Lab 1: Getting Prepared for Digital Forensic Programming

Purpose

This lab will help you prepare your environment for digital forensic programming using Python. By the end of this session, you should be able to:

- Use **VS Code** with the integrated terminal.
- Create and activate a **Python virtual environment**.
- Install and verify required packages inside the virtual environment.
- Confirm that **Git** is installed and available.
- Run a simple forensic Python program.

Section A – University Lab Machines (No Admin Rights)

Important Notes

- You **do not have administrator rights** on lab machines.
- All packages must be installed inside the **virtual environment**.
- Do **not** try to install globally — it will fail.
- IT staff have provided a pre-compiled wheel file (`pytsk3-20250312-cp311-cp311-win_amd64.whl`) so you don't need to compile from source.

Step 1 – Check Environment in VS Code Terminal

If python is not recognized or shows the wrong version on Windows lab machines, run these PowerShell commands in the VS Code terminal (no admin needed), then proceed with the checks below.

1. Use the installed Python immediately in this terminal:

```
Set-Alias -Name python -Value "C:\Program Files\Python313\python.exe"; Set-Alias -Name python3 -Value "C:\Program Files\Python313\python.exe"; python --version
```

2. Make it take precedence in this terminal:

```
$env:Path = "C:\Program Files\Python313;C:\Program Files\Python313\Scripts;" + $env:Path; Get-Command python; python --version
```

3. Make it permanent for your user (restart VS Code after running):

```
[Environment]::SetEnvironmentVariable('Path', "C:\Program
Files\Python313;C:\Program Files\Python313\Scripts;" +
[Environment]::GetEnvironmentVariable('Path','User'), 'User')
```

4. Open App execution aliases and toggle off “App Installer python.exe” and “python3.exe”:

```
Start-Process "ms-settings:apps-advanced-app-settings"
```

```
python --version
pip --version
git --version
```

- `python` and `pip` must respond with a version.
- `git` should also respond with a version (if not, Git is not available on this lab machine).

If pip is missing or broken on Windows lab machines, run these in the VS Code PowerShell terminal:

```
& "C:\Program Files\Python313\python.exe" -m pip --version
# If pip is missing, install/repair it and upgrade pip
& "C:\Program Files\Python313\python.exe" -m ensurepip --upgrade; & "C:\Program
Files\Python313\python.exe" -m pip install --upgrade pip
```

Then use the PATH steps above to prioritize Python 3.13 in this terminal and make it permanent if needed.

Step 2 – Create a Virtual Environment

1. In VS Code, open the folder where you will store your lab work.
2. Create a virtual environment:

   ```
   python -m venv venv
   ```

   (On macOS, use `python3` instead of `python` if needed.)

3. Activate the virtual environment:

   - **Windows (PowerShell inside VS Code)**:

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

If you encounter an execution policy error, run the following command to temporarily bypass it, then try activating again:

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

- **macOS/Linux**:

```
source venv/bin/activate
```

4. Your terminal prompt should now show `(venv)` at the start.

Step 3 – Test Pip Functionality

Before installing the complex `pytsk3` package, let's verify that pip works correctly in your virtual environment.

1. Install a simple test package:

```
pip install requests
```

2. Verify the installation:

```
pip list
```

You should see `requests` and its dependencies listed.

3. Test that the package works:

```
python -c "import requests; print('Pip and virtual environment working correctly!')"
```

If this prints the success message, your pip and virtual environment are functioning properly.

Step 4 – Install Packages from Local Wheel File

Since lab machines cannot compile packages, use the wheel provided by IT staff.

1. Ensure the wheel file (`pytsk3-20250312-cp311-cp311-win_amd64.whl`) is copied into your lab folder.

2. With the virtual environment activated, install using:

```
pip install pytsk3-20250312-cp311-cp311-win_amd64.whl
```

Or, if the file is in another directory, use the full path:

```
pip install "C:\path\to\pytsk3-20250312-cp311-cp311-win_amd64.whl"
```

3. Verify installation:

```
pip list
```

You should see `pytsk3` listed.

Step 5 – Verify Pytsk3 Installation

Create a simple test script to verify that `pytsk3` is working correctly.

1. Create a file called `test_pytsk3.py` with the following content:

```
#!/usr/bin/env python3
"""
Simple test script to verify pytsk3 installation and basic functionality.
"""

import re
import sys

def test_pytsk3_import():
    """Test if pytsk3 can be imported successfully."""
    try:
        import pytsk3
        print("✓ pytsk3 imported successfully")
        print(f"  Version: {pytsk3.TSK_VERSION_STR}")
        return True
    except ImportError as e:
        print(f"✗ Failed to import pytsk3: {e}")
        return False

def test_regex_functionality():
    """Test basic regex functionality for forensic pattern matching."""
    try:
        # Test email pattern matching (common in forensics)
        email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]
{2,}\b'
        test_text = "Found emails: john.doe@example.com and
admin@forensics.lab"
```

```
        emails = re.findall(email_pattern, test_text)

        if len(emails) == 2:
            print("✓ Regex functionality working correctly")
            print(f"  Found emails: {emails}")
            return True
        else:
            print(f"✗ Regex test failed - expected 2 emails, found
{len(emails)}")
            return False
    except Exception as e:
        print(f"✗ Regex test failed: {e}")
        return False

def main():
    """Run all tests."""
    print("Testing Digital Forensics Environment Setup...")
    print("=" * 50)

    pytsk3_ok = test_pytsk3_import()
    regex_ok = test_regex_functionality()

    print("=" * 50)

    if pytsk3_ok and regex_ok:
        print("🎉 All tests passed! Your environment is ready for digital
forensics work.")
        return 0
    else:
        print("❌ Some tests failed. Please check your installation.")
        return 1

if __name__ == "__main__":
    sys.exit(main())
```

2. Run the test script:

```
python test_pytsk3.py
```

You should see output indicating that both pytsk3 and regex functionality are working correctly.

Section B – Personal Laptops (With Admin Rights)

Step 1 – Install Required Software

1. **Visual Studio Code (VS Code)**
    - Download from: https://code.visualstudio.com

- Install the **Python extension**.

2. **Python 3.11 (Recommended)**
   - The provided wheel file (`pytsk3-20250312-cp311-cp311-win_amd64.whl`) is compiled for Python 3.11 (`cp311`).
   - Download Python 3.11 from: https://www.python.org/downloads/
   - If you use a different Python version, the wheel file may not install correctly.
   - Download from: https://www.python.org/downloads/
   - Tick **Add Python to PATH** during installation.
   - Verify installation:

```
python --version
pip --version
```

3. **Git**
   - Download from: https://git-scm.com/downloads
   - Verify installation:

```
git --version
```

Step 2 – Create and Use a Virtual Environment

Even with admin rights, it is **best practice** to use a virtual environment.

```
python -m venv venv
.\venv\Scripts\Activate      # Windows
source venv/bin/activate     # macOS/Linux
```

Step 3 – Test Pip Functionality

Before installing the complex `pytsk3` package, verify that pip works correctly.

```
pip install requests
pip list
python -c "import requests; print('Pip and virtual environment working
correctly!')"
```

Step 4 – Install Packages

You have **two options**:

Option A (Recommended for Consistency – Use Local Wheel File)
If you have been given the same wheel file (`pytsk3-20250312-cp311-cp311-win_amd64.whl`):

```
pip install pytsk3-20250312-cp311-cp311-win_amd64.whl
```

Why use the local wheel file?
Installing `pytsk3` directly from PyPI often fails on Windows because it requires a C++ compiler (such as Microsoft Visual C++ Build Tools) to build the package from source. Most users do not have these build tools installed, and the build process can be complex. The pre-built wheel file avoids this problem by providing a ready-to-install package.

Option B (Install from PyPI – Requires Compiler and Internet)

```
pip install pytsk3
```

Note: If you try this option and see an error like
`error: Microsoft Visual C++ 14.0 or greater is required`,
it means you need to use the local wheel file instead.

Confirm installation with:

```
pip list
```

Step 5 – Verify Pytsk3 Installation

Create and run the same test script as described in Section A:

1. Create `test_pytsk3.py` (see Section A, Step 5 for the complete code).
2. Run the verification:

   ```
   python test_pytsk3.py
   ```

   You should see output indicating that both pytsk3 and regex functionality are working correctly.

Student Checklist

For Everyone

- [ ] Open VS Code and terminal.
- [ ] Confirm `python`, `pip`, and `git` are installed.
- [ ] Create and activate a virtual environment.
- [ ] Test pip functionality by installing `requests` package.
- [ ] Install `pytsk3` (from **wheel file** or from **PyPI**, depending on setup).
- [ ] Run `pip list` and confirm both `requests` and `pytsk3` are installed.
- [ ] Create and run `test_pytsk3.py` to verify pytsk3 and regex functionality.

Additional for Personal Laptops

- [ ] Install VS Code, Python, and Git if not already installed.
- [ ] Ensure **Python is added to PATH** during installation.

Deliverable for Lab 1: Show your instructor that you can:

1. Activate your virtual environment.
2. Run `pip list` and display both `requests` and `pytsk3`.
3. Execute `test_pytsk3.py` and show successful verification output.