Weekly Report – 3rd week

Name :- Vaibhav Bhardwaj
Class :- MCA 3rd Semester
Roll No. :- 2222888

Day 10:-

- Queue

```
class Patient:

    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender
        self.next = None
        self.previous = None

    def show(self):
        # print("Name:", self.name, "Age:", self.age, "Gender:", self.gender)
        print("Name:{}, Age:{}, Gender:{}".format(self.name, self.age, self.gender))


class PatientQueue:

    def __init__(self):
        self.head = None
        self.tail = None
        self.size = 0

    def enqueue(self, patient):

        self.size += 1

        if self.head is None:
            self.head = patient
            self.tail = patient
        else:
            self.tail.next = patient
            patient.previous = self.tail

            # Add a New Patient in the end of Queue
            self.tail = patient
```

```
def dequeue(self):
    if self.size != 0:
        self.size -= 1
        self.head = self.head.next
    else:
        print("Queue is Empty. Cannot DeQueue")

def iterate(self):

    if self.size > 0:
        temp = self.head
        while True:
            temp.show()
            temp = temp.next

            if temp is None:
                break
    else:
        print("Queue is Empty. Cannot Iterate")
```

Day 11:-

- Stack

```
class ScreenInterface:

    def __init__(self, title=""):
        self.title = title
        self.next = None
        self.previous = None

    def show(self):
        print(">> {}".format(self.title))


class Stack:

    def __init__(self):
        self.head = None
        self.tail = None
        self.size = 0

    def push(self, interface):

        self.size += 1

        if self.head is None:
            self.head = interface
            self.tail = interface
        else:
            self.tail.next = interface
            interface.previous = self.tail

            # Any newly added interface will be tail :)
            self.tail = interface

    def pop(self):
        if self.size != 0:
            self.size -= 1
            self.tail = self.tail.previous
```

```
        else:
            print("STACK is EMPTY :)")

    def iterate(self):
        if self.size != 0:
            temp = self.tail
            while True:
                temp.show()
                temp = temp.previous

                if temp is None:
                    break
```

Day 12:-

- Multi-Value Containers
  - List

```
numbers = list(range(10, 101, 10))
print("1. numbers:", numbers)
print("Type of numbers:", type(numbers))

numbers.append(30)
numbers.append(77)
numbers.append(30)
numbers.append(95)

print("2. numbers:", numbers)
print("Sum:", sum(numbers))
print("Min:", min(numbers))
print("Max:", max(numbers))
print("Length:", len(numbers))

reverse_numbers = list(reversed(numbers))
print("reverse_numbers:", reverse_numbers)

print("Index of 50 is:", numbers.index(50))
print("Count of 30 in numbers is:", numbers.count(30))

data = [70, 30, 50, 90, 20]
print("data is:", data)
data.sort()
print("data in sorted arrangement is:", data)
```

- Set

```
names = ["john", "anna", "sia", "angel", "kim"]
names.sort()
print("names:", names)
print("Min:", min(names))
print("Min:", max(names))

names.remove("sia")
data.remove(30)

# del names[2]

print(names)
print(data)

data = [10, 20, 30, 40, 50]
# data.pop()
# data.pop(0)
# data.clear()

data.insert(2, 77)
# data.insert(len(data), 99)
data.insert(-1, 99)

print(data)
```

- Dictionary

```
john_followers = {"fionna", "sia", "jack", "joe", "george"}
jake_followers = {"anna", "jack", "leo", "joe", "harry", "mike"}
fionna_followers = {"sia", "joe"}

print("john_followers:", john_followers)
print("jake_followers:", jake_followers)
print("fionna_followers:", fionna_followers)

followers = john_followers.intersection(jake_followers)
followers = john_followers.intersection(jake_followers).intersection(fionna_followers)
print("followers:", followers)

print("issubset:", fionna_followers.issubset(john_followers))
print("issuperset:", john_followers.issuperset(fionna_followers))

A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

C = A - B
print("C is:", C)

D = A & B
print("D is:", D)

E = A ^ B
print("E is:", E)

F = A | B
print("F is:", F)

# Explore what is symmetric_difference
G = A.symmetric_difference(B)
print("G is:", G)
```

Day 13:-

- Strings in Python

```
cafe_name = 'John\'s Cafe'
print("cafe_name is:", cafe_name, type(cafe_name), hex(id(cafe_name)))

print("Length:", len(cafe_name))
print("Min:", min(cafe_name))
print("Max:", max(cafe_name))

# Multi Line String
johns_cafe_name = """John's Cafe Delight
- An Authentic Italian Restaurant"""

print(johns_cafe_name)

# RAW String
quote = r'Be \nExceptional'
print(quote)
```

```
names = "John, Jennie, Jim, Jack, Joe"
# Indexing, Neg Indexing, Slicing, Multiplicity, Concatenation, Membership Testing
print(names[1])
print(names[-1])
print(names[1:5])

new_names = names * 2
print(new_names)

print(names, id(names))
names = names + ", Kia"
print(names, id(names))

print("Kia" in names)
print("George" not in names)
```

Day 14:-

- Virtual Environment
- Database connectivity with MySQL

```
import mysql.connector as db

class Customer:

    def __init__(self):
        self.name = input("Enter Customer Name: ")
        self.phone = input("Enter Customer Phone: ")
        self.email = input("Enter Customer Email: ")


def main():

    customer = Customer()
    print(vars(customer))

    # DataBase Connectivity

    # Step1: Create Connection with Database
    connection = db.connect(user='root',
                            password='',
                            host='127.0.0.1',
                            database='gw2023pds1')

    # Step2: Obtain Cursor to perform SQL operations :)
    cursor = connection.cursor()

    # Step3: Create SQL Statement
    sql = "insert into Customer values" \
          "(null, '{name}', '{phone}', '{email}');".format_map(vars(customer))

    # Step4: Execute SQL Command
    cursor.execute(sql)
    connection.commit()
          print("Customer Inserted...")


     if __name__ == "__main__":
          main()
```