IQRA NATIONAL UNIVERSITY

Department of Computer Science

MODREN PROGRAMMING LANGUAGE

Name: Ali Muqadas

ID: 13131

Submitted to: Sir Fahim Ullah

Assignment: FINAL ASSIGNMENT

JQ1: Restaurant: Make a class called Restaurant. The __init__ () method for Restaurant should store two attributes: a restaurant_name and a cuisine_type. Make a method called describe_restaurant () that prints these two pieces of information, and a method called open_restaurant () that prints a message indicating that the restaurant is open. Make an instance called restaurant from your class. Print the two attributes individually, and then call both methods.

Answer:

```
class Restaurant():
        def __init__(self,name,cuisine_type):
                self.name = name
                self.cuisine_type = cuisine_type
        def describe_restaurant(self):
                print(self.name ,self.cuisine_type)
        def open_restaurant(self):
                print('the',self.name, 'is opening')

restaurant = Restaurant('zct','food')
restaurant.describe_restaurant()
restaurant.open_restaurant()

#9-3
class User():
        def __init__(self,first_name,last_name,**user_info):
                self.first_name = first_name
                self.last_name = last_name
                self.user_info = user_info
        def greet_user(self):
                print('Hello,',self.first_name.title())
        def describe_user(self):
                print('first_name is ',self.first_name)
                print('last_name is ',self.last_name)
                for k,v in self.user_info.items():
                        print(k,'is',v)

user1 = User('xiaoli','li',age=12,sex='F')
user1.describe_user()
user1.greet_user()
#9-4
class Restaurant():
        def __init__(self,name,cuisine_type):
```

```
                self.name = name
                self.cuisine_type = cuisine_type
                self.number_served = 0
        def describe_restaurant(self):
                print(self.name ,self.cuisine_type,self.number_served)
        def set_number_served(self,served_num):
                self.number_served = served_num
        def increment_number_served(self,inc_served_num):
                self.number_served += inc_served_num

restaurant = Restaurant('zct','food')
restaurant.describe_restaurant()
restaurant.set_number_served(56)
restaurant.describe_restaurant()
restaurant.increment_number_served(12)
restaurant.describe_restaurant()
#9-5
class User():
        def __init__(self,first_name,last_name,**user_info):
                self.first_name = first_name
                self.last_name = last_name
                self.login_attempts = 0
                self.user_info = user_info
        def greet_user(self):
                print('Hello,',self.first_name.title())
        def describe_user(self):
                print('first_name is ',self.first_name)
                print('last_name is ',self.last_name)
                print('login_attempts is ',self.login_attempts)
                for k,v in self.user_info.items():
                        print(k,'is',v)
        def increment_login_attempts(self):
                self.login_attempts+=1
        def     reset_login_attempts(self):
                self.login_attempts = 0

user1 = User('xiaoli','li',age=12,sex='F')
user1.describe_user()
user1.increment_login_attempts()
user1.increment_login_attempts()
```

```
user1.describe_user()
user1.reset_login_attempts()
user1.describe_user()
#9-6
class IceCreamStand(Restaurant):
        def __init__(self,name,cuisine_type,flavors):
                super().__init__(name,cuisine_type)
                self.flavors = flavors
        def show_flavors(self):
                for flavor in  self.flavors:
                        print(flavor)
iceCream=IceCreamStand('aa','bb',['aaa','bbb','ccc'])
iceCream.show_flavors()
class User():
        def __init__(self,first_name,last_name,**user_info):
                self.first_name = first_name
                self.last_name = last_name
                self.user_info = user_info
        def greet_user(self):
                print('Hello,',self.first_name.title())
        def describe_user(self):
                print('first_name is ',self.first_name)
                print('last_name is ',self.last_name)
                for k,v in self.user_info.items():
                        print(k,'is',v)
#9-7
class Admin(User):
        def __init__(self,first_name,last_name):
                super().__init__(first_name,last_name)
                self.privileges = ['can add post','can delete post','can ban
user']

        def show_privileges(self):
                for privilege in self.privileges:
                        print(self.first_name,self.last_name,privilege)

admin = Admin('adminf','adminl')
admin.show_privileges()

#9-8
```

```
class Privileges():
	def __init__(self):
		self.privileges = ['can add post','can delete post','can ban
user']


class Admin(User):
	def __init__(self,first_name,last_name):
		super().__init__(first_name,last_name)
		self.privileges = Privileges()

	def show_privileges(self):
		for privilege in self.privileges.privileges:
			print(self.first_name,self.last_name,privilege)

admin = Admin('adminf','adminl')
admin.show_privileges()
#9-9
class Car():
	def __init__(self, make, model, year):
		self.make = make
		self.model = model
		self.year = year
		self.odometer_reading = 0
	def get_descriptive_name(self):
		long_name = str(self.year) + ' ' + self.make + ' ' + self.model
		return long_name.title()
	def read_odometer(self):
		print("This car has " + str(self.odometer_reading) + " miles
on it.")
	def update_odometer(self, mileage):
		if mileage >= self.odometer_reading:
			self.odometer_reading = mileage
		else:
			print("You can't roll back an odometer!")
	def increment_odometer(self, miles):
		self.odometer_reading += miles
class Battery():
	"""A simple attempt to simulate an electric car battery """
	def __init__(self, battery_size=70):
```

```
                """ Initialize the battery's properties """
                self.battery_size = battery_size
        def describe_battery(self):
                """ print a message describing the battery capacity """
                print("This car has a " + str(self.battery_size) + "-kWh
battery.")
        def get_range(self):
                """ print a message indicating the battery's cruising range """
                if self.battery_size == 70:
                        range = 240
                elif self.battery_size == 85:
                        range = 270
                message = "This car can go approximately " + str(range)
                message += " miles on a full charge."
                print(message)
        def upgrade_battery(self):
                """ print a message describing the battery capacity """
                if self.battery_size!=85:
                        self.battery_size=85
                print("This car upgrade to " + str(self.battery_size) + "-kWh
battery.")
class ElectricCar(Car):
        """The uniqueness of electric cars """
        def __init__(self, make, model, year):
                """ initializes the properties of the parent class, and then
initializes the electric car-specific property """
                super().__init__(make, model, year)
                self.battery = Battery()
my_tesla = ElectricCar('tesla', 'model s', 2016)
print(my_tesla.get_descriptive_name())
my_tesla.battery.upgrade_battery()
my_tesla.battery.get_range()
#9-14
from random import randint


class Die():
        def __init__(self,sides=6):
                self.sides = sides
        def roll_die(self):
```

```
                print(randint(1,self.sides))
die_six = Die(6)
for i in range(10):
        die_six.roll_die()
die_ten = Die(10)
for i in range(10):
        die_ten.roll_die()
die_20 = Die(20)
for i in range(10):
        die_20.roll_die()
```

Q2:

Album: Write a function called make_album () that builds a dictionary describing a music album. The function should take in an artist name and an album title, and it should return a dictionary containing these two pieces of information. Use the function to make three dictionaries representing different albums. Print each return value to show that the dictionaries are storing the album information correctly.

Add an optional parameter to make_album () that allows you to store the number of tracks on an album. If the calling line includes a value for the number of tracks, add that value to the album's dictionary. Make at least one new function call that includes the number of tracks on an album.

Answer:

```
 def make_album(artist, title, tracks=0):
"""Build a dictionary containing information about an album."""
album_dict = {
    'artist': artist.title(),
    'title': title.title(),
    }
if tracks:
    album_dict['tracks'] = tracks
return album_dict

# Prepare the prompts.
title_prompt = "\nWhat album are you thinking of? "
artist_prompt = "Who's the artist? "

# Let the user know how to quit.
print("Enter 'quit' at any time to stop.")
```

```
while True:
    title = input(title_prompt)
    if title == 'quit':
        break

    artist = input(artist_prompt)
    if artist == 'quit':
        break

    album = make_album(artist, title)
    print(album)

print("\nThanks for responding!")
```

Q3:

a) Rental Car: Write a program that asks the user what kind of rental car they would like. Print a message about that car, such as “Let me see if I can find you a Subaru.”

b) Restaurant Seating: Write a program that asks the user how many people are in their dinner group. If the answer is more than eight, print a message saying they’ll have to wait for a table. Otherwise, report that their table is ready.

c) Multiples of Ten: Ask the user for a number, and then report whether the number is a multiple of 10 or not.

d) Multiples of Ten: Ask the user for a number, and then report whether the number is a multiple of 20 or not.

e) Multiples of Ten: Ask the user for a number, and then report whether the number is a multiple of 30 or not.

Multiples of Ten: Ask the user for a number, and then report whether the number is a multiple of 140 or not.

Answer:

A
```
car = input("What kind of car would you like? ")
print("Let me see if I can find you a " + car.title() + ".")
```

B
```
party_size = input("How many people are in your dinner party tonight? ")
party_size = int(party_size)
```

```
if party_size > 8:
    print("I'm sorry, you'll have to wait for a table.")
else:
    print("Your table is ready.")
C number = input("Give me a number, please: ")
number = int(number)

if number % 10 == 0:
    print(str(number) + " is a multiple of 10.")
D number = input("Give me a number, please: ")
number = int(number)

if number % 20 == 0:
    print(str(number) + " is a multiple of 20.")

else:
    print(str(number) + " is not a multiple of 20.")

E  number = input("Give me a number, please: ")
number = int(number)

if number % 30 == 0:
    print(str(number) + " is a multiple of 30.")

F number = input("Give me a number, please: ")
number = int(number)

if number % 140 == 0:
    print(str(number) + " is a multiple of 140.")
```

Q4: Pizza Toppings: Write a loop that prompts the user to enter a series of pizza toppings until they enter a 'quit' value. As they enter each topping, print a message saying you'll add that topping to their pizza. Also add each topping to a list. After all the toppings are added

Answer:

```
prompt = "\nWhat topping would you like on your pizza?"
prompt += "\nEnter 'quit' when you are finished: "
```

```
while True:
    topping = input(prompt)
    if topping != 'quit':
        print("  I'll add " + topping + " to your pizza.")
    else:
        break
```

Q5: Rivers: Make a dictionary containing Ten major rivers and the country each river runs through. One key-value pair might be 'river 1': 'country 1'. Use separate loops for each of the following

- **Use a loop to print a sentence about each river, such as The River 1 runs through Country 1. Name of the river and country must be titled when printing the sentence.**
- **Use a loop to print the name of each river included in the dictionary.**
- **Use a loop to print the name of each country included in the dictionary**

Answer:

```
rivers = {
    'nile': 'egypt',
    'volga':'europe',
    'mississippi': 'united states',
    'zambezi':'africa',
    'fraser': 'canada',
    'nile':'east africa',
    'kuskokwim': 'alaska',
    'amazon':'peru',
    'yangtze': 'china',
    'danube':'germany',
    }

for river, country in rivers.items():
    print(f"The {river.title()} flows through {country.title()}.")

print("\nThe following rivers are included in this data set:")
for river in rivers.keys():
    print(f"- {river.title()}")
```

```
print("\nThe following countries are included in this data set:")
for country in rivers.values():
    print(f"- {country.title()}")
```