Python Image Generation Project Setup Guide

This guide walks through creating a clean Python project from scratch for AI image generation scripts. It covers installing Python, creating a virtual environment, installing packages, organizing folders, and running scripts correctly.

1. Install Python

Download Python from python.org and install it.

IMPORTANT:
- Check 'Add Python to PATH'
- Restart VS Code after installation

2. Create a Project Folder

Create a new folder anywhere on your computer.

Example:
C:\Users\YOURNAME\Desktop\ai-image-project

3. Open the Folder in VS Code

Open VS Code.
Go to File → Open Folder
Select your project folder you created in step 2.

4. Open the Terminal

In VS Code:
Terminal → New Terminal

5. Create a Virtual Environment (.venv)

```
python -m venv .venv
```

6. Activate the Virtual Environment

```
.\.venv\Scripts\Activate.ps1
```

7. Verify Activation

Your terminal should now begin with:

(.venv)

8. Install Required Packages

```
pip install requests fal-client python-dotenv
```

9. Create Project Folders

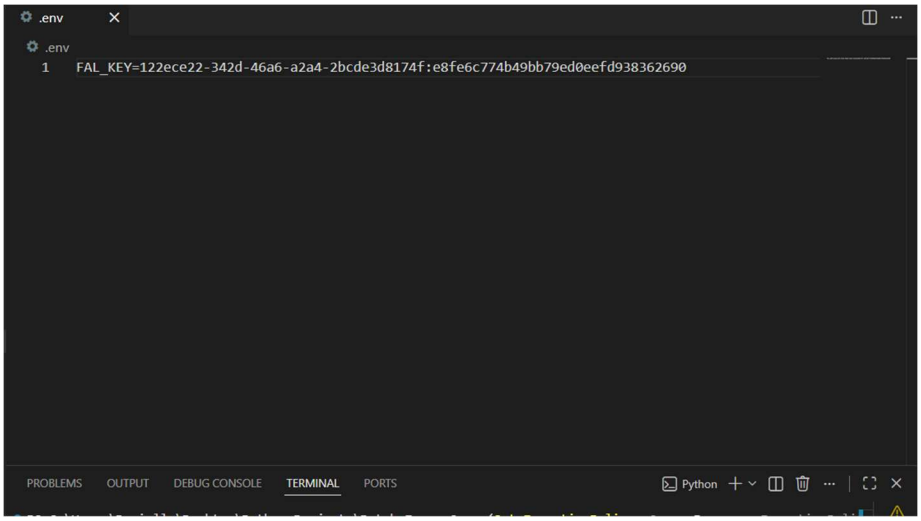
```
mkdir prompts
mkdir output
mkdir scripts
```

10. Example Project Structure

```
ai-image-project/
│
├── .venv/
├── .env
├── prompts/
│   └── prompts.txt
├── output/
├── scripts/
│   └── generate_basic.py
└── requirements.txt
```

11. Create a .env File

Create a file named .env in the project root. This is where your API key will go. It will look like this:



Example:
FAL_KEY=your_api_key_here OR

NOTE: I went with Fal.ai for my API key. You can add money and use credits as needed. You can add as little as $1 to get started and that will go pretty far for testing. With most pictures being less than a penny to generate. You can use any API key you choose or feel is best for your needs. You will have to update the script and .env for those changes.

12. Create prompts.txt

Inside prompts/prompts.txt place one prompt per line.

Example:
A cinematic chalkboard AI control room
Minimal sci-fi interface
Abstract server system

13. Create generate_basic.py

Place your image generation script inside the scripts folder. Script below

Change the style lock and add any negative prompts inside the brackets.

14. Run the Script

```
python scripts\generate_basic.py
```

15. Recommended Workflow

1. Open project folder
2. Activate .venv
3. Install packages if needed
4. Edit prompts
5. Run script
6. Generated images save into the output folder

Final Notes

Always activate the virtual environment before installing packages or running scripts. This keeps your project isolated and prevents dependency issues.

Script:

```
import os
from pathlib import Path

import requests
from dotenv import load_dotenv
import fal_client

ROOT = Path(__file__).resolve().parents[1]
PROMPTS_FILE = ROOT / "prompts" / "prompts.txt"
OUT_DIR = ROOT / "outputs"

load_dotenv(ROOT / ".env")

key = os.getenv("FAL_KEY", "").strip()
if not key:
    raise SystemExit("FAL_KEY missing. Add it to .env like: FAL_KEY=your_key_here")

os.environ["FAL_KEY"] = key

if not PROMPTS_FILE.exists():
    raise SystemExit("Missing prompts/prompts.txt")

prompts = [
    p.strip()
    for p in PROMPTS_FILE.read_text(encoding="utf-8").splitlines()
    if p.strip()
]

if not prompts:
    raise SystemExit("prompts.txt is empty")

OUT_DIR.mkdir(parents=True, exist_ok=True)

MODEL = "fal-ai/flux/schnell"
WIDTH = 1280
```

```
HEIGHT = 720



STYLE_LOCK = (
    "photo realistic"
    "vintage 2000s"
    "film grain"
)

NEGATIVE_PROMPT = (

)

def download_image(url: str, out_path: Path) -> None:
    response = requests.get(url, timeout=120)
    response.raise_for_status()
    out_path.write_bytes(response.content)

for i, prompt in enumerate(prompts, start=1):
    full_prompt = f"{prompt}. {STYLE_LOCK}"

    #print(f"Generating {i}/{len(prompts)}: {prompt}")

    handler = fal_client.submit(
        MODEL,
        arguments={
            "prompt": full_prompt,
            "negative_prompt": NEGATIVE_PROMPT,
            "image_size": {
                "width": WIDTH,
                "height": HEIGHT,
            },
            "output_format": "png",
        },
    )

    result = handler.get()
    image_url = result["images"][0]["url"]

    out_path = OUT_DIR / f"scene_{i:03d}.png"
    download_image(image_url, out_path)

    print(f"Saved {out_path.name}")

print(f"\nDone. Generated {len(prompts)} images in {OUT_DIR}")
```