**Name**: Nico Kranni

**Pair:** No pair

**Amount of completed tasks:**

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

Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

Doing this exercise, I learned…

I am still wondering…

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

1. Multiple choice:

a. The _____ method is automatically called when an object is created.

i. __init__

b. The _____ programming practice is centered on creating functions that are

separated from the data that they work on.

ii. procedural

c. The _____ programming practice is centered on creating objects.

iv. object-oriented

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

iii. data attribute

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

iii. begin the name of the attribute with two underscores

f. A(n) _____ method stores a value in the data attribute or changes its value in

some other way.

iii. mutator

2. Explain the following terms:

a. Super class

Super class is the class you make temporary object of using super()

b. Sub class

It is a class that has inherited something from parent classes. Also known as child class.

c. Base class

Is usually referred as parent class. It may be the class that you inherit from, or you use super on it, which makes it Super class I guess.

d. Derived class

Same as sub class or child class

e. “Is a” relationship

Relationship is called connection between classes. Connection for example can be inheritance.

## 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** | Amount of dices | Person 1 wins<br>Person 2 wins<br>Person 3 wins | Person 1 wins<br>Person 2 wins<br>Person 3 wins |
| **3** | Number of dices | 1<br>1<br>1<br>3 | 5<br>6<br>6<br>6 |
| | | | |
| **4** | Mr=Person("Nico","Kranni",11)<br><br>Mrs=Person("Reina","Ketonen",9) | Player ID: 9<br>Name: Reina Ketonen<br>Dice number: 5 | Player ID: 11<br>Name: Nico Kranni<br>Dice number: 4 |
| | | | |
| **5** | kitty=Mammal("1","Cat","Cindy","2x5m","600g")<br>kitty2=Mammal("5","Cat","Mesi","2x5m","500g")<br><br>Mr=Person("Nico", "Kranni", 11)<br>Mrs=Person("Reina", "Ketonen", 9) | Student ID: 9<br>Name: Reina Ketonen<br>Pet name: Mesi<br>Pet species: Cat | Student ID: 11<br>Name: Nico Kranni<br>Pet name: Cindy<br>Pet species: Cat |
| | | | |
| **6** | List of mammals | Student ID: 9<br>Name: Reina Ketonen<br>Roll: 11<br>Pet name: Moto Moto<br>Pet species: hippopotamus | Student ID: 11<br>Name: Nico Kranni<br>Roll: 10<br>Pet name: Koda<br>Pet species: bear |
| | | | |
| **7** | Code game, where each player draws thee cards, and highest one wins. | Person 1 wins<br>Person 2 wins<br>Person 3 wins | Person 3 wins<br>Person 2 wins<br>Person 1 wins |
| | | | |
| | | | |
| | | | |
| | | | |

Task 3

```
# Description: Create four Dice objects and store them in a list.



import dice



def main():
    # Get a list of Dice objects.

    dice_list1=make_list()
    dice_list2=make_list()
    dice_list3=make_list()

    dices_list=[dice_list1,dice_list2,dice_list3]



    for list in dices_list:
        display_list(list)



    for list in dices_list:
        for item in list:
            item.roll_dice(6)
        display_list(list)

    sum1=0
    sum2=0
    sum3=0

    for value in range(0,roll_dices):
        sum1+=dice_list1[value].get_value()
    print("First person dices:",sum1)

    for value in range(0,roll_dices):
        sum2+=dice_list2[value].get_value()
    print("Second person dices:",sum2)

    for value in range(0,roll_dices):
        sum3+=dice_list3[value].get_value()
    print("Third person dices:",sum3)


    if sum1 > sum2 and sum1 >sum3:
        print("Person 1 wins")
```

```
    if sum2 > sum3 and sum2 >sum1:
        print("Person 2 wins")
    if sum3 > sum2 and sum3 >sum1:
        print("Person 3 wins")

# The display_list function acceps a list containing
# objects as an argument displays the
# data stored in each object.



def display_list(a_list):
    for item in a_list:
        print(item)
    print()

def make_list():

    # Create an empty list.
    dice_list=[]
    #Add four Dice objects to the list.
    for count in range(1,roll_dices+1):

        # dice_list.append(dice.Dice())
        # Create a new Dice object in memory and
        # assing it to the dice variable
        new_dice = dice.Dice()


        # Add the object to the list.
        dice_list.append(new_dice)

    return dice_list
roll_dices=int(input("How many dices to roll? "))
main()
```

Task 4

```
from dice import Dice

class Player:
    def __init__(self,first_name,last_name,player_id):
        self.first_name=first_name
        self.last_name=last_name
        self.player_id=player_id


    #STR
    def __str__(self):
       for key in myPlayer:
            return "\nPlayer ID: {0}\nName: {1} {2}\nDice number:
{3}\n".format(key,myPlayer[key][0],myPlayer[key][1],myPlayer[key][2])

    #Accessors
    def getFirst(self):
        print("My first name is: ", self.first_name)

    def getLast(self):
        print("My last name is: ", self.last_name)

    def getID(self):
        print(self.first_name+"'s ID is: ", self.player_id)

    def getFull(self):
        print("My name is: ", self.first_name, self.last_name)

    #Mutators
    def setFirst(self):
        self.first_name=input("Change first name to: ")

    def setLast(self):
        self.last_name=input("Change last name to: ")

    def setID(self):
        self.id=input("Set new player ID: ")


    # Creates dictionary with player_id as key
    def store(self,dice):
        self.storage={self.player_id:[self.first_name,self.last_name,dice.value]}

    def playerInformation(self):
        for key in myPlayer:
```

```
            print("Player
ID:",key,"\nName:",myPlayer[key][0],myPlayer[key][1],"\nDice
number:",myPlayer[key][2])
            print()

# New dices and roll them
dice1=Dice()
dice2=Dice()
dice1.roll_dice(6)
dice2.roll_dice(6)

# Create players
Mr=Player("Nico","Kranni",11)
Mrs=Player("Reina","Ketonen",9)

# Create dictionary and include value of dice into it
Mr.store(dice1)
Mrs.store(dice2)

# Created empty dictionary to print players info
myPlayer={}
myPlayer.update(Mr.storage)
myPlayer.update(Mrs.storage)

Mr.playerInformation()
```

Task 5

```
from mammal import Mammal
class Person:
    def __init__(self,first_name,last_name,student_id):
        self.first_name=first_name
        self.last_name=last_name
        self.student_id=student_id
        self.storage={}


    #STR
    def __str__(self):
        for key in myStudent:
            return "\nPlayer ID: {0}\nName: {1} {2}\nDice number:
{3}\n".format(key,myStudent[key][0],myStudent[key][1],myStudent[key][2])

    #Accessors
    def getFirst(self):
        print("My first name is: ", self.first_name)

    def getLast(self):
        print("My last name is: ", self.last_name)

    def getID(self):
        print(self.first_name+"'s ID is: ", self.student_id)

    def getFull(self):
        print("My name is: ", self.first_name, self.last_name)

    #Mutators
    def setFirst(self):
        self.first_name=input("Change first name to: ")

    def setLast(self):
        self.last_name=input("Change last name to: ")

    def setID(self):
        self.id=input("Set new student ID: ")

    def store(self,pet):
        self.storage={self.student_id:[self.first_name,self.last_name,pet.name,pet.
species]}

    def studentInformation(self):
        for key in myStudent:
```

```
            print("Student
ID:",key,"\nName:",myStudent[key][0],myStudent[key][1],"\nPet
name:",myStudent[key][2],"\nPet species:",myStudent[key][3])
            print()

kitty=Mammal("1","Cat","Cindy","2x5m","600g")
kitty2=Mammal("5","Cat","Mesi","2x5m","500g")

Mr=Person("Nico", "Kranni", 11)
Mrs=Person("Reina", "Ketonen", 9)

Mr.store(kitty)
Mrs.store(kitty2)

myStudent={}
myStudent.update(Mr.storage)
myStudent.update(Mrs.storage)

Mr.studentInformation()
```

Task 6

```
from dice import Dice
from mammal import *
class Person:
    def __init__(self,first_name,last_name,student_id):
        self.first_name=first_name
        self.last_name=last_name
        self.student_id=student_id
        self.storage={}


    #STR
    def __str__(self):
        for key in myStudent:
            return "\nPlayer ID: {0}\nName: {1} {2}\nDice number:
{3}\n".format(key,myStudent[key][0],myStudent[key][1],myStudent[key][2])

    #Accessors
    def getFirst(self):
        print("My first name is: ", self.first_name)

    def getLast(self):
        print("My last name is: ", self.last_name)

    def getID(self):
        print(self.first_name+"'s ID is: ", self.student_id)

    def getFull(self):
        print("My name is: ", self.first_name, self.last_name)

    #Mutators
    def setFirst(self):
        self.first_name=input("Change first name to: ")

    def setLast(self):
        self.last_name=input("Change last name to: ")

    def setID(self):
        self.id=input("Set new student ID: ")

    def store(self,roll,name,species):
        self.storage={self.student_id:[self.first_name,self.last_name,roll,name,spe
cies]}

    def studentInformation(self):
        for key in myStudent:
```

```
            print("Student
ID:",key,"\nName:",myStudent[key][0],myStudent[key][1],"\nRoll:",myStudent[key][2],
"\nPet name:",myStudent[key][3],"\nPet species:",myStudent[key][4])
            print()


#Create students
Mr=Person("Nico", "Kranni", 11)
Mrs=Person("Reina", "Ketonen", 9)

#Create two dice
noppa1=Dice()
noppa2=Dice()


# We created empty variable to store the rolls
# Then we roll the dices and store them to sum
sum1=0
noppa1.roll_dice(6)
noppa2.roll_dice(6)
sum1+=noppa1.get_value()
sum1+=noppa2.get_value()

# We take the sum and take corresponding animal/slot from list
animal=lista[sum1-1]
Mr.store(sum1,animal[2], animal[1])

sum1=0
noppa1.roll_dice(6)
noppa2.roll_dice(6)
sum1+=noppa1.get_value()
sum1+=noppa2.get_value()
animal=lista[sum1-1]



Mrs.store(sum1,animal[2], animal[1])

myStudent={}
myStudent.update(Mr.storage)
myStudent.update(Mrs.storage)
Mr.studentInformation()
```

Task 7

```
# File:         card.py
# Author:       Nico Kranni
# Description:  Card class

import random
class Card:
    def __init__(self,suit,val):
        self.suit=suit
        self.value=val
    def __str__(self):
        return "Card is {} {} ".format(self.value, self.suit)
    def show_card(self):
        print("{} of {}".format(self.value, self.suit))
```

```
# File:         card.py
# Author:       Nico Kranni
# Description:  Deck class

import random
import card
class Deck:
    def __init__(self):
        self.cards=[]
        self.build()


    def build(self):
        for suit in ["Spades","Clubs","Diamonds","Hearts"]:
            for rank in range(1,14):
                self.cards.append(card.Card(suit, rank))

    def show_deck(self):
        for c in self.cards:
            c.show_card()

    def shuffle_deck(self):
        print("Shuffling the deck....")
        for item in range(len(self.cards)-1,0,-1):
            r=random.randint(0,item)
            self.cards[item],self.cards[r] = self.cards[r], self.cards[item]

    def draw_card(self):
        return self.cards.pop()
```

```
# File:         main.py
# Author:
# 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.

    #Highest of 3 game
    def game():
        # Created empty variable to store cards
        player1=0

        # Then we draw 3 cards and add it to variable
```

```
        for i in range(1,4):
            card1=my_deck.draw_card()
            player1+=card1.value

        player2=0
        for i in range(1,4):
            card1=my_deck.draw_card()
            player2+=card1.value

        player3=0
        for i in range(1,4):
            card1=my_deck.draw_card()
            player3+=card1.value

        # Check who has highest
        if player1 > player2 and player1 >player3:
            print("Person 1 wins")
        if player2 > player3 and player2 >player1:
            print("Person 2 wins")
        if player3 > player2 and player3 >player1:
            print("Person 3 wins")
    game()

# Calling the main function here, do not change...
main()
```