# Soldier Documentation

***Release 0.1***

**Yash Mehrotra**

**Jul 08, 2023**

# Contents

Contents:

CHAPTER 1

# Overview

soldier is a library designed for executing and managing system processes with ease.

It is written on top of the subprocess module and has a much user-friendly API compared to subprocess.

With soldier, you can execute **shell commands**, get their output, status code and stderr

You can even run **processes** through soldier where you can set their timeouts, start them synchronously or asynchronously (and kill them whenever you want), and you can even execute root commands.

CHAPTER 2

# Installation

The installation is the simplest part.

```
$ pip install soldier
```

or you can install from source:

```
$ pip install git+https://github.com/yashmehrotra/soldier.git
```

# CHAPTER 3

# Usage

**Philosophy**: The reason for creating soldier is to provide a user-friendly and pythonic API on top the subprocess module in order to manage system processes in python.

After successfully installing soldier, you will now be able to communicate with system processes without making your eyes bleed.

**Get output of pwd command**

```
>> import soldier
>> current_path = soldier.run('pwd')
>> print(current_path.output)
/home/pythonista
# Status Code
>> print(current_path.exit_code)
0
```

**Run a process in background, and later terminate it**

```
>> firefox_proc = soldier.run('firefox', background=True)
# Get pid of firefox process
>> print(firefox_proc.pid)
18673
# Kill firefox
>> firefox_proc.kill()
>> firefox_proc.is_alive()
>> False
```

**Run a root command**

```
>> soldier.run('service nginx start', sudo='my_password')
```

**Run a command with timeout**

```
>> soldier.run('./infinite_loop_script.py', timeout=10, kill_on_timeout=True)
```

(continues on next page)

(continued from previous page)

```
# Handle timeouts
>> try
       soldier.run('./my_script.py', timeout=5, kill_on_timeout=False)
   except soldier.ProcessTimeoutError:
       print("Your script timed out")
Your script timed out
```

# CHAPTER 4

# API Documentation

soldier.**run**(*command*, *background=False*, *std_id=''*, *sudo=None*, *timeout=0*, *kill_on_timeout=False*, *stream=False*, *suppress_std_err=False*, *shell=False*)

The main run command which executes the system process

**Parameters**

- **background** (*bool*) – Set this true if you want to run the command asynchronously
- **std_in** (*string*) – The standard input to be given to the process
- **sudo** (*string*) – If you want to execute the command as root, this argument should be your password
- **cwd** (*string*) – Working directory of the command being executed
- **env** (*dict(str, str)*) – Environment variables that would be available when the command is being executed
- **stream** (*bool*) – When set to true, the output of your command will be streamed. (It does not work with piped commands)
- **suppress_std_err** (*bool*) – When set to true, the output from stderr would not be printed.
- **timeout** (*int*) – The timeout for the process in seconds
- **kill_on_timeout** (*bool*) – If set to true, your process will killed when the time is up, and if it is False, it will throw a `soldier.ProcessTimeoutError`
- **shell** (*bool*) – Set this to true if you want to execute the process in the /bin/sh environment

**Returns** A *soldier.Soldier* object

**Return type** *soldier.Soldier*

**Warning:** Passing `shell=True` can be a security hazard if combined with untrusted input.

**class** soldier.**Soldier**

The class object which is returned with the *soldier.run()* method.

**Methods**

**kill**()

This function is used to kill the current process

**Return type** None

**is_alive**()

This function checks whether process is active or not

**Returns** A `bool` specifying whether the process is running or not

**Return type** `bool`

**Properties**

- pid - Returns the pid of the process
- exit_code - Returns the exit code of the process
- status_code - Returns the exit code of the process (To be deprecated, prefer exit_code)
- output - Returns the stdout (standard output) of the process
- error - Returns the stderr (standard error) of the process
- start_ts - Returns the start time (`datetime.datetime` object) of the process
- end_ts - Returns the end time (`datetime.datetime` object) of the process
- duration - Returns the total duration(`datetime.timedelta` object) of the process

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search

# Index

## S