Terminal Commands, Git, and Flask Tutorial

Terminal Commands

Basic Navigation

```
# Show current directory location
pwd

# List directory contents
ls
ls -la  # Show hidden files and detailed information

# Change directories
cd /path/to/directory
cd ..   # Go up one directory
cd ~    # Go to home directory
cd -    # Go to previous directory
```

File Operations

```
# Create files and directories
touch newfile.txt
mkdir new_directory

# Copy, move, and rename
cp file.txt backup_file_copy.txt
mv oldname.txt newname.txt
mv file.txt different_directory/

# View file contents
cat file.txt
less file.txt  # For larger files

# Delete files and directories
rm file.txt
rm -r directory/  # Remove directory and contents
```

Useful Shortcuts and Commands

```
# Clear terminal
clear

# Find files
find . -name "*.txt"

# Search file contents
grep "search term" file.txt
grep -r "search term" directory/

# Command history
history
```

Git and Flask Tutorial

Setting Up a Flask Project

1. Create a new directory for the project

```
mkdir flask-demo
cd flask-demo
```

2. Create a virtual environment

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

3. Install Flask

```
pip install flask
```

4. Create the application structure

```
touch app.py
mkdir templates
touch templates/index.html
```

5. Add code to app.py

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html', message='Hello from Flask!')

if __name__ == '__main__':
    app.run(debug=True)
```

6. Add code to templates/index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Flask Demo</title>
</head>
<body>
    <h1>{{ message }}</h1>
    <p>This is a simple Flask application for Git demonstration.</p>
</body>
</html>
```

7. Create a requirements.txt file

```
pip freeze > requirements.txt
```

Setting Up Git with the Flask App

```
# Initialize a new repository
git init

# Create a .gitignore file for Python/Flask
touch .gitignore
echo "venv/" >> .gitignore
echo "__pycache__/" >> .gitignore
echo "*.pyc" >> .gitignore
echo ".env" >> .gitignore

# Check repository status
git status
```

Initial Commit of Flask App

```
# Stage all files (except those in .gitignore)
git add .

# Commit the initial Flask application
git commit -m "Initial commit of Flask application"

# View commit history
git log
git log --oneline # Compact view
```

Making Changes to the Flask App

```
# Add a new route to app.py
@app.route('/about')
def about():
    return render_template('about.html', title='About Us')

# Create about.html template
touch templates/about.html
```

Add the following content to templates/about.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>{{ title }}</title>
</head>
<body>
    <h1>About Us</h1>
    <p>This is the about page of our Flask application.</p>
    <a href="/">Back to Home</a>
</body>
</html>
```

```
# Stage and commit changes
git add app.py templates/about.html
git commit -m "Add about page to Flask application"
```

Working with Remote Repositories

First, create a new repository on GitHub through the web interface

```
# Add a remote repository
git remote add origin https://github.com/username/flask-demo.git

# View remote repositories
git remote -v

# Push changes to remote
git push -u origin main

# Pull changes from remote
git pull origin main
```

Branching and Merging with the Flask App

```
# Create a feature branch for a new Flask feature
git checkout -b feature-contact-form

# Now we'll add a contact form feature
```

Add the following to app.py:

```
@app.route('/contact', methods=['GET', 'POST'])
def contact():
    if request.method == 'POST':
        name = request.form.get('name')
        email = request.form.get('email')
        message = request.form.get('message')
        # Here you would typically save this to a database
        # or send an email
        return render_template('contact.html',
                               success=True,
                               name=name)
    return render_template('contact.html', success=False)
```

Create and add the following content to templates/contact.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Contact Us</title>
</head>
<body>
    <h1>Contact Us</h1>

    {% if success %}
        <p>Thank you, {{ name }}! Your message has been received.</p>
    {% else %}
        <form method="POST">
            <div>
                <label for="name">Name:</label>
                <input type="text" id="name" name="name" required>
            </div>
            <div>
                <label for="email">Email:</label>
                <input type="email" id="email" name="email" required>
            </div>
            <div>
                <label for="message">Message:</label>
                <textarea id="message" name="message" rows="5" required></textarea>
            </div>
            <button type="submit">Send Message</button>
        </form>
    {% endif %}

    <p><a href="/">Back to Home</a></p>
</body>
</html>
```

```
# Stage and commit changes
git add app.py templates/contact.html
git commit -m "Add contact page with form"
```

Don't forget to add the import for request at the top of app.py:

```
from flask import Flask, render_template, request
```

```
# Stage and commit changes
git add app.py templates/contact.html
git commit -m "Add contact form feature"

# Switch back to main branch
git checkout main

# Merge the feature branch
git merge feature-contact-form

# Delete the feature branch after merging
git branch -d feature-contact-form
```

Handling Merge Conflicts

```
# Create two branches with conflicting changes
git checkout -b update-home-page

# Edit index.html to change the heading
```

Change the h1 in index.html to:

```
<h1 style="color: blue;">{{ message }}</h1>
```

```
git add templates/index.html
git commit -m "Update home page heading with blue color"

git checkout main

# Edit index.html again but differently
```

Change the h1 in index.html to:

```
<h1 style="font-size: 24px;">{{ message }}</h1>
```

```
git add templates/index.html
git commit -m "Change home page heading size"

# Try to merge and handle the conflict
git merge update-home-page
```

You will get a merge conflict. Open index.html and you'll see something like:

```
<<<<<<< HEAD
<h1 style="font-size: 24px;">{{ message }}</h1>
=======
<h1 style="color: blue;">{{ message }}</h1>
>>>>>>> update-home-page
```

Edit the file to resolve the conflict, combining both changes:

```
<h1 style="font-size: 24px; color: blue;">{{ message }}</h1>
```

```
# After resolving the conflict
git add templates/index.html
git commit -m "Resolve merge conflict in home page"
```

Handling Sensitive Information

```
# Accidentally commit a config file with secrets
echo "SECRET_KEY = 'secret'" > config.py
git add config.py
git commit -m "Add config file"

# Remove sensitive file that was committed
git rm --cached config.py
echo "config.py" >> .gitignore
git add .gitignore
git commit -m "Remove sensitive config file and update gitignore"

# Create a safe example config file
echo "# Use environment variables for secrets" > config_example.py
echo "# SECRET_KEY = os.environ.get('SECRET_KEY')" >> config_example.py
git add config_example.py
git commit -m "Add example config file without secrets"
```

Common Git Commands Reference

Basic Commands

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
git config --list

# Repository Operations
git init                        # Initialize a new repository
git clone [url]                 # Clone an existing repository
git remote add origin [url]     # Connect to a remote repository
git remote -v                   # View remote repositories

# Basic Workflow
git status                      # Check repository status
git add [file]                  # Stage a file
git add .                       # Stage all files
git commit -m "[message]"       # Commit changes
git push origin [branch]        # Push to remote
git pull origin [branch]        # Pull from remote

# View History
git log                         # View commit history
git log --oneline               # Compact history view
git log --graph                 # Graphical history
git diff                        # Show unstaged changes
git diff --staged               # Show staged changes
```

Advanced Commands

```
# Branching
git branch                      # List branches
git branch [name]               # Create a branch
git checkout [branch]           # Switch to a branch
git checkout -b [branch]        # Create and switch to a branch
git merge [branch]              # Merge a branch into current
git branch -d [branch]          # Delete a branch

# Undoing Changes
git restore [file]              # Discard changes (Git 2.23+)
git restore --staged [file]     # Unstage changes (Git 2.23+)
git reset HEAD~1                # Undo last commit
git reset --hard HEAD~1         # Undo commit and discard changes
git commit --amend              # Modify the last commit

# Stashing
git stash                       # Temporarily store changes
git stash list                  # List stashes
git stash apply                 # Apply most recent stash
git stash drop                  # Remove most recent stash
git stash pop                   # Apply and remove stash
```

Recommended Resources

- Pro Git Book - Comprehensive guide to Git
- GitHub Documentation - GitHub-specific guides
- Oh Chord Sheet - Quick reference for Git commands
- Flask Documentation - Official Flask documentation
- Flask Mega-Tutorial - In-depth Flask tutorial series