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	Let's test that a single card works... 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.	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

		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.	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

		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	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
```

```python
# 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
```

```python
# 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
```

```python
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
```

```python
# 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
```

```python
# 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
```

```python
# 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
```

```python
# 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()
```