Python Blackjack Game

Write a Python program called `blackjack.py` that plays an abbreviated game of Blackjack. You will need to `import random` to get random cards from a deck you will construct, and so your program will need to accept a `-s|--seed` that will set `random.seed()` with the value that is passed in so that the test suite will work. The other arguments you will accept are two flags (Boolean values) of `-p|--player_hits` and `-d|--dealer_hits`. As usual, you will also have a `-h|--help` option for usage statement.

To play the game, the user will run the program and will see a display of what cards the dealer has (noted "D") and what cards the player has (noted "P") along with a sum of the values of the cards. In Blackjack, number cards are worth their value, face cards are worth 10, and the Ace will be worth 1 for our game (though in the real game it can alternate between 1 and 11).

To create your 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].

Combine these with the numbers 2-10 and the letters "A", "J", "Q," and "K" (hint: look at `itertools.product`). Because your game will use randomness, you will need to sort your deck and then use the `random.shuffle` method so that your cards will be in the correct order to pass the tests.

When you make the initial deal, keep in mind how cards are actually dealt – first one card to each of the players, then one to the dealer, then the players, then the dealer, etc. You might be tempted to use `random.choice` or something like that to select your cards, but you need to keep in mind that you are modeling an actual deck and so selected cards should no longer be present in the deck. If the `-p|--player_htis` flag is present, deal an additional card to the player; likewise with the `-d|--dealer_hits` flag.

After displaying the hands, the code should:

1. Check if the player has more than 21; if so, print 'Player busts! You lose, loser!' and exit(0)
2. Check if the dealer has more than 21; if so, print 'Dealer busts.' and exit(0)
3. Check if the player has exactly 21; if so, print 'Player wins. You probably cheated.' and exit(0)
4. Check if the dealer has exactly 21; if so, print 'Dealer wins!' and exit(0)
5. If the either the dealer or the player has less than 18, you should indicate "X should hit."

NB: Look at the Markdown format to see the actual output as the suites won't display in the PDF version!

```
$ ./blackjack.py
D [11]:  J  A
```

```
P [18]:  8  10
Dealer should hit.
$ ./blackjack.py
D [13]:  3  J
P [16]:  6  10
Dealer should hit.
Player should hit.
$ ./blackjack.py -s 5
D [ 5]:  4  A
P [19]:  10  9
Dealer should hit.
$ ./blackjack.py -s 3 -p
D [19]:  K  9
P [22]:  3  9  J
Player busts! You lose, loser!
$ ./blackjack.py -s 15 -p
D [19]:  10  9
P [21]:  10  8  3
Player wins. You probably cheated.
```

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 13 items

test.py::test_usage PASSED                                               [  7%]
test.py::test_play01 PASSED                                              [ 15%]
test.py::test_play02 PASSED                                              [ 23%]
test.py::test_play03 PASSED                                              [ 30%]
test.py::test_play04 PASSED                                              [ 38%]
test.py::test_play05 PASSED                                              [ 46%]
test.py::test_play06 PASSED                                              [ 53%]
test.py::test_play07 PASSED                                              [ 61%]
test.py::test_play08 PASSED                                              [ 69%]
test.py::test_play09 PASSED                                              [ 76%]
test.py::test_play10 PASSED                                              [ 84%]
```

```
test.py::test_play11 PASSED                                              [ 92%]
test.py::test_play12 PASSED                                              [100%]

========================== 13 passed in 0.82 seconds ===========================
```