Unit Tests for Auction Class

Assignment

1. Create a unit test class named **`auction_test.py`** to test that the Auction methods obey the rules of the auction and the behavior specified in the docstring comments for each method. This means you should try to verify the auction *behavior as well as auction methods*.

2. Use descriptive test method names such as test_bid_when_auction_stopped.
   Not a descriptive name: test_borderline

3. Please **do not** write tests for obvious type errors such as bid(None, "1.5") where amount is not a number and name not a string. Type errors like that are the programmer's fault.

4. Use a setUp method to create an Auction object as a *test fixture* (an attribute) instead of creating an auction in each test. It is OK for some test(s) to create their own auction object for a special case.

Evaluation

Your test code will be used to test several variations of the Auction class code. One variation is correct and the others all have a defect. Some may have multiple defects.

Auction Rules

1. Each bid must have an amount > 0 and non-blank bidder name (string). If not, a ValueError is raised.

2. A bid amount must be at least the current best bid *plus a minimum increment*. The minimum increment is specified as a parameter in the Auction constructor. with a default *minimum increment* is 1 (that is, a bid must be at least 1 more than the highest bid so far.) If bid is too low, an AuctionError is raised.

3. Bids allowed only when an auction is active. Auction has methods start( ) and stop() to enable and disable bidding. stop and start may be called many times to pause the auction.

4. The participant with the highest valid bid (best bid) wins.

5. The method best_bid() always returns the highest valid bid so far, and winner() returns the name of the bidder with the highest bid. These methods can be invoked at any time, even while the auction is active.

6. To mitigate accidental typing errors, the bidder name is always put into standard form. This means surrounding white space is removed and words are converted to title case.
So, if a bidder has name " donald   DuCK ", the name will be converted to "Donald Duck".

Documentation

See the Docstring comments in code. To view them in a Python shell, use:

```
>>> from auction import Auction
>>> help(Auction)
>>> help(Auction.method_name)
```

What to Submit - *This will be explained in class.*

Push your auction test code and README.md file to Github.

In **`README.md`**, you must complete a table that describes each defect you find by testing, and try to identify the likely cause of the error. Since you can't look at the source code for the Auction variants, it isn't always possible to identify the cause of the defect.

Example Auction

You should not copy this example. Create your own test cases.

`auction = Auction(` `  "TDD in Python, 2E")`	Create auction for a great testing book!
`auction.start( )`	Start accepting bids.
`auction,bid("Jim", 200)`	Jim bids 200 Baht.
`auction.bid("mai", 250)`	Mai bids 250 Baht.
`auction.bid("", 1000)` `ValueError: Invalid bidder name`	Anonymous bids are not allowed.
`auction.bid("Jim", 250.10)` `AuctionError: Bid is too low`	What a cheap-stake! Only 10 *satang* higher! But it is not allowed. The default bid increment is 1.

`auction.best_bid()`
`250`
`auction.winner( )`
`'Mai'`
	You can call these methods at any time to see the best bid so far and name of the top bidder.
	Names are normalized to Title Case.

`auction.bid("Jittat", 260)`	Ajarn Jittat wants the book, too!
`auction.stop( )`	No bidding allowed now. Call start() to allow bidding.
`auction.bid("Jim", 265)` `AuctionError: bidding not allowed now`	Too late! Auction has stopped.
`auction.winner( )` `'Jittat'`	And Aj. Jittit gets "*TDD in Python*" at a bargain price!