progrock

Release 0.3.1

October 31, 2014

Contents

A multi-progressbar implementation to complement multiprocessing.Process.

CHAPTER 1

Installation

progrock is available on the Python Package Index and can be installed via `pip` or `easy_install`:

```
pip install
```

CHAPTER 2

Requirements

There are no requirements outside of the Python standard library.

CHAPTER 3

Screenshot

The following image shows the example code listing in action:

CHAPTER 4

API Documentation

4.1 API

The `progrock.MultiProgress` class is used in conjunction with the methods exposed at the module level such as `progrock.increment()` to create a full-screen experience allowing the user to track the progress of individual processes as they perform their work.

This module is meant as a complement to `multiprocessing.Process` and provide an easy to use, yet opinionated view of child process progress bars.

class `progrock.`**`MultiProgress`**(*title=None*, *steps=None*, *value=0*)

The MultiProgress class is responsible for rendering the progress screen using curses. In addition, it can wrap the creation of processes for you to automatically pass in the `multiprocessing.Queue` object that is used to issue commands for updating the UI.

If you do not pass in a `title` for the application, the Python file that is being run will be used as a title for the screen.

If you pass in `steps`, a progress bar will be centered in the footer to display the overall progress of an application. The bar can be incremented from the parent process using `MultiProgress.increment_app()` or if you're incrementing from a child process, you can call `progrock.increment_app()` passing in `ipc_queue`.

Parameters

- **title** (*str*) – The application title
- **steps** (*int*) – Overall steps for the application
- **value** (*int*) – Overall progress value for the application

`add_process`(*process*, *status='Initializing'*, *steps=100*, *value=0*)

Add a process to the MultiProgress display. The process must already have been started proior to invoking this method.

Parameters

- **multiprocessing.Process** – The process to add
- **status** (*str*) – The status text for the process box
- **steps** (*int|float*) – The number of steps for the progress bar
- **value** (*int|float*) – Current progress value for the process

`increment_app`(*value=1*)

If using the application progress bar, increment the progress of the bar.

Parameters **value** (*int*) – The value to increment by. Default: 1

initialize()
Initialize the MultiProgress screen. Should only be invoked if not using the MultiProgress instance as a context manager. If the instance MultiProgress instance is used as a context manager, this is done automatically.

new_process(*target*, *name=None*, *args=None*, *kwargs=None*, *status='Initializing'*, *steps=100*, *value=0*)
Create and start new multiprocessing.Process instance, automatically appending the update queue to the positional arguments passed into the target when the process is started. Once the process is created, it is added to the stack of processes in MultiProgress, bypassing the need to invoke MutiProgress.add_process().

Parameters

- **target** (*method*) – The method to invoke when the process starts
- **name** (*str*) – Process name
- **args** (*tuple*) – Positional arguments to pass into the process
- **kwargs** (*dict*) – Keyword arguments to pass into the process
- **status** (*str*) – The status text for the process box
- **steps** (*int|float*) – The number of steps for the progress bar
- **value** (*int|float*) – Current progress value for the process

Returns multiprocessing.Process

shutdown()
Shutdown MultiProgress screen. Must be called if the MultiProgress instance is not being used as a context manager. If the instance MultiProgress instance is used as a context manager, this is done automatically.

progrock.**increment**(*ipc_queue*, *value=1*)
Increment the progress value for the current process, passing in the queue exposed by MultiProgress.ipc_queue and automatically passed into the target function when creating the process with MultiProgress.new_process.

Parameters

- **ipc_queue** (*multiprocessing.Queue*) – The IPC command queue
- **value** (*int*) – The value to increment by. Default: 1

progrock.**increment_app**(*ipc_queue*, *value=1*)
Increment the progress value for the application, passing in the queue exposed by MultiProgress.ipc_queue.

Parameters

- **ipc_queue** (*multiprocessing.Queue*) – The IPC command queue
- **value** (*int*) – The value to increment by. Default: 1

progrock.**reset_start_time**(*ipc_queue*)
Restart the start time of a process, passing in the queue exposed by MultiProgress.ipc_queue and automatically passed into the target function when creating the process with MultiProgress.new_process.

Parameters **ipc_queue** (*multiprocessing.Queue*) – The IPC command queue

progrock.**reset_value**(*ipc_queue*)
Reset the progress value for the current process, passing in the queue exposed by `MultiProgress.ipc_queue` and automatically passed into the target function when creating the process with `MultiProgress.new_process`.

Parameters **ipc_queue** (*multiprocessing.Queue*) – The IPC command queue

progrock.**set_app_step_count**(*ipc_queue*, *steps*)
Set the number of steps for the application, passing in the queue exposed by `MultiProgress.ipc_queue`.

Parameters

- **ipc_queue** (*multiprocessing.Queue*) – The IPC command queue
- **steps** (*int*) – The number of steps for the application.

progrock.**set_status**(*ipc_queue*, *status*)
Set the status of current process, passing in the queue exposed by `MultiProgress.ipc_queue` and automatically passed into the target function when creating the process with `MultiProgress.new_process`.

Parameters

- **ipc_queue** (*multiprocessing.Queue*) – The IPC command queue
- **status** (*str*) – The status text for the current process

progrock.**set_step_count**(*ipc_queue*, *steps*)
Set the number of steps for current process, passing in the queue exposed by `MultiProgress.ipc_queue` and automatically passed into the target function when creating the process with `MultiProgress.new_process`.

Parameters

- **ipc_queue** (*multiprocessing.Queue*) – The IPC command queue
- **steps** (*int*) – The number of steps for the current process

progrock.**set_value**(*ipc_queue*, *value*)
Set the progress value for the current process, passing in the queue exposed by `MultiProgress.ipc_queue` and automatically passed into the target function when creating the process with `MultiProgress.new_process`.

Parameters

- **ipc_queue** (*multiprocessing.Queue*) – The IPC command queue
- **value** (*int*) – The value to set for the process

4.2 Examples

The following example will create a process for each CPU core on the system that it is run on, displaying the `progrock.MultiProgress` screen, using `progrock.MultiProgress` as a context manager. The child processes will iterate 100 times, updating their progress bar and then sleeping up to 1 second.

```
import multiprocessing
import progrock
import random
import time

def example_runner(ipc_queue):
    # Update the processes status in its progress box
```

```
    progrock.set_status(ipc_queue, 'Running')

    # Increment the progress bar, sleeping up to one second per iteration
    for iteration in range(1, 101):
        progrock.increment(ipc_queue)
        time.sleep(random.random())

processes = []

# Create the MultiProgress instance
with progrock.MultiProgress('Example') as progress:

    # Spawn a process per CPU and append it to the list of processes
    for proc_num in range(0, multiprocessing.cpu_count()):
        processes.append(progress.new_process(example_runner))

    # Wait for the processes to run
    while any([p.is_alive() for p in processes]):
        time.sleep(1)
```

This example performs the exact same tasks as the previous one, however it does not use `progrock.MultiProgress` as a context manager. In this example you will notice that the screen must be initialized on startup and shutdown when done.

```
import multiprocessing
import progrock
import random
import time

def example_runner(ipc_queue):
    # Update the processes status in its progress box
    progrock.set_status(ipc_queue, 'Running')

    # Increment the progress bar, sleeping up to one second per iteration
    for iteration in range(1, 101):
        progrock.increment(ipc_queue)
        time.sleep(random.random())

processes = []
cpu_count = multiprocessing.cpu_count()

# Create the MultiProgress instance
progress = progrock.MultiProgress('Example', cpu_count)

# Initialize the screen
progress.initialize()

# Spawn a process per CPU and append it to the list of processes
for proc_num in range(0, cpu_count):
    processes.append(progress.new_process(example_runner))
    progress.increment_app()
    time.sleep(random.random())

# Wait for the processes to run
while any([p.is_alive() for p in processes]):
    time.sleep(1)

# Shutdown the screen
```

```
progress.shutdown()
```

CHAPTER 5

Version History

See `history`

CHAPTER 6

Issues

Please report any issues to the Github project at https://github.com/gmr/progrock/issues

CHAPTER 7

Source

progrock source is available on Github at https://github.com/gmr/progrock

CHAPTER 8

License

progrock is released under the 3-Clause BSD license.

CHAPTER 9

Indices and tables

- *genindex*
- *modindex*
- *search*

p

A

I

M

N

P

R

S