Aspect-oriented Programming - Diving deep into Decorators

The aspect-oriented programming paradigm can support the separation of cross-cutting concerns such as logging, caching, or checking of permissions. This can improve code modularity and maintainability. Python offers decorator to implement re-usable code for cross-cutting tasks.

This tutorial is an in-depth introduction to decorators. It covers their usage and how to implement simple and more advanced solutions. Use cases demonstrate how to work apply aspect-oriented programming. In addition to showing how functions can use closures to create decorators, the tutorial introduces callable class instances as alternative. Class decorators can solve problems that used be to be tasks for metaclasses. The tutorial provides uses cases for class decorators.

While the focus is on best practices and practical applications, the tutorial also provides deeper insight how Python works behind the scene. After the tutorial participants will feel comfortable with functions that take functions and return new functions.

Audience

This tutorial is for intermediate Python programmers who want to dive deeper. Solid working knowledge of functions and classes basics is required.

Outline

Session 1

- Examples of using decorators (15 min)
  - from the standard library
  - from third-party packages
- Closures for decorators (10 min)
- Write a simple decorator (15 min)
- Best Practice (10 min)
- Use case: Caching (10 min)
- Use case: Logging (10 min) x * Exercise (20 min)

Break

Session 2

- Parameterizing decorators (10 min)
- Chaining decorators (5 min)
- Callable instances instead of functions (10 min)
- Use case: Argument Checking (10 min)
- Use case: Registration (10 min)
- Class decorators (15 min)

- Exercise (20 min)
- Wrap-up and questions (10 min)

Format

The tutorial will be hands on. While the students will receive a comprehensive PDF with all course content, I will not distribute pre-filled Notebooks. Instead, I will start with a blank Notebook for each topic and develop the content step-by-step. The participants are encouraged to type along. My typing speed is usually appropriate and allows participants to follow. In addition, the supplied PDF contains all needed code to get back on track, if I should be too fast. More complex example are also provide as Python source code files. I also explicitly ask for feedback if I am too fast or things are unclear. I encourage questions at any time. In fact, questions and my answers are often an important part of my teaching, making the learning experience much more lively and typically more useful.

So the participants will be active throughout the whole tutorial. In addition, there will be exercises that each participant has to do on her/his own (or in collaboration with a neighbor) during the tutorial. We will look at the solutions in the tutorial. I also supply a solutions PDF after the tutorial.

Software

You only need a current Python installation. There are no dependencies.