Ready, Set, Deploy: Share Your Python Magic

Take Your Code from Local to Global

Michael Borck

Table of contents

Why Share Your Project? - Collaborate with others - Get feedback and contributions - Showcase your work

Using GitHub to Share Your Project

Why Use GitHub? - Version control with Git - Share code with the world - Collaborate on projects

Setting Up a GitHub Repository

Step-by-Step Guide 1. Create a GitHub account 2. Create a new repository 3. Clone the repository to your local machine 4. Add your project files 5. Commit and push your changes

Example: Git Commands

Initialize and Push to GitHub

```
# Initialize git in your project directory
git init

# Add your files to the repository
git add .

# Commit your changes
git commit -m "Initial commit"

# Add the remote repository URL
git remote add origin https://github.com/yourusername/yourrepository.git

# Push your changes to GitHub
git push -u origin master
```

Creating a README.md

Why Include a README.md? - Provide an overview of your project - Explain how to install and use it - Highlight key features and dependencies

Example: README.md

```
# Project Title

## Overview
Brief description of your project.

## Installation
```bash
pip install your_project
```

Usage

```
from your_project import your_function
result = your_function()
print(result)
```

Features

- Feature 1
- Feature 2

License

MIT
```

```
# Using PyInstaller

**What is PyInstaller?**
- Convert Python scripts into standalone executables
- No need for users to install Python or dependencies

**Installing PyInstaller**
```bash
pip install pyinstaller
```
```

Creating an Executable with PyInstaller

Simple Command to Create Executable

```
pyinstaller --onefile your_script.py
```

- Generates an executable in the `dist` directory
- Users can run the executable without installing Python

Packaging Your Python Project

Why Package Your Project? - Make it easy to distribute - Ensure dependencies are managed - Allow others to install it easily

Creating setup.py

Example: setup.py

```
from setuptools import setup, find_packages

setup(
    name="your_project",
    version="0.1.0",
    packages=find_packages(),
    install_requires=[
        "pandas",
        "numpy",
    ],
    entry_points={
        "console_scripts": [
            "your_command=your_module:main_function",
        ],
    },
)
```

Uploading to PyPI

Why Upload to PyPI? - Share your project with the Python community - Make it easy to install using pip

Steps to Upload 1. Register on PyPI 2. Build your package 3. Upload using twine

Example: Uploading to PyPI

Build and Upload Commands

```
# Install necessary tools
pip install setuptools wheel twine

# Build your package
python setup.py sdist bdist_wheel

# Upload to PyPI
twine upload dist/*
```

Summary

- Share your project on GitHub
- Use PyInstaller to create executables
- Package and upload to PyPI for easy distribution