Environment Variables

Environment variables allow developers to customize the environment in which an application is operating.

Benefits

Security

Sometimes applications need to reference secret passwords, tokens, and other values. This is especially the case when the app is authenticating to some other service on behalf of a given user.

But hard-coding these sensitive values into a program's source code would be irresponsible from a security standpoint, especially when sharing the source code online. You don't want your passwords on GitHub for everyone to see.

Environment variables provide a way of separating these secret values from a program's source code.

Collaboration and Customization

Sometimes developers need to run an application in different environments. For example, two developers may want to use the same program to download their respective social media posts from an API.

But each developer has their own private API key which provides access to their own private social media posts. It would be ineffective for them to use the exact same source code, and inefficient to maintain two slightly different versions of the application's source code.

Environment variables allow developers to share the same source code while specifying different values at run-time.

Testing and Delivery

Environment variables allow developers to specify customized environments in which to develop, test and deliver their application.

Environment variable customization allows an application to perform differently in a "test" environment than it would in a user-facing "production" environment. For example, developers can use the application to manipulate example data in a "test" environment without affecting real user data in the "production" environment.

Usage

Setting

Environment variables can be set "globally", in which case they are accessible by any program running on that given computer. Or they can be set "locally", in which case they are only accessible by programs located in a specific directory.

After setting an environment variable using one of the approaches below, reference the section on "Getting" to see if the variable was set properly.

Setting on Mac or Git Bash

Mac users (or Git Bash users on Windows) should be able to manage global environment variables using a hidden file called "~/.bash_profile". Open the file with your text editor (e.g. `code ~/.bash_profile`), and place inside the following contents:

```
# ~/.bash_profile
export MY_SECRET_MESSAGE="SecretPassword123"
# or ...
# export MY_SECRET_MESSAGE=SecretPassword123
```

Then exit and re-open your Terminal for the changes to take effect.

Setting on Windows

Windows users can set local environment variables from the command-line using the `set` keyword:

```
# Windows Command Prompt:
set MY_SECRET_MESSAGE="SecretPassword123"
# or ...
# set MY_SECRET_MESSAGE=SecretPassword123
```

> NOTE: if you close your command prompt and re-open it, you will need to re-set the environment variable.

Setting Locally via the Command Line

To set a script-specific environment variable on either Mac or Windows, its possible to prefix the environment variable before invoking your Python script. For example:

```
MY_SECRET_MESSAGE="SecretPassword123" python path/to/my_script.py
```

To access environment variables from within a Python program, use the `os` module.

Setting Locally Using Dotenv File

To set project-specific local environment variables on either Mac or Windows, consider using the "dotenv" approach. Create a special file in your project named ".env" and place inside content like the following:

```
# my-secure-project/.env

MY_SECRET_MESSAGE="SecretPassword123"
```

To load these variables from the ".env" file into a Python program, use the `dotenv` package, then access them using the `os` module.

Getting

You will know you have successfully set an environment variable when you can access its value from the command-line:

```
# Mac Terminal:
echo $MY_SECRET_MESSAGE #> SecretPassword123

# Windows Command Prompt:
echo %MY_SECRET_MESSAGE% #> SecretPassword123
```

To access environment variables from within a Python program, use the `os` module.