Weekly Report – 2nd week

Name :- Vaibhav Bhardwaj
Class :- MCA 3rd Semester
Roll No. :- 2222888
GitHub:- https://github.com/R34L-D34TH/college-report

Day 5:-

- Loops

```
flag = False

# for idx in range(0,6,1): # idx : 0, 1, 2, 3, 4, 5
for idx in range(6): # idx : 0, 1, 2, 3, 4, 5

    print("Matching", name, "with", employees[idx])

    if name == employees[idx]:
        flag = True
        break

if flag:
    print("Name Found...")
else:
    print("Name Not Found...")
```

- Functions

```
total_bricks = int(input("Enter Number of Bricks: "))
bricks_in_wall = 0

for idx in range(1, total_bricks):
    john_bricks = idx
    bricks_in_wall += john_bricks

    if bricks_in_wall >= total_bricks:
        difference = bricks_in_wall - total_bricks
        print("John Placed the Last Brick:", (john_bricks - difference))
        break

    jack_bricks = idx * 2
    bricks_in_wall += jack_bricks

    if bricks_in_wall >= total_bricks:
        difference = bricks_in_wall - total_bricks
        print("Jack Placed the Last Brick:", (jack_bricks-difference))
        break


print("bricks_in_wall is:", (bricks_in_wall-difference))
```

Day 6:-

- main() in python

```
def main(): # main thread :)
    a = 10
    b = 2*a
    print("b is:", b)

    print("NAME:", __name__)


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

- Functions in Memory

```
def square(num):
    print("[square] num is:", num, id(num))
    num = num * num
    print("[square] num now is:", num, id(num))


def main():
    a = 10
    print("[main] a is:", a, id(a))
    square(a)
    print("[main] a now is:", a, id(a))


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

- Recursion

```python
def get_max(numbers, length):
    if length == 1:
        return numbers[0]
    else:
        result = get_max(numbers, length-1)
        if result > numbers[length-1]:
            return result
        else:
            return numbers[length-1]


def main():
    data = [20, 30, 10]
    max_number = get_max(data, len(data))
    print("MAX NUMBER:", max_number)


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

Day 7:-

- OOPS

```
class Dish:
    # CONSTRUCTOR
    def __init__(self, name="", price=0, ratings=4.0):
        self.name = name
        self.price = price
        self.ratings = ratings

    def show(self):
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        print("NAME:", self.name)
        print("PRICE:", self.price)
        print("RATINGS:", self.ratings)
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")


dish1 = Dish("Noodles", 300, 4.5)
dish2 = Dish("Burger", 100, 4.3)
dish3 = Dish(name="Fries")
```

- Principle of OOPS
    1. Identify Object and what data goes inside object
    2. Write its representation in the code i.e. class
    3. From the class create a real object in memory

Day 8:-

Relationship Mapping

1 to 1

1 User has 1 Address
1 User has 1 Profile
1 Restaurant has 1 Menu

1 to many

1 Menu has many Dishes
1 User has many Addresses
1 Teacher has many students
1 Meeting has many participants
1 YT Channel has many subscribers

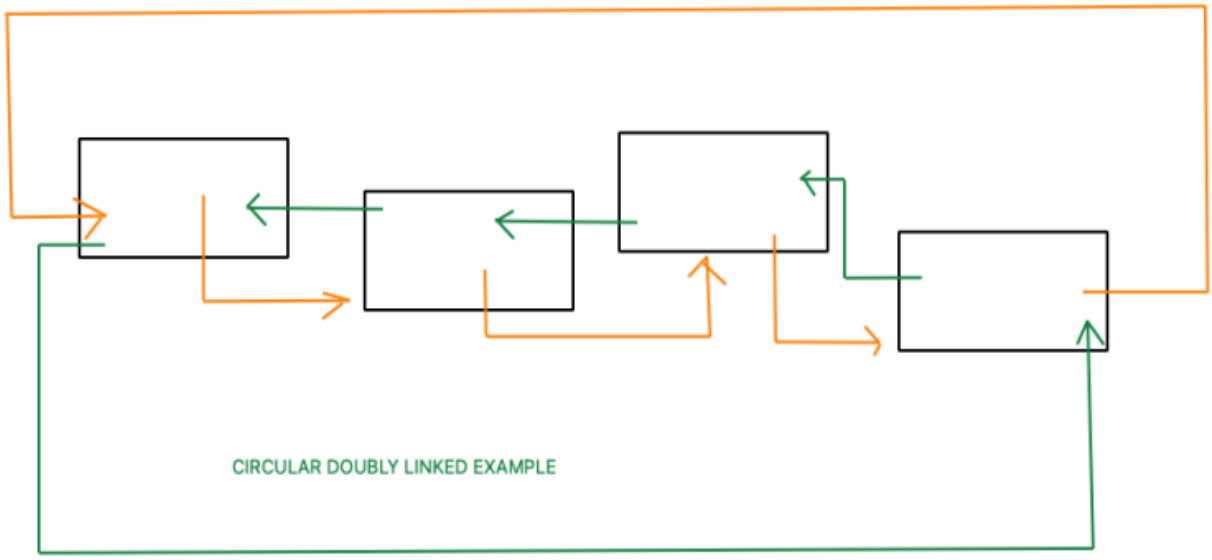
many to many

Many YT Channels has many Sunscribers

```
class Menu:
    def __init__(self, name, categories):
        self.name = name
        self.categories = categories

    def show(self):
        print(self.name, end=" | ")

    def show_categories(self):
        for idx in range(len(self.categories)):
            self.categories[idx].show()



class Product:

    def __init__(self, name, brand, price, rating):
        self.name = name
        self.brand = brand
        self.price = price
        self.rating = rating

    def show(self):
        print("-----------------------")
        print(self.name, " | ", self.brand)
        print(self.price, " | ", self.rating)
        print("-----------------------")
```

Day 9:-

- Data Structures




```
class Song:

    def __init__(self, track, artists, duration):
        self.track = track
        self.artists = artists
        self.duration = duration
        self.next = None
        self.previous = None

    def show(self):
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        print(self.track)
        print(self.artists)
        print(self.duration)
        print("CURRENT:", self, "NEXT:", self.next, "PREVIOUS:", self.previous)
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
```

```
class PlayList:

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

    def append(self, song):

        self.size += 1

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

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

            # CIRCULAR
            self.head.previous = self.tail
            self.tail.next = self.head

       def iterate(self, direction=0):

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

                    if temp == self.head:
                        break
            else:
                temp = self.tail
                while True:
                    temp.show()
                    temp = temp.previous

                    if temp == self.tail:
                        break
```