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

- **9/12 Solution Walkthrough - Card and Deck classes**
  - -> this notebook is a walk through of their solutions
  - -> this is an example solution -> it can be done in multiple different ways

  - **-> in the .ipynb file**
    - there are 12 steps to the solution
    - -> importing the global variables
      - ones which can be referenced in other objects
      - -> shuffling the deck prior to dealing

    - **-> the first stage to their solution is to initialise the entire thing -> set tuples defining the suits, ranks and values**
    - -> she pastes the entire thing into a cell
    - -> **then -> in all of the game examples defines a boolean which checks if the game is being played or not (e.g if the player wishes to quit the game)**

  - **-> then the card class**
    - **class Card:**
      - **def __init__(self): <- she first initialises the class with the different attributes which the class has**
        - **self.suit = suit**
        - **self.rank = rank**
      - **def __str__(self): <- then is defining the different functions (methods) for use on that string, this is for the string representation**
        - **return self.rank+ " of "+self.suit**

    - -> then storing the cards as part of a list which can be shuffled around
    - -> it looks similar to the previous example -> where there is another class called Deck
    - -> the cards are initialised -> and in this case they are the same each time
    - -> then she iterates through the different suits
      - -> in one of these sections it's self.deck.append(Card(suit, rank))
    - **-> she is defining methods in Deck which -> shuffle the cards in the deck and deal them**
      - **-> random.shuffle(self.deck) <- shuffle sets the list equal to the list shuffled (the list changes and the result of the entire thing is NoneType)**
      - -> the deal method (function) in the Deck class uses the pop method

  - **-> she tests the deck class**
    - -> test_deck = Deck()
    - -> she also runs a line of code which prints which card types we are dealing with
    - -> and then pint(test_deck)

  - **-> thought process for this lecture**
    - -> she sets up global variables -> suits, ranks and values
    - -> the values take a dictionary form, to convert between the string and a number format

    - **-> then there is another class for cards**
      - -> a suit and a rank
      - -> this allows us to print out cards

- **-> then the Deck class was defined**
  - -> this had an empty list
  - -> then the list was populated with cards of each suit and iterating through different numbers
  - **-> the cards are objects because they have been defined in this example as part of a class**
  - -> then there is a method defined which pops off certain results