```
==== compile_to_pdf.py ====
import os
from pathlib import Path
from fpdf import FPDF

# Extensions to include
FILE_EXTENSIONS = [".py", ".yaml", ".yml", ".json", ".txt"]

class CodePDF(FPDF):
    def __init__(self):
        super().__init__()
        self.set_auto_page_break(auto=True, margin=15)
        self.add_page()
        self.set_font("Courier", size=8)

    def add_code_file(self, filepath):
        self.set_font("Courier", size=8)
        self.multi_cell(0, 5, f"\n==== {filepath} ====\n")
        try:
            with open(filepath, 'r', encoding='utf-8', errors='ignore') as f:
                for line in f:
                    clean_line = ''.join(c if 0x20 <= ord(c) <= 0x7E or c in '\t\n\r' else '?' for c in line)
                    self.multi_cell(0, 5, clean_line.rstrip())
        except Exception as e:
            self.multi_cell(0, 5, f"[Error reading {filepath}: {e}]\n")

def gather_files(root_dir, extensions):
    return [
        f for f in Path(root_dir).rglob("*")
         if f.is_file() and f.suffix.lower() in extensions and "venv" not in f.parts and "__pycache__" not in
f.parts
    ]

def main(root=".", output="symbolic_manifesto.pdf"):
    pdf = CodePDF()
    files = gather_files(root, FILE_EXTENSIONS)

    if not files:
        print("[!] No matching files found.")
        return

    for file in sorted(files):
        pdf.add_code_file(file)

    pdf.output(output)
    print(f"[?] Compiled {len(files)} files into: {output}")

if __name__ == "__main__":
    main()

==== modelscan.py ====
import torch
from pathlib import Path
```

```
model_dir = Path(".")
model_files = sorted(model_dir.glob("*.pt"))

for model_file in model_files:
    print(f"\n? Inspecting: {model_file.name}")
    try:
        data = torch.load(model_file, map_location="cpu")
        if isinstance(data, dict):
            print(f"  Keys: {list(data.keys())}")
            if 'model_state_dict' in data:
                layers = data['model_state_dict']
                print(f"  Layers: {len(layers)}")
                for k, v in list(layers.items())[:5]:
                    print(f"    ?? {k}: {tuple(v.shape)}")
        else:
            print(f"  Type: {type(data)}")
    except Exception as e:
        print(f"  [ERROR] Failed to load: {e}")
```