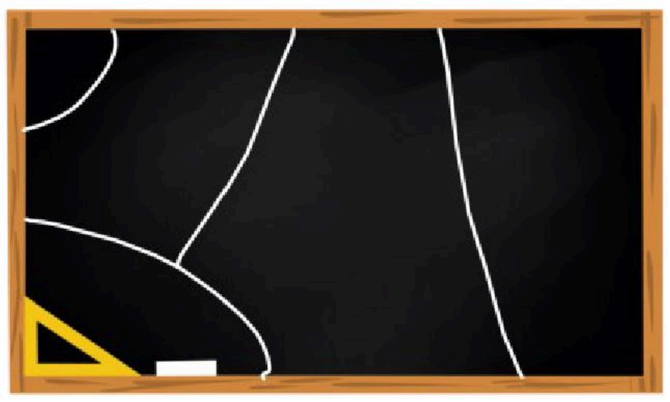
效果不太对呀！怎么是关掉一个弹窗才出现下一个呢？
如何实现铺天盖地弹窗出现的震撼效果呢？

这里要用到一个新的标准库 —— **threading库**

threading库主要用于实现多线程编程。多线程编程是并发编程的一种形式，它允许程序同时执行多个任务或操作

简单给大家讲讲什么是“多线程”？

想象一下，你在上美术课，老师让大家画一张大海报。海报被分成很多小块，每块都要画上不同图案



单线程：如果只有你一个人画这张海报，你要按顺序画完一块，再画下一块。这就是单线程工作，你要完全按照一个接一个的步骤来完成整个任务

多线程：老师让一个小组负责画海报，每个同学都负责海报的一小块，那么所有同学可以同时开始画自己的那一部分。每个同学就像是一个线程，可以同时工作

1.2 弹窗轰炸

回到我们的任务。我们首先需要导入threading库

1．导入threading库

```
# 导入threading库，用于创建和管理线程
import threading
```

2．创建线程对象t，将tanchuang函数作为目标函数

【thread】：线

```
t = threading.Thread(target=tanchuang)
```

【target】：目标

详细解析：

1. threading.Thread() 用于表示一个线程的执行；
2. target=tanchuang 指定了线程开始执行时调用的函数；
3. t = 这部分代码将新创建的Thread对象赋值给变量t，通过变量t来引用和控制这个线程

这段代码想要正常工作，必须确保以下几点：

1. 确保已经导入了 threading 模块；
2. 确保 tanchuang 函数已经被定义；
3. 创建好Thread对象后，通常要调用它的start()方法来启动线程

3. 启动线程

```
t.start()
```

4. 100条线程，挨个启动

```
for i in range(100):
    t = threading.Thread(target=tanchuang)
    t.start()
```

如果想要让弹窗以相同时间间隔弹出，该如何实现呢？

这里想要实现延迟效果，需要用到前面学过的 time 模块，以便能够使用 sleep() 函数

【sleep】：
睡眠

```
time.sleep(0.1)
```

别忘了导入
time 模块

举例，帮助理解 time.sleep(0.1)

```
import time
for i in range(10):
    print(i)
    # 打印后暂停0.1秒，再打印下一个
    time.sleep(0.1)
```

1.2 弹窗轰炸

time.sleep(0.1) 这行代码，该写在哪里呢？

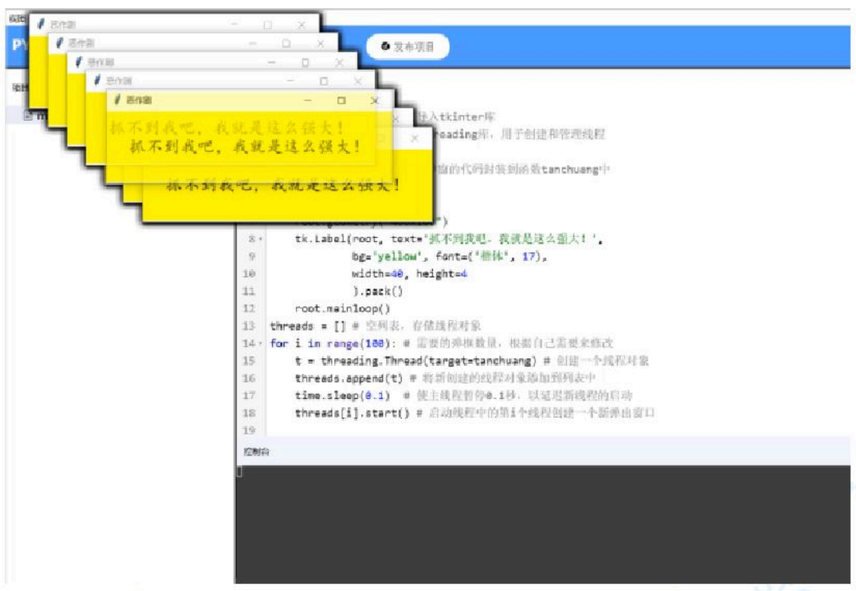
```
for i in range(100):
    t = threading.Thread(target=tanchuang)
    time.sleep(0.1)    # 使主线程暂停0.1秒，以延迟新线程的启动
    t.start()
```

任务二新增代码如下：

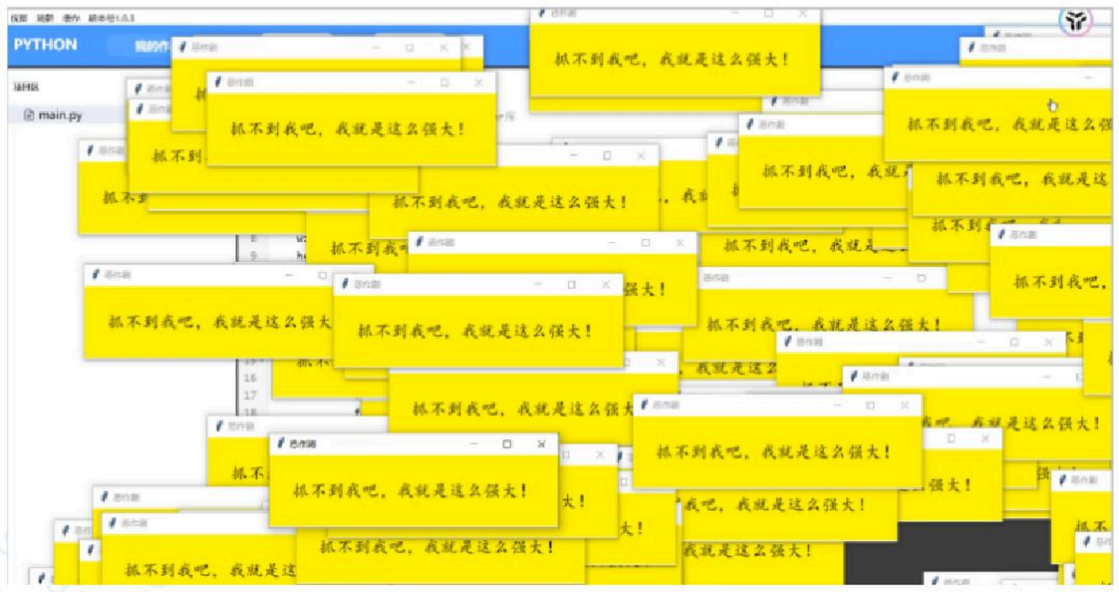
```
import threading     # 导入threading库，用于创建和管理线程
import time      # 导入时间库
def tanchuang():    # 将单个弹窗的代码封装到函数tanchuang中
    root = tk.Tk()
    root.title('恶作剧')
    root.geometry("400x100")
    tk.Label(root, text='抓不到我吧，我就是这么强大！',
             bg='yellow', font=('楷体', 17),
             width=40, height=4
             ).pack()
    root.mainloop()
for i in range(100):  # 需要的弹框数量，根据自己需要来修改
    t = threading.Thread(target=tanchuang) # 创建一个线程对象
    time.sleep(0.1)      # 使主线程暂停0.1秒，以延迟新线程的启动
    t.start()  # 启动线程，创建一个新弹出窗口
```

1.3 满屏弹窗

任务二效果演示



完整效果演示



对比可以发现：

我们的弹窗还需要每次出现在随机位置，才能实现满屏弹窗的效果

1. 生成随机的x坐标，y坐标

① 导入random模块

```
import random
```

② 获取屏幕的宽度和高度

【screenwidth】：屏幕宽度

```
width = root.winfo_screenwidth()
height = root.winfo_screenheight()
```

【screenheight】：屏幕高度

③ 生成随机的x坐标、y坐标

x坐标：
y坐标：

```
x = random.randint(0, width)
y = random.randint(0, height)
```

1.3 满屏弹窗

2. 设置弹窗出现的位置

格式：窗口.geometry("widthxheight+x+y")

```
root.geometry(f"400x100+{x}+{y}")
```

任务三新增代码如下：

```
import random       # 导入随机库
def tanchuang():
    root = tk.Tk()
    width = root.winfo_screenwidth()   # 获取屏幕的宽度
    height = root.winfo_screenheight()  # 获取屏幕的高度
    x = random.randint(0, width)   # 随机生成的宽度
    y = random.randint(0, height)   # 随机生成的高度
    root.title('恶作剧')
    root.geometry(f"400x100+{x}+{y}")
    tk.Label(root,
             text='抓不到我吧，我就是这么强大！',
             bg='yellow',
             font=('楷体', 17), width=40, height=4
             ).pack()
    root.mainloop()
```

完整代码

```
import tkinter as tk     # 导入tkinter库
import random      # 导入随机库
import threading     # 导入threading库，用于创建和管理线程
import time     # 导入时间库
def tanchuang():
    root = tk.Tk()     # 创建窗口对象
    width = root.winfo_screenwidth()  # 获取屏幕的宽度
    height = root.winfo_screenheight()  # 获取屏幕的高度
    x = random.randint(0, width)  # 随机生成宽度
    y = random.randint(0, height)  # 随机生成高度
    root.title('恶作剧')    # 定义窗口的标题
    root.geometry(f"400x100+{x}+{y}")    # 窗口尺寸位置
    tk.Label(root, text='抓不到我吧，我就是这么强大！', # 显示文字
             bg='yellow',  # 背景颜色
             font=('楷体', 17), # 字体和字体大小
             width=40, height=4  # 标签宽高
             ).pack()  # 配置标签位置
    root.mainloop()
for i in range(100):  # 需要的弹框数量，根据自己需要来修改
    # 创建一个线程对象，将tanchuang函数作为目标函数
    t = threading.Thread(target=tanchuang)
    # 使主线程暂停0.1秒，以延迟新线程的启动。确保窗口出现有一定间隔
    time.sleep(0.1)
    # 启动线程，调用tanchuang函数并创建一个新弹出窗口
    t.start()
```

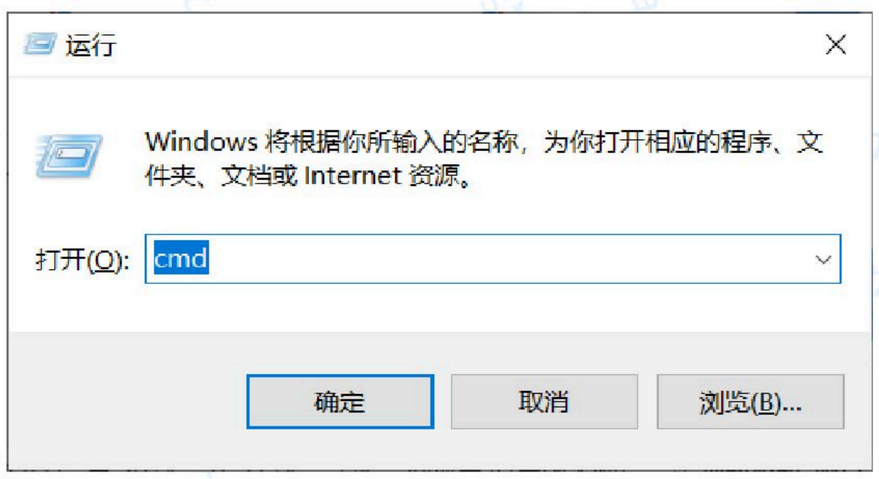
拓展任务

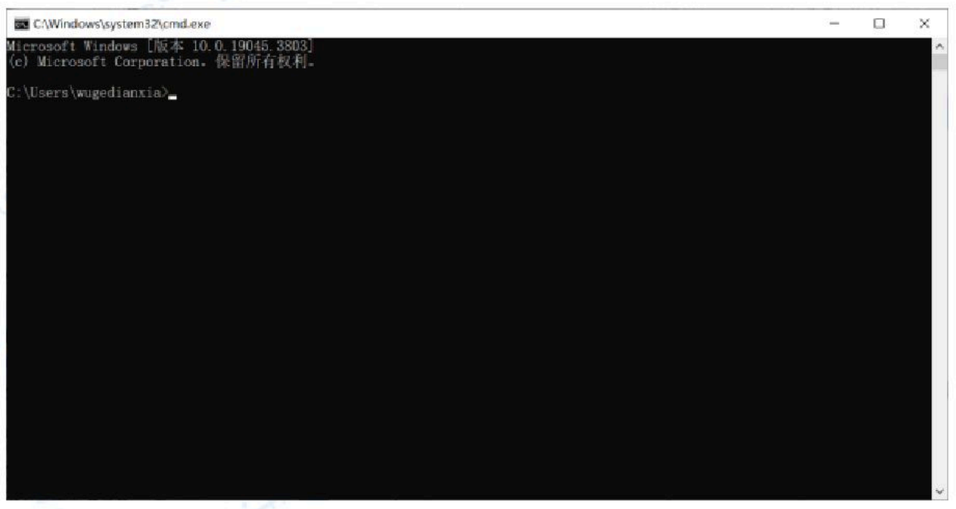
exe文件的打包

1. 打开命令提示符

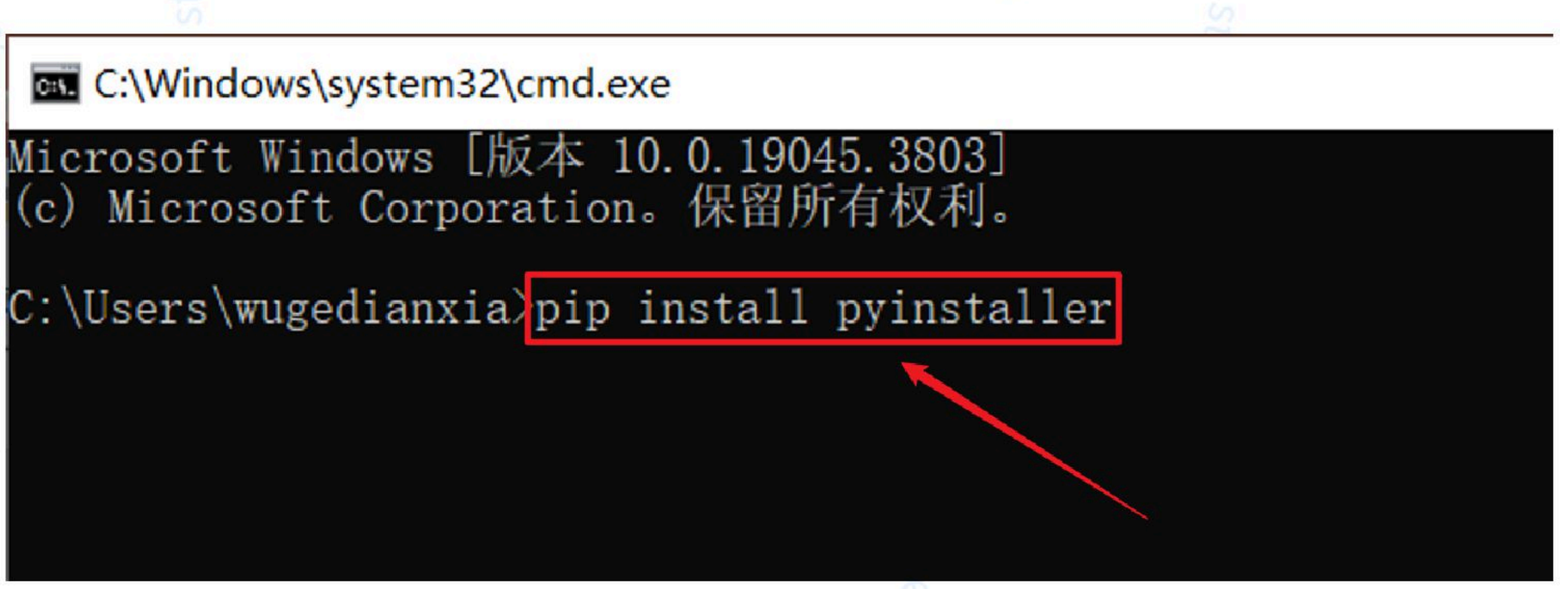
① 按下键盘上的Windows徽标键和字母R键。这将打开"运行"对话框

② 在运行对话框中，输入"cmd"（不包括引号），单击“确定”按钮





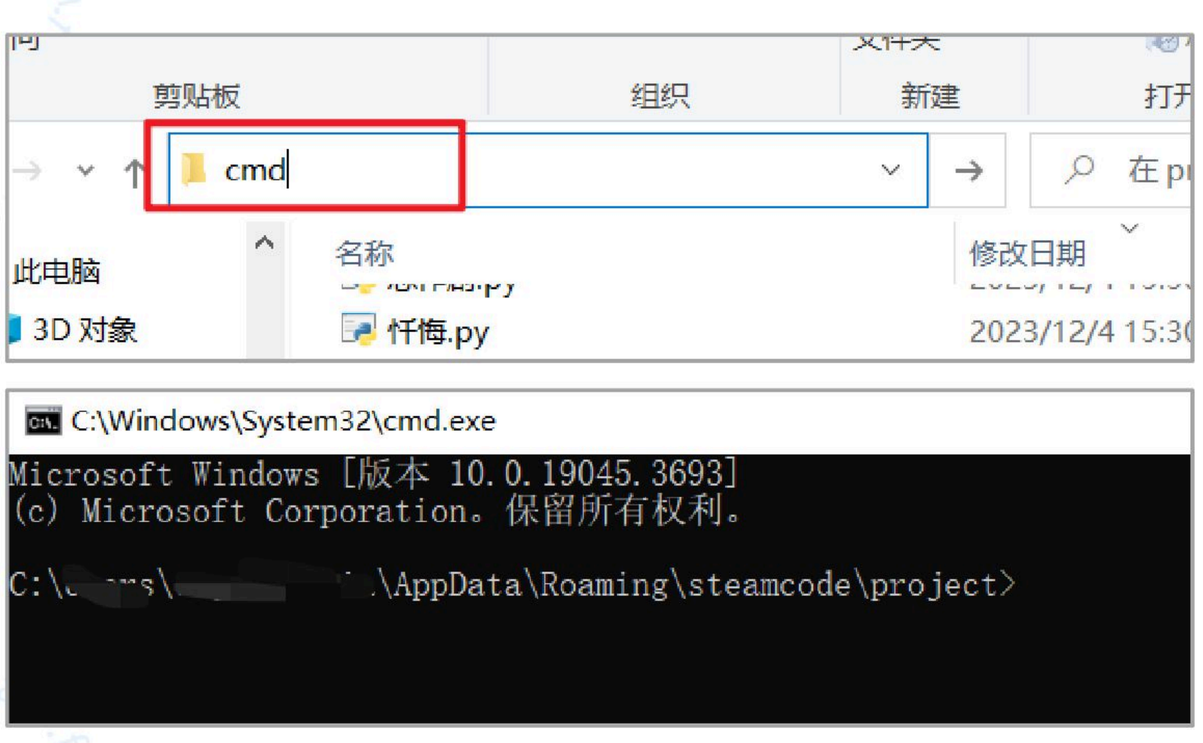
2. 如果没有安装pyinstaller，则在命令提示符输入 pip install pyinstaller



3. 如何到达，主程序文件所在位置呢？

① 点击...进入主程序所在的文件夹（针对客户端，进入文件位置）

② 进入后在红框处地址都删掉，写入 cmd ，就可以直接跳到对应目录下



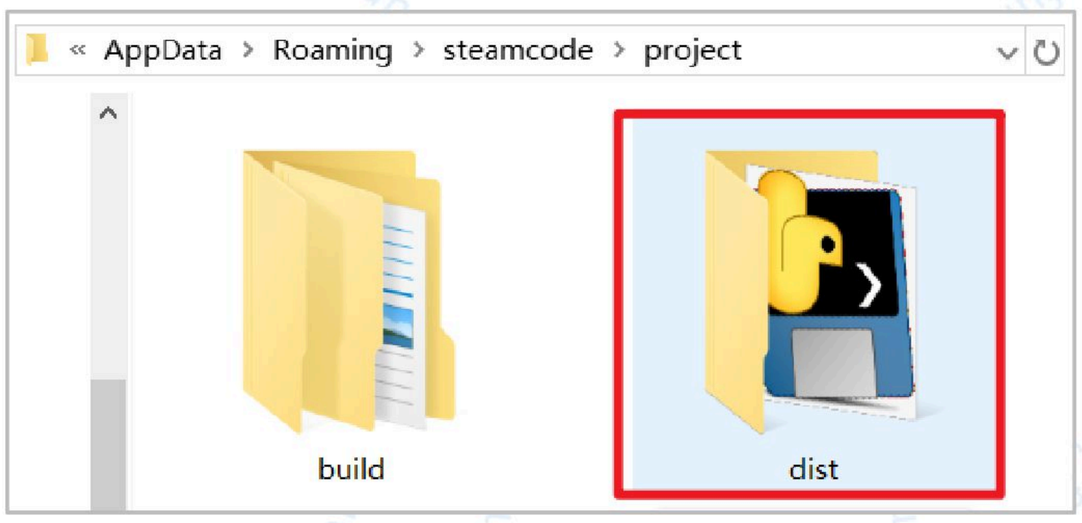
拓展任务

4. 接着，在此文件目录下，输入 `pyinstaller -F 恶作剧.py --noconsole`

这里跟自己的文件名一致即可


```
# -F参数：将程序打包成一个.exe文件
# --noconsole参数：执行.exe文件时不显示cmd命令窗
```

5. 成功后在同级目录下会出现2个文件夹，就是 build 和 dist ，其中dist 是我们要的内容，找到 dist 中的 .exe 文件就可以发送给别人了



2.强化练习

1．在Python中，以下哪个函数或方法可以将线程暂停一段时间？（　　）

A. threading.sleep()　　B. thread.sleep()

C. time.sleep()　　D. sleep()

2．在Python中，多线程编程可以使用哪个库？（　　）

A. tkinter　　B. random

C. time　　D. threading

3．以下哪个方法用于启动线程？（　　）

A. run()　　B. start()

C. begin()　　D. initiate()

4．在tkinter中，用于获取屏幕宽度的方法是？（　　）

A. winfo_screenlength()　　B. winfo_screenbreadth()

C. winfo_screenwidth()　　D. winfo_screenheight()

5．Python中，random.randint(a, b)函数的作用是？（　　）

A．生成a到b之间的随机小数

B．生成a到b之间的随机整数，包括a和b

C．生成a到b之间的随机整数，不包括a和b

D．生成一个随机的布尔值

3.术语箱

thread	线	screenwidth	屏幕宽度
target	目标	screenheight	屏幕高度
sleep	睡眠		

4.课后挑战

弹窗木马

模拟本节课作品，完成视频中的轰炸效果。标签文字、颜色可以自定义。

提示：

想要设置标签文字颜色，可以使用 fg=''