Name: Nico

Pair:None

Amount of completed tasks:9

Which tasks were left undone or incomplete:8

Self-assessment:

This exercise was difficult and long as heck!

Doing this exercise, I learned not sure what, but something

I am still wondering how I managed to get it to almost work

I did not manage to do the one with colors.

1.Explain the following terms:

a. Abstraction (in programming)

It is about only showing relevant information to user. Public and private identifiers are used to hide, with classes.

b. Accessor and mutator methods

Accessor get(). It can get information from hidden object/data. It can only get the data, not change the data.

Mutator set(). This one alters the hidden data within object.

c. Public and private methods

_(underscore) is used to create method as private. It protects the method.

Public is quite the opposite; it does not use _ and it is accessible.

It should be noted, underscore needs to be at the front to make it private. After variable it only makes sure it does not conflict with python keywords.

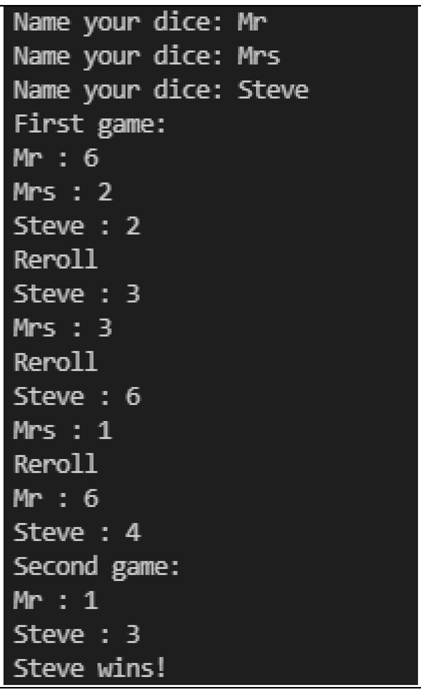
d. __str__ method (in Python)

It is used to print information about the object. But __str__ has to be defined in code beforehand.

Same usage as __repr__ but repr and str can be both called from __repr__ where __str__ can be only called by str

Test report

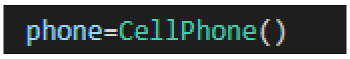
Task	**Input / action**	**Desired output**	**Actual output (use red color if desired output != actual output)**
2	User runs the program <Run the program a couple of times so that you get **each side up** at least once>	This side is up: Heads Tossing the coin… Now this side is up: <Tails> <Heads, Tails, Rabbit hole, Upright, wormhole>	This side up: Heads This side up: Tails
2	User runs the program <Run the program a couple of times so that you get **each currency** at least once.>	Currency is: Euro <Euro, Pound, Dollar, Ruble, Yen>	Current currency: Euro New currency: Ruble
3	User runs the program <Run the program a couple of times so that you get **each currency** at least once.>	Currency (original): Euro Currency (new): <Dollar> <Euro, Pound, Dollar, Ruble, Yen>	Current currency: Euro New currency: Ruble
5	User runs the program <Run the program a couple of times so that you get **each number, color and that extra feature** at least once>	Rolling the dice … number: 4 color: red extra feature: xx <1..6, all the colors, all the feature values>	Give your dice a colour: Black Name your dice: Mr This side up currently: 3 You roll the dice… This side up currently: 1 My name is: Mr
6	User runs the program <Run the program a couple of times so that you get **each possible sum** at least once>	Rolling the dice1 … number: 4 Rolling the dice2 … number: 2 The sum is: 6 The sum is <2..12>	Give your dice a colour: Blue Name your dice: Mrs Give your dice a colour: Black Name your dice: Mr You roll the dice... This side up currently: 2 You roll the dice... This side up currently: 2 Sum of the dices is: 4

7	User runs the program <Run the program a couple of times so that you get **every player to win** at least once.>	Dice rolling game First round … Player1: 6 Player2: 4 Player3: 5 Second round … Player1: 3 Player3: 5 The winner is: Player3 <Player1, Player2, Player3>	 Name your dice: Mr Name your dice: Mrs Name your dice: Steve First game: Mr : 6 Mrs : 2 Steve : 2 Reroll Steve : 3 Mrs : 3 Reroll Steve : 6 Mrs : 1 Reroll Mr : 6 Steve : 4 Second game: Mr : 1 Steve : 3 Steve wins!
8	User runs the program <Run the program a couple of times so that you get **every player to win** at least once.>	Dice rolling game First round … Player1: 6 Player2: 6 Player3: 5 Player 2 is out because of red dice. Second round … Player1: 3 Player3: 6 The winner is: Player3 <Player1, Player2, Player3>	
9	User runs the program <Write test case depending on your implementation.>	Here is the data that you provided : Manufacturer: <Apple> Model number: <iPhone 7> Retail price: <500.0>	Enter the manufactor: Apple Enter the model number: iPhone 7 Change price to: 500 Here is the data that you provided: Manufacturer: Apple Model number: iPhone 7 Retail price: 500

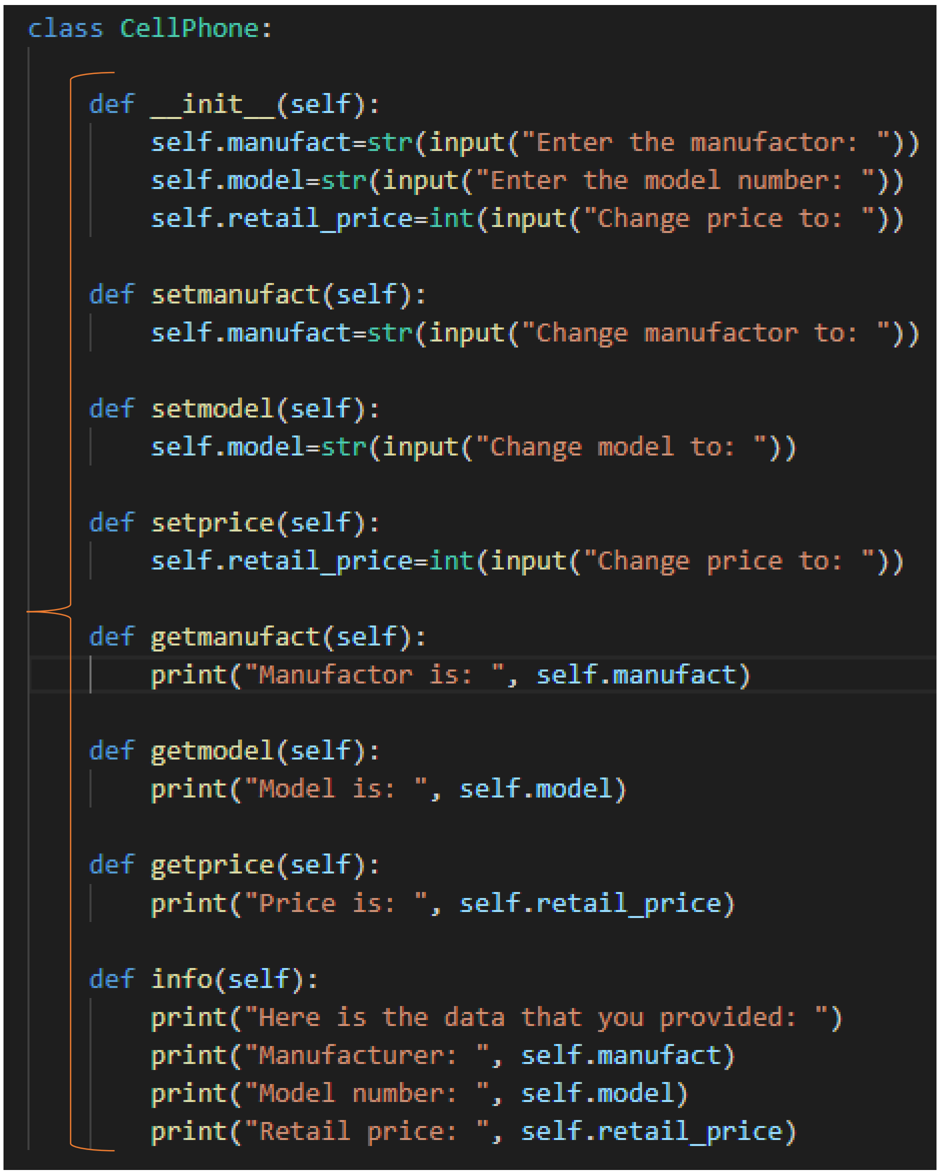
Task 4:

It does not change since the attribute is private and can not be changed. Outside of the object.

Task 10:

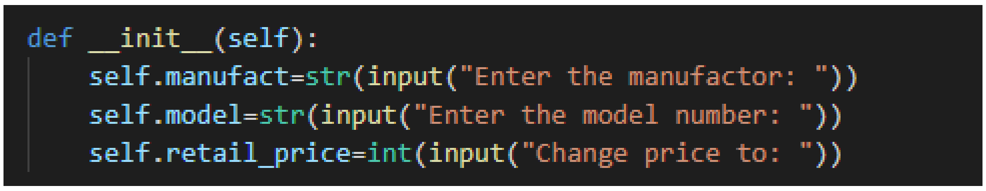
a) 

```
phone=CellPhone()
```



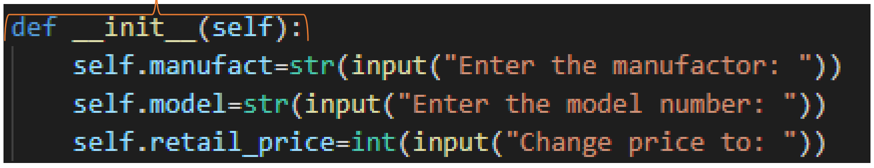
```python
class CellPhone:

    def __init__(self):
        self.manufact=str(input("Enter the manufactor: "))
        self.model=str(input("Enter the model number: "))
        self.retail_price=int(input("Change price to: "))

    def setmanufact(self):
        self.manufact=str(input("Change manufactor to: "))

    def setmodel(self):
        self.model=str(input("Change model to: "))

    def setprice(self):
        self.retail_price=int(input("Change price to: "))

    def getmanufact(self):
        print("Manufactor is: ", self.manufact)

    def getmodel(self):
        print("Model is: ", self.model)

    def getprice(self):
        print("Price is: ", self.retail_price)

    def info(self):
        print("Here is the data that you provided: ")
        print("Manufacturer: ", self.manufact)
        print("Model number: ", self.model)
        print("Retail price: ", self.retail_price)
```

b) Hard to line the encapsulation, but it is index after class



```python
def __init__(self):
    self.manufact=str(input("Enter the manufactor: "))
    self.model=str(input("Enter the model number: "))
    self.retail_price=int(input("Change price to: "))
```

c)
d) No hidden attributes?
e) Everything is public method
f) No private methods

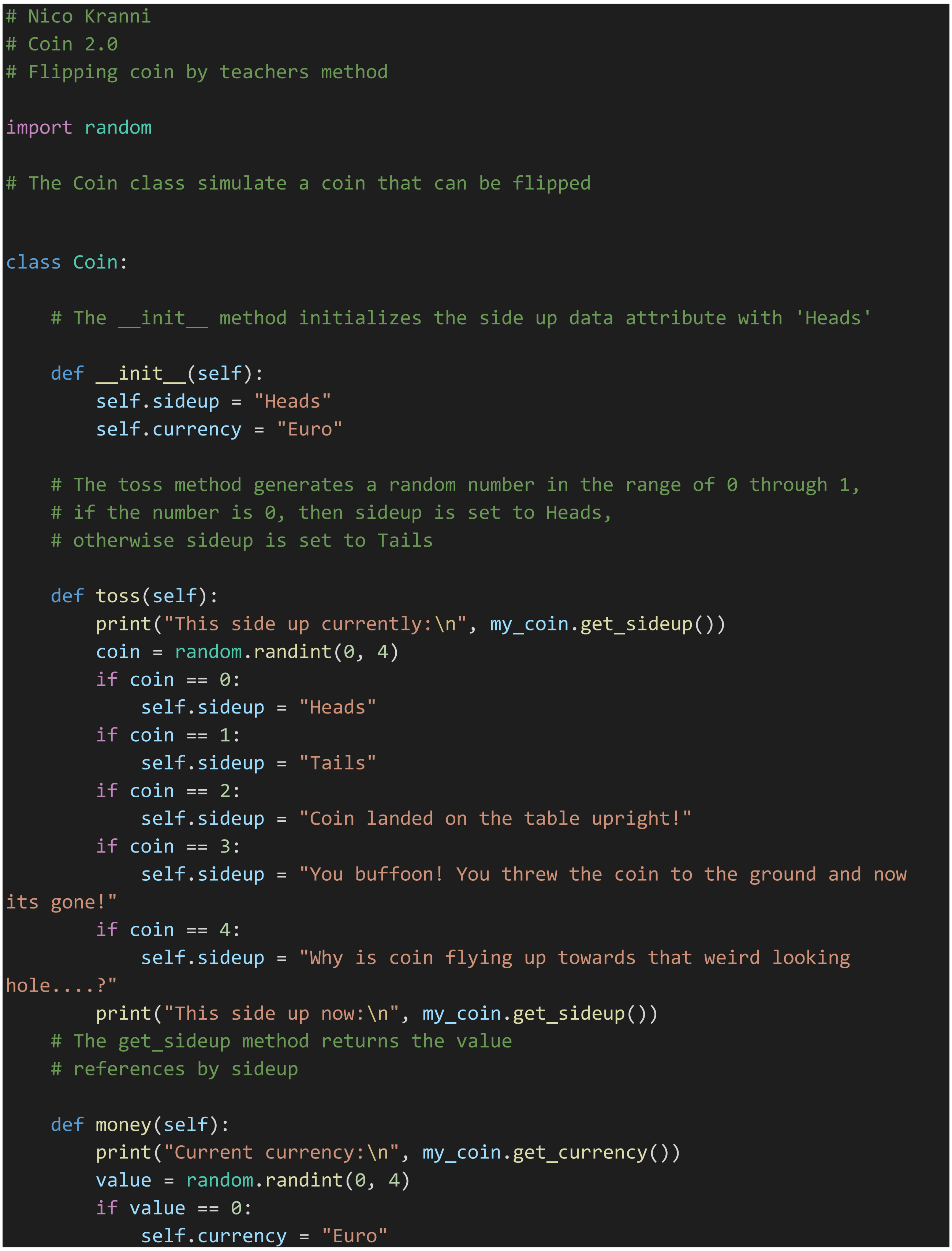


```python
def __init__(self):
    self.manufact=str(input("Enter the manufactor: "))
    self.model=str(input("Enter the model number: "))
    self.retail_price=int(input("Change price to: "))
```

g)

Screen captures of all code here

Coin



```
# Nico Kranni
# Coin 2.0
# Flipping coin by teachers method

import random

# The Coin class simulate a coin that can be flipped


class Coin:

    # The __init__ method initializes the side up data attribute with 'Heads'

    def __init__(self):
        self.sideup = "Heads"
        self.currency = "Euro"

    # The toss method generates a random number in the range of 0 through 1,
    # if the number is 0, then sideup is set to Heads,
    # otherwise sideup is set to Tails

    def toss(self):
        print("This side up currently:\n", my_coin.get_sideup())
        coin = random.randint(0, 4)
        if coin == 0:
            self.sideup = "Heads"
        if coin == 1:
            self.sideup = "Tails"
        if coin == 2:
            self.sideup = "Coin landed on the table upright!"
        if coin == 3:
            self.sideup = "You buffoon! You threw the coin to the ground and now
its gone!"
        if coin == 4:
            self.sideup = "Why is coin flying up towards that weird looking
hole....?"
        print("This side up now:\n", my_coin.get_sideup())
    # The get_sideup method returns the value
    # references by sideup

    def money(self):
        print("Current currency:\n", my_coin.get_currency())
        value = random.randint(0, 4)
        if value == 0:
            self.currency = "Euro"
```

```python
        if value == 1:
            self.currency = "Pound"
        if value == 2:
            self.currency = "Dollar"
        if value == 3:
            self.currency = "Ruble"
        if value == 4:
            self.currency = "Yen"
        print("New currency:\n", my_coin.get_currency())
    # The get_sideup method returns the value
    # references by sideup

    def get_sideup(self):
        return self.sideup

    def get_currency(self):
        return self.currency
# The main function.


'''def main():

    # Create an object from the Coin class.

    my_coin = Coin()

    # Display the side of the coin
    print("This side up:\n", my_coin.get_sideup())
    print("This currency:\n", my_coin.get_currency())
    # Toss the coin
    my_coin.toss()
    my_coin.money()

    # Display the side of the coin
    print("This side up:\n", my_coin.get_sideup())
    print("This currency:\n", my_coin.get_currency())
'''



my_coin=Coin()

my_coin.toss()
my_coin.sideup="Test"
my_coin.toss()
```

Dice

```
# Nico Kranni
# Dice
# Rolling dice the game, with properties.


import random
import inspect
class Dice:

    # The __init__ method initializes the side up data attribute with 'Heads'

    def __init__(self):
        self.sideup = random.randint(1, 6) # First it gives random value for the
dice
        #self.colour=str(input("Give your dice a colour: ")) # You can set color
for the dice.
        self.name=str(input("Name your dice: ")) #You can name the dice when
created
        #self.material=str(input("What material is your dice made of: "))


    def rematerial(self): #To change material of dice
        self.material=str(input("What material is your new material: "))


    def roll(self): #This rolls the dice and saves it to .sideup
        print("You roll the dice...")
        dice = random.randint(1, 6)
        self.sideup=dice
        print("This side up currently:\n", self.sideup)
    def get_sideup(self): #Gets current side that is up
        print("This side up currently:", self.sideup)




    def changeColour(self): #If you want to change colour
        self.colour=str(input("Change colour to: "))
        print ("New colour is:", self.colour)
    def getColour(self): #To check what colour you had
        print("My colour is:", self.colour)





    def changeName(self): #Change your dices name
```

```python
        self.name=str(input("Give dice a new name: "))
        print("New name is: ", self.name)
    def getName(self): #Forgot what your dice name was?
        print("My name is:", self.name)

    def rollTheDices(self):
        first_dice.roll()
        second_dice.roll()
        print("Sum of the dices is: ", first_dice.sideup+second_dice.sideup)

    def theGame(self):
        print("First game:")

        first_dice.sideup=random.randint(1,6)
        print(first_dice.name, ":", first_dice.sideup)

        second_dice.sideup=random.randint(1,6)
        print(second_dice.name, ":", second_dice.sideup)

        third_dice.sideup=random.randint(1,6)
        print(third_dice.name, ":", third_dice.sideup)

        while first_dice.sideup == second_dice.sideup or
first_dice.sideup==third_dice.sideup or second_dice.sideup==third_dice.sideup:
            print("Reroll")
            if first_dice.sideup == second_dice.sideup:

                first_dice.sideup= random.randint(1, 6)
                print(first_dice.name,":", first_dice.sideup)

                second_dice.sideup= random.randint(1, 6)
                print(second_dice.name,":", second_dice.sideup)

            if first_dice.sideup==third_dice.sideup:

                first_dice.sideup= random.randint(1, 6)
                print(first_dice.name,":", first_dice.sideup)

                third_dice.sideup= random.randint(1, 6)
                print(third_dice.name,":", third_dice.sideup)

            if second_dice.sideup==third_dice.sideup:

                third_dice.sideup= random.randint(1, 6)
                print(third_dice.name,":", third_dice.sideup)

                second_dice.sideup= random.randint(1, 6)
                print(second_dice.name,":", second_dice.sideup)
        if first_dice.sideup < second_dice.sideup and first_dice.sideup <
third_dice.sideup:

            print("Second game:")

            second_dice.sideup=random.randint(1,6)
            print(second_dice.name, ":", second_dice.sideup)

            third_dice.sideup=random.randint(1,6)
            print(third_dice.name, ":", third_dice.sideup)


            while second_dice.sideup==third_dice.sideup:
                print("Reroll")
                if second_dice.sideup==third_dice.sideup:

                    third_dice.sideup= random.randint(1, 6)
                    print(third_dice.name,":", third_dice.sideup)

                    second_dice.sideup= random.randint(1, 6)
                    print(second_dice.name,":", second_dice.sideup)


            if second_dice.sideup < third_dice.sideup:
                print (third_dice.name, "wins!")
            else:
                print (second_dice.name, "wins!")

        if second_dice.sideup < first_dice.sideup and second_dice.sideup <
third_dice.sideup:

            print("Second game:")

            first_dice.sideup=random.randint(1,6)
            print(first_dice.name, ":", first_dice.sideup)

            third_dice.sideup=random.randint(1,6)
            print(third_dice.name, ":", third_dice.sideup)

            while first_dice.sideup==third_dice.sideup:
                print("Reroll")
                if first_dice.sideup==third_dice.sideup:

                    first_dice.sideup= random.randint(1, 6)
                    print(first_dice.name,":", first_dice.sideup)

                    third_dice.sideup= random.randint(1, 6)
                    print(third_dice.name,":", third_dice.sideup)
            if first_dice.sideup < third_dice.sideup:
                print (third_dice.name, "wins!")
            else:
                print (first_dice.name, "wins!")


        if third_dice.sideup < first_dice.sideup and third_dice.sideup < 
second_dice.sideup:

            print("Second game:")

            first_dice.sideup=random.randint(1,6)
            print(first_dice.name, ":", first_dice.sideup)

            second_dice.sideup=random.randint(1,6)
            print(second_dice.name, ":", second_dice.sideup)


            while first_dice.sideup == second_dice.sideup:
                print("Reroll")

                if first_dice.sideup == second_dice.sideup:

                    first_dice.sideup= random.randint(1, 6)
                    print(first_dice.name,":", first_dice.sideup)

                    second_dice.sideup= random.randint(1, 6)
                    print(second_dice.name,":", second_dice.sideup)

            if first_dice.sideup < second_dice.sideup:
                print (second_dice.name, "wins!")
            else:
                print (first_dice.name, "wins!",)



    # The get_sideup method returns the value
    # references by sideup


first_dice=Dice()
second_dice=Dice()
third_dice=Dice()
first_dice.theGame()
```

Phone

```
class CellPhone:

    def __init__(self):
        self.manufact=str(input("Enter the manufactor: "))
        self.model=str(input("Enter the model number: "))
        self.retail_price=int(input("Change price to: "))

    def setmanufact(self):
        self.manufact=str(input("Change manufactor to: "))

    def setmodel(self):
        self.model=str(input("Change model to: "))

    def setprice(self):
        self.retail_price=int(input("Change price to: "))

    def getmanufact(self):
        print("Manufactor is: ", self.manufact)

    def getmodel(self):
        print("Model is: ", self.model)

    def getprice(self):
        print("Price is: ", self.retail_price)

    def info(self):
        print("Here is the data that you provided: ")
        print("Manufacturer: ", self.manufact)
        print("Model number: ", self.model)
        print("Retail price: ", self.retail_price)

phone=CellPhone()
phone.info()
```