

**Usman Institute of Technology**

**Department of Computer Science Fall 2022**

Name: Muhammad Waleed

Roll no: 20B-115-SE

Course: Operating Systems (CS312)

Course Instructor: Ma’am Shabina Mushtaq

Date: 14-Jan-2023

# Banker’s Algorithm (With Resource Request method):

class Banker:

    def \_\_init\_\_(self, \_max, allocation, available):

        self.\_max = \_max

        self.allocation = allocation

        self.available = available

        self.need = []

        self.num\_resources = len(\_max[0])

        self.calculate\_need()

    def calculate\_need(self):

        for i in range(len(self.\_max)):

            for j in range(self.num\_resources):

                self.need.append(self.\_max[i][j]-self.allocation[i][j])

        self.need = [self.need[i:i+self.num\_resources] for i in range(0, len(self.need), self.num\_resources)]

    def safe\_state(self):

        work = self.available

        finish = [False for i in range(len(self.need))]

        sequence = []

        while False in finish:

            for i in range(len(self.need)):

                for j in range(len(self.need[i])):

                    if self.need[i][j] <= work[j] and finish[i] == False:

                        work[j] += self.allocation[i][j]

                        finish[i] = True

                        sequence.append(f"p{i+1}")

        return sequence

    def request\_resources(self, process, request):

        if request[0] > self.need[process][0] or request[1] > self.need[process][1] or request[2] > self.need[process][2]:

            print("Error: Request exceeds need")

            return False

        elif request[0] > self.available[0] or request[1] > self.available[1] or request[2] > self.available[2]:

            print("Error: Request exceeds available")

            return False

        else:

            for i in range(self.num\_resources):

                self.available[i] -= request[i]

                self.allocation[process][i] += request[i]

                self.need[process][i] -= request[i]

            if self.safe\_state() == False:

                for i in range(self.num\_resources):

                    self.available[i] += request[i]

                    self.allocation[process][i] -= request[i]

                    self.need[process][i] += request[i]

                print("Error: Request results in unsafe state")

                return False

            else:

                print("Request granted")

                return True

    def table(self):

        print("Process\t\tMax\t\tAllocation\tNeed\t\tAvailable")

        for i in range(len(self.\_max)):

            print(f"P{i+1}\t\t{self.\_max[i]}\t{self.allocation[i]}\t{self.need[i]}\t{self.available}")

if \_\_name\_\_ == "\_\_main\_\_":

    \_max = [[7, 5, 3], [3, 2, 2], [9, 0, 2], [2, 2, 2], [4, 3, 3]]

    allocation = [[0, 1, 0], [2, 0, 0], [3, 0, 2], [2, 1, 1], [0, 0, 2]]

    available = [3, 3, 2]

    banker = Banker(\_max, allocation, available)

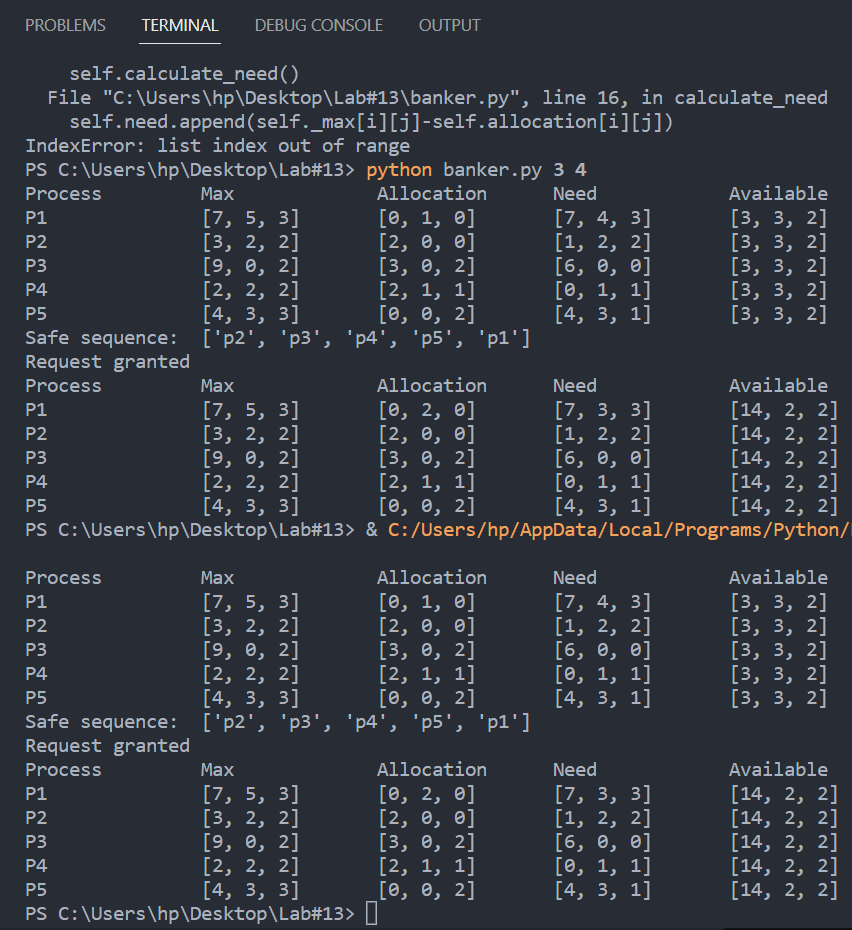
    banker.table()

    print("Safe sequence: ", banker.safe\_state())

    banker.request\_resources(0, [0, 1, 0])

    banker.table()

Output:



# Bash Script:

echo "Enter number of processes:"

read num\_processes

echo "Enter number of resources:"

read num\_resources

python3 modified\_banker.py $num\_processes $num\_resources

Changes in banker.py

import sys

class Banker:

    def \_\_init\_\_(self, num\_resources, num\_processes, \_max, allocation, available):

        self.\_max = \_max

        self.allocation = allocation

        self.available = available

        self.need = []

        self.num\_resources = num\_resources

        self.num\_processes = num\_processes

        self.calculate\_need()

... rest of code

if \_\_name\_\_ == "\_\_main\_\_":

    num\_resources = int(sys.argv[1])

    num\_processes = int(sys.argv[2])

    \_max = [[7, 5, 3], [3, 2, 2], [9, 0, 2], [2, 2, 2], [4, 3, 3]]

    allocation = [[0, 1, 0], [2, 0, 0], [3, 0, 2], [2, 1, 1], [0, 0, 2]]

    available = [3, 3, 2]

    banker = Banker(num\_resources, num\_processes,\_max, allocation, available)

    banker.table()

    print("Safe sequence: ", banker.safe\_state())

    print("Safe sequence: ", banker.safe\_state())

    banker.request\_resources(0, [0, 1, 0])

    banker.table()

# Output:

