

```
import time

from rich.progress import Progress

with Progress() as progress:

    task1 = progress.add_task("[red]Downloading...", total=1000)
    task2 = progress.add_task("[green]Processing...", total=1000)
    task3 = progress.add_task("[cyan]Cooking...", total=1000)

    while not progress.finished:
        progress.update(task1, advance=0.5)
        progress.update(task2, advance=0.3)
        progress.update(task3, advance=0.9)
        time.sleep(0.02)
```

The `total` value associated with a task is the number of steps that must be completed for the progress to reach 100%. A *step* in this context is whatever makes sense for your application; it could be number of bytes of a file read, or number of images processed, etc.

16.2.1 Starting and stopping

The context manager is recommended if you can use it. If you don't use the context manager, be sure to call `start()` to start the progress display, and `stop()` to stop it.

Here's an example that doesn't use the context manager:

```
import time

from rich.progress import Progress

progress = Progress()
progress.start()

try:
    task1 = progress.add_task("[red]Downloading...", total=1000)
    task2 = progress.add_task("[green]Processing...", total=1000)
    task3 = progress.add_task("[cyan]Cooking...", total=1000)

    while not progress.finished:
        progress.update(task1, advance=0.5)
        progress.update(task2, advance=0.3)
        progress.update(task3, advance=0.9)
        time.sleep(0.02)
finally:
    progress.stop()
```

Note the use of the `try / finally`, to ensure that `stop()` is called.

16.2.2 Updating tasks

When you call `add_task()` you get back a *Task ID*. Use this ID to call `update()` whenever you have completed some work, or any information has changed. Typically you will need to update `completed` every time you have completed a step. You can do this by setting `completed` directly or by setting `advance` which will add to the current `completed` value.