

This PDF is generated from: <https://www.kalelabellium.eu/Mon-28-Apr-2025-32427.html>

Title: Super large solar panel water pump

Generated on: 2026-03-12 00:10:02

Copyright (C) 2026 KALELA SOLAR. All rights reserved.

For the latest updates and more information, visit our website: <https://www.kalelabellium.eu>

`super()` lets you avoid referring to the base class explicitly, which can be nice. But the main advantage comes with multiple inheritance, where all sorts of fun stuff can happen.

"super" object has no attribute "`__sklearn_tags__`". This occurs when I invoke the `fit` method on the `RandomizedSearchCV` object. I suspect it could be related to compatibility ...

In fact, multiple inheritance is the only case where `super()` is of any use. I would not recommend using it with classes using linear inheritance, where it's just useless overhead.

As for chaining `super::super`, as I mentionned in the question, I have still to find an interesting use to that. For now, I only see it as a hack, but it was worth mentioning, if only for the differences ...

I wrote the following code. When I try to run it as at the end of the file I get this stacktrace: `AttributeError: "super" object has no attribute do_something` class `Parent`: `def ...`

`super()` is a special use of the `super` keyword where you call a parameterless parent constructor. In general, the `super` keyword can be used to call overridden methods, ...

What is the difference between `List<T>` `super T` and `List<T>` `extends T` ? I used to use `List<T>` `extends T`, but it does not allow me to add elements to it `list.add(e)`, whereas the `Li...`

The automatic insertion of `super ()` by the compiler allows this. Enforcing `super` to appear first, enforces that constructor bodies are executed in the correct order which would ...

Web: <https://www.kalelabellium.eu>

Super large solar panel water pump

Source: <https://www.kalelabellium.eu/Mon-28-Apr-2025-32427.html>

Website: <https://www.kalelabellium.eu>

