# attentive

***Release 0.1.4***

**Sep 12, 2017**

# Contents

```
from attentive import StoppableThread, quit
from random import randint


class Man(StoppableThread):
    def __init__(self, name):
        StoppableThread.__init__(self)
        self.name = name

    def run(self):
        print('{} has quickened'.format(self.name))
        while not self.stopped:
            self.sleep(randint(1, 10))
            print('{} throws a {}'.format(self.name, randint(1, 6)))

        print('{} expires'.format(self.name))


with Man('Trump'), Man('Wang'), Man('Erdoğan'):
    while not quit.is_set():
        quit.wait(1)
```

CHAPTER 1

# Example Run

```
Trump has quickened
Wang has quickened
Erdoğan has quickened
Wang throws a
Trump throws a
Erdoğan throws a
Wang throws a
Wang throws a
Erdoğan throws a
Erdoğan throws a
Trump throws a
^CErdoğan throws a
  Erdoğan expires
Wang throws a
  Wang expires
Trump throws a
  Trump expires
```

Use `attentive` if you need to wire up a some worker threads that needs to cleanly shut themselves down on a SIG_INT or SIG_TERM.

`StoppableThread` is a context managed thread that lives on while in context. Once it exists context it sets its internal stopped flag that are periodically checked for state. This signals thread state allowing the thread to cleanly exit.

External state is controlled by a signal event, exiting the main context loop.

Internally use the StoppableThread.sleep() method that is interrupted when stop()ed during sleep.

# CHAPTER 2

# Install

Install from pypi

```
$ pip install attentive
```

Install from source

```
$ pip install .
```

# CHAPTER 3

# Versioning

Current version is 0.1.3