# Puzzle

October 9, 2021

## 1 Puzzle

ENGR 195E

Melissa Iraheta
Hasitha Gampa
Tania Moran Hernandez
Eduardo Velazco

[1]:
```
import os
import stat
#os.chmod("Puzzle.ipynb",stat.S_IRWXO|stat.S_IRWXU|stat.S_IRWXG) #allow anyone␣
↪to read and write this file
```

[2]:
```
import pyper as pr
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
import time
import numpy as np
import random
import csv
import string
import time
scores = {"Emotions":0, "Expressions":0, "Social":0,"School":0,"Patterns":0}␣
↪#initialize dictionary for points - outside class so can be used in any class
prize_image = [Image.open('Prize_Images/first_pp.png'),Image.open('Prize_Images/
↪two_pp.png'), Image.open('Prize_Images/three_pp.png'),
               Image.open('Prize_Images/four_pp.png'), Image.open('Prize_Images/
↪five_pp.png'), Image.open('Prize_Images/final_pp.png')]
```

### 1.1 Puzzle Game

[ ]:
```
import pynput
```

[ ]:
```
import pyper as pr
import pandas as pd
import matplotlib.pyplot as plt
```

```
from PIL import Image
import time
import numpy as np
import random
import csv
import string
import time
scores = {"Emotions":0, "Expressions":0, "Social":0,"School":0,"Patterns":0}␣
 ↪#initialize dictionary for points - outside class so can be used in any class
activities_passed = 0 #sets base count of activities passed

# The images used to print out if the score is greater thatn 80%
prize_image = [Image.open('Prize_Images/first_pp.png'), Image.
 ↪open('Prize_Images/two_pp.png'), Image.open('Prize_Images/three_pp.png'),
               Image.open('Prize_Images/four_pp.png'), Image.open('Prize_Images/
 ↪five_pp.png'), Image.open('Prize_Images/final_pp.png')]


def print_menu():

    print(30 * "-","\033[1m"+"   Welcome to Puzzle  "+"\033[0m", 30 * "-")

    print("""
        Which game would you like to play:

                    A) Patterns Game
                    B) Social Situation Game
                    C) Facial Expressions
                    D) School Situation Game
                    E) Emotions Game
                    F) Leave Puzzle Game
                    """)
    print(73 * "-")
    print()

loop = True


while loop:
    timestamp = time.strftime("%m/%d/%Y")
    print_menu()
    while True:
        userID = (input('what is your 4-digit ID: '))
        if len(userID) > 4:
            print('invalid ID sorry')
            continue
        elif len(userID) < 4 or userID in string.ascii_letters:
            print('Sorry, it has to be 4 digits!')
        else:
            break


    choice = input ("Please enter your choice [A-F]: ")
    # Patterns game begin here
    if choice == "A" or choice =="a":
        print(
         "\033[1m"+ " --- Patterns Game!--- "+"\033[0m")

        # Patterns Game
        class Patterns:
            def __init__(self, images, question, ok_answers, correct_answer,␣
↪points):
                self.images = images
                self.question = question #set questions
                self.ok_answers = ok_answers # answer options
                self.correct_answer = correct_answer #correct answer
                self.points = points #points

            # Creating the questions
        question = [
                Patterns(Image.open("Patterns/Shapes.png"),"What shape comes␣
↪next? Finish the pattern", ["Rectangle","Triangle","Circle", "I don't know"],
                    1, 1),
                Patterns(Image.open("Patterns/Animals.png"),"Which animal comes␣
↪next? Finish the pattern", ["Elephant","Monkey","Giraffe", "I don't know"],
                    2 ,1),
                Patterns(Image.open("Patterns/Colors.png"),"What color comes␣
↪next? Finish the pattern.", ["Orange ","Green","Red", "I don't know"],
                    0, 1),
                Patterns(Image.open("Patterns/Numbers.png"),"What number is␣
↪missing? Finish the pattern.",[ "1","2","3", "I don't know"],
                    1, 1)
                ]


        #Where answers will be placed
        user_answers = {}
        p_test = random.shuffle(question) #Randomize questions


        def run_pattern(p_test):
            global user_answers #access global list of user_answers
            global activities_passed #acess global list of activities_passed
            score = 0   #sets base score
            user_answers = {"Incorrect Answers":0, "Invalid Answers":0,␣
↪"Correct Answers":0}
            #adds dictionary to store puzzles
            for question_all in question: #runs through each question
                print("")
                print(question_all.question) #goes through all acceptable␣
↪questions

                for image in question_all.ok_answers: #Calls images assigned to␣
↪each question
                    get_image = np.asarray(question_all.images)
                    image_plot = plt.imshow(get_image)
                    plt.xticks([])
                    plt.yticks([])
                plt.show()

                q = [x for x in range(len(question_all.ok_answers))] #runs␣
↪through answers
                random.shuffle(q)
                correct_answer = q.index(question_all.correct_answer)
                for ix in range(len(question_all.ok_answers)):
                    print("({:s}) {:s}".format("ABCD"[ix],question_all.
↪ok_answers[q[ix]]))
                not_answer = True
                while not_answer:
                    sel = input("Your answer is: ").lower().strip()
                    if len(sel) != 1:
                        print("Please, answers should be 1 character (ex:a,b,c␣
↪or d)")
                        user_answers["Invalid Answers"] += 1
                        continue
                    if not (sel in "abcd"):
                        print("Oops, that's not an option!")
                        user_answers["Invalid Answers"] += 1
                        continue
                    if sel == "abcd"[correct_answer]:
                        print("Great job! You get one point!")
                        not_answer = False
                        score += 1
                        user_answers["Correct Answers"] += 1
                        scores["Patterns"] += 1
                    else:
                        print("Nope, better luck next time!")
                        user_answers["Incorrect Answers"] += 1
                        not_answer = False
            with open('puzzle_patternsscores.csv', 'a', newline='') as out_file:
                writer = csv.writer(out_file)
                #writer.writerow(["Date", "User ID", "Correct Answers",␣
↪"Incorrect Answers", "Invalid Answers"])
                writer.writerow([timestamp,userID, user_answers["Correct␣
↪Answers"], user_answers["Incorrect Answers"], user_answers["Invalid␣
↪Answers"]])

            print("Your final score is: " + str(score) + "/" +␣
↪str(sum(quiz_points.points for quiz_points in question)))

            if user_answers["Correct Answers"] >= 3:
                activities_passed += 1
                plt.imshow(prize_image[activities_passed])
                plt.xticks([])
                plt.yticks([])
                plt.show()
                print("You unlocked a puzzle piece!")
            elif user_answers["Correct Answers"] <= 2:
                print("Sorry! Could not unlock a puzzle piece. Try again!")

        run_pattern(p_test)
        user_answers
        scores
        loop = True
        print()

    elif choice == "B" or choice =="b":
        print("\033[1m"+ " --- Social Situations Game--- "+"\033[0m")

        #Creating a class called Social Question for the Social
        class social_question:
            def __init__(self,images, question,ok_answers, correct_answer,␣
↪points):
                self.images = images
                self.question = question
                self.ok_answers = ok_answers
                self.correct_answer = correct_answer
                self.points = points
        # Here we create the questions and open the file
        question = [
                social_question(Image.open("Social_Qs/dooropen.jpg"),"What if␣
↪you found the door open when you got home?", [ "You go to a trusted␣
↪neighbors house","You go get vanilla ice cream","You fall asleep on the␣
↪lawn","I don't know"],
                 0, 1),
                    social_question(Image.open("Social_Qs/laughing.jpg"),"If␣
↪someone is making jokes to you but you do not like it", ["Ask nicely to␣
↪stop","You kick them","You go to the swings", "I don't know"],
                0 ,1),
                social_question(Image.open("Social_Qs/following.jpg"),"You are␣
↪walking and someone is following you", ["Ask a trusted adult to walk you␣
↪home","You climb a tree to the top","You go pet a dog", "I don't know"],
                0, 1),
                    social_question(Image.open("Social_Qs/vase.jpg"),"You␣
↪accidentally break your mom's favorite vase while playing inside the house␣
↪(something you are not allowed to do)",[ "Be honest that you broke it","You␣
↪do jumping jacks","You sit and watch TV", "I don't know"],
                0, 1),
                    social_question(Image.open("Social_Qs/jacket.jpg"),"You see␣
↪your friend wearing a jacket you lost a week ago",["Notify your parent its␣
↪yours","You go play the piano","You eat a cookie","I don't know"],
                0, 1),
                    social_question(Image.open("Social_Qs/toys.jpg"),"You found␣
↪a new toy on the playground",["You take it to the lost and found","You make␣
↪a sand castle","You dig a hole with your hands","I don't know"],
                0, 1),
                    social_question(Image.open("Social_Qs/dog.jpg"),"You see a␣
↪dog in front of your house",["Stay inside the house","You play hide and␣
↪seek","You play with ballons","I don't know"],
                0, 1),
                    social_question(Image.open("Social_Qs/night.jpg"),"You hear␣
↪the front door opening in the middle of the night, what do you do? ",["Go to␣
↪your parents room","You eat some cake","You sing a song","I don't know"],
                0, 1),
                    social_question(Image.open("Social_Qs/oven.jpg"),"You see␣
↪the oven smoking",["Tell an adult","You go into a pool","You play tag", "I␣
↪don't know"],
                0, 1)]

            #Where the answers will be placed
        user_answers = {}

        #Shuffling the questions in order for them to be randomized
        test_social = random.shuffle(question)

            #This class is called run_test
        def run_social(test_social):
            global user_answers #access global list of user_answers
            global activities_passed #acess global list of activities_passed
            score = 0   #Starting score is at 0
            user_answers = {"Incorrect Answers":0,"Invalid Answers":0, "Correct␣
↪Answers":0}

            for question_all in question: #it runs through the number of␣
↪questions
                print("")
                print(question_all.question)

                    #Goes through the acceptable questions
                # Code to get the corresponding image with the corresponding␣
↪question
                for image in question_all.ok_answers:
                    get_image = np.asarray(question_all.images)
                    image_plot = plt.imshow(get_image)
                    plt.xticks([]) # removes plot ticks
                    plt.yticks([])
                plt.show()

                    # Goes through all the questions and gets the correct␣
↪choice depending on index
                q = [x for x in range(len(question_all.ok_answers))]
                random.shuffle(q)
                correct_answer = q.index(question_all.correct_answer)
                for ix in range(len(question_all.ok_answers)):
                    print("({:s}) {:s}".format("ABCD"[ix],question_all.
↪ok_answers[q[ix]]))

                # If it is not true then it gives an error message
                not_answer = True
                while not_answer:
                    sel = input("Your answer is: ").lower().strip()
                    if len(sel) != 1:
                        print("Not an acceptable answer, sorry! (ex:a,b,c, or␣
↪d)")
                        user_answers["Invalid Answers"] += 1
                        continue

                        # If the the answer is not part of the selection
                    if not (sel in "abcd"):
                        print("Sorry the letter input is not a choice")
                        user_answers["Invalid Answers"]+= 1
                        continue

                    # If the answer is correct
                    if sel == "abcd"[correct_answer]:
                        print("Great job! You got one point.")
                        not_answer = False
                        score += 1
                        user_answers["Correct Answers"] += 1
                        scores["Social"] += 1

                    # When the answer is incorrect
                    else:
                        print("Nope, better luck next time!")
                        user_answers["Incorrect Answers"] += 1
                        not_answer = False

            with open('Social_Qscores.csv', 'a', newline='') as out_file:
                writer = csv.writer(out_file)
                #writer.writerow(["Date", "User ID", "Correct Answers",␣
↪"Incorrect Answers", "Invalid Answers"])
                writer.writerow([timestamp,userID, user_answers["Correct␣
↪Answers"], user_answers["Incorrect Answers"], user_answers["Invalid␣
↪Answers"]])

            print("Your final score is: " + str(score) + "/" +␣
↪str(sum(quiz_points.points for quiz_points in question)))
            print()

            if user_answers["Correct Answers"] >= 7:
                activities_passed += 1
                plt.imshow(prize_image[activities_passed])
                plt.xticks([])
                plt.yticks([])
                plt.show()
                print("Puzzle Unlocked!")
            elif user_answers["Correct Answers"] <= 6:
                print("Sorry, no puzzle piece unlocked. Try again!")
                print()

        run_social(test_social)
        user_answers
        scores
        loop = True
        print()

    elif choice == "C" or choice =="c":
        print( "\033[1m"+ " --- Facial Game!--- "+"\033[0m")
        # Facial Expressions Game
        class facial_expressions:
            def __init__(self, images, question, ok_answers, correct_answer,␣
↪points):
                self.images = images
                self.question = question #set questions
                self.ok_answers = ok_answers # answer options
                self.correct_answer = correct_answer #correct answer
                self.points = points #points

            # Creating the questions
        question = [
                    facial_expressions(Image.open('FacialExp_images/Tired.
↪png'),"How does Deedee feel today?", [ "Happy","Tired","Sad","I don't know"],
               1, 1),
                    facial_expressions(Image.open('FacialExp_images/Excited.
↪png'),"How does Bob feel today?", ["Excited","Sad","Embarassed","I don't␣
↪know"],
                0, 1),
                    facial_expressions(Image.open('FacialExp_images/Happy.
↪png'),"How does Nico feel today?", ["Nervous","Afraid","Happy","I don't␣
↪know"],
                2, 1),
                    facial_expressions(Image.open('FacialExp_images/Embarassed.
↪png'),"How does Quinn feel today?", [ "Happy","Afraid","Embarassed","I don't␣
↪know"],
               2, 1),
                    facial_expressions(Image.open('FacialExp_images/Afraid.
↪png'),"How does Akeno feel today?", ["Afraid","Tired","Surprised","I don't␣
↪know"],
               0, 1),
                    facial_expressions(Image.open('FacialExp_images/Surprised.
↪png'),"How does Alberto feel today?", ["Sad","Surprised","Happy","I don't␣
↪know"],
               1,1),
                    facial_expressions(Image.open('FacialExp_images/Sad.
↪png'),"How does Minnie feel today?", ["Tired","Happy","Sad","I dont know"],
               2,1),
                    facial_expressions(Image.open('FacialExp_images/Disgusted.
↪png'),"How does Gray feel today? ", ["Disgusted","Happy","Afraid","I don't␣
↪know"],
               0,1)]

               #Where answers will be placed
        user_answers = {}
        f_test = random.shuffle(question) #Randomize questions

        user_answers = {"Incorrect Answers":0,"Invalid Answers":0, "Correct␣
↪Answers":0}
        def run_fexpressions(f_test):
            global user_answers
            global activities_passed
            score = 0   #sets base score
            for question_all in question: #runs through each question
                print("")
                print(question_all.question) #goes through all acceptable␣
↪questions

                for image in question_all.ok_answers: #Calls images assigned to␣
↪each question
                    get_image = np.asarray(question_all.images)
                    image_plot = plt.imshow(get_image)
                    plt.xticks([])
                    plt.yticks([])
                plt.show()

                q = [x for x in range(len(question_all.ok_answers))] #runs␣
↪through answers
                random.shuffle(q)
                correct_answer = q.index(question_all.correct_answer)
                for ix in range(len(question_all.ok_answers)):
                    print("({:s}) {:s}".format("ABCD"[ix],question_all.
↪ok_answers[q[ix]]))
                not_answer = True
                while not_answer:
                    sel = input("Your answer is: ").lower().strip()
                    if len(sel) != 1:
                        print("Please, answers should be 1 character (ex:a,b, c␣
↪or d)")

                        user_answers["Invalid Answers"] += 1
                        continue
                    if not (sel in "abcd"):
                        print("Oops, that's not an option!")
                        user_answers["Invalid Answers"] += 1
                        continue
                    if sel == "abcd"[correct_answer]:
                        print("Great job! You get one point!")
                        not_answer = False
                        score += 1
                        user_answers["Correct Answers"] += 1
                        scores["Expressions"] += 1
                    else:
                        print("Nope, better luck next time!")
                        user_answers["Incorrect Answers"] += 1
                        not_answer = False


            with open('puzzle_FExscores.csv', 'a', newline='') as out_file:
                writer = csv.writer(out_file)
                #writer.writerow(["Date", "User ID", "Correct Answers",␣
↪"Incorrect Answers", "Invalid Answers"])
                writer.writerow([timestamp,userID, user_answers["Correct␣
↪Answers"], user_answers["Incorrect Answers"], user_answers["Invalid␣
↪Answers"]])

            print("Your final score is: " + str(score) + "/" +␣
↪str(sum(quiz_points.points for quiz_points in question)))

        # checks to see if user meets criteria
            if user_answers["Correct Answers"] >= 6:
                activities_passed += 1
                #shows image
                plt.imshow(prize_image[activities_passed])
                plt.xticks([])
                plt.yticks([])
                plt.show()
            elif user_answers["Correct Answers"] <= 5:
                print("Sorry! Could not unlock a puzzle piece. Try again!")


        run_fexpressions(f_test)
        user_answers
        scores
        loop = True
        print()

    elif choice == "D" or choice =="d":
        print("\033[1m"+ " --- School Situation Game! ---"+"\033[0m")
        class quest:
            def __init__(self,image,questions,acceptable_answers,␣
↪correct_answer, points, user_answers):
                self.image = image
                self.questions = questions
                self.acceptable_answers = acceptable_answers
                self.correct_answer = correct_answer
                self.points = points
                self.user_answers = user_answers
        questions = [
            quest(Image.open("school_social/homework_prob1.jpg"),"What to do␣
↪with hard homework ?",[ "Ask for help", "Guess", "Ignore it","I don't know"],
                 0, 1,{}),
            quest(Image.open("school_social/group_game.jpg"),"If you want to␣
↪play in a group game." ,["Ask to join in","Stand and observe","Run in", "I␣
↪don't know"],
            0 ,1,{}),
            quest(Image.open("school_social/ask_for.jpg"), "When you ask for␣
↪something, you should say" ,["Please","Hola","Goodbye", "I don't know"],
                  0, 1,{}),
            quest(Image.open("school_social/injured.jpg"),"If you scrape your␣
↪knee at school, who can help?" ,[ "A teacher","a dog","the president", "I␣
↪don't know"],
                0, 1,{}),
            quest(Image.open("school_social/teach_stud.jpg"),"What would you do␣
↪if you get in trouble",["Say 'sorry' ","Ignore it","Skip class", "I don't␣
↪know"],
                 0, 1,{})
            ]

        answers = []
        quest_test = random.shuffle(questions)

        user_answers = {"Incorrect Answers":0,"Invalid Answers":0, "Correct␣
↪Answers":0}
        def run_test(quest_test):
            global activities_passed
            global user_answers
            score = 0
            #add the dictionary here to store user answer#
            for question_all in questions: #asks user each question
                #user_answer = input(question_all.questions).lower().strip()
                print("")
                print(question_all.questions)
                for image in question_all.acceptable_answers:
                    get_image = np.asarray(question_all.image)
                    image_plot = plt.imshow(get_image)
                    plt.xticks([])
                    plt.yticks([])
                plt.show()
                q = [x for x in range(len(question_all.acceptable_answers))]
                random.shuffle(q)
                correct_answer = q.index(question_all.correct_answer)
                for ix in range(len(question_all.acceptable_answers)):
                    print("({:s}) {:s}".format("ABCD"[ix],question_all.
↪acceptable_answers[q[ix]]))
                not_answer = True
                while not_answer:
                    sel = input("Your answer:").lower().strip()
                    if len(sel) != 1:
                        print("answers should be 1 character")
                        user_answers["Invalid Answers"] += 1
                        continue
                    if not (sel in "abcd"):
                        print("letter not one of choices")
                        user_answers["Invalid Answers"] += 1
                        continue
                    if sel == "abcd"[correct_answer]:
                        print("correct!")
                        not_answer = False
                        score += 1
                        scores["School"] += 1
                        user_answers["Correct Answers"] += 1
                    else:
                        print("Better Luck Next Time")
                        user_answers["Incorrect Answers"] += 1
                        not_answer = False
            with open('puzzle_SSscores.csv', 'a', newline='') as out_file:
                writer = csv.writer(out_file)
                #writer.writerow(["Date", "User ID", "Correct Answers",␣
↪"Incorrect Answers", "Invalid Answers"])
                writer.writerow([timestamp,userID, user_answers["Correct␣
↪Answers"], user_answers["Incorrect Answers"], user_answers["Invalid␣
↪Answers"]])

            print("Your score is: " + str(score) + "/" + str(sum(quiz_points.
↪points for quiz_points in questions)))

            if user_answers["Correct Answers"] >= 4:
                activities_passed += 1
                #shows image
                plt.imshow(prize_image[activities_passed])
                plt.xticks([])
                plt.yticks([])
                plt.show()
            elif user_answers["Correct Answers"] <= 3:
                print("Sorry! Could not unlock a puzzle piece. Try again!")

        run_test(quest_test)
        user_answers
        scores
        loop = True
        print()

    elif choice == "E" or choice == "e":
        print( "\033[1m"+ "--- Emotions Game! ---"+"\033[0m")
        class e_quest:
            def __init__(self, Emotion_qs, images, acceptable_answers,␣
↪correct_answer, points):
                self.Emotion_qs = Emotion_qs
                self.images = images
                self.acceptable_answers = acceptable_answers
                self.correct_answer = correct_answer #correct answer
                self.points = points
                    #add correct answer
        Emotion_qs = [
                    e_quest("When I feel grumpy, I can: ", Image.open("Images/
↪Emotion qs/grumpyEmoji.jpg"),["Throw a tantrum", "Take a nap", "Punch a␣
↪wall", "I don't know"], 1, 1),
                    e_quest("When I feel worried, I can: ", Image.open("Images/
↪Emotion qs/anxiousEmoji.jpg"), ["Count to 10 and take deep breaths", "Run␣
↪away", "Scream", "I don't know"],0, 1),
                    e_quest("When I feel bored, I can: ", Image.open("Images/
↪Emotion qs/boredEmoji.jpg"), ["Play a game", "Do nothing", "Cry", "I don't␣
↪know"],0, 1),
                    e_quest("When I feel caring, I can: ", Image.open("Images/
↪Emotion qs/lovingEmoji.jpg"), ["Shout at someone", "Walk away", "Give a␣
↪hug", "I don't know"],2, 1),
                    e_quest("When I feel curious, I can: ", Image.open("Images/
↪Emotion qs/curiousEmoji.jpg"), ["Ask a question", "Bite my nails", "Ignore␣
↪the task", "I don't know"],0, 1),
                    e_quest("When I feel safe, I can: ", Image.open("Images/
↪Emotion qs/safeEmoji.jpg"), ["Relax", "Run away", "Scream", "I don't␣
↪know"],0, 1),
                    e_quest("When I feel jealous, I can: ", Image.open("Images/
↪Emotion qs/jealousEmoji.jpg"), ["List out things I'm good at", "Think about␣
↪what you don't have", "Scream", "I don't know"],0, 1),
                    e_quest("When I feel lazy, I can: ", Image.open("Images/
↪Emotion qs/lazyEmoji.jpg"), ["Procrastinate", "Watch Youtube/Netflix",␣
↪"Cry", "I don't know"], 1, 1)
                ]

        Emotion_a = []
        e_test = random.shuffle(Emotion_qs)

        user_answers = {"Incorrect Answers":0,"Invalid Answers":0, "Correct␣
↪Answers":0}
        def run_emotions(e_test):
            global activities_passed
            global user_answers
            score = 0
            #add the dictionary here to store user answer#
            for question_all in Emotion_qs: #asks user each question
                 #user_answer = input(question_all.questions).lower().strip()
                print("")
                print(question_all.Emotion_qs)
                q = [x for x in range(len(question_all.acceptable_answers))]
                random.shuffle(q)
                 #iterate through all answers and display image
                for image in question_all.acceptable_answers:
                    get_image = np.asarray(question_all.images)
                    image_plot = plt.imshow(get_image)
                    plt.xticks([])
                    plt.yticks([])
                plt.show()
                q = [x for x in range(len(question_all.acceptable_answers))]
                random.shuffle(q)
                correct_answer = q.index(question_all.correct_answer)
                for ix in range(len(question_all.acceptable_answers)):
                    print("({:s}) {:s}".format("ABCD"[ix],question_all.
↪acceptable_answers[q[ix]]))
                not_answer = True
                while not_answer:
                    sel = input("Your answer:").lower().strip()
                    if len(sel) != 1:
                        print("answers should be 1 character")
                        user_answers["Invalid Answers"] += 1
                        continue
                    if not (sel in "abcd"):
                        print("Oops, that's not an option!")
                        continue
                        user_answers["Invalid Answers"] += 1
                    if sel == "abcd"[correct_answer]:
                        print("Great job! You get one point!")
                        not_answer = False
                        score += 1
                        scores["Emotions"] += 1
                        user_answers["Correct Answers"] += 1
                    else:
                        print("Nope, better luck next time!")
                        user_answers["Incorrect Answers"] += 1
                        not_answer = False
            with open('puzzle_Escores.csv', 'a', newline='') as out_file:
                writer = csv.writer(out_file)
                #writer.writerow(["Date", "User ID", "Correct Answers",␣
↪"Incorrect Answers", "Invalid Answers"])
                writer.writerow([timestamp,userID, user_answers["Correct␣
↪Answers"], user_answers["Incorrect Answers"], user_answers["Invalid␣
↪Answers"]])

            print("Your score is: " + str(score) + "/" + str(sum(quiz_points.
↪points for quiz_points in Emotion_qs)))
```

```
        # checks to see if user meets criteria
            if user_answers["Correct Answers"] >= 6:
                activities_passed += 1
                #shows image
                plt.imshow(prize_image[activities_passed])
                plt.xticks([])
                plt.yticks([])
                plt.show()
            elif user_answers["Correct Answers"] <= 5:
                print("Sorry! Could not unlock a puzzle piece. Try again!")

        run_emotions(e_test)
        user_answers
        scores
        loop = True
        print()


    elif choice=="F" or choice=="f":
        print()
        print("Leaving Puzzle Now.. Thank you for playing!!")
        print()
        loop = False
    else:
        print("You must only select A - F")
        print("Please try again")
        print(print_menu())

print(print_menu())
```

```
------------------------------    Welcome to Puzzle
------------------------------

        Which game would you like to play:

                      A) Patterns Game
                      B) Social Situation Game
                      C) Facial Expressions
                      D) School Situation Game
                      E) Emotions Game
                      F) Leave Puzzle Game

-------------------------------------------------------------------------

what is your 4-digit ID: 22
Sorry, it has to be 4 digits!
```

## 2 Reward System Process

```
[ ]: #Process for creating puzzle images for different scenarios
     im = Image.open('Prize_Images/Prize.png')
     plt.imshow(im)
     ax = plt.gca()

     ax.axes.xaxis.set_visible(False)
     ax.axes.yaxis.set_visible(False)

     plt.savefig('Prize_Images/final_pp', dpi = 100)
```

```
[ ]: #puzzle piece
     im = Image.open('Prize_Images/Prize.png')
     plt.imshow(im)
     ax = plt.gca()

     #Adds fifth puzzle piece
     puzzle_piece = patches.Rectangle((0,0),
                      414,
                      624,
                      fill = True, color='firebrick')

     ax.add_patch(puzzle_piece)
     #plt.text(200,350,'Emotions\nGame',color = 'white')

     #Removes the axis
     ax.axes.xaxis.set_visible(False)
     ax.axes.yaxis.set_visible(False)
     plt.savefig('Prize_Images/five_pp.png')
     plt.show()
```

```
[ ]: #puzzle piece
     im = Image.open('Prize_Images/Prize.png')
     plt.imshow(im)
     ax = plt.gca()

     #Adds fifth puzzle piece
     puzzle_piece = patches.Rectangle((0,0),
                      414,
                      624,
                      fill = True, color='firebrick')

     ax.add_patch(puzzle_piece)
     #plt.text(200,350,'Emotions\nGame',color = 'white')
```

```
#Adds fourth puzzle piece
puzzle_piece = patches.Rectangle((414,0),
                 414,
                 624,
                 fill = True, color='blue')


ax.add_patch(puzzle_piece)
#Removes the axis
ax.axes.xaxis.set_visible(False)
ax.axes.yaxis.set_visible(False)
plt.savefig('Prize_Images/four_pp.png')
plt.show()
```

[ ]:
```
#os.chmod("Emotions_pp.png",0o666)
#Emotions puzzle piece
im = Image.open('Prize_Images/Prize.png')
plt.imshow(im)
ax = plt.gca()

#Adds fifth puzzle piece
puzzle_piece = patches.Rectangle((0,0),
                 414,
                 624,
                 fill = True, color='firebrick')

ax.add_patch(puzzle_piece)
#plt.text(200,350,'Emotions\nGame',color = 'white')
#Removes the axis
ax.axes.xaxis.set_visible(False)
ax.axes.yaxis.set_visible(False)

#Adds fourth puzzle piece
puzzle_piece = patches.Rectangle((414,0),
                 414,
                 624,
                 fill = True, color='blue')


ax.add_patch(puzzle_piece)

#Adds third puzzle piece
puzzle_piece = patches.Rectangle((828,0),
                 414,
                 624,
                 fill = True, color='blueviolet')
```

```python
ax.add_patch(puzzle_piece)
plt.savefig('Prize_Images/three_pp.png')
plt.show()
```

[ ]:
```python
#Opens original image
im = Image.open('Prize_Images/Prize.png')
plt.imshow(im)
ax = plt.gca()

#Adds fifth puzzle piece
puzzle_piece = patches.Rectangle((0,0),
                 414,
                 624,
                 fill = True, color='firebrick')

ax.add_patch(puzzle_piece)
#plt.text(200,350,'Emotions\nGame',color = 'white')

#Adds fourth puzzle piece
puzzle_piece = patches.Rectangle((414,0),
                 414,
                 624,
                 fill = True, color='blue')


ax.add_patch(puzzle_piece)

#Adds third puzzle piece
puzzle_piece = patches.Rectangle((828,0),
                 414,
                 624,
                 fill = True, color='blueviolet')
ax.add_patch(puzzle_piece)

#Adds second puzzle piece
puzzle_piece = patches.Rectangle((0,624),
                 622,
                 622,
                 fill = True, color='seagreen')

ax.add_patch(puzzle_piece)

#Adds last puzzle piece


#Removes the axis
ax.axes.xaxis.set_visible(False)
ax.axes.yaxis.set_visible(False)
plt.savefig('Prize_Images/two_pp.png')
plt.show()
```

[ ]:
```
#Opens original image
im = Image.open('Prize_Images/Prize.png')
plt.imshow(im)
ax = plt.gca()

#Adds fifth puzzle piece
puzzle_piece = patches.Rectangle((0,0),
                 414,
                 624,
                 fill = True, color='firebrick')
ax.add_patch(puzzle_piece)
#plt.text(200,350,'Emotions\nGame',color = 'white')

#Adds fourth puzzle piece
puzzle_piece = patches.Rectangle((414,0),
                 414,
                 624,
                 fill = True, color='blue')
ax.add_patch(puzzle_piece)

#Adds third puzzle piece
puzzle_piece = patches.Rectangle((828,0),
                 414,
                 624,
                 fill = True, color='blueviolet')
ax.add_patch(puzzle_piece)

#Adds second puzzle piece
puzzle_piece = patches.Rectangle((0,624),
                 622,
                 622,
                 fill = True, color='seagreen')
ax.add_patch(puzzle_piece)

puzzle_piece = patches.Rectangle((624,624),
                 622,
                 1244,
                 fill = True, color='coral')
ax.add_patch(puzzle_piece)

#Removes the axis
ax.axes.xaxis.set_visible(False)
ax.axes.yaxis.set_visible(False)
```

```
plt.savefig('Prize_Images/first_pp.png')
plt.show()
```