# animation Documentation

***Release 0.0.6***

**Blake Printy**

**Dec 06, 2018**

# Contents


**1 Content:** **1**

1.1 Installation . . . . . 1

1.2 Usage . . . . . 1

1.3 API . . . . . 2

**2 Indices and tables** **5**

CHAPTER 1

Content:

## 1.1 Installation

### 1.1.1 Through pip

```
$ pip install animation
```

### 1.1.2 Via GitHub

```
$ git clone http://github.com/bprinty/animation.git
$ cd animation
$ python setup.py install
```

## 1.2 Usage

The animation module provides decorators for doing terminal-based wait animations. To add a wait animation to a function that requires some processing time, simply decorate the function with the wait animation you want to use.

Here is an example of how to use it in a project:

```
import animation
import time

@animation.simple_wait
def long_running_function():
    ... 5 seconds later ...
    return
```

This will print an animated waiting message like this (the elipses at the end of the text grow and shrink while the function executes):

```
waiting ...
```

The animation types provided by default are:

- bar (simple bar that slides back and forth)
- spinner (a spinning line)
- dots (dots that move around in a sqare)
- elipses (elipses that grow and shrink)
- text with elipses (elipses with text in front of them)

And you can use any of these built-in animations like so:

```
import animation
import time

@animation.wait('bar')
def long_running_function():
    ... 5 seconds later ...
    return

@animation.wait('spinner')
def long_running_function():
    ... 5 seconds later ...
    return
```

In addition to these default types, the module also supports custom animations. For example, to create an animation with a counter-clockwise spinning wheel:

```
wheel = ('-', '/', '|', '\\')
@animation.wait(wheel)
def long_running_function():
    ... 5 seconds later ...
    return
```

If you want to manually start and stop the wait animation, you can use the `animation.Wait` class:

```
wait = animation.Wait()
wait.start()
long_running_function()
wait.stop()
```

### 1.2.1 Questions/Feedback

File an issue in the GitHub issue tracker.

## 1.3 API

`animation.`**`simple_wait`**(*func*)

Decorator for adding simple text wait animation to long running functions.

### Examples

```
>>> @animation.simple_wait
>>> def long_running_function():
>>>     ... 5 seconds later ...
>>>     return
```

animation.**wait**(*animation='elipses'*, *text=''*, *speed=0.2*)

Decorator for adding wait animation to long running functions.

**Parameters**

- **animation** (*str, tuple*) – String reference to animation or tuple with custom animation.
- **speed** (*float*) – Number of seconds each cycle of animation.

### Examples

```
>>> @animation.wait('bar')
>>> def long_running_function():
>>>     ... 5 seconds later ...
>>>     return
```

**class** animation.**Wait**(*animation='elipses'*, *text='waiting'*, *speed=0.2*)

Class for managing wait animations.

**Parameters**

- **animation** (*str, tuple*) – String reference to animation or tuple with custom animation.
- **text** (*str*) – Optional text to print before animation.
- **speed** (*float*) – Number of seconds each cycle of animation.

### Examples

```
>>> animation = Wait()
>>> animation.start()
>>> long_running_function()
>>> animation.stop()
```

**start**()

Start animation thread.

**stop**()

Stop animation thread.

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search

# Index

## S

simple_wait() (in module animation), 2
start() (animation.Wait method), 3
stop() (animation.Wait method), 3

## W

Wait (class in animation), 3
wait() (in module animation), 3