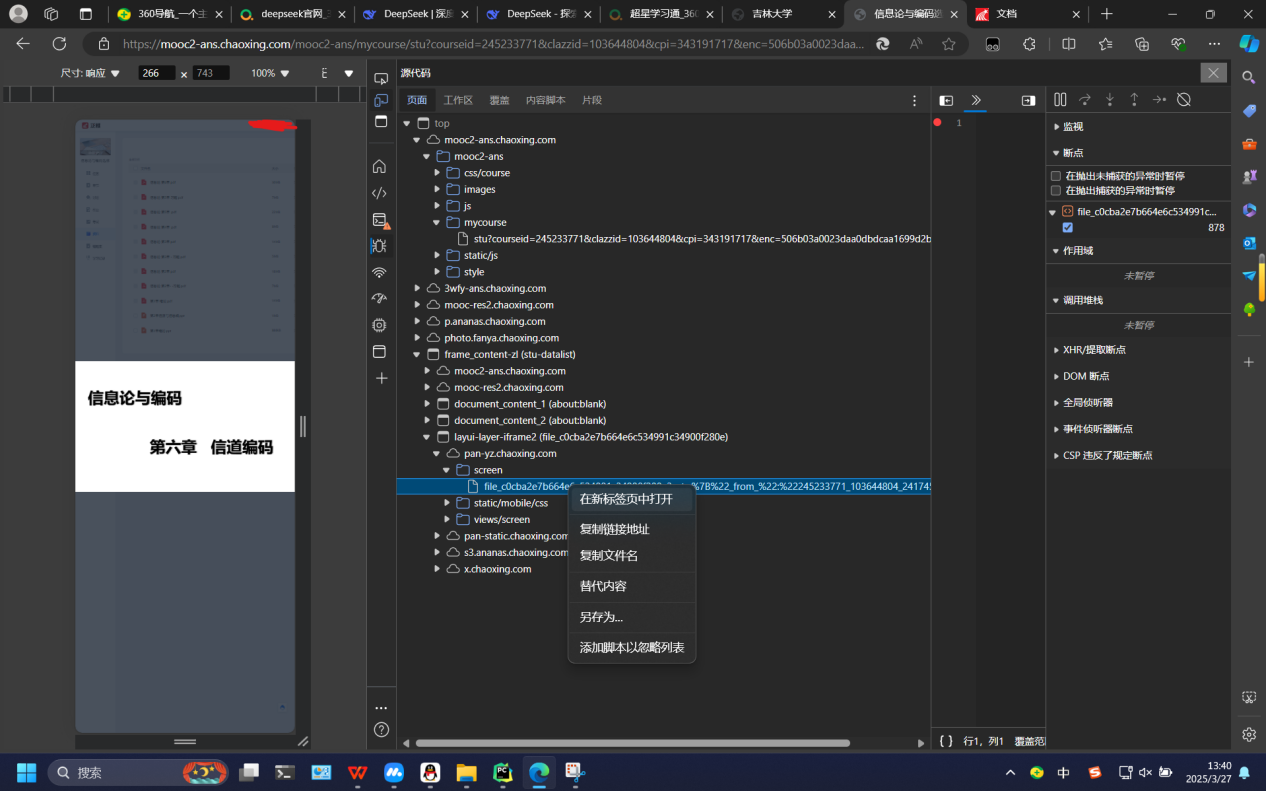
本工具使用编译环境为PyCharm 2021.1.3，使用Python3.10可正常运行。本工具仅限于获取学习文件使用，方便使用平板等工具的同学进行批注学习。仅可在超星学习通上使用。

使用方法：

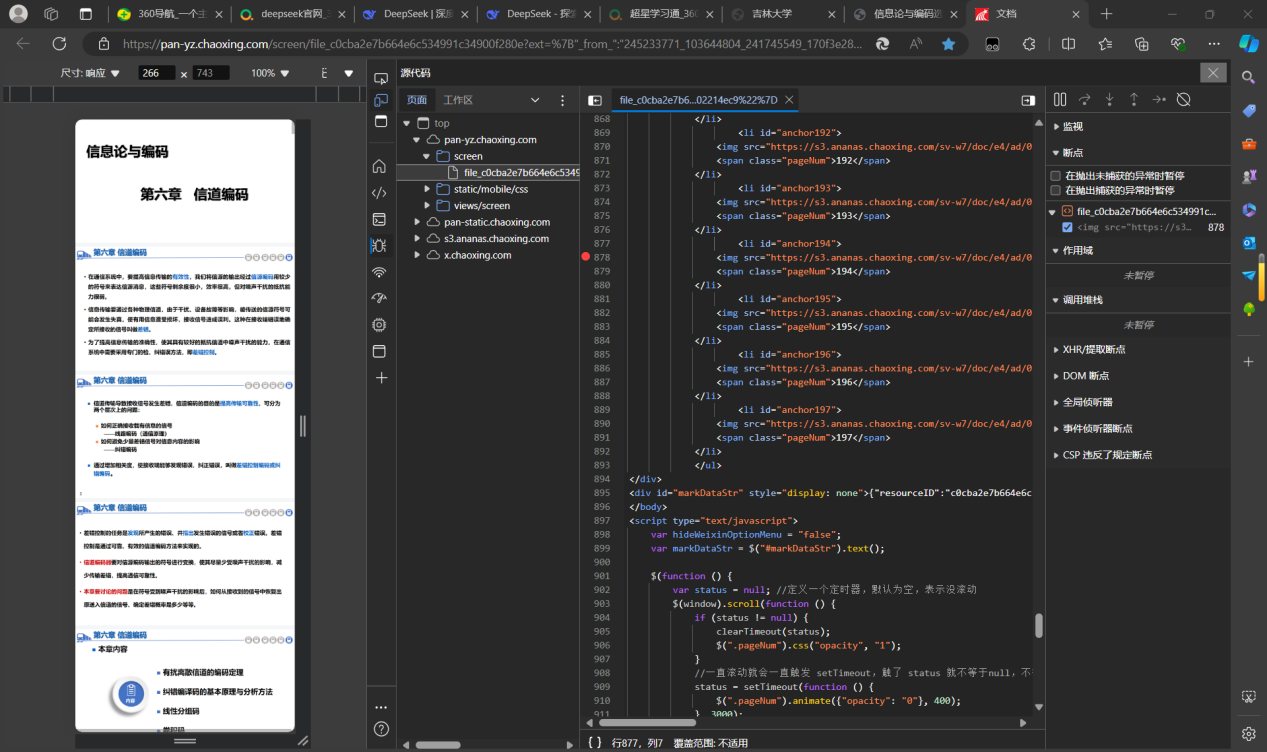
1.打开不可直接下载的ppt



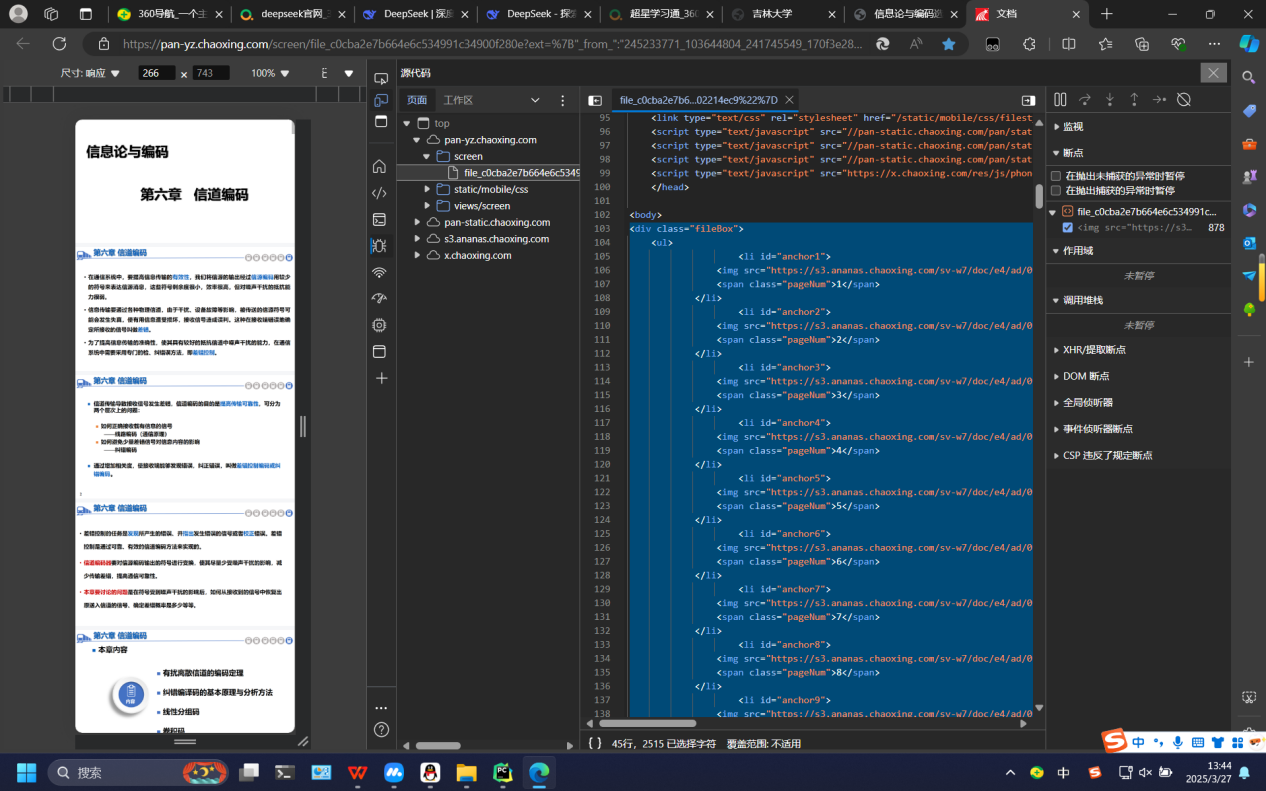
2.点击F12，进入源代码界面，找到”file\_”开头的文件，右键，选择“在新标签页打开”。

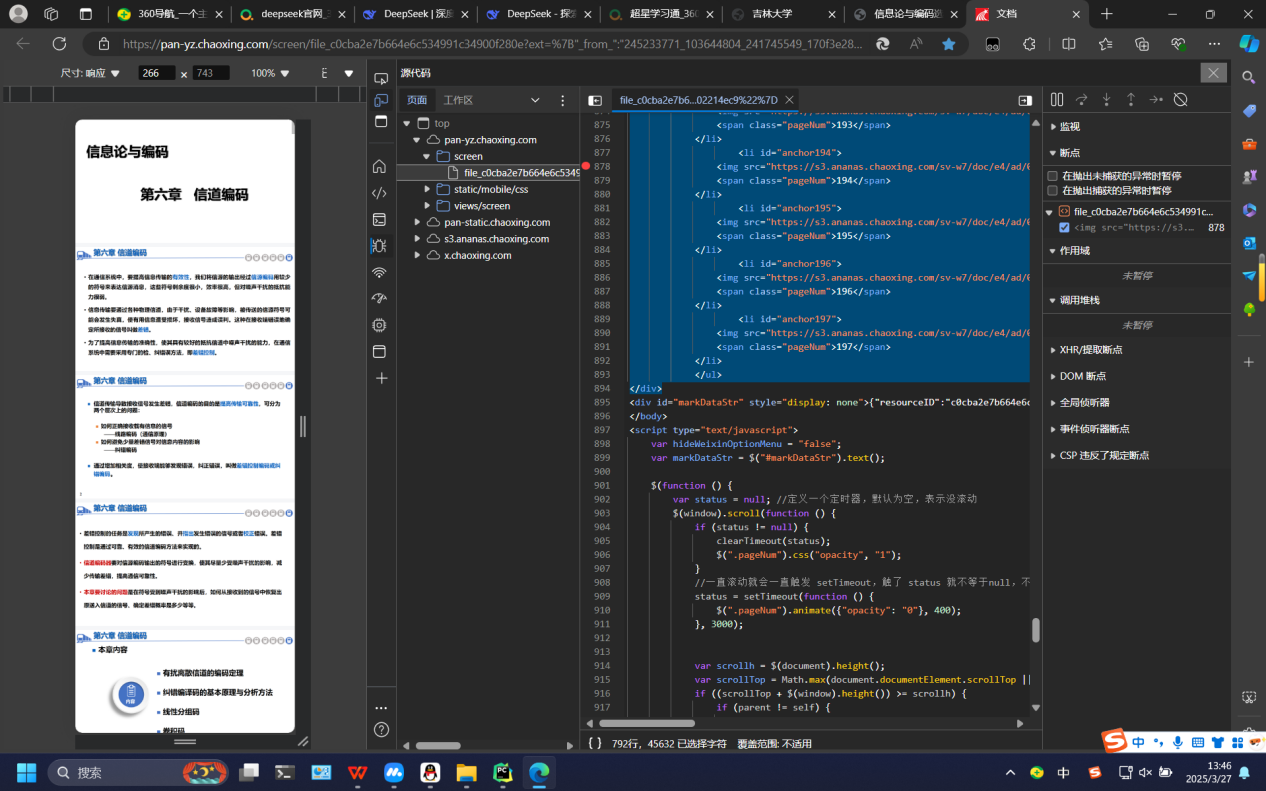


3.执行完2后，ppt会在一个网页单独打开。再按F12进入开发者模式，点开”file\_”开头的文件进入该界面。

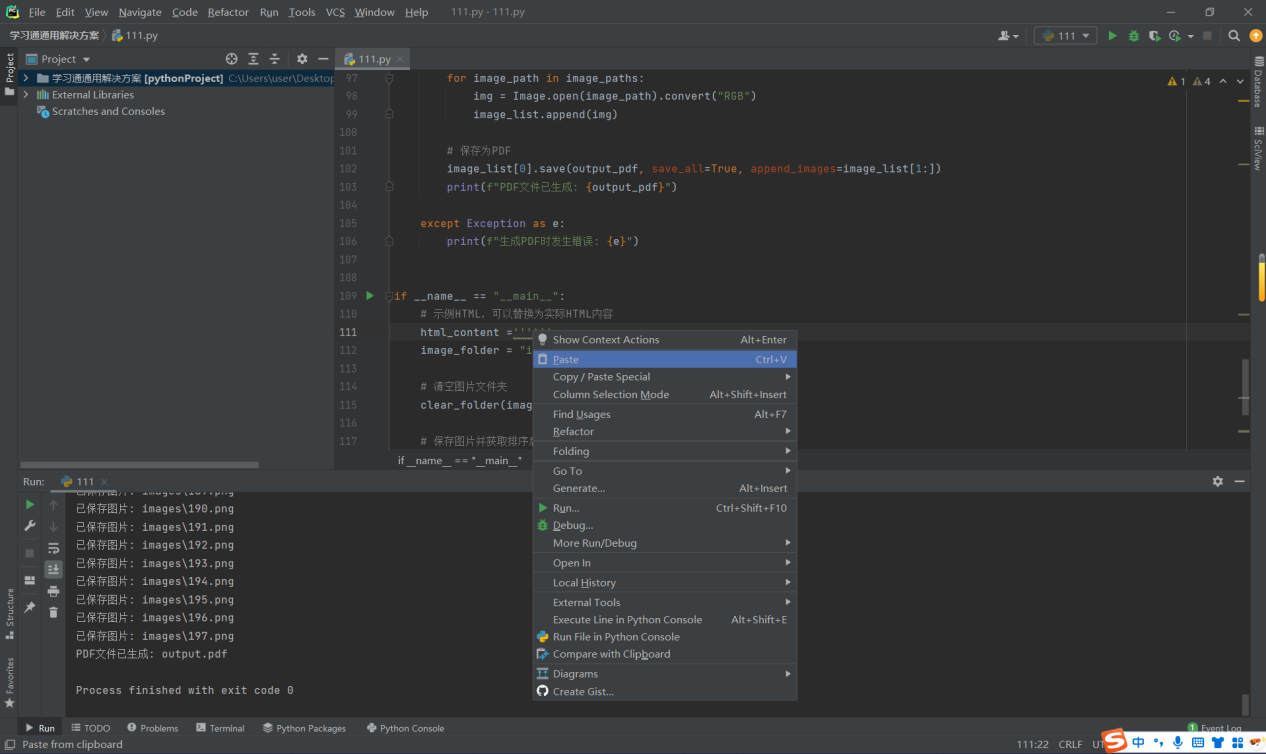


4.在代码区找到以<div class="fileBox">开头的行，由第一个图所示位置开始框选至第二个图所示的位置。框选完成后，复制。

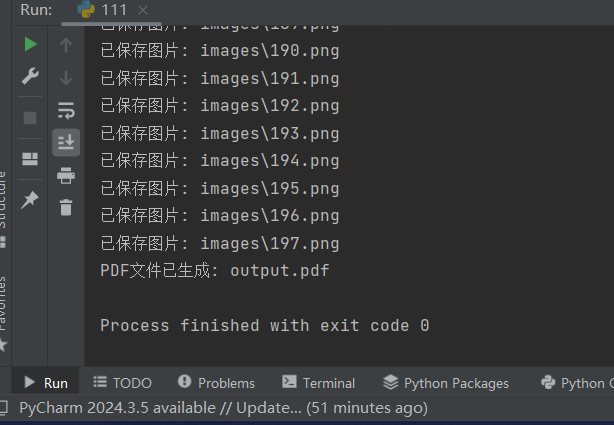


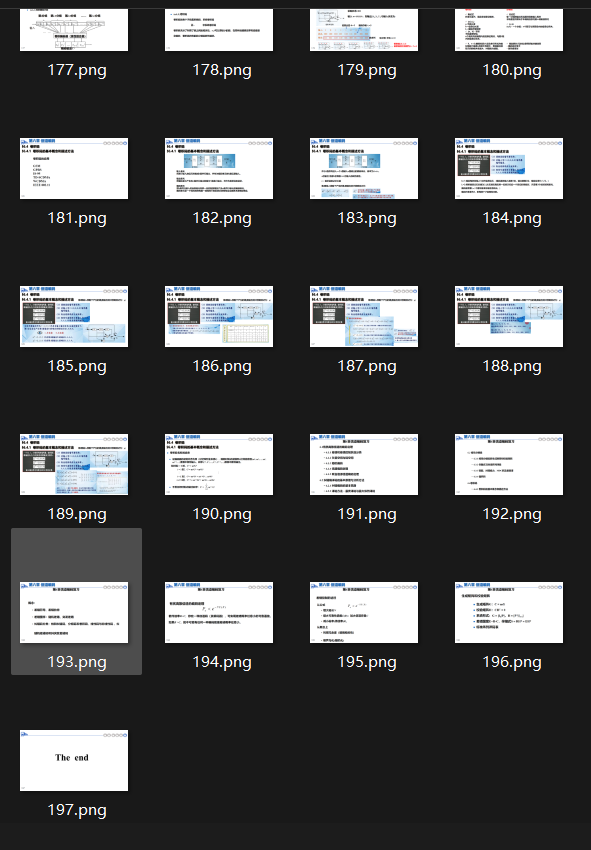


5.进入PyCharm，在Python代码找到如下位置，将复制内容粘贴至’’’(复制的内容)’’’，保存文件，配置好环境后开始运行，得到output.pdf即为学习通里不可下载的PPT



1. 成功示例





代码：

import os

import requests

from bs4 import BeautifulSoup

from urllib.parse import urljoin

from PIL import Image

def clear\_folder(folder\_path):

"""

清空指定文件夹。

参数：

folder\_path (str): 需要清空的文件夹路径。

"""

if os.path.exists(folder\_path):

for file in os.listdir(folder\_path):

file\_path = os.path.join(folder\_path, file)

try:

if os.path.isfile(file\_path):

os.unlink(file\_path) # 删除文件

elif os.path.isdir(file\_path):

os.rmdir(file\_path) # 删除子文件夹

except Exception as e:

print(f"无法删除 {file\_path}: {e}")

def save\_images\_from\_html(html\_content, save\_folder="images"):

"""

从HTML内容中提取图片并下载到本地文件夹。

参数：

html\_content (str): HTML内容。

save\_folder (str): 保存图片的本地文件夹路径。

"""

try:

# 解析HTML

soup = BeautifulSoup(html\_content, 'html.parser')

# 查找所有图片标签及其对应的页码

img\_tags = soup.find\_all('li')

# 创建保存图片的文件夹

if not os.path.exists(save\_folder):

os.makedirs(save\_folder)

images = []

# 下载图片并保存到本地

for img\_tag in img\_tags:

img\_url = img\_tag.find('img').get('src')

page\_num = img\_tag.find('span', class\_='pageNum').text.strip()

if not img\_url:

continue

# 获取图片内容

try:

img\_data = requests.get(img\_url).content

img\_name = f"{page\_num}.png" # 使用页码作为文件名

save\_path = os.path.join(save\_folder, img\_name)

# 保存图片到本地

with open(save\_path, 'wb') as img\_file:

img\_file.write(img\_data)

images.append((page\_num, save\_path)) # 保存页码和路径

print(f"已保存图片: {save\_path}")

except Exception as e:

print(f"无法下载图片 {img\_url}: {e}")

# 按照页码排序图片

images.sort(key=lambda x: int(x[0])) # 按照数字顺序排序

return [img[1] for img in images] # 返回排序后的图片路径列表

except Exception as e:

print(f"发生错误: {e}")

return []

def images\_to\_pdf(image\_paths, output\_pdf="output.pdf"):

"""

将指定的图片路径列表转换为单个PDF文件。

参数：

image\_paths (list): 图片路径列表。

output\_pdf (str): 生成的PDF文件路径。

"""

try:

# 确保有图片可供转换

if not image\_paths:

print("没有找到图片文件，无法生成PDF。")

return

# 打开图片并转换为PDF格式

image\_list = []

for image\_path in image\_paths:

img = Image.open(image\_path).convert("RGB")

image\_list.append(img)

# 保存为PDF

image\_list[0].save(output\_pdf, save\_all=True, append\_images=image\_list[1:])

print(f"PDF文件已生成: {output\_pdf}")

except Exception as e:

print(f"生成PDF时发生错误: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

# 示例HTML，可以替换为实际HTML内容

html\_content =''''''

image\_folder = "images"

# 清空图片文件夹

clear\_folder(image\_folder)

# 保存图片并获取排序后的图片路径列表

image\_paths = save\_images\_from\_html(html\_content, save\_folder=image\_folder)

# 将排序后的图片转换为PDF

images\_to\_pdf(image\_paths, output\_pdf="output.pdf")