**NAME** : Tejesh Santosh Yewale

**ROLL NO. :** A-61

**PRACTICAL NO. A1**

**CODE:**

# STEP1: Accept size of bucket

n = int(input("Enter size of bucket:"))

bucket=[]

for i in range(0,n):

bucket.append(-1)

bucket

# STEP2: Design suitable hash function using division method, folding method etc. key=tel mod bucket\_size 12345%10 = 5

def hashDiv(telephone):

key=int(telephone) % n

return(key)

hashDiv(12345678)

# STEP3: Accept elements less then or equal to size of bucket

def acceptKeys():

print(n)

n1=0

while(n1<=0):

n1=int(input("Enter nos of keys:"))

if(n1>n):

n1=0

else:

break

l1=[]

for i in range(0,n1):

t="Enter "+ str(i+1)+" telephone number:"

l1.append(input(t))

return(l1,n1)

acceptKeys()

# Step4: Generate key-index using above hash function and store number in bucket.

bucket.clear()

for i in range(0,n):

bucket.append(-1)

keys,n1=acceptKeys()

print("Bucket Contents:",bucket)

#store the keys in bucket

for i in range(0,n1):

key=hashDiv(keys[i])

bucket[key]=keys[i]

print("Bucket Contents:",bucket)

#store the keys in bucket

bucket.clear()

complexity=[]

for i in range(0,n):

bucket.append(-1)

complexity.append(-1)

coll=[]

#keys,n1=acceptKeys()

for i in range(0,n1):

key=hashDiv(keys[i])

if(bucket[key]==-1):

bucket[key]=keys[i]

complexity[key]='O(1)'

else:

print("Collision at index:",key, "for :",keys[i])

coll.append(keys[i])

print("Bucket Contents:",bucket)

print("insertion Complexity:",complexity)

density=(n-bucket.count(-1))/n\*100

print("Density of the bucket:",density)

print('Collision:',coll)

def isCollision(ind,bucket):

if(bucket[ind]==-1):

return(0)

return(1)

def linear\_probing(coll,bucket,complexity):

offset=1

for keys in coll:

key=hashDiv(keys)

while(isCollision(key,bucket)):

key=hashDiv((int(keys)+offset))

offset=offset+1

bucket[key]=keys

complexity[key]='O('+str(offset)+')'

print('key ',keys,' inserted successfully at index:',key)

offset=1

return

def quadratic\_probing(coll,bucket,complexity):

offset=1

for keys in coll:

key=hashDiv(keys)

while(isCollision(key,bucket)):

key=hashDiv(int(keys)+(offset\*offset))

offset=offset+1

if(offset>n):

break

bucket[key]=keys

complexity[key]='O('+str(offset)+')'

print('key ',keys,' inserted successfully at index:',key)

offset=1

return

print("Collision:",coll)

while(1):

print("1. Linear probing\n2.Quadratic probing")

ch=int(input("Enter your choice:"))

print("bucket:",bucket,"\n",bucket)

if(ch==1):

linear\_probing(coll,bucket,complexity)

print('-'\*10,"Linearing Probing",'-'\*10)

break

if(ch==2):

quadratic\_probing(coll,bucket,complexity)

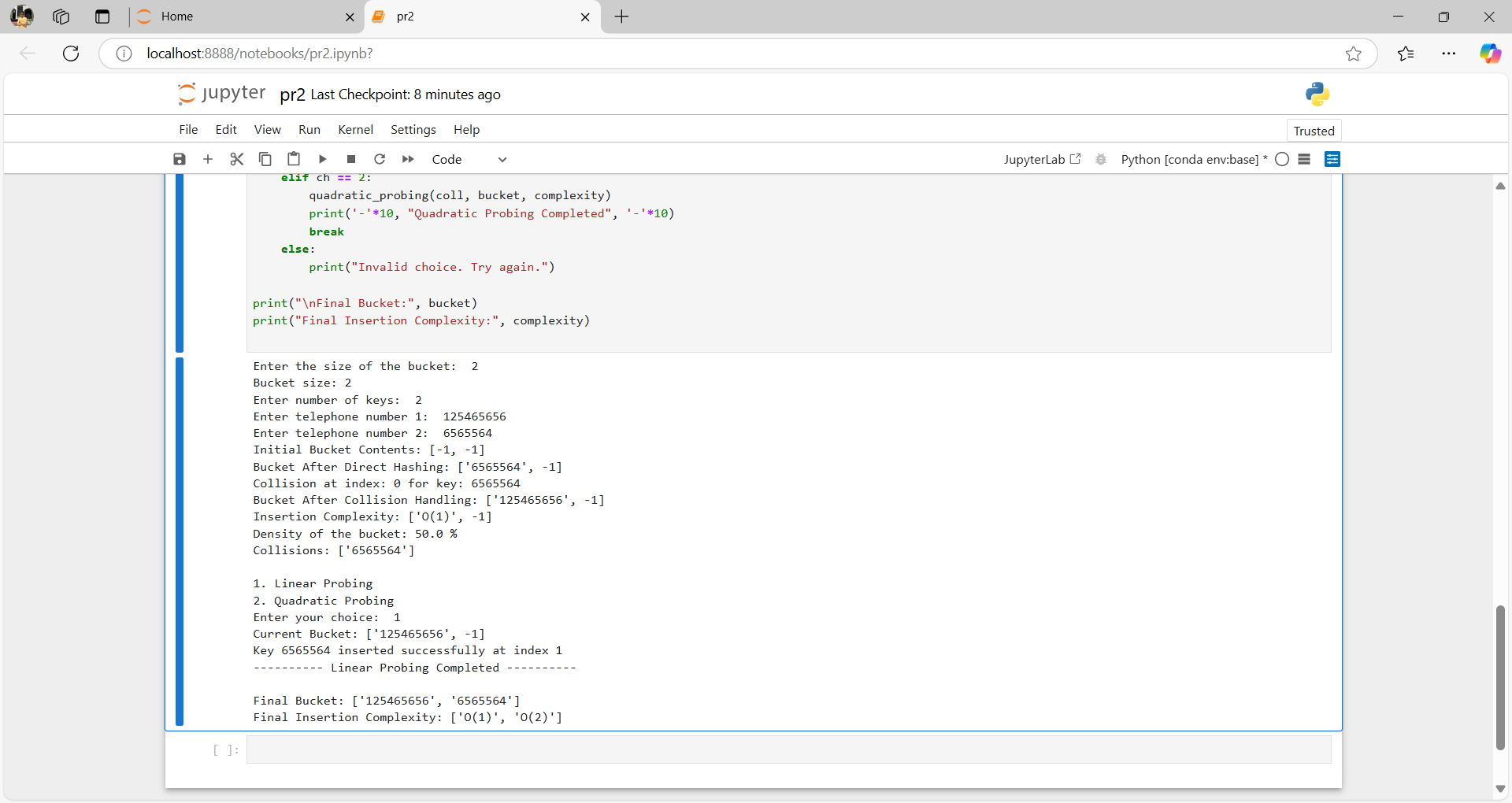
print('-'\*10,"Qaudratic Probing",'-'\*10)

break

#print(bucket,"\n",bucket)

print(bucket,"\n",complex

**OUTPUT:**

****

