Django is popular among web developers because it takes less effort and time to set up, making it a good choice for quicker deployment and smaller projects. It is also particularly good at having a well made framework that makes it streamlined to be secure, fast, and readable/written with best practices in mind.

Companies that use Django

- Mozilla uses Django for their support.mozilla.org support site - They are a web browser.
- National Geographic uses Django for their main website and then builds their education website on Django CMS - They are an educational magazine.
- Udemy uses Django for their website because they have to handle huge amounts of content - They offer educational courses for tech and other job skills.
- Coursera also is built on Django due to the amount of content and its scalability - They are similar to udemy, but a B corp that offers courses and online degrees in tech and other job skills..
- Indeed uses Django to handle the large flow of postings, inquiries and responses - They are a job posting board.

Scenarios

- You need to develop a web application with multiple users.

There is not much to say for this one, because this is more or less any living web application. Django is totally prepared to be up to the task for building a web application and having scalability for handling just a few users all the way up to multitudes of users.

- You need fast deployment and the ability to make changes as you proceed.

This would probably depend on the nature of the changes that I expect to make. Django would be a great option due to its facilitation of a quick deployment through the amount of pre-coding in the background. It could be well suited because it will have the tools, scalability and community support to tackle changes that have to do with the implementing of new features or dealing with larger data flows. It's also particularly easy to add new components in Django. However, if the changes may require a lot of customization and trial and error with different approaches or things of that sort. Django will likely be too structured to deal with those sorts of changes.

- You need to build a very basic application, which doesn't require any database access or file operations.

I would not use Django for this because Django comes with a lot of things having to do with database interaction and file operations built in and it would be better to go with a framework or approach that would be more lightweight.

- You want to build an application from scratch and want a lot of control over how it works.

I would use something very lightweight like flask for this, which as I understand, would have the benefit of encouraging me to be thoughtful when choosing which or what additional tools I want to import into my ecosystem to use. I would not use Django for this as it seems to be very opinionated and there is a "Django way" of doing things that I wouldn't want to be constrained by in this scenario.

- You’re about to start working on a big project and are afraid of getting stuck and needing additional support.

I would definitely consider using Django for this because Django is open source and has an active and strong community. Plus, the benefit of something like writing in Python and in a known opinionated framework like the Django Way means that people who might be supporting me are more likely to be able to understand my code or be able to help me by pointing to the exact Django Way of doing what I am trying to do.