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

- **3/12 Deck Class**
  - -> a class deck which can be shuffled etc
  - **-> the first milestone project was combining functions together -> this one is combining classes**

  - **-> the deck class**
    - needing a new deck -> create 52 required card objects and hold them as a list
      - -> as an attribute of the deck class (not just stored as an element in an array)
      - -> each element represents a card object
    - to be able to shuffle the deck (the shuffle function in the random library)
    - -> then to deal cards from the deck object (using the pop method)
    - -> the deck class holds a list of card objects
    - -> holding cards / card objects

  - **-> in the .ipynb file**
    - she is reusing the code from the previous lecture videos -> tuples with suits, ranks and values
    - then we also have the definition of the card class

    - **for the deck class**
      - -> def __init__(self):
        - -> create a deck
        - -> we are defining its attributes
        - self.all_cards = [] <- an empty list with no input from the user
        - for suit in suits: <- for suits / diamonds / etc
          - for rank in ranks:
            - -> then creates the card object
            - created_card = Card(suit, rank) <- this creates a new deck
            - self.all_cards.append(created_card) <- appending to the list of cards (so we're building a deck by iterating through all the suits and ranks)

    - **-> then creates decks**
      - new_deck = Deck()
      - first_card = new_deck.all_cards[0]

    - **-> print(first_card)**
      - hearts is the first suit and two is the first rank
      - -> the two of hearts should have been at the top of the deck (from the input in this example)
      - -> [-1] rather than [0] wold have returned the last card in the deck

    - **-> for card_object in new_deck.all_cards:**
      - -> and then prints out the card objects
      - -> the card objects return back the rank of the suit

    - **-> creating a shuffle method**
      - this is in the Deck class
      - **-> we have a list and we want to shuffle it**
        - -> random.shuffle(mylist)
        - -> from random import shuffle <- we are using the shuffle function to randomly shuffle the deck

            - -> this sets the deck equal to the shuffled version
    - -> see the modules and package series of lectures

    - **-> the shuffle method for the deck**
        - internally shuffle all the cards
        - -> random.shuffle(self.all_cards)
        - -> we want to know that the deck is shuffled
            - she prints out the card at the top of the deck, shuffles the entire thing and then prints out the card at the bottom of the deck
            - -> the cards on the top and bottom of the deck are no longer the same

- **-> removing cards from the deck**
    - -> def deal_one(self):
        - return self.all_cards.pop <- this is in the definition of the class
        - -> removing one of the class objects and returning it
        - -> mycard = new_deck.deal_one()
        - -> then she checks the length of the new deck
            - and it has reduced by one
            - -> you can get an error if you pop all of the cards off of the list (deck of card objects)
            - -> you can run a for loop

- -> the deal one method