War Card Game in Python

> The generation of random numbers is too important to be left to chance. – Robert R. Coveyou

Create a Python program called "war.py" that plays the card game "War." The program will use the `random` module to shuffle a deck of cards, so your program will need to accept a `-s|--seed` argument (default: `None`) which you will use to call `random.seed`, if present.

First you program will need to create a deck of cards. You will need to use the Unicode symbols for the suites ( ) [which won't display in the PDF, so consult the Markdown file] and combine those with the numbers 2-10 and the letters "J", "Q," "K," and "A." (hint: look at `itertools.product`).

```
>>> from itertools import product
>>> a = list('ABC')
>>> b = range(3)
>>> list(product(a, b))
[('A', 0), ('A', 1), ('A', 2), ('B', 0), ('B', 1), ('B', 2), ('C', 0), ('C', 1), ('C', 2)]
```

NB: You must sort your deck and then use the `random.shuffle` method so that your cards will be in the correct order to pass the tests!

In the real game of War, the cards are shuffled and then dealt one card each first to the non-dealer, then to the dealer, until all cards are dealt and each player has 26 cards. We will not be modeling this behavior. When writing your version of the game, simply `pop` two cards off the deck as the cards for player 1 and player 2, respectively. Compare the two cards by ignoring the suite and evaluating the value where 2 is the lowest and Aces are the highest. When two cards have the same values (e.g., two 5s or two Jacks), print "WAR!" In the real game, this initiates a sub-game of War which is a "recursive" algorithm which we will not bother modeling. Keep track of which player wins each round where no points are awarded in a tie. At the end, report the points for each player and state the winner. In the event of a tie, print "DRAW."

```
$ ./war.py -h
usage: war.py [-h] [-s int]

"War" cardgame

optional arguments:
  -h, --help          show this help message and exit
  -s int, --seed int  Random seed (default: None)
$ ./war.py -s 1
 9  J P2
 A  5 P1
 4  8 P2
```

```
 6  3 P1
 5  3 P1
 K 10 P1
 7  7 WAR!
 2  4 P2
 2 10 P2
 6  5 P1
 2  6 P2
 4  8 P2
 J  9 P1
10  Q P2
 8  7 P1
 K  Q P1
10  2 P1
 9  9 WAR!
 8  J P2
 3  5 P2
 Q  4 P1
 6  A P2
 K  7 P1
 Q  3 P1
 A  K P1
 A  J P1
P1 14 P2 10: Player 1 wins
$ ./war.py -s 2
 4  6 P2
 K  J P1
 J  4 P1
 7  4 P1
 Q 10 P1
 5  3 P1
 K  9 P1
 2  Q P2
 7  A P2
 3  A P2
 5  8 P2
 2 10 P2
10  K P2
 2  3 P2
 Q  8 P1
 6  J P2
 6  8 P2
 8  7 P1
 5  2 P1
 6  J P2
 9  9 WAR!
```

```
 K  A P2
10  Q P2
 7  5 P1
 9  A P2
 4  3 P1
P1 11 P2 14: Player 2 wins
$ ./war.py -s 10
 J  3 P1
 2  5 P2
 Q 10 P1
10  4 P1
 6  5 P1
 3  J P2
 K  8 P1
 5  8 P2
 5  3 P1
 J 10 P1
10  J P2
 A  7 P1
 K  Q P1
 7  A P2
 9  9 WAR!
 2  6 P2
 K  A P2
 6  Q P2
 8  9 P2
 3  7 P2
 8  Q P2
 6  4 P1
 7  2 P1
 4  4 WAR!
 9  2 P1
 K  A P2
P1 12 P2 12: DRAW
```

Test Suite

A passing test suite looks like this:

```
$ make test
pytest -v test.py
============================= test session starts ==============================
platform darwin -- Python 3.6.8, pytest-4.2.0, py-1.7.0, pluggy-0.8.1 -- /anaconda3/bin/pyth
cachedir: .pytest_cache
```

```
rootdir: /Users/kyclark/work/python/practical_python_for_data_science/ch09-python-games/exer
plugins: remotedata-0.3.1, openfiles-0.3.2, doctestplus-0.2.0, arraydiff-0.3
collected 4 items

test.py::test_usage PASSED                                               [ 25%]
test.py::test_play01 PASSED                                              [ 50%]
test.py::test_play02 PASSED                                              [ 75%]
test.py::test_play03 PASSED                                              [100%]

========================== 4 passed in 0.32 seconds ==========================
```