SECTION 11: MILESTONE PROJECT - 2 hours 18 minutes, 12 sections

- **10/12 Solution Walkthrough - Hand and Chip Classes**
    - -> a hand class and a chips class
    - -> there are global variables called suits, ranks and values
        - pass in the rank of the card and get back the object
    - -> she's written a class called deck -> with attributes that shuffle and deal the cards

    - **-> then she's defining a hand class**
        - -> this is to add cards to someones' hand
        - -> adding a value attribute to the hand

        - **-> in the hand class -> methods**
            - adding cards -> the arguments of this method when it's defined are self and and card
            - -> then the card is being appended into the hand
            - -> self.cards.append(card)
            - -> the card object has a rank
            - -> the card is being appended to the current list of cards
            - -> then being added to self.value / the current value of the hand

        - **-> then she tests the code before the ace (which can have two possible values (a 1 or 11)**
            - -> test_deck = Deck
            - -> test_deck.shuffle()
            - -> then
                - test_player = Hand()
                - pulled_card = test_deck.deal <- **shift tab returns the different methods which are available for use**
                - test_player.add_card

    - **-> so the thought process is**
        - **after defining the Hand class**
            - -> she's defined a deck
            - -> shuffled the deck
            - -> created a test player with a hand object -> dealt cards from it
            - -> then printed the card which was pulled from the deck
            - -> each card has a string representation
            - -> then printed out the test player's value

        - **-> then she's doing the same thing in less lines / more efficiently**
            - test_player.add_card(test_deck.deal())
            - test_player.value <- then the value is printed

    - **-> the Aces**
        - -> the value of the aces can be a 1 or an 11 -> depending on what the user wants
        - -> she's adding this into the hand class
        - -> if card == 'Ace'

        - **-> then while self.value > 21 and self.aces:**
            - self.value -= 10 <- **subtract 10 from the current value**
            - self.aces -=1
            - -> we're not adjusting an ace equal to 21

            - -> the logic is -> change the ace from an 11 to a 1 depending on how close to 21 the current score is
                - **truthiness -> representing an integer as a boolean value**
                - if zero:
                    - print("TRUE")
                    - -> **zero is treated as false (in this example which she's done)**
            - -> another one you can do is -
                - -> while self.value > 21 and self.aces > 0:
                - -> and then set the value of the ace

    - **-> a chip class**
        - -> the attributes are total and bet
            - the bets of the user and the total score which we have
        - -> then the methods (functions) in the definition of the class are
            - -> the methods are that they win or loose the bet

    - **-> summary of the thought process**
        - -> the hand class is a representation of the player
            - -> the computer or the human player
        - -> then aces are kept track of and cards are added to the deck
        - -> then we are adjusting for the ace -> i.e depending on the value of the current count, its value is 1 or 11