# Understanding Plugin Architecture for Python Packages

**Ashish Shukla**

Lead Software Engineer

EPAM Systems

# Overview

## How plugins work in Python

- What and Why Plugins ?
- Some Examples
- Plugin Loader from SQLAlchemy
- Python package entrypoints and importlib.metadata

## How you can make it work for your package

- Creating a hello world package
- Creating plugin structure and plugin loader utils
- Extending hello world package with hello-world-kannada-plugin

# What and Why Plugins ?

- What ?

> Plugins are a way to extend something.

- Why ?
    i. Keeping core package slim
    ii. Let the users decide what they want
    iii. Extensibility
    iv. Decouple core logic and implementations
    v. Get contributions !

# Some Examples

## SQLAlchemy Dialects

You can install additional dialects in SQLAlchemy like Clickhouse, GSheet, Druid etc. and extend the set of default supported dialects.

## Airflow Providers

You can extend airflow's supported operators, hooks, connections etc. by installing new providers. eg. Google, AWS etc.

# Plugin Loader from SQLAlchemy

```
from importlib import metadata as importlib_metadata

class PluginLoader:
    def __init__(self, group):
        self.group = group
        self.impls = {}

    def load(self, name):
        if name in self.impls:
            return self.impls[name]()

        entrypoints = importlib_metadata.entry_points()
        impls = entrypoints.select(group=self.group)
        for impl in impls:
            if impl.name == name:
                self.impls[impl.name] = impl.load
                return impl.load()

        raise ValueError("No such plugin!")
```

# Python Package Entrypoints and Importlib Metadata

## Entrypoints

(From Python packaging user guide]
(https://packaging.python.org/en/latest/specifications/entry-points/)

> Entry points are a mechanism for an installed distribution to advertise components it provides to be discovered and used by other code.

# Python Package Entrypoints and Importlib Metadata

## How to declare it in pyproject.toml

```
[project.entry-points.group_name]
name = path_to_object
```

## How to declare it in setup.py

```
entry_points={
    'group_name': [
        'key = path.to:Callable',
    ],
},
```

# How you can make it work for your package

## Using the same logic for our own plugins

1. Let others know how they can create code for you
    - Plugin Interface
2. Give them an identifier so that they can tell their code is for your package
    - Entrypoint Group Name
3. Write some code that can load `their code`
    - Plugin Loader
4. Use others' code !

# How you can make it work for your package

**Coding time !!**

# Conclusion

1. Plugins = Extensibility
2. Entrypoints = Python native way of implementing plugins
3. Result = Long Live your future package !

# References


1. SQLAlchemy - https://github.com/zzzeek/sqlalchemy
    i. Dialects
    ii. External Dialects List
    iii. Create Engine Implementation
2. Airflow - https://github.com/apache/airflow
3. Python Packaging Guide - https://packaging.python.org/en/latest/

# Thanks !

# Questions ?

Code and slides at:

https://github.com/ash2shukla/pycon2025

Connect with me:

https://linkedin.com/in/ash2shukla