**Python Packaging Formats**

**1. Source Distribution (.tar.gz)**

* The **original format** for distributing Python code.
* Contains your raw source code + setup.py.
* Installed using:
* pip install mypackage-0.1.tar.gz
* Advantage: Human-readable, portable.
* Disadvantage: Slower install (code must be built on user’s machine).

**Build Example:**

# Inside your package folder (with setup.py)

python setup.py sdist

➡ Creates dist/mypackage-0.1.tar.gz

**2. Egg File (.egg)**

* Introduced by **Setuptools** (older system).
* Similar to JAR in Java.
* Includes metadata + Python bytecode.
* Installed using easy\_install (now deprecated).
* Rarely used today — replaced by wheels.

**Build Example:**

python setup.py bdist\_egg

➡ Creates dist/mypackage-0.1-py3.egg

**3. Wheel (.whl)**

* **Modern standard** (recommended).
* Built distribution (binary).
* Faster install than source (tar.gz) because it doesn’t need to be built locally.
* Cross-platform (can be platform-specific if C extensions are included).

**Build Example:**

# Install wheel first

pip install wheel

# Build wheel

python setup.py bdist\_wheel

| **Format** | **Type** | **Usage Today** | **Pros** | **Cons** |
| --- | --- | --- | --- | --- |
| **tar.gz** | Source distribution | Still used | Portable, readable | Requires building on install |
| **egg** | Binary distribution (Setuptools) | Obsolete | Supported old tooling | Deprecated, not recommended |
| **wheel (.whl)** | Binary distribution | Standard | Fast install, pip-compatible | May need platform-specific builds |

**Docker vs Podman (Python Deployment Context)**

| **Feature** | **Docker** | **Podman** |
| --- | --- | --- |
| **Daemon** | Needs dockerd (runs as root by default) | Daemonless, rootless by default |
| **CLI Compatibility** | Widely used standard | Mostly Docker-compatible |
| **Security** | Root daemon can be security risk | Safer (runs as user) |
| **Systemd Integration** | Requires extra setup | Native support |
| **Kubernetes** | Needs docker-shim (deprecated in K8s v1.24+) | Podman generates Kubernetes YAML directly |
| **Python Workflow** | Works great (Dockerfile + docker run) | Same commands work; better for rootless deployments |

 Both **Docker & Podman** let you deploy Python services in containers.

 Docker = more popular, Podman = more secure (rootless, daemonless).

 Same workflow: **write Dockerfile → build → run**.

 From Python, use **docker SDK** or call Podman/Docker CLI

**Assignment Day 2:**

**Mini Project Question: Banking System Application**

**Project Title:**  
 *Console-based Banking System in Python*

**Problem Statement:**  
Design and implement a **Banking System application** that allows users to create accounts, perform transactions, and manage their banking operations. The system should use **variables, data types, functions, and file handling** to store and retrieve customer data.

**Requirements:**

1. **User Account Management**
   * Create a new account (Name, Account Number, Initial Balance, PIN).
   * Login using Account Number and PIN.
2. **Banking Operations**
   * Deposit money.
   * Withdraw money (check for sufficient balance).
   * Check balance.
   * Transfer money between accounts.
3. **File Handling**
   * Save all accounts and transactions to a file (accounts.txt).
   * Retrieve account details from the file when the program starts.
4. **Extra Features (Optional for bonus marks):**
   * Transaction history for each account.
   * Simple interest calculation on balance.
   * Close an account.

**Expected Outcome:**  
A user should be able to run the program, register or log in, and perform basic banking functions securely. Account data must persist across runs using files.