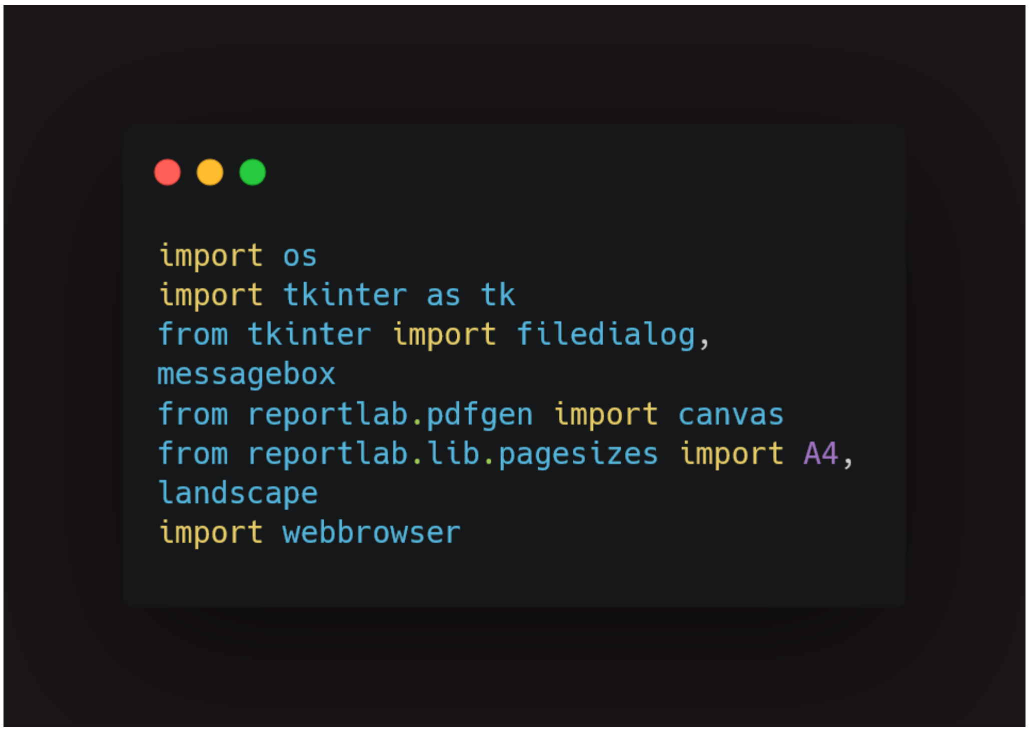
Presentation:

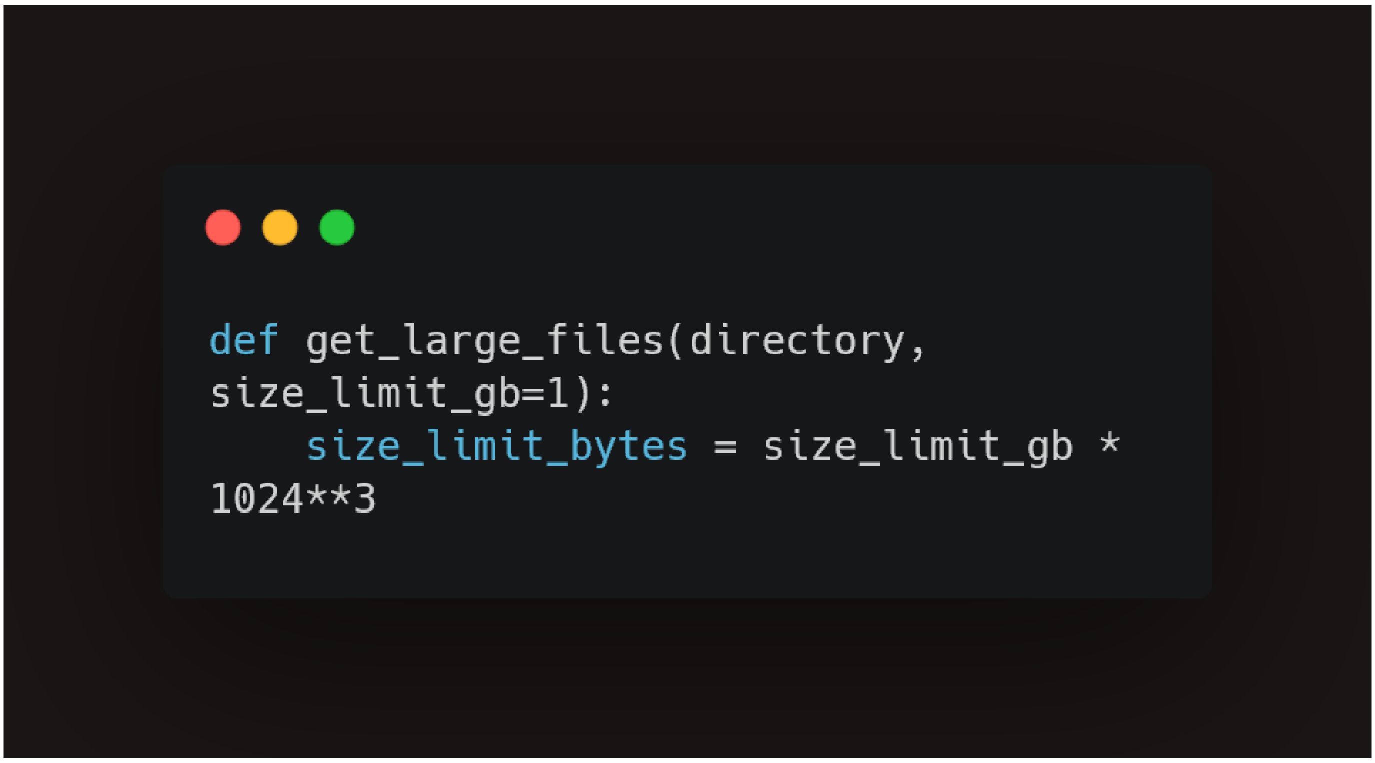
Imports



```python
import os
import tkinter as tk
from tkinter import filedialog,
messagebox
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4,
landscape
import webbrowser
```

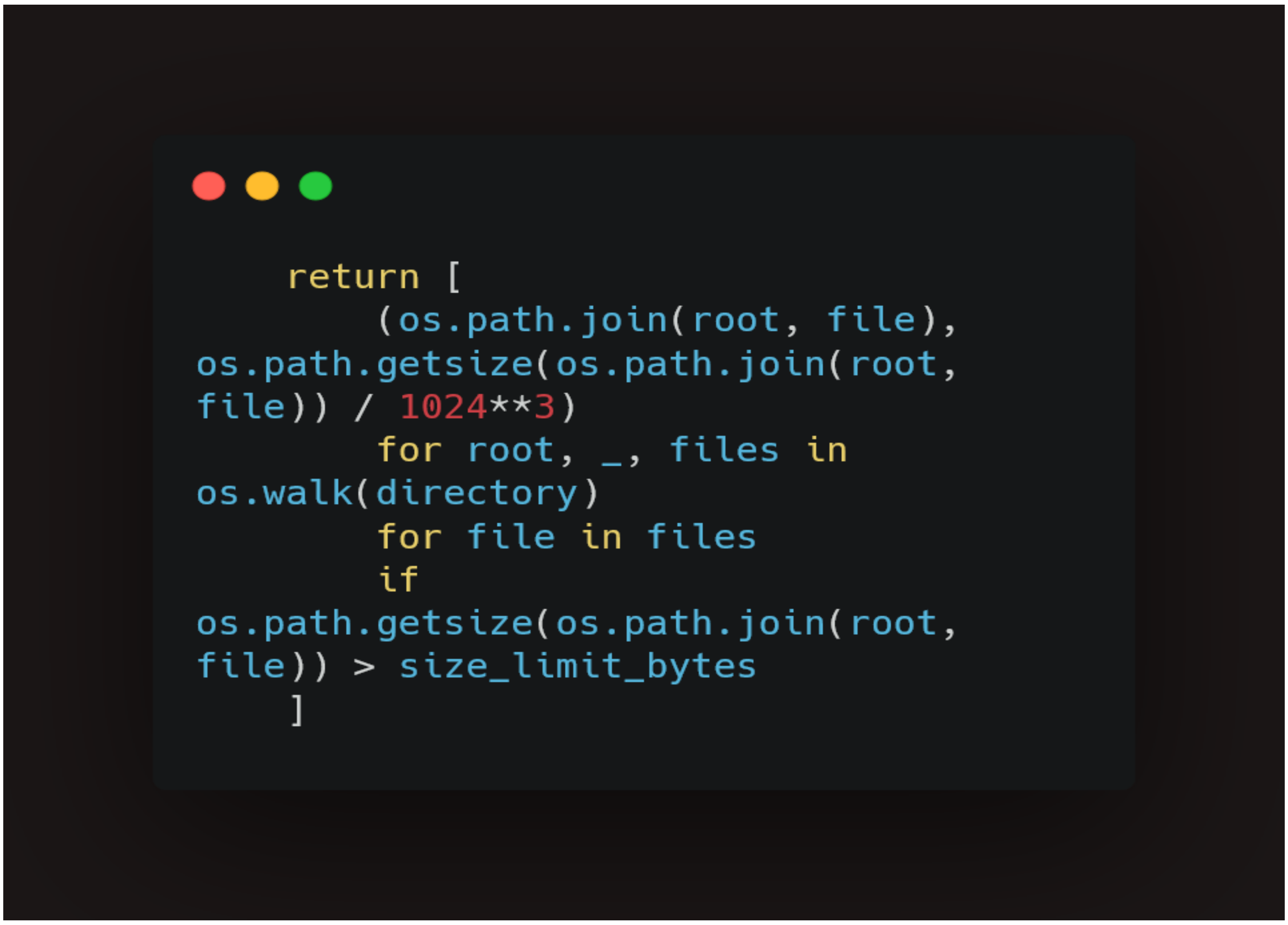
- **`os`**: Handles file system operations like walking directories and getting file sizes.
- **`tkinter`**: Provides the graphical interface for user dialogs.
- **`filedialog` & `messagebox`**: Modules from `tkinter` used to select directories and display messages.
- **`reportlab`**: A library to create PDF files programmatically.
    - **`canvas`**: Used for drawing and generating PDF content.
    - **`A4` & `landscape`**: Define page dimensions.
- **`webbrowser`**: Opens the generated PDF in the default browser or PDF viewer.

Function: `get_large_files`



```python
def get_large_files(directory,
size_limit_gb=1):
    size_limit_bytes = size_limit_gb *
1024**3
```

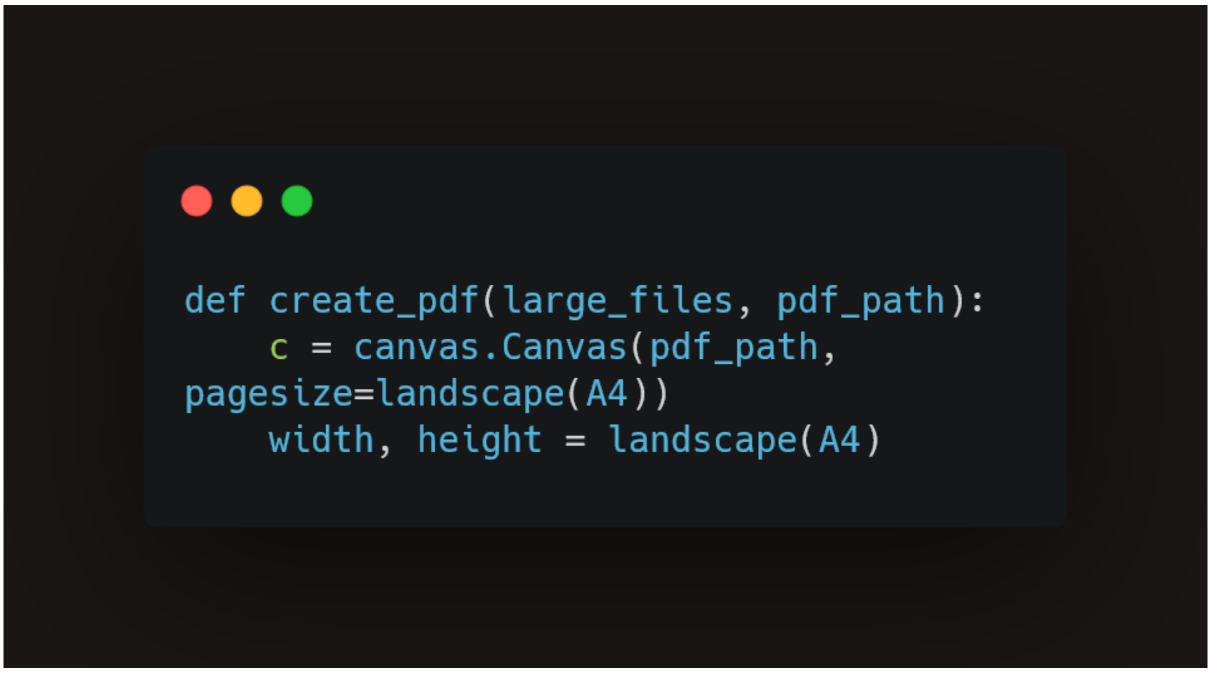
Converts the size limit from gigabytes to bytes (`1 GB = 1024`^{3} `bytes`)



```
    return [
        (os.path.join(root, file),
os.path.getsize(os.path.join(root,
file)) / 1024**3)
        for root, _, files in
os.walk(directory)
        for file in files
        if
os.path.getsize(os.path.join(root,
file)) > size_limit_bytes
    ]
```

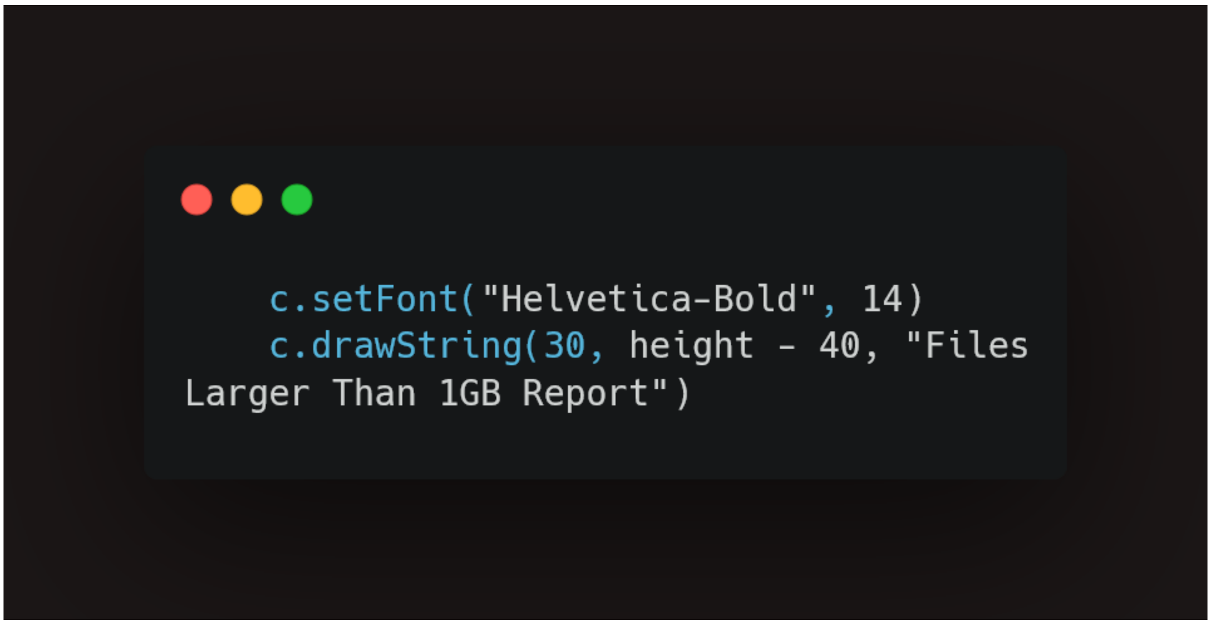
- **`os.walk(directory)`**: Recursively iterates through all subdirectories and files.
- **`os.path.join(root, file)`**: Constructs the full path of each file.
- **`os.path.getsize(path)`**: Gets the file size in bytes.
- Filters files larger than the specified size and calculates their size in GB.
- Returns a list of tuples containing the file path and size.

Function: `create_pdf`



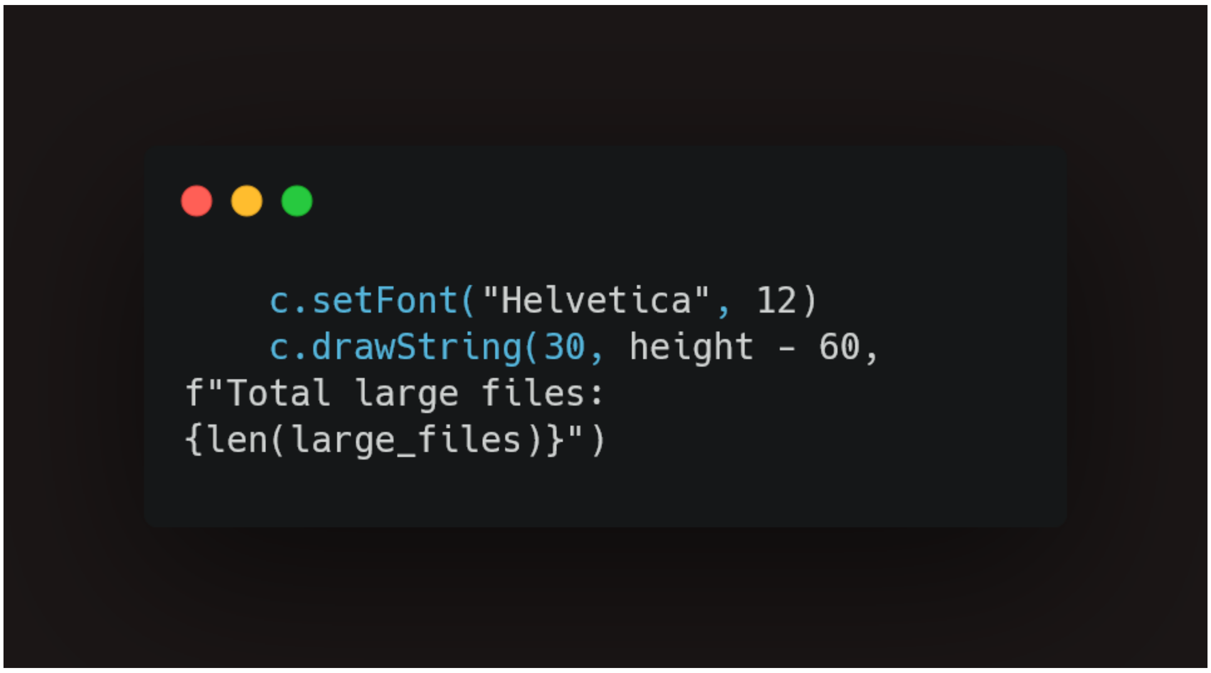
```python
def create_pdf(large_files, pdf_path):
    c = canvas.Canvas(pdf_path,
pagesize=landscape(A4))
    width, height = landscape(A4)
```

- **`canvas.Canvas`**: Creates a new PDF file.
- **`landscape(A4)`**: Sets the page orientation to landscape with A4 size.
- Stores page width and height for positioning content.



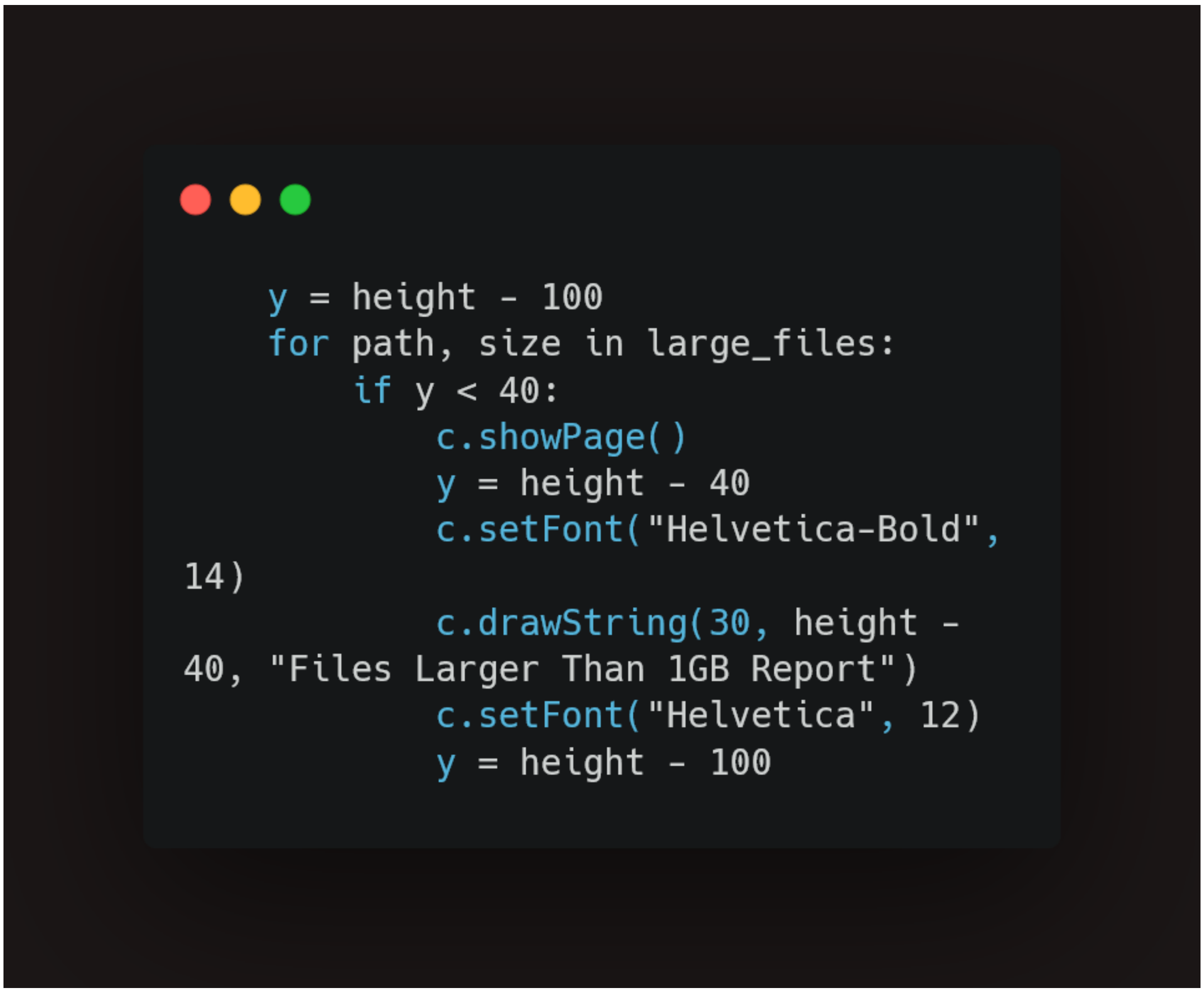
```
    c.setFont("Helvetica-Bold", 14)
    c.drawString(30, height - 40, "Files
Larger Than 1GB Report")
```

- **`setFont`**: Sets the font and size.
- **`drawString`**: Writes a title at a specific position on the page.



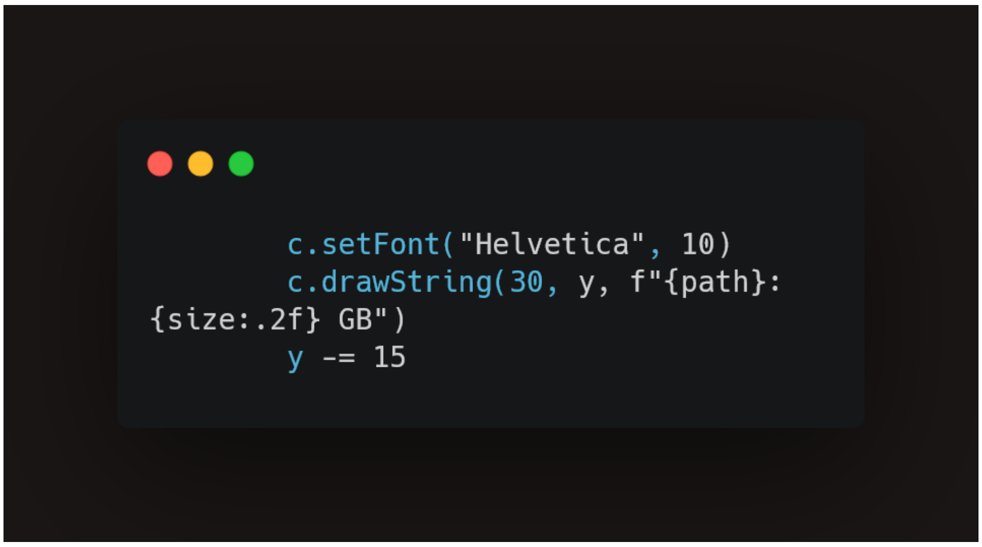
```
    c.setFont("Helvetica", 12)
    c.drawString(30, height - 60,
f"Total large files:
{len(large_files)}")
```

Adds a subtitle displaying the number of large files.



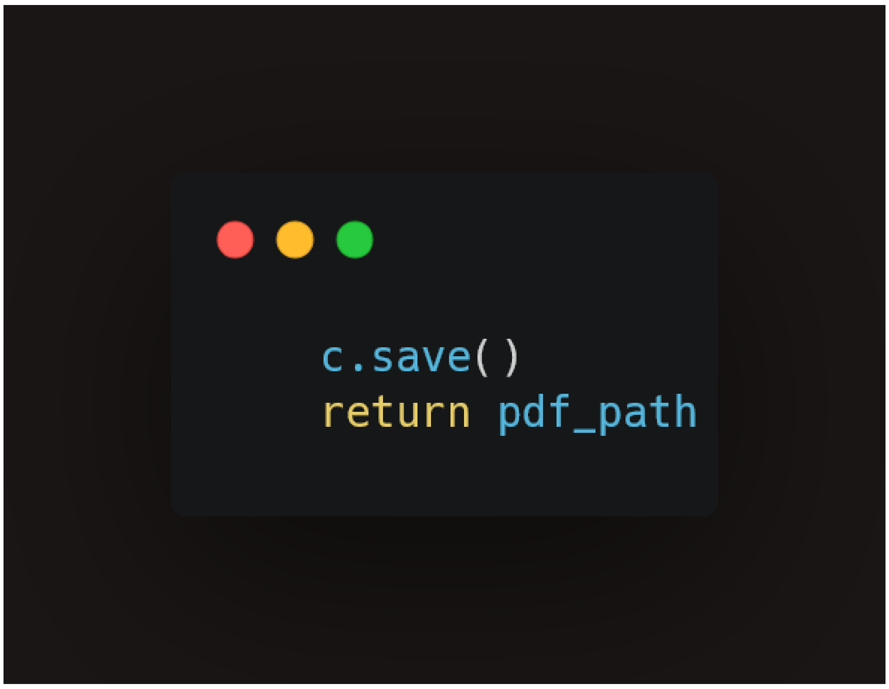
```
    y = height - 100
    for path, size in large_files:
        if y < 40:
            c.showPage()
            y = height - 40
            c.setFont("Helvetica-Bold",
14)
            c.drawString(30, height -
40, "Files Larger Than 1GB Report")
            c.setFont("Helvetica", 12)
            y = height - 100
```

- Loops through the list of large files and prints each file's path and size.
- If the page fills up (`y < 40`), starts a new page using `showPage`.



```
        c.setFont("Helvetica", 10)
        c.drawString(30, y, f"{path}:
{size:.2f} GB")
        y -= 15
```

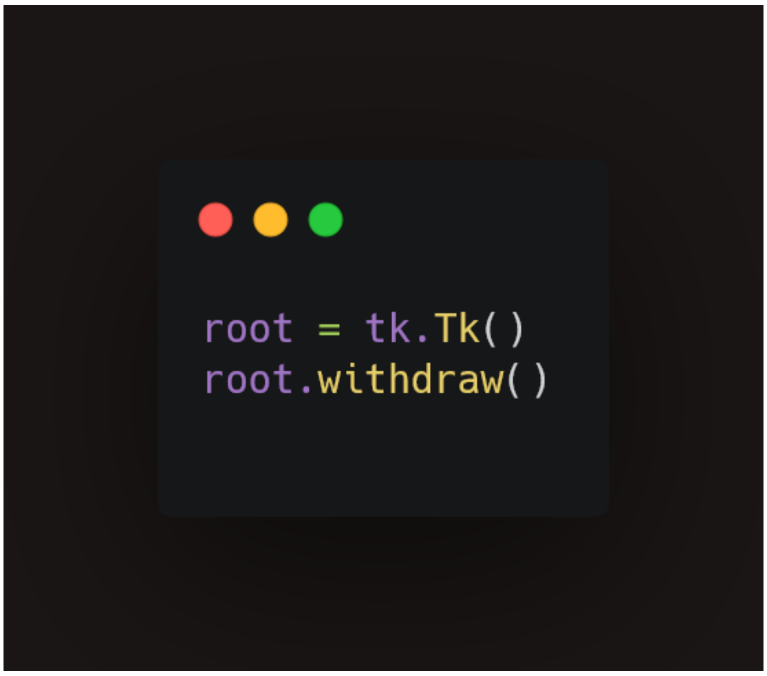
Writes each file's details on a new line. Moves the y-coordinate for the next line.



```
c.save()
return pdf_path
```

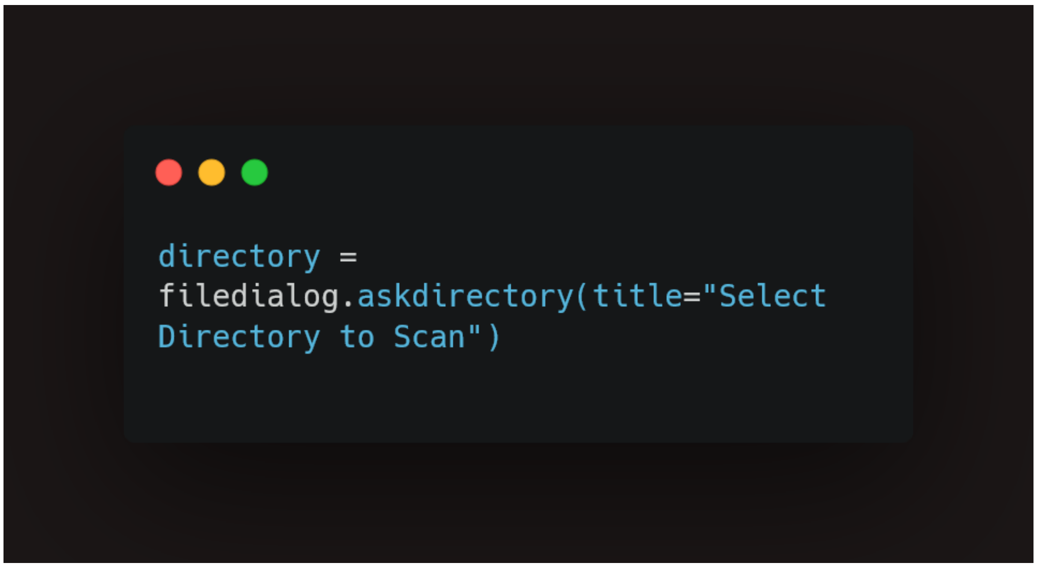
Saves the PDF and returns its path.

Main Program



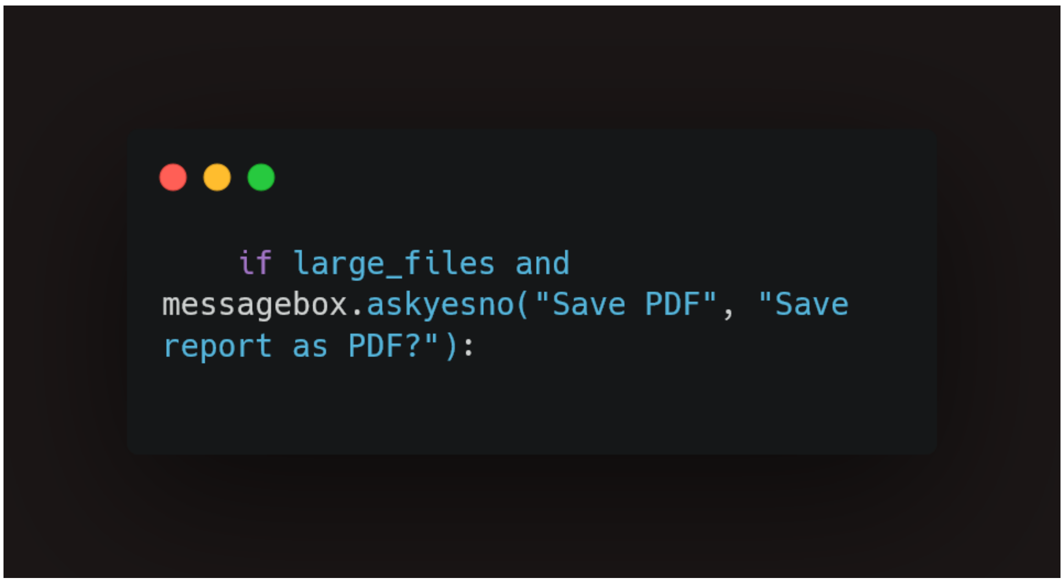
```
root = tk.Tk()
root.withdraw()
```

Initializes a hidden `Tkinter` window for dialogs.



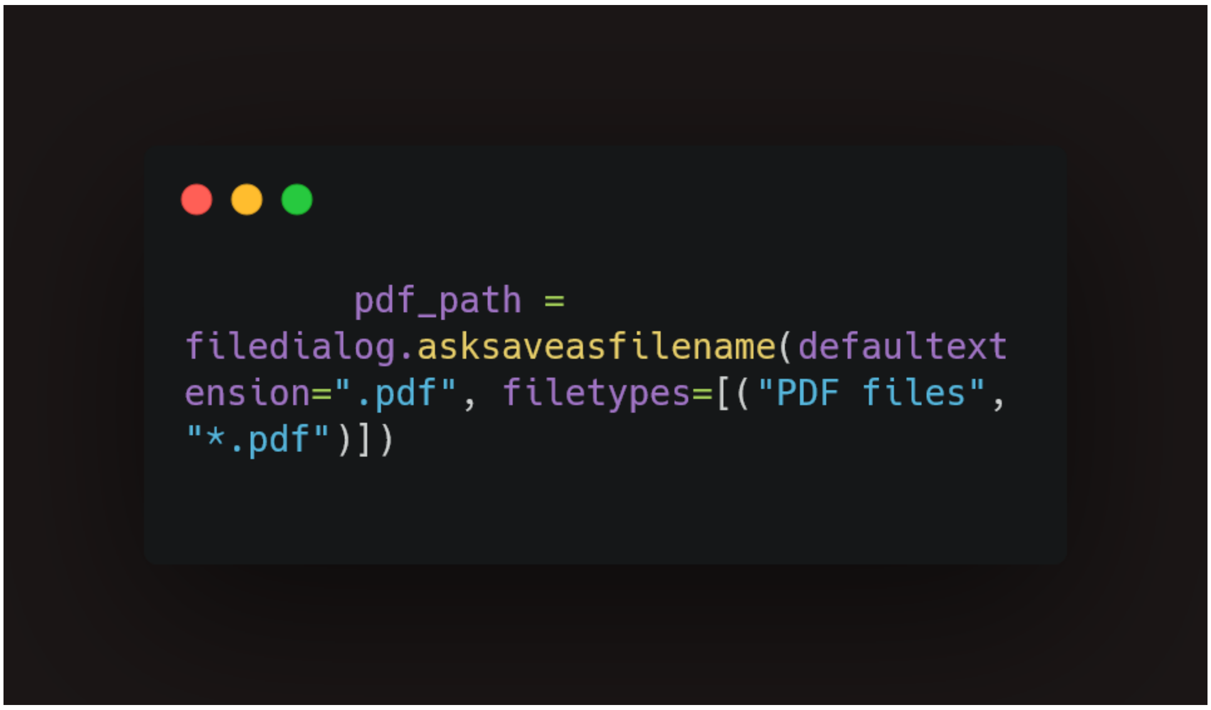
```
directory =
filedialog.askdirectory(title="Select
Directory to Scan")
```

If the user selects a directory, scans it for large files using `get_large_files`.



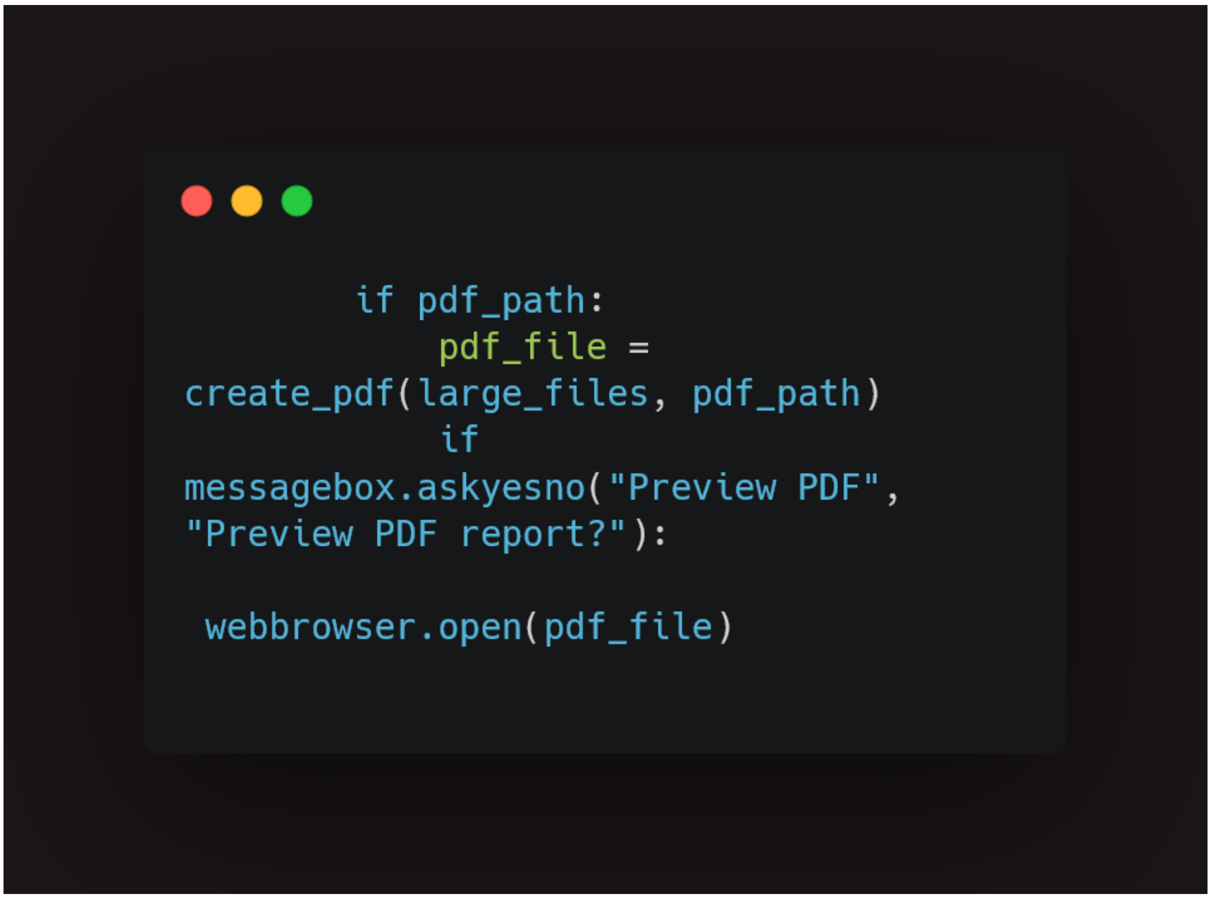
```
    if large_files and
messagebox.askyesno("Save PDF", "Save
report as PDF?"):
```

If large files are found, prompts the user to save the report as a PDF.



```
        pdf_path =
filedialog.asksaveasfilename(defaultext
ension=".pdf", filetypes=[("PDF files",
"*.pdf")])
```

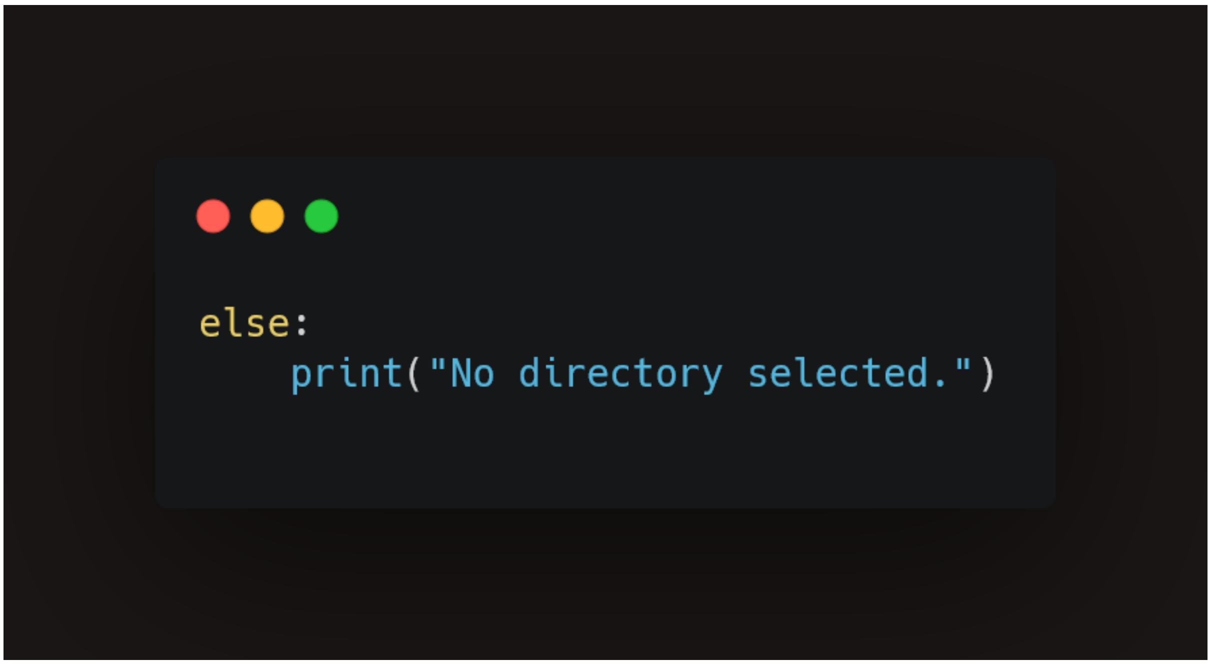
Opens a "Save As" dialog to choose the PDF file name and location.



```
        if pdf_path:
            pdf_file =
create_pdf(large_files, pdf_path)
            if
messagebox.askyesno("Preview PDF",
"Preview PDF report?"):

 webbrowser.open(pdf_file)
```

If a save location is selected, generates the PDF and optionally opens it in the default viewer.



```python
else:
    print("No directory selected.")
```

Prints a message to the console if no directory is selected.