UIT2502 – Data Analytics and Visualization Lab

Game Application using numpy.

Vigneshwaran R

3122 21 5002 122

Aim:

To work with numpy and create a game application.

Code:

import numpy as np

import tkinter as tk

root2=tk.Tk()

root2.title('Game of Cards')

root2.geometry('700x300')

prompt1 = tk.Label(root2, text = 'Rules are: ')

prompt1.pack()

prompt2 = tk.Label(root2, text = '1. Its a 2 player game of cards where each player randomly gets 26 cards...')

prompt2.pack()

prompt3 = tk.Label(root2, text = '2. All the Symbols on the cards are considered equal and the real game is based on number on the face of the cards')

prompt3.pack()

prompt4 = tk.Label(root2, text = '3. Two cards are released by both the players at the same time.. The player whose cards has the lesser should take both the cards')

prompt4.pack()

prompt5 = tk.Label(root2, text = '4. If the number on the cards match, the cards are discarded...')

prompt5.pack()

prompt6 = tk.Label(root2, text = '5. The player who wins the first 50 hands wins first!!!')

prompt6.pack()

prompt7 = tk.Label(root2, text = '6. The player who reaches 0 cards before reaching 20 fins wins ...')

prompt7.pack()

root2.mainloop()

"""A class for creation of card objects"""

symbol = np.array(['Hearts','Clubs','Spade','Diamond'])

rank = np.array(['2','3','4','5','6','7','8','9','10','J','Q','K','A'])

class CardObject:

def \_\_init\_\_(self,rank,symbol):

self.rank=rank

self.symbol=symbol

def \_\_str\_\_(self):

return self.rank+" of "+self.symbol

class Cards:

"""A class that creates and stores all card objects"""

def \_\_init\_\_(self):

self.cards=[]

for i in symbol:

for j in rank:

self.cards.append(CardObject(j,i))

self.cards=np.array(self.cards)

def shuffle(self):

np.random.shuffle(self.cards)

class Players:

"""A class that creates players and allows us to plays the game """

def \_\_init\_\_(self):

self.Player1=np.array([])

self.Player2=np.array([])

cardslist=Cards()

cardslist.shuffle()

for i in range(0,26):

self.Player1=np.append(self.Player1,cardslist.cards[i])

self.Player2=np.append(self.Player2,cardslist.cards[i+26])

def shufflehands(self):

np.random.shuffle(self.Player1)

np.random.shuffle(self.Player2)

def play(self):

Player1\_win=0

Player2\_win=0

lis=[["Player1 Card","Player2 Card"]]

while True:

if len(self.Player1)==0 or Player1\_win == 20:

newwindow=tk.Tk()

newwindow.title('Results')

text\_widget = tk.Text(newwindow, width = 50, height =len(lis), font =('Arial',12))

for i in lis:

text\_widget.insert(tk.END, i[0]+" "+i[1]+"\n")

text\_widget.insert(tk.END,'Player 1 wins')

text\_widget.pack(pady = 20)

newwindow.mainloop()

return 1

elif len(self.Player2)==0 or Player2\_win == 20:

newwindow=tk.Tk()

newwindow.title('Results')

text\_widget = tk.Text(newwindow, width = 40 , height =len(lis), font =('Arial',12))

for i in lis:

text\_widget.insert(tk.END, i[0]+" "+i[1]+"\n")

text\_widget.insert(tk.END,'Player 2 wins')

text\_widget.pack(pady = 20)

newwindow.mainloop()

return 2

Player1\_Card=self.Player1[0]

self.Player1=self.Player1[1:]

Player2\_Card=self.Player2[0]

self.Player2=self.Player2[1:]

lis.append([Player1\_Card.symbol+Player1\_Card.rank,Player2\_Card.symbol+Player2\_Card.rank])

if np.where(rank == Player1\_Card.rank) > np.where(rank == Player2\_Card.rank):

self.Player2=np.append(self.Player2,Player1\_Card)

self.Player2=np.append(self.Player2,Player2\_Card)

Player1\_win+=1

elif np.where(rank == Player1\_Card.rank) < np.where(rank == Player2\_Card.rank):

self.Player1=np.append(self.Player1,Player1\_Card)

self.Player1=np.append(self.Player1,Player2\_Card)

Player2\_win+=1

else:

self.shufflehands()

def exit\_game(win):

win.destroy()

def play\_againf(win):

global entry

win.destroy()

root = tk.Tk()

root.title('Card Game...')

root.geometry('300x150')

prompt = tk.Label(root, text = 'Predict who will win this game 1 or 2: ')

prompt.pack(pady = 10)

entry = tk.Entry(root, width =20)

entry.pack()

submit = tk.Button(root, text = 'Play', command = lambda: get\_input(root))

submit.pack(pady = 10)

root.mainloop()

def get\_input(win):

global entry

user\_input = entry.get()

win.destroy()

game = Players()

if int(user\_input) == game.play():

root1 = tk.Tk()

root1.title('Card Game...')

root1.geometry('300x150')

prompt = tk.Label(root1, text = 'You have predicted correctly!!! ')

prompt.pack(pady = 10)

exitbutton = tk.Button(root1, text = 'Exit', command = lambda: exit\_game(root1))

play\_again = tk.Button(root1, text = 'Play Again', command = lambda: play\_againf(root1))

exitbutton.pack(pady = 10)

play\_again.pack(pady = 10)

root1.mainloop()

else:

root2 = tk.Tk()

root2.title('Card Game...')

root2.geometry('300x150')

prompt = tk.Label(root2, text = 'Better luck next time!!!')

prompt.pack(pady = 10)

exitbutton = tk.Button(root2, text = 'Exit', command = lambda: exit\_game(root2))

play\_again = tk.Button(root2, text = 'Play Again', command = lambda: play\_againf(root2))

exitbutton.pack(pady = 10)

play\_again.pack(pady = 10)

root2.mainloop()

root = tk.Tk()

root.title('Card Game...')

root.geometry('300x150')

prompt = tk.Label(root, text = 'Predict who will win this game: ')

prompt.pack(pady = 10)

entry = tk.Entry(root, width =20)

entry.pack()

submit = tk.Button(root, text = 'Play', command = lambda: get\_input(root))

submit.pack(pady = 10)

root.mainloop()

Input Output Screenshots

A card game with text

Description automatically generated

A screenshot of a computer game

Description automatically generated

Test Case 1

A list of sports teams

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Test 2

A list of sports teams

Description automatically generated

A list of sports teams

Description automatically generated

A screenshot of a computer

Description automatically generated

Result

Thus, a game has been developed using numpy.