**Name**: Steve Hommy

**Pair: -**

**Amount of completed tasks: 7**

**Which tasks were left undone or incomplete: 0**

Self-assessment:

This exercise was easy for me because I have been using dictionary for a while now so there was no issue for me. Doing this exercise, I learned how to loop through dictionary and printing key and value. I understood everything and I managed to do everything.

## Test report

Write the test report yourself to each coding task (task number, input/action, desired output and then the testing evidence (actual output)). Add rows if necessary. Include answers to theoretical questions and pseudocode to this return document as well in addition to code screen captures. Actual output can be a screen capture of the terminal showing the output.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Input / action** | **Desired output** | **Actual output (use red color if desired output != actual output)** |
| **3** | User inputs: 3 | How many dices will be rolled? 3  Player1 rolling...  Player1 dices are  1 2 3  Player2 rolling...  Player2 dices are  6 1 6  Player1 sum of dices are 6  Player2 sum of dices are 13  Player2 has won! | How many dices will be rolled? 3  Player1 rolling...  Player1 dices are  1 2 3  Player2 rolling...  Player2 dices are  6 1 6  Player1 sum of dices are 6  Player2 sum of dices are 13  Player2 has won! |
| **4** | Run program | Player1 status:  First name: Steve  Last name: Hommy  ID: 1  Player2 status:  First name: Tom  Last name: Cruise  ID: 2  Player1 and player2 roll their dices...  player with ID: 1  Rolled: 2  player with ID: 2  Rolled: 3 | Player1 status:  First name: Steve  Last name: Hommy  ID: 1  Player2 status:  First name: Tom  Last name: Cruise  ID: 2  Player1 and player2 roll their dices...  player with ID: 1  Rolled: 2  player with ID: 2  Rolled: 3 |
| **5** | Run program | Student1 status:  First name: Steve  Last name: Hommy  ID: 1  Student2 status:  First name: Tom  Last name: Cruise  ID: 2  Student with ID: 1  Has this mammal:  Name: Bob  Species: Dog  Size: 60  Weight: 30  ID: 1  Student with ID: 2  Has this mammal:  Name: Snuf  Species: Cat  Size: 40  Weight: 20  ID: 2 | Student1 status:  First name: Steve  Last name: Hommy  ID: 1  Student2 status:  First name: Tom  Last name: Cruise  ID: 2  Student with ID: 1  Has this mammal:  Name: Bob  Species: Dog  Size: 60  Weight: 30  ID: 1  Student with ID: 2  Has this mammal:  Name: Snuf  Species: Cat  Size: 40  Weight: 20  ID: 2 |
| **6** | Run program | Student1 status:  First name: Steve  Last name: Hommy  ID: 1  Student2 status:  First name: Tom  Last name: Cruise  ID: 2  Steve  Rolled: 8  Tom  Rolled: 6  Steve Rolled higher number so it will get heavier mammal  Steve mammal is:  Name: Bob  Species: Dog  Size: 60  Weight: 30  ID: 1  Tom mammal is:  Name: Snuf  Species: Cat  Size: 40  Weight: 20  ID: 2 | Student1 status:  First name: Steve  Last name: Hommy  ID: 1  Student2 status:  First name: Tom  Last name: Cruise  ID: 2  Steve  Rolled: 8  Tom  Rolled: 6  Steve Rolled higher number so it will get heavier mammal  Steve mammal is:  Name: Bob  Species: Dog  Size: 60  Weight: 30  ID: 1  Tom mammal is:  Name: Snuf  Species: Cat  Size: 40  Weight: 20  ID: 2 |
| **7** | Run program | Let's test that a single card works...  12 of Hearts  Card is 12 of Hearts  Single card testing is over.  Let's test that a deck of card is created...  1 of Spades  2 of Spades  3 of Spades  4 of Spades  5 of Spades  6 of Spades  7 of Spades  8 of Spades  9 of Spades  10 of Spades  11 of Spades  12 of Spades  13 of Spades  1 of Clubs  2 of Clubs  3 of Clubs  4 of Clubs  5 of Clubs  6 of Clubs  7 of Clubs  8 of Clubs  9 of Clubs  10 of Clubs  11 of Clubs  12 of Clubs  13 of Clubs  1 of Diamonds  2 of Diamonds  3 of Diamonds  4 of Diamonds  5 of Diamonds  6 of Diamonds  7 of Diamonds  8 of Diamonds  9 of Diamonds  10 of Diamonds  11 of Diamonds  12 of Diamonds  13 of Diamonds  1 of Hearts  2 of Hearts  3 of Hearts  4 of Hearts  5 of Hearts  6 of Hearts  7 of Hearts  8 of Hearts  9 of Hearts  10 of Hearts  11 of Hearts  12 of Hearts  13 of Hearts  Card deck testing is over.  Let's shuffle the deck.  Let's test that a deck of card is shuffled...  4 of Clubs  3 of Diamonds  7 of Hearts  9 of Spades  1 of Hearts  13 of Spades  11 of Clubs  12 of Clubs  10 of Hearts  10 of Spades  8 of Diamonds  11 of Diamonds  13 of Hearts  1 of Spades  7 of Clubs  2 of Clubs  9 of Clubs  5 of Diamonds  5 of Spades  4 of Hearts  4 of Spades  9 of Hearts  13 of Clubs  3 of Clubs  2 of Hearts  5 of Hearts  10 of Clubs  8 of Hearts  8 of Clubs  6 of Clubs  2 of Diamonds  3 of Hearts  5 of Clubs  6 of Spades  1 of Clubs  3 of Spades  4 of Diamonds  11 of Spades  10 of Diamonds  6 of Hearts  13 of Diamonds  12 of Spades  11 of Hearts  6 of Diamonds  1 of Diamonds  8 of Spades  2 of Spades  9 of Diamonds  7 of Spades  7 of Diamonds  12 of Hearts  12 of Diamonds  Cards should be suffled now.  Let's draw 2 cards and show them.  You draw:  12 of Diamonds  Your opponent draw:  12 of Hearts  Draw 3 cards and highest value wins  Here are the cards that have been draw  7 of Diamonds  7 of Spades  9 of Diamonds  Winner is  9 of Diamonds | Let's test that a single card works...  12 of Hearts  Card is 12 of Hearts  Single card testing is over.  Let's test that a deck of card is created...  1 of Spades  2 of Spades  3 of Spades  4 of Spades  5 of Spades  6 of Spades  7 of Spades  8 of Spades  9 of Spades  10 of Spades  11 of Spades  12 of Spades  13 of Spades  1 of Clubs  2 of Clubs  3 of Clubs  4 of Clubs  5 of Clubs  6 of Clubs  7 of Clubs  8 of Clubs  9 of Clubs  10 of Clubs  11 of Clubs  12 of Clubs  13 of Clubs  1 of Diamonds  2 of Diamonds  3 of Diamonds  4 of Diamonds  5 of Diamonds  6 of Diamonds  7 of Diamonds  8 of Diamonds  9 of Diamonds  10 of Diamonds  11 of Diamonds  12 of Diamonds  13 of Diamonds  1 of Hearts  2 of Hearts  3 of Hearts  4 of Hearts  5 of Hearts  6 of Hearts  7 of Hearts  8 of Hearts  9 of Hearts  10 of Hearts  11 of Hearts  12 of Hearts  13 of Hearts  Card deck testing is over.  Let's shuffle the deck.  Let's test that a deck of card is shuffled...  4 of Clubs  3 of Diamonds  7 of Hearts  9 of Spades  1 of Hearts  13 of Spades  11 of Clubs  12 of Clubs  10 of Hearts  10 of Spades  8 of Diamonds  11 of Diamonds  13 of Hearts  1 of Spades  7 of Clubs  2 of Clubs  9 of Clubs  5 of Diamonds  5 of Spades  4 of Hearts  4 of Spades  9 of Hearts  13 of Clubs  3 of Clubs  2 of Hearts  5 of Hearts  10 of Clubs  8 of Hearts  8 of Clubs  6 of Clubs  2 of Diamonds  3 of Hearts  5 of Clubs  6 of Spades  1 of Clubs  3 of Spades  4 of Diamonds  11 of Spades  10 of Diamonds  6 of Hearts  13 of Diamonds  12 of Spades  11 of Hearts  6 of Diamonds  1 of Diamonds  8 of Spades  2 of Spades  9 of Diamonds  7 of Spades  7 of Diamonds  12 of Hearts  12 of Diamonds  Cards should be suffled now.  Let's draw 2 cards and show them.  You draw:  12 of Diamonds  Your opponent draw:  12 of Hearts  Draw 3 cards and highest value wins  Here are the cards that have been draw  7 of Diamonds  7 of Spades  9 of Diamonds  Winner is  9 of Diamonds |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Multiple choice:

a. The \_\_\_\_\_ method is automatically called when an object is created.

i. \_\_init\_\_

ii. init

iii. \_\_str\_\_

iv. \_\_object\_\_

b. The \_\_\_\_\_ programming practice is centered on creating functions that are separated from the data that they work on.

i. modular

ii. procedural

iii. functional

iv. object-oriented

c. The \_\_\_\_\_ programming practice is centered on creating objects.

i. object-centric

ii. objective

iii. procedural

iv. object-oriented

d. A(n) \_\_\_\_\_ is a component of a class that references data

i. method

ii. instance

iii. data attribute

iv. module

e. By doing this, you can hide a class’s attribute from code outside the class.

i. avoiding using the self-parameter to create the attribute

ii. begin the attribute’s name with private\_\_

iii. begin the name of the attribute with two underscores

iv. begin the name of the attribute with the symbol #

f. A(n) \_\_\_\_\_ method stores a value in the data attribute or changes its value in some other way.

i. modifier

ii. constructor

iii. mutator

iv. Accessor

2. Explain the following terms:

a. Super class

**The Python super() method lets you access methods in a parent class. You can think of super() as a way to jump up to view the methods in the class from which another class is inherited. The super() method does not accept any arguments.**

b. Sub class

**A class which inherits from a superclass is called a subclass, also called heir class or child class.**

c. Base class

**base class is the class being inherited from, also called parent class. They don’t contain implementation. Instead, they provide an interface and make sure that derived classes are properly implemented.**

d. Derived class

**Derived class is the class that inherits from another class, also called child class**

e. “Is a” relationship

**This means that when you have a Derived class that inherits from a Base class, you created a relationship where Derived is a specialized version of Base. Classes are represented as boxes with the class name on top.**

# Code

# File name: diceClass.py

# Author: Steve Hommy

# Description: Create a Dice Class

import random

class Dice:

    def \_\_init\_\_(self):

        self.number = 1

    def roll\_the\_dice(self):

        random\_number = random.randint(1, 6)

        self.number = random\_number

    def get\_number(self):

        return self.number

# File name: main.py

# Author: Steve Hommy

# Description: Main function file

from diceClass import Dice

number\_of\_dices = int(input("How many dices will be rolled? "))

# Rolling the dice and then looping through the list

def player1\_roll():

    global player1\_dice

    print("Player1 rolling...")

    print("Player1 dices are")

    player1\_dice = []

    for i in range(number\_of\_dices):

        dice1 = Dice()

        dice1.roll\_the\_dice()

        player1\_dice.append(dice1.number)

    for dices1 in player1\_dice:

        print(dices1, end=" ")

def player2\_roll():

    global player2\_dice

    print("\n\nPlayer2 rolling...")

    print("Player2 dices are")

    player2\_dice = []

    for i in range(number\_of\_dices):

        dice2 = Dice()

        dice2.roll\_the\_dice()

        player2\_dice.append(dice2.number)

    for dices2 in player2\_dice:

        print(dices2, end=" ")

# Sum the list

def player1\_dice\_sum():

    player1\_sum\_of\_dices = sum(player1\_dice)

    print("\n\nPlayer1 sum of dices are", player1\_sum\_of\_dices)

    return player1\_sum\_of\_dices

def player2\_dice\_sum():

    player2\_sum\_of\_dices = sum(player2\_dice)

    print("Player2 sum of dices are", player2\_sum\_of\_dices)

    return player2\_sum\_of\_dices

# While the loop is true keep running until it meets break statement

def main():

    while True:

        player1\_roll()

        player2\_roll()

        if player1\_dice\_sum() == player2\_dice\_sum():

            print("It's a tie. Both players have to roll again\n")

        elif player1\_dice\_sum() > player2\_dice\_sum():

            print("Player1 has won!")

            break

        else:

            print("Player2 has won!")

            break

main()

# File name: diceClass.py

# Author: Steve Hommy

# Description: Create a Dice Class

import random

class Dice:

    def \_\_init\_\_(self):

        self.\_\_number = 1

    def roll\_the\_dice(self):

        random\_number = random.randint(1, 6)

        self.\_\_number = int(random\_number)

    def get\_number(self):

        return self.\_\_number

# File name: playerClass.py

# Author: Steve Hommy

# Description: Create a Player Class

class Player:

    def \_\_init\_\_(self, first\_name, last\_name, id):

        self.\_\_first\_name = first\_name

        self.\_\_last\_name = last\_name

        self.\_\_id = int(id)

    def \_\_str\_\_(self):

        return f"""

        First name: {self.\_\_first\_name}

        Last name: {self.\_\_last\_name}

        ID: {self.\_\_id}

        """

    def set\_first\_name(self, first\_name):

        self.\_\_first\_name = first\_name

    def set\_last\_name(self, last\_name):

        self.\_\_last\_name = last\_name

    def set\_id(self, id):

        self.\_\_id = id

    def get\_first\_name(self):

        return self.\_\_first\_name

    def get\_last\_name(self):

        return self.\_\_last\_name

    def get\_id(self):

        return self.\_\_id

# File name: main.py

# Author: Steve Hommy

# Description: Main function file

from playerClass import Player

from diceClass import Dice

def main():

    # Giving values to object

    player1 = Player("Steve", "Hommy", 1)

    player2 = Player("Tom", "Cruise", 2)

    print("Player1 status:", player1)

    print("Player2 status:", player2)

    dice1 = Dice()

    dice2 = Dice()

    dice1.roll\_the\_dice()

    dice2.roll\_the\_dice()

    print("Player1 and player2 roll their dices...\n")

    # Creating dictionary where player id is the key and dice number is value

    player\_dict = {

        player1.get\_id(): dice1.get\_number(),

        player2.get\_id(): dice2.get\_number()

    }

    # looping through dictionary and printing out the key and the value

    for key in player\_dict:

        print("player with ID:", key, "\nRolled:", player\_dict[key])

main()

# File name: mammalClass.py

# Author: Steve Hommy

# Description: Create a Mammal Class

class Mammal:

    def \_\_init\_\_(self, name, species, size, weight, id):

        self.\_\_name = name

        self.\_\_species = species

        self.\_\_size = int(size)

        self.\_\_weight = int(weight)

        self.\_\_id = int(id)

    def \_\_str\_\_(self):

        return f"""

        Name: {self.\_\_name}

        Species: {self.\_\_species}

        Size: {self.\_\_size}

        Weight: {self.\_\_weight}

        ID: {self.\_\_id}

        """

    def set\_name(self, name):

        self.\_\_name = name

    def set\_species(self, species):

        self.\_\_species = species

    def set\_size(self, size):

        self.\_\_size = size

    def set\_weight(self, weight):

        self.\_\_weight = weight

    def set\_id(self, id):

        self.\_\_id = id

    def get\_name(self):

        return self.\_\_name

    def get\_species(self):

        return self.\_\_species

    def get\_size(self):

        return self.\_\_size

    def get\_weight(self):

        return self.\_\_weight

    def get\_id(self):

        return self.\_\_id

# File name: studentClass.py

# Author: Steve Hommy

# Description: Create a Student Class

class Student:

    def \_\_init\_\_(self, first\_name, last\_name, id):

        self.\_\_first\_name = first\_name

        self.\_\_last\_name = last\_name

        self.\_\_id = int(id)

    def \_\_str\_\_(self):

        return f"""

        First name: {self.\_\_first\_name}

        Last name: {self.\_\_last\_name}

        ID: {self.\_\_id}

        """

    def set\_first\_name(self, first\_name):

        self.\_\_first\_name = first\_name

    def set\_last\_name(self, last\_name):

        self.\_\_last\_name = last\_name

    def set\_id(self, id):

        self.\_\_id = id

    def get\_first\_name(self):

        return self.\_\_first\_name

    def get\_last\_name(self):

        return self.\_\_last\_name

    def get\_id(self):

        return self.\_\_id

# File name: main.py

# Author: Steve Hommy

# Description: Main function file

from studentClass import Student

from mammalClass import Mammal

def main():

    # Giving values to object

    student1 = Student("Steve", "Hommy", 1)

    student2 = Student("Tom", "Cruise", 2)

    print("Student1 status:", student1)

    print("Student2 status:", student2)

    mammal1 = Mammal("Bob", "Dog", 60, 30, 1)

    mammal2 = Mammal("Snuf", "Cat", 40, 20, 2)

    # Creating dictionary where student id is the key and mammal is value

    student\_dict = {

        student1.get\_id(): mammal1,

        student2.get\_id(): mammal2

    }

    # looping through dictionary and printing out the key and the value

    for key in student\_dict:

        print("Student with ID:", key, "\nHas this mammal:", student\_dict[key])

main()

# File name: diceClass.py

# Author: Steve Hommy

# Description: Create a Dice Class

import random

class Dice:

    def \_\_init\_\_(self):

        self.\_\_number = 1

    def roll\_the\_dice(self):

        random\_number = random.randint(1, 6)

        self.\_\_number = int(random\_number)

    def get\_number(self):

        return self.\_\_number

# File name: mammalClass.py

# Author: Steve Hommy

# Description: Create a Mammal Class

class Mammal:

    def \_\_init\_\_(self, name, species, size, weight, id):

        self.\_\_name = name

        self.\_\_species = species

        self.\_\_size = int(size)

        self.\_\_weight = int(weight)

        self.\_\_id = int(id)

    def \_\_str\_\_(self):

        return f"""

        Name: {self.\_\_name}

        Species: {self.\_\_species}

        Size: {self.\_\_size}

        Weight: {self.\_\_weight}

        ID: {self.\_\_id}

        """

    def set\_name(self, name):

        self.\_\_name = name

    def set\_species(self, species):

        self.\_\_species = species

    def set\_size(self, size):

        self.\_\_size = size

    def set\_weight(self, weight):

        self.\_\_weight = weight

    def set\_id(self, id):

        self.\_\_id = id

    def get\_name(self):

        return self.\_\_name

    def get\_species(self):

        return self.\_\_species

    def get\_size(self):

        return self.\_\_size

    def get\_weight(self):

        return self.\_\_weight

    def get\_id(self):

        return self.\_\_id

# File name: studentClass.py

# Author: Steve Hommy

# Description: Create a Student Class

class Student:

    def \_\_init\_\_(self, first\_name, last\_name, id):

        self.\_\_first\_name = first\_name

        self.\_\_last\_name = last\_name

        self.\_\_id = int(id)

    def \_\_str\_\_(self):

        return f"""

        First name: {self.\_\_first\_name}

        Last name: {self.\_\_last\_name}

        ID: {self.\_\_id}

        """

    def set\_first\_name(self, first\_name):

        self.\_\_first\_name = first\_name

    def set\_last\_name(self, last\_name):

        self.\_\_last\_name = last\_name

    def set\_id(self, id):

        self.\_\_id = id

    def get\_first\_name(self):

        return self.\_\_first\_name

    def get\_last\_name(self):

        return self.\_\_last\_name

    def get\_id(self):

        return self.\_\_id

# File name: main.py

# Author: Steve Hommy

# Description: Main function file

from studentClass import Student

from mammalClass import Mammal

from diceClass import Dice

def dice\_roll():

    global student1\_dice, student2\_dice

    student1\_dice = 0

    student2\_dice = 0

    # Looping twice and rolling the dice twice

    for i in range(0, 2):

        dice1 = Dice()

        dice2 = Dice()

        dice1.roll\_the\_dice()

        dice2.roll\_the\_dice()

        student1\_dice += dice1.get\_number()

        student2\_dice += dice2.get\_number()

# Creating dictionary where student name is the key and sum of the value is value

def student\_dictionary():

    student\_dict = {

        student1.get\_first\_name(): student1\_dice,

        student2.get\_first\_name(): student2\_dice

    }

    # looping through dictionary and printing out the key and the value

    for key in student\_dict:

        print(key, "\nRolled:", student\_dict[key])

def main():

    global student1, student2

    # Giving values to object

    student1 = Student("Steve", "Hommy", 1)

    student2 = Student("Tom", "Cruise", 2)

    print("Student1 status:", student1)

    print("Student2 status:", student2)

    mammal1 = Mammal("Bob", "Dog", 60, 30, 1)

    mammal2 = Mammal("Snuf", "Cat", 40, 20, 2)

    dice\_roll()

    student\_dictionary()

    # While the loop is true keep running until it meets break statement

    while True:

        if student1\_dice == student2\_dice:

            print("It's a tie. Both players have to roll again\n")

            dice\_roll()

            student\_dictionary()

        elif student1\_dice > student2\_dice:

            print(student1.get\_first\_name(), "Rolled higher number so it will get heavier mammal")

            print(student1.get\_first\_name(), "mammal is:", mammal1)

            print(student2.get\_first\_name(), "mammal is:", mammal2)

            break

        else:

            print(student2.get\_first\_name(), "Rolled higher number so it will get heavier mammal")

            print(student1.get\_first\_name(), "mammal is:", mammal2)

            print(student2.get\_first\_name(), "mammal is:", mammal1)

            break

main()

# File: card.py

# Author: Steve Hommy

# Description: Create a Card Class

class Card:

    def \_\_init\_\_(self, suit, val):

        self.suit = suit

        self.value = val

    def \_\_str\_\_(self):

        return f"Card is {self.value} of {self.suit}"

    def show\_card(self):

        print("{} of {}".format(self.value, self.suit))

# File: deck.py

# Author: Steve Hommy

# Description: Create a Deck Class

from card import Card

import random

class Deck:

    def \_\_init\_\_(self):

        self.cards = []

        self.build()

    def build(self):

        for s in ["Spades", "Clubs", "Diamonds", "Hearts"]:

            for v in range(1, 14):

                self.cards.append(Card(s, v))

    def show\_deck(self):

        for c in self.cards:

            c.show\_card()

    def shuffle\_deck(self):

        for i in range(len(self.cards)-1, 0, -1):

            r = random.randint(0, i)

            self.cards[i], self.cards[r] = self.cards[r], self.cards[i]

    def draw\_card(self):

        return self.cards.pop()

# File: main.py

# Author: Steve Hommy

# Description: Deck of cards and card games.

import card

import deck

def main():

    print("Let's test that a single card works...")

    my\_card = card.Card("Hearts", 12)

    my\_card.show\_card()

    print(my\_card)

    print("Single card testing is over.\n")

    print("Let's test that a deck of card is created...")

    my\_deck = deck.Deck()

    my\_deck.show\_deck()

    print("Card deck testing is over.\n")

    print("Let's shuffle the deck.")

    my\_deck.shuffle\_deck()

    print("Let's test that a deck of card is shuffled...")

    my\_deck.show\_deck()

    print("Cards should be suffled now.\n")

    print("Let's draw 2 cards and show them.")

    print("You draw:")

    card1 = my\_deck.draw\_card()

    card1.show\_card()

    print("Your opponent draw:")

    card1 = my\_deck.draw\_card()

    card1.show\_card()

    # Code your Exercise 5 taks 7 game here.

    print("Draw 3 cards and highest value wins")

    while True:

        print("Here are the cards that have been draw ")

        draw1 = my\_deck.draw\_card()

        draw2 = my\_deck.draw\_card()

        draw3 = my\_deck.draw\_card()

        draw1.show\_card(), draw2.show\_card(), draw3.show\_card()

        if draw1.value == draw2.value == draw3.value:

            print("We have to re-draw because there is a tie")

        elif draw1.value > draw2.value and draw3.value:

            print("Winner is"), draw1.show\_card()

            break

        elif draw2.value > draw1.value and draw3.value:

            print("Winner is"), draw2.show\_card()

            break

        else:

            print("Winner is"), draw3.show\_card()

        break

# Calling the main function here, do not change...

main()