Barnacle Documentation

Release 0.8.1

Paul van Gent

May 14, 2019

Index

Barnacle offers fun and customisable progress bars for your Python programs. Use them to fun-up all of your and others' Python applications that require time to execute. It started off as a Keras plugin (and it still is!) to liven up my deep learning sessions, but has since become more general in scope.

Get your own Barnacle today!

We have zombies:

Gib:

Sunglasses:

We hate tables:

Tables hate us:

We LOVE Keras:

Keras Plugin included

CHAPTER 1

What else?

Many more presets are available and Barnacle offers the flexibility to design your own as well. Barnacle aims to be a simple rendering engine for progress bars of all types. See the docs (coming soon) on how to easily design your own custom progress bars.

So shut up and take my money already!

(Except it's free forever!!1!)

* of course we also have customisable regular progress bars (but you don't really care about that, do you?)

1.1 Quickstart

1.1.1 Installation

github

Download latest release here

```
python setup.py install
```

1.1.2 Basic Example

The package is easy to use. Load the module, select a preset, and it's good to go.

```
import barnacle #import module

bar = barnacle.random() #either initialize bar object with a random animation
bar = barnacle.load('zombie') #or with a specific preset (see docs for full list)

for i in range(0, 101): #whatever loop you run
        bar.draw(i, 100) #give it current step, total steps, and the bar draws itself.
        #some time consuming task here
```

`draw` expects two arguments:

- *currentstep*: The current step that needs to be drawn
- *totalsteps*: The total steps expected to be taken

Alternatively you can also get the bar as a string, in case you want more control:

```
for i in range(0, 101): #whatever loop you run
        bar_string = bar.update_bar(i, 100) #give it current step, total steps, and␣
↪the bar draws itself.
        #now you are free to do anything you want with the string object, print it,␣
↪eat it, cook it, whatever!

        #some time consuming task here
```

1.1.3 Keras Plugin

Barnacle started off as a Keras plugin to make my long model fitting hours more bearable. This functionality is still available. Usage is also simple:

```
import keras #import keras first
from barnacle import barnacle_keras #import barnacle keras plugin

#you can select a preset and mode
barnacle_keras.Progbar.preset = 'tableflip' #see docs for full list of presets,
↪'random' for random
barnacle_keras.Progbar.random = True #whether to select a random progress bar every␣
↪epoch

#override Keras progbar class in memory
keras.callbacks.Progbar = barnacle_keras.Progbar

#build your model and fit as you usually would
```

1.1.4 Presets

Find out what presets are available:

```
import barnacle
#what presets are included?
barnacle.print_presets()
```

1.2 Designing a custom progress bar

Barnacle offers the ability to easily design your own pogress bar. This section of the documentation details how this works.

1.2.1 Types of bars available

Several types of progress bars are available, and each can be customised. The types are:

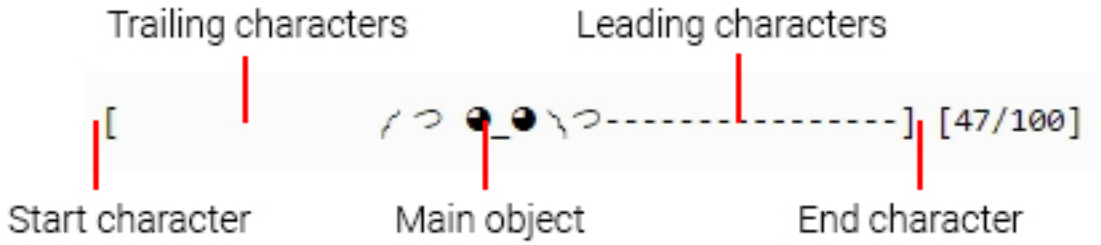
- Simple objectslider
- Animated objectslider
- Interaction objectslider
- Textscroller
- Standard progressbar

1.2.2 Simple objectslider

The *class: simple_objectslider* that will slide a defined object across the progress bar, with settable leading and trailing characters and possible animations. As an example of the simplest form, consider this preset:

Example 1:

This bar is built from several parts:



To define this progress bar, a standardised `dict{}` object is used (the 'start' and 'end' characters are left to their default values:

```
{'icons': ['  _ ', '  _ ', ' ', '-']}
```

This dictionary contains a list of icons from which the progress bar will be built up. The icons need to be in the correct order, which is:

- `icons`: a list containing four elements at fixed indices:

0. the object to slide across the progress bar
1. the end-state of the object (what it looks like when it reaches the end of the progress bar)
2. trailing character: what to draw behind the object, use ' ' for blank
3. leading character: what to draw in front of the object, use ' ' for blank

So to use this progress bar, simply do:

```
import barnacle
import time

bar = barnacle.simple_objectslider()

custom_bar = {'icons': ['  _ ', '  _ ', ' ', '-']}
bar.load(custom_bar)

for i in range(0,51):
        bar.draw(i, 50) #update bar each step
        time.sleep(0.2) #simulate time consuming loop
```

Example 2:

As an example of the extra arguments in the dictionary, consider this preset:

Several things are happening here:

- The dude moving across the bar sometimes whistles
- A table is at the end of the bar
- When the dude reaches the end of the bar the table is flipped and the dude throws up his arms
- For example's sake let's use a custom bar terminator and custom trailing character

The preset dict is defined as:

```
{'icons': ['( °-°) ', '(°°)', '.', ' ', '( °-°)♪'],
'target_icons': ['    ', ' ~~  '], 'extra_odds': 0.2,
    'start': '<', 'end': '>'}
```

Several arguments are in the dict:

- just like before, `icons`: a list containing four elements at fixed indices:

0. the object to slide across the progress bar
1. the end-state of the object (what it looks like when it reaches the end of the progress bar)
2. trailing character: what to draw behind the object, here we use '.' for the example's sake
3. leading character: what to draw in front of the object, use ' ' for blank
4. **extra**: the 'extra_icon', the dude whistling

- `target_icons`: list containing two objects to be placed at the end of the progress bar. The elements are:

0. target object default state: while the bar is progressing this is drawn (here: normal table)
1. target end state, when the bar reaches the last step, this is drawn in stead of the 0th index

- `extra_odds`: the odds that the last element in 'icons' is drawn when the :draw(): method is called (here: dude whistling). Needs to be 0 <= extra_odds <= 1.0, with 0.5 representing a 50% chance each draw update.
- `start`: the start character of the bar
- `end`: the end character of the bar

Full example:

```
import barnacle
import time

bar = barnacle.simple_objectslider()

custom_bar = {'icons': ['( °-°) ', '(°°)', '.', ' ', '( °-°)♪'],
                          'target_icons': ['     ', ' ~~  '], 'extra_odds': 0.2,
                          'start': '<', 'end': '>'}
bar.load(custom_bar)

for i in range(0,51):
        bar.draw(i, 50) #update bar each step
        time.sleep(0.2) #simulate time consuming loop
```

1.2.3 Docs on other bar types coming ASAP