class `patiencebar.`**`Patiencebar`**`(`*valmax=100, barsize=None, title=None, bar=True, up_every=2*`)`

Bases: `object`

Provides a terminal-friendly single-thread progress bar

Args:

- valmax (float): the finish value of the progress bar. Default is 100.
- barsize (int >0): the size of the bar in the opened terminal. If `None`, the bar will automatically fit the width of the window.
- title (str): the title, printed one line above the progress bar
- bar (bool): whether the bar should be displayed or not. If `False`, only the text given at each *`update()`* will be printed
- up_every (int [0-100]): if `bar` is `True`, the progress bar will be updated every `up_every` percent of progress. Setting `up_every` = 0 updates the progress bar at each *`update()`*

```
>>> import patiencebar as PB
>>> n_calc = 34
>>> pb = PB.Patiencebar(valmax=n_calc, barsize=50, title="Test bar")
>>> for i in range(n_calc):
>>>     do_stuff()
>>>     pb.update()
```

`bar`

`barsize`

`reset``(`*valmax=None, barsize=None, title=None, bar=None, up_every=None*`)`

Resets the progress bar with initialization values, unless new values are given

Args:

- valmax (float): the finish value of the progress bar. Default is 100.
- barsize (int >0): the size of the bar in the opened terminal. If `None`, the bar will automatically fit the width of the window.
- title (str): the title, printed one line above the progress bar.
- bar (bool): whether the bar should be displayed or not. If `False`, only the text given at each *`update()`* will be printed.
- up_every (int [0-100]): if `bar` is `True`, the progress bar will be updated every `up_every` percent of progress. Setting `up_every` = 0 updates the progress bar at each *`update()`*.

```
>>> import patiencebar as PB
>>> n_calc = 34
>>> pb = PB.Patiencebar(valmax=n_calc, barsize=50, title="Test bar")
>>> for i in range(n_calc):
>>>     do_stuff()
>>>     pb.update()
>>> pb.reset(title="Second trial", barsize=70)
>>> for i in range(n_calc):
>>>     do_stuff()
>>>     pb.update()
```

`running`

`title`

`up_every`

update (*step=None*)

Updates the progress bar to a newer value

Args:

- step (None): adds 1 to the progress of the bar, where `valmax` is the finish value.
- step (float): sets the progress of the bar to the `step` value, where `valmax` is the finish value.
- step (str): displays `step` on a new line. For this to work, `bar` must be `False` (no progress bar displayed) otherwise the update instruction is ignored.

valmax

class patiencebar.**Patiencebarmulti** (*valmax=100*, *barsize=None*, *title=None*, *bar=True*, *up_every=2*)

Bases: *patiencebar.Patiencebar*

Provides a terminal-friendly multi-thread progress bar

Args:

- valmax (float): the finish value of the progress bar. Default is 100.
- barsize (int >0): the size of the bar in the opened terminal. If `None`, the bar will automatically fit the width of the window.
- title (str): the title, printed one line above the progress bar
- bar (bool): whether the bar should be displayed or not. If `False`, only the text given at each *update()* will be printed
- up_every (int [0-100]): if `bar` is `True`, the progress bar will be updated every `up_every` percent of progress. Setting `up_every` = 0 updates the progress bar at each *update()*

```
>>> import patiencebar as PB
>>> from threading import Thread
>>> n_calc = 34
>>>
>>> def worker(pbm, otherarg, anotherarg):
>>>     do_stuff(otherarg, anotherarg)
>>>     pbm.update()
>>>
>>> pbm = PB.Patiencebarmulti(n_calc, 50, "Test bar")
>>> for i in range(n_calc):
>>>     ttt = Thread(target=worker, args=(pbm, otherarg, anotherarg))
>>>     ttt.daemon = True
>>>     ttt.start()
```

reset (*valmax=None*, *barsize=None*, *title=None*, *bar=None*, *up_every=None*)

Resets the progress bar with initialization values, unless new values are given

Args:

- valmax (float): the finish value of the progress bar. Default is 100.
- barsize (int >0): the size of the bar in the opened terminal. If `None`, the bar will automatically fit the width of the window.
- title (str): the title, printed one line above the progress bar.
- bar (bool): whether the bar should be displayed or not. If `False`, only the text given at each *update()* will be printed.
- up_every (int [0-100]): if `bar` is `True`, the progress bar will be updated every `up_every` percent of progress. Setting `up_every` = 0 updates the progress bar at each *update()*.

stop ()

Stops the progress bar, in case it didn't stop naturally

`update`(*step=None*)

Updates the progress bar to a newer value

Args:

- step (None): adds 1 to the progress of the bar, where `valmax` is the finish value.
- step (float): sets the progress of the bar to the `step` value, where `valmax` is the finish value.
- step (str): displays `step` on a new line. For this to work, `bar` must be `False` (no progress bar displayed) otherwise the update instruction is ignored.