however, the not familiar with the game, can add to and/or overwrite those instance variables and methods.

It's best learnt through an example (or at least I think so), so we're going to learn how to create a deck of playing cards through OOP.

Basic list comprehensions

List comprehensions are a shorter, (usually) more readable version of a for loop. Not all for loops can be written as a list comprehension, but things over. It is much better memory-wise for large arrays, but slightly harder to implement.

*Durstenfeld improved this algorithm slightly so that the elements of the array get swapped, rather than creating a new array and copying things over. It is much better memory-wise for large arrays, but slightly harder to implement.

### your code here ###

```python
Card
```
like.

finally, we get to the engine of the game. most of it is done for you, but there are 3 pieces of code you'll need to add yourself. this next part can be confusing, so take your time! we're going to create a

```
class Player
    # your code here!
```

```python
def __init__(self)
    # boolean flag to indicate when the player either passes or goes bust
    self.userPass = False

    self.current_hand = []

    self.get_status()  # get the player's current status

    if userPass:
        return

    self.add_points()  # increase the player's score

    while
        print_status(self)

        # check neither party broke 21 bc they drew two aces
        if userPass and self.dealer.get_status() < 17:
            self.end_result = True
            return

        # start of game stuff
        self.add_card(self)  # add a card
        self.add_card(self)  # add a second card

        # if the user passed (i.e. didn't bust) and the dealer has cleared 17, we see the result
        if userPass and self.dealer.get_status() >= 17:
            self.end_result = True
            return

        # if the dealer broke, we see the result
        if self.dealer.get_status() > 21:
            self.end_result = True
            return

        self.output()  # output the player's current status
```

```python
def all_cards
    return self.current_hand
```

```python
def add_points()
    # your code here!
```

```python
def get_status()
    # your code here!
```

```python
def add_card(self, c)
    # your code here!
```

```python
def get_points(self)
    # your code here!
```

```
# copy over the
```