Contents

1 blackjack package 3
  1.1 Subpackages 3
  1.2 Submodules 5
  1.3 Module contents 7

2 Indices and tables 9

Python Module Index 11
1.1 Subpackages

1.1.1 blackjack.packages package

Submodules

class Card (input_card)

Class of a single card in a traditional deck of 52 cards. Cards are 2 - 10 and have a value matching their integer name. or the are a ‘face’ card (Jack, Queen, King) and valued at 10. Ace cards are valued at 1 or 11 depending on the state of the rest of the hand.

Parameters input_card (tuple) – A 3-value tuple composed of: card_name : string

The name of the card from the traditional set of playing cards.

suit [string] One of ('H', 'D', 'C', 'S') representing the four possible suits.

orig_loc [int] The card’s original location in the shoe at “construction” time

card_name

string

The “rank” of given card instance

suit

string

The “suit”

orig_loc

int

The integer representing the location in the original shoe, before any shuffling

value

int

The integer value (from 1 - 11) of a card’s worth in the game of Blackjack.

assign_value ()
flip_ace()
For imposing a new value if the status of the game requires it.

class Dealer (shoe)
    Bases: blackjack.packages.player.Player

A subclass of the class Player, with a method to determine if he hits or not in a given situation.

hits()

class Player (shoe)
    The hand of cards for a given player

    shoe [Instance of Shoe Class] The current shoe initialized in Blackjack.sitting()

    hand_of_cards
        list

A list of cards in the current hand.

    max_target_score
        int

A constant set to 21 per the rules of blackjack

    status
        dict

A dictionary of bools, representing: “bj” : Score a Blackjack? “bust” : Bust (go over 21) yet? “interested” : Interested in receiving another card when offered

    get_card()
        Gets a new card from the shoe

    pretty_output()
        Creates a string of the current cards for output

    score()
        Calculates the current score of the hand

    update_status()
        Updates a players status based on score of current hand_of_cards

        Returns As evidence of executing

        Return type True

class User (shoe)
    Bases: blackjack.packages.player.Player

A subclass of the class Player, with a method to determine if he hits or not in a given situation.

hits()

class Shoe (num_decks=1)
    A shoe of cards made out of a number of traditional decks. User will be asked for the number of decks to put in the shoe.

    Parameters num_decks (int) – The number of decks supplied by the user. Defaults to 1 deck.

num_decks
    int

Number of 52 card decks to use
cards
    list
    List of instances of card object returned by the constructor method

__len__ ()
    Overloading Builtin: Assigns the value of len(list of cards) to the length of the object

cards_left ()
    Not currently accessed by any part of the script, other than the testing suite.
    Returns  The length of list that is self.cards
    Return type  int

constructor ()
    
    Builds a shoe of playing card decks. Each card in a given deck will be unique.
    cards
        list
        A list of the cards currently available to the players

suits
    list
    A list of strings representing the 4 suits possible in a deck
    
    Returns  A list of cards that is a (num_decks) multiple of standard 52 card decks.
    Return type  list

hand_out_card ()
    Takes the top card, aka the last card in the list of cards and removes it from the list and returns it.
    
    Returns  Based on success of execution.
    Return type  bool

merge_sort (lst)
    Not currently accessed by any part of the script, other than the testing suite.
    Parameters  lst (list) – A list of “orderable” objects.
    Returns  A new list of the same elements - sorted.
    Return type  list

shuffle_shoe ()
    Shuffles the order of the cards in the list that constitutes the shoe.
    
    Returns  For evidence of execution
    Return type  True

Module contents

1.2  Submodules

blackjack.py Author: Cole Howard
A simple implementation of the game of blackjack.
The user is tasked with defining the number of decks of 52 cards to be used in the shoe. The game can then be played until the shoe runs out or the user opts out. Dealer always wins ties.

### 1.2.1 Usage

Copy the folder Blackjack (capital B) and run the script:

```bash
$ python3 blackjack/bj_main.py
```

All interaction will be via stdout on the command line.

Tests can be run from the same directory via

```bash
$ nosetests
```

or

```bash
$ py.test
```

**Dependencies:** Python 3.0 or newer random.shuffle Card class from card.py Shoe class from shoe.py User and Dealer (subclasses of Player) from player.py

**Blackjack**

An environment for playing blackjack. User will define the size of the deck shoe (in numbers of 52 card decks) and can then play hands of blackjack until the shoe runs out, or they opt out

```python
def dealer_victories
    int
    Number of hands won by the dealer

def player_victories
    int
    Number of hands won by the player
```

**sitting()**

```python
class Game (shoe)
    Initialize game of blackjack

    shoe [Instance of Shoe Class] The current shoe initialized in Blackjack.sitting()

dealer
    instance of Player class

player
    instance of Player class

initial_hand_size
    int
    Number of cards initially dealt

max_target_score
    int
    A constant set to 21 per the rules of blackjack

dealer_trigger
    int
    A constant set to 17, the score at which the dealer will stop accepting additional cards, per the rules of blackjack
```
deal ()
    Deals the set # of starting cards (defined in the init)

eval_state ()
    Evaluates the status of the game to the stdout on command line
    
    Returns  “d”, “p”, or None (if out of cards or unresolved)

    Return type  string

get_winner ()
    Determines the winner if both players pass
    
    Returns  “d” (dealer) or “p” (player) for victor

    Return type  string

play ()
    Plays a single round of blackjack

    Return type  instance of Player object

report (finished=False)
    Reports the status of the game to the stdout on command line

report_results ()
    Report final outcome of the hand

    Returns  As notification of execution

    Return type  True

1.3  Module contents
CHAPTER 2

Indices and tables

• genindex
• modindex
• search
b
blackjack, 7
blackjack.bj_main, 5
blackjack.packages, 5
blackjack.packages.card, 3
blackjack.packages.player, 4
blackjack.packages.shoe, 4
Symbols

__len__() (Shoe method), 5

A
assign_value() (Card method), 3

B
Blackjack (class in blackjack.bj_main), 6
blackjack (module), 7
blackjack.bj_main (module), 5
blackjack.packages (module), 5
blackjack.packages.card (module), 3
blackjack.packages.player (module), 4
blackjack.packages.shoe (module), 4

C
Card (class in blackjack.packages.card), 3
card_name (Card attribute), 3
cards (Shoe attribute), 4, 5
cards_left() (Shoe method), 5
constructor() (Shoe method), 5

d
deal() (Game method), 6
Dealer (class in blackjack.packages.player), 4
dealer (Game attribute), 6
dealer_trigger (Game attribute), 6
dealer_victories (Blackjack attribute), 6

e
eval_state() (Game method), 7

F
flip_ace() (Card method), 3

G
Game (class in blackjack.bj_main), 6
get_card() (Player method), 4
get_winner() (Game method), 7

H
hand_of_cards (Player attribute), 4
hand_out_card() (Shoe method), 5
hits() (Dealer method), 4
hits() (User method), 4

I
initial_hand_size (Game attribute), 6

M
max_target_score (Game attribute), 6
max_target_score (Player attribute), 4
merge_sort() (Shoe method), 5

N
num_decks (Shoe attribute), 4

O
orig_loc (Card attribute), 3

P
play() (Game method), 7
Player (class in blackjack.packages.player), 4
player (Game attribute), 6
player_victories (Blackjack attribute), 6
pretty_output() (Player method), 4

R
report() (Game method), 7
report_results() (Game method), 7

S
score() (Player method), 4
Shoe (class in blackjack.packages.shoe), 4
shuffle_shoe() (Shoe method), 5
sitting() (Blackjack method), 6
status (Player attribute), 4
suit (Card attribute), 3
suits (Shoe attribute), 5
U
update_status() (Player method), 4
User (class in blackjack.packages.player), 4

V
value (Card attribute), 3