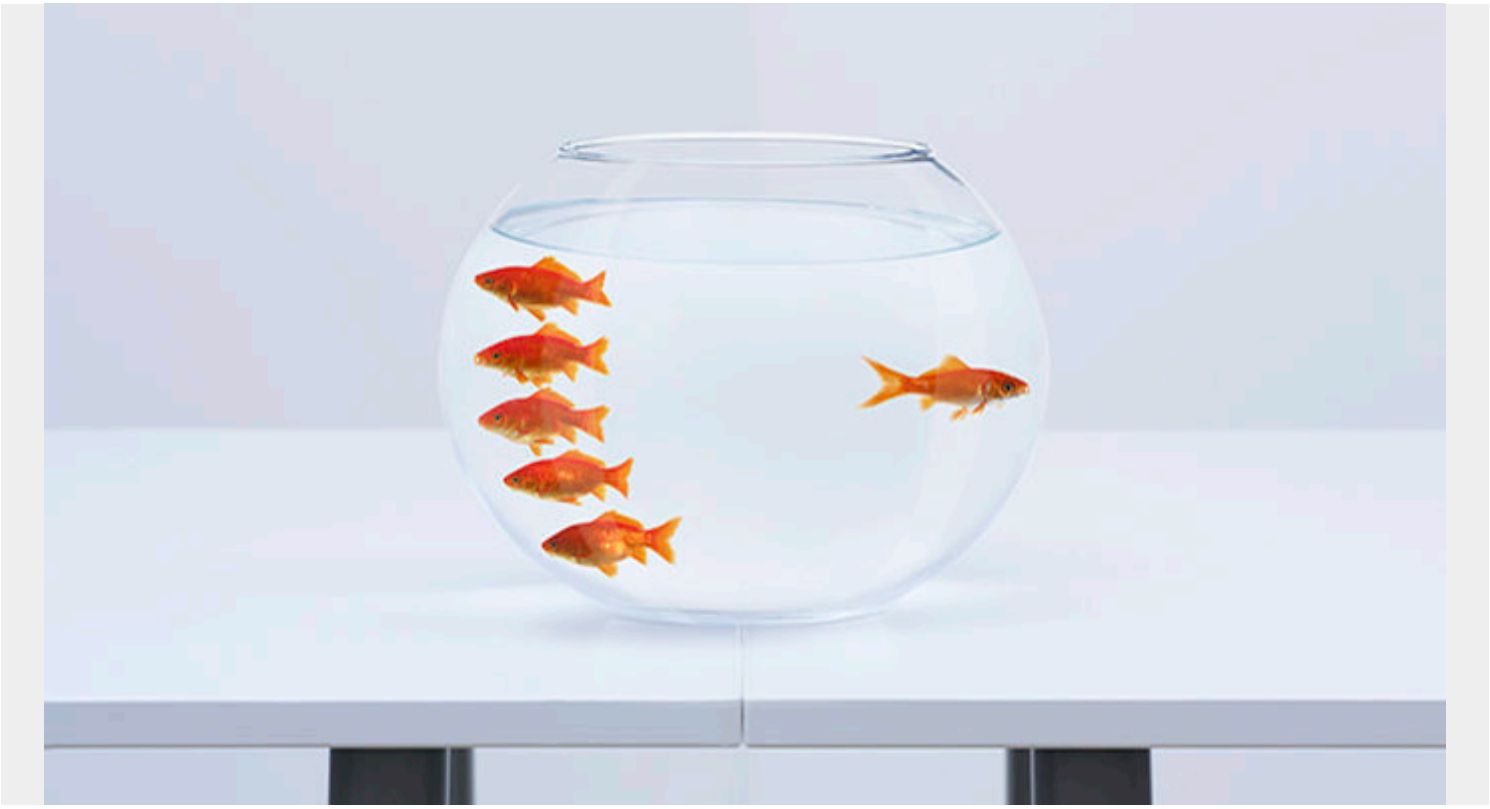
# ASYNCHRONOUS PROGRAMMING: A BEGINNER'S GUIDE



Asynchronous programming helps a user flow smoothly through an application. Let's take a look at this programming practice with:

- Definitions
- Workflows
- Functions
- Use cases
- When to use it
- And more!

## What is asynchronous programming?

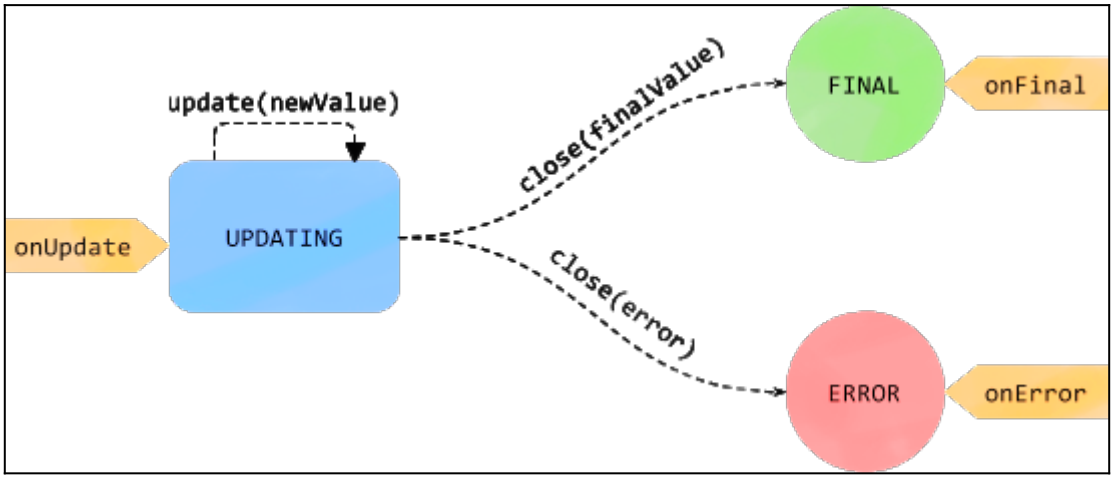
It takes time for a function to fetch data from an API. Asynchronous programming was devised to accommodate for the lag between when a function is called to when the value of that function is returned.

Without asynchronous programming, apps would spend a long time on loading screens. A loading screen might appear:

- When a user signs in, waiting for all their user data to be returned from the database.
- During the user experience, waiting for the data to load at each new screen.

Asynchronous programming allows a user to go about his business in an application, while

processes run in the background, thus enhancing the user experience.




(Source)

Here's an example: Data may take long a long time to submit to a database. With asynchronous programming, the user can move to another screen while the function continues to execute. When a photo is loaded and sent on Instagram, the user does not have to stay on the same screen waiting for the photo to finish loading. The user can continue in the app or leave the app while the photo loads.

# How asynchronous programming works

The easiest way to see how asynchronous programming works is to compare it to synchronous programming. And, we'll use a yummy example.

## Synchronous programming

Synchronous programming follows a strict set of sequences. When the code runs in a synchronous program, it will follow each step of an algorithm. It does so in order and will wait for the present operation to finish before continuing on to the next.

Synchronous programming follows a "Bake a cake" algorithm.

1. Measure the ingredients.
2. Mix flour, eggs, and sugar.
3. Heat oven and bake.
4. Eat.

Each step must happen in order. The ingredients must be measured, mix must be mixed, before the mix is baked. And, to taste like a cake, it should be baked before it is eaten. Because only one person is doing all the work, you must complete one task fully before starting the next.

Synchronous programming has a one-track mind. It follows the guide step by step.

## Asynchronous programming

Asynchronous cake baking, by contrast, allows multiple people to be working on the task at once. One person can gather and measure the ingredients while another person begins mixing the ingredients together. Asynchronous programming allows multiple processes to be started, lets the

processes do their work, and when their job is finished, it gets the result and puts it through the steps.

If the oven finishes heating before the cake mix is fully prepared, asynchronous programming says that is okay. Synchronous programming would never have started the oven without the mix having been prepared. When the mix is completed, it sends an update to the algorithm to come back and pick up the result of the mix and push it through the process. Now, when the cake mix is prepared, it can be passed into a heated oven that is already heated to the right temperature, ready to bake the cake.

Unfortunately, asynchronous programming won't help you eat your cake, but it will help get the cake down the line faster. The baking must still happen before you can eat it. (And, if the eater is called to eat before the cake is ready, like how the oven was heating before the cake mix was ready, the one eating can pace the kitchen hungrily.)

## Asynchronous functions

Asynchronous functions are often found in front end applications and used particularly in independent, high volume, IO tasks. Front end applications benefit from its use because it enhances the flow of an application.

Backend processes may use asynchronous functions to run many tasks or make lots of network calls. In the backend, asynchronous programming allows the computer to do more, faster. It calls a lot of functions whose response times are indefinite and processing the results.

An example is web scraping, then storing the result in a database: the process is routine, and it doesn't matter what order the results get written to the directory—they just need to have a file name.

Asynchronous programming exists in:

- Java
- JavaScript
- Typescript
- Dart

The typical function is written with the async/await combo.

```
async function foo() {
    const value = await somePromise();
    return value;
}
```

## Common use cases

The most common use of the asynchronous function is to make a call to an API. Because network times and retrievals are uncertain, asynchronous functions say, "Get me the data from a website (or REST API), and when it gets here, insert that fetched data back into my script."

Async functions are used to:

- Interact with an API
- Slow down an application's UX

They can also be used to create delays in a user's activity. Why would you want to slow down an app? Because computers can do things incredibly fast, and, when executed, it is jarring to a user.

So, designers intentionally slow down the application. A message can be sent almost instantly to another user. Often, the loading circle is not a necessary circle saying the message is taking time to send. Instead, it's there because it helps a user understand what is going on and feel more comfortable using the app.

Yes, sometimes it takes a message a couple seconds to send because of network delays. Encoding a message will also take a little bit of time to happen on a user's device before it is sent over a network.

Items on the screen can appear and disappear instantly, and it is through animations that helps a user follow what is happening on the screen. Animations can be asynchronous because, while they perform their operation over a period of time, other functions can be operating in the background.

## When to use asynchronous functions

Asynchronous is not always the best way to go. Asynchronous programs add more complexity and make the code more unreadable. Young programmers will often use async functions too much because they think it acts as a safeguard to ensure their code works at run-time. A general rule for when to use async functions:

- **Good for:** Tasks that may take a while; high iteration.
- **Bad for:** Simplicity.

Max Galka, mapping founder of blueshift, said it best:

> *Asynchronous loops are necessary when there is a large number of iterations involved or when the operations within the loop are complex. But for simple tasks like iterating through a small array, there is no reason to overcomplicate things by using a complex recursive function. A simple synchronous for/while loop works just fine, and will also be faster and more readable.*

## Additional resources

For more on this topic, explore these resources:

- BMC DevOps Blog
- What is Systems Programming?
- What Is Pair Programming?
- Resilience Engineering: An Introduction
- Python Development Tools: Your Python Starter Kit