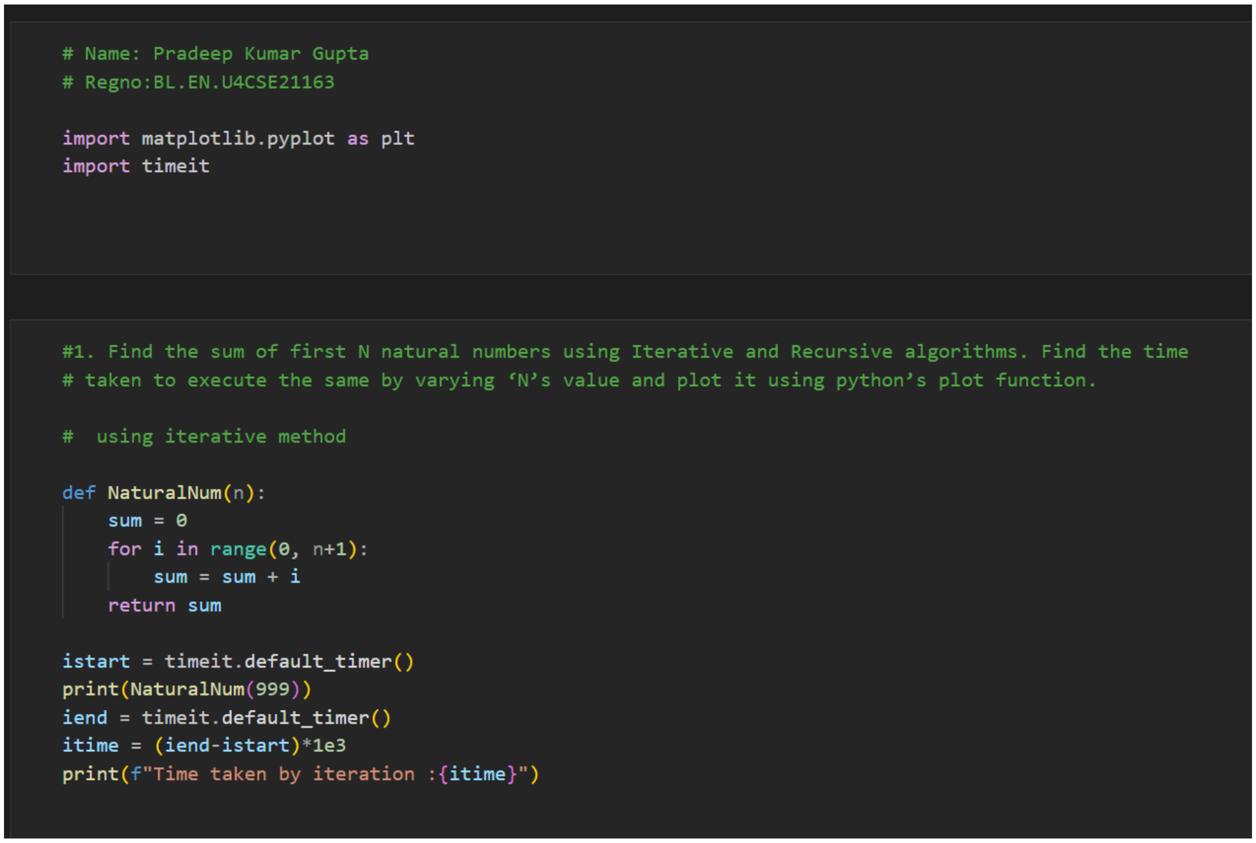
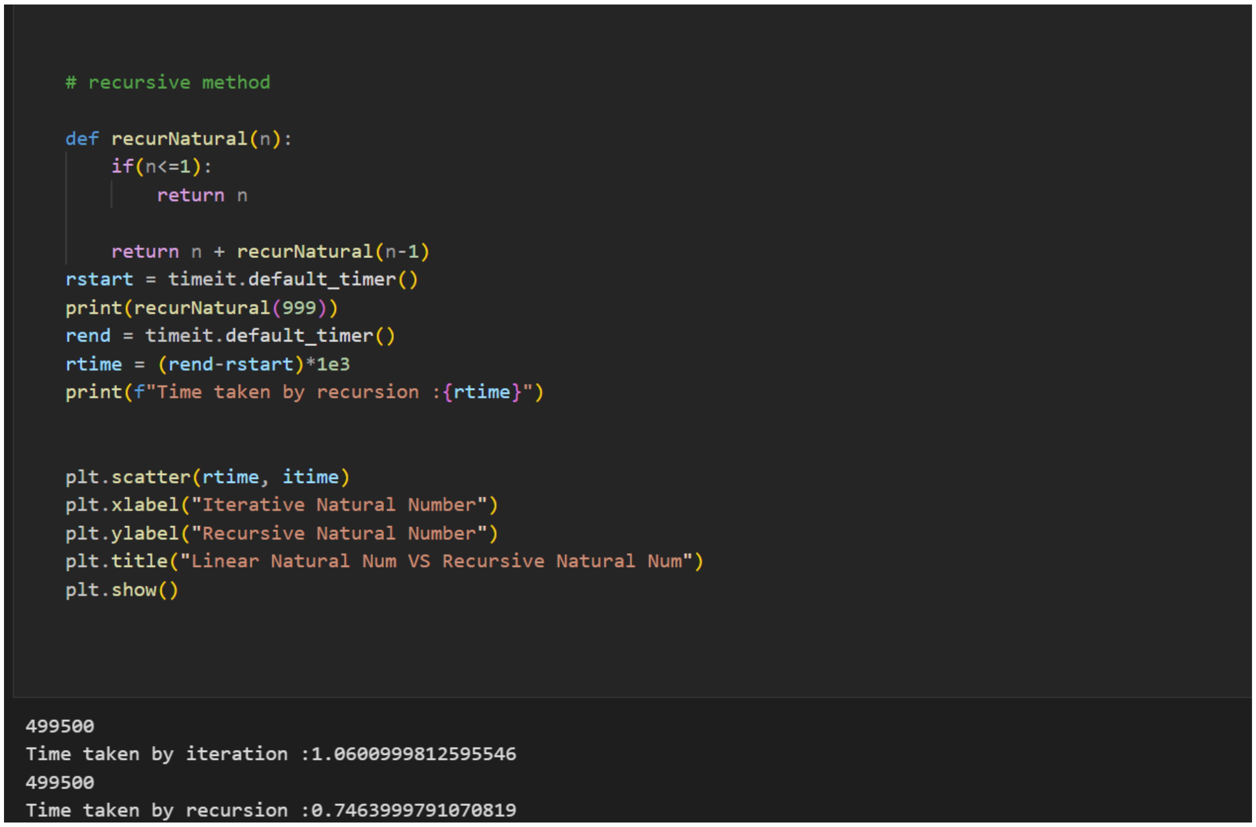
Name: Pradeep Kumar Gupta
Regno: BL.EN. U4CSE21163



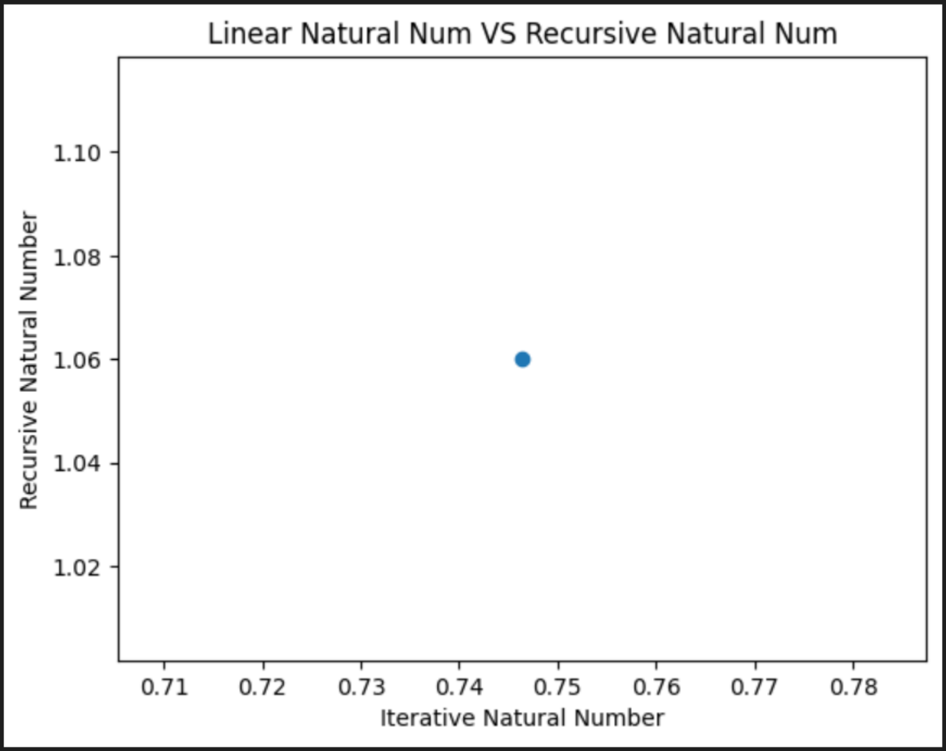
```python
# Name: Pradeep Kumar Gupta
# Regno:BL.EN.U4CSE21163

import matplotlib.pyplot as plt
import timeit
```

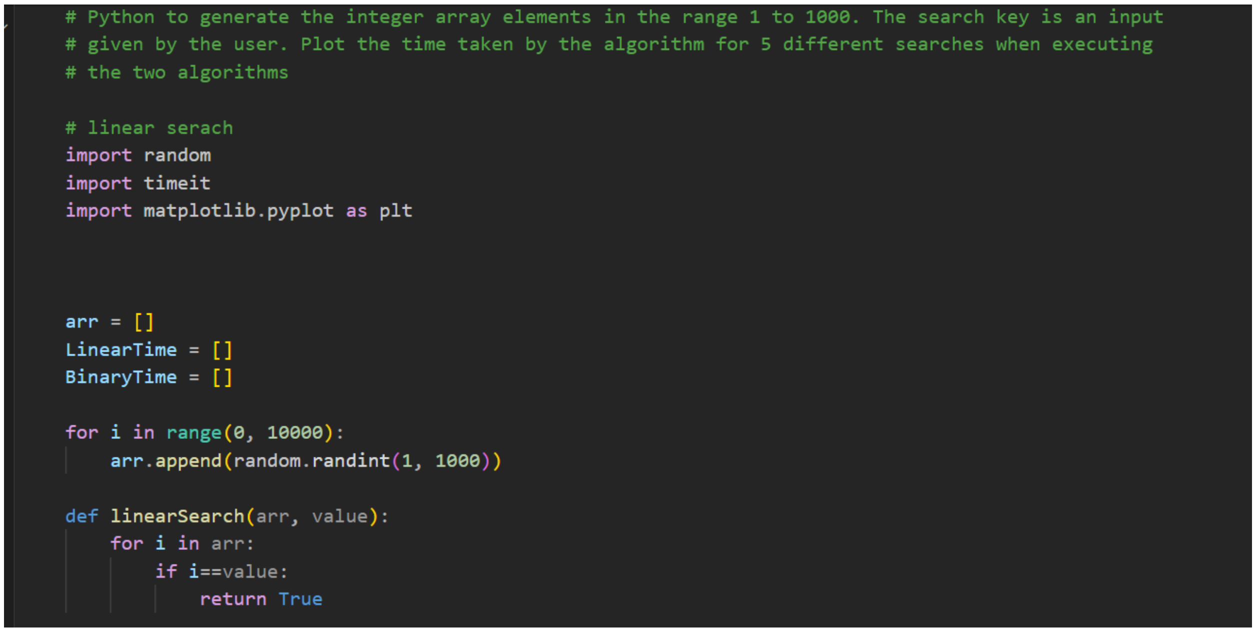
```python
#1. Find the sum of first N natural numbers using Iterative and Recursive algorithms. Find the time
# taken to execute the same by varying 'N's value and plot it using python's plot function.

#  using iterative method

def NaturalNum(n):
    sum = 0
    for i in range(0, n+1):
        sum = sum + i
    return sum

istart = timeit.default_timer()
print(NaturalNum(999))
iend = timeit.default_timer()
itime = (iend-istart)*1e3
print(f"Time taken by iteration :{itime}")
```

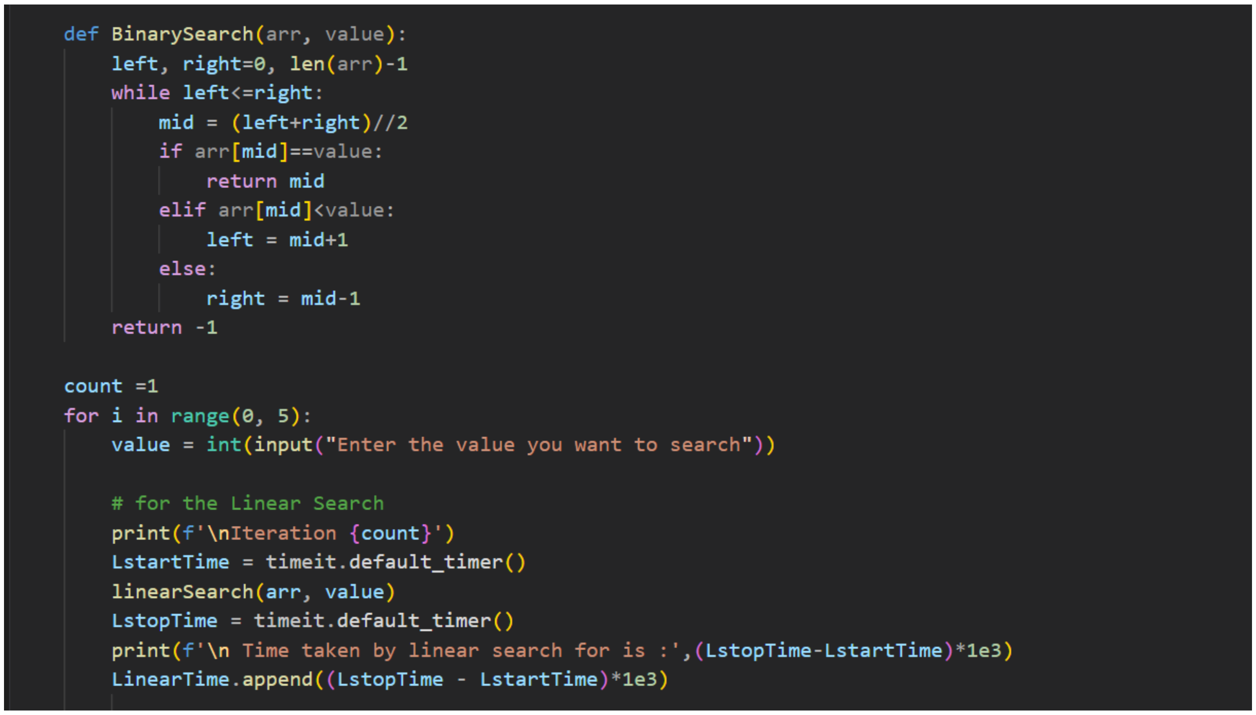


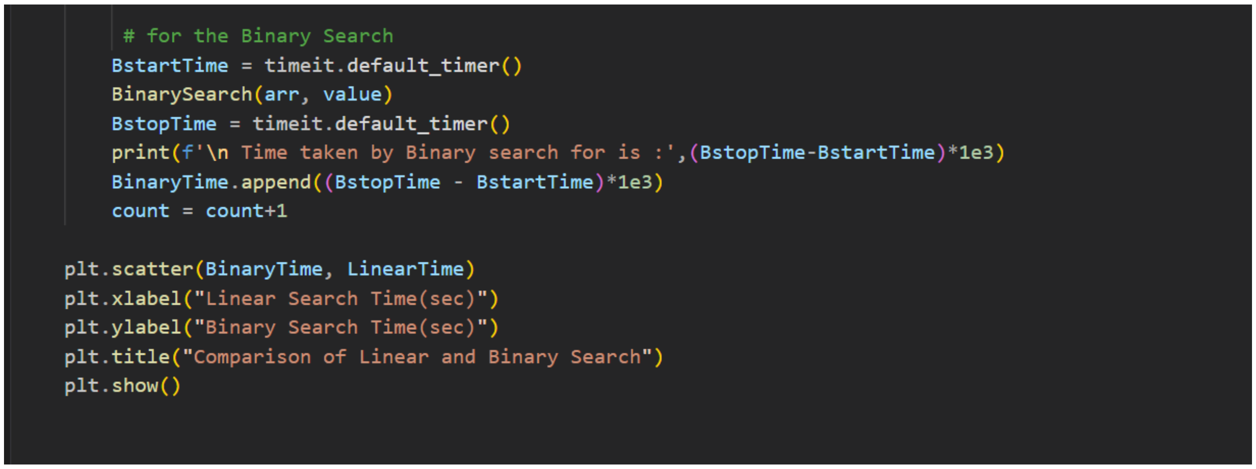
```python
# recursive method

def recurNatural(n):
    if(n<=1):
        return n

    return n + recurNatural(n-1)
rstart = timeit.default_timer()
print(recurNatural(999))
rend = timeit.default_timer()
rtime = (rend-rstart)*1e3
print(f"Time taken by recursion :{rtime}")


plt.scatter(rtime, itime)
plt.xlabel("Iterative Natural Number")
plt.ylabel("Recursive Natural Number")
plt.title("Linear Natural Num VS Recursive Natural Num")
plt.show()
```

```
499500
Time taken by iteration :1.0600999812595546
499500
Time taken by recursion :0.7463999791070819
```

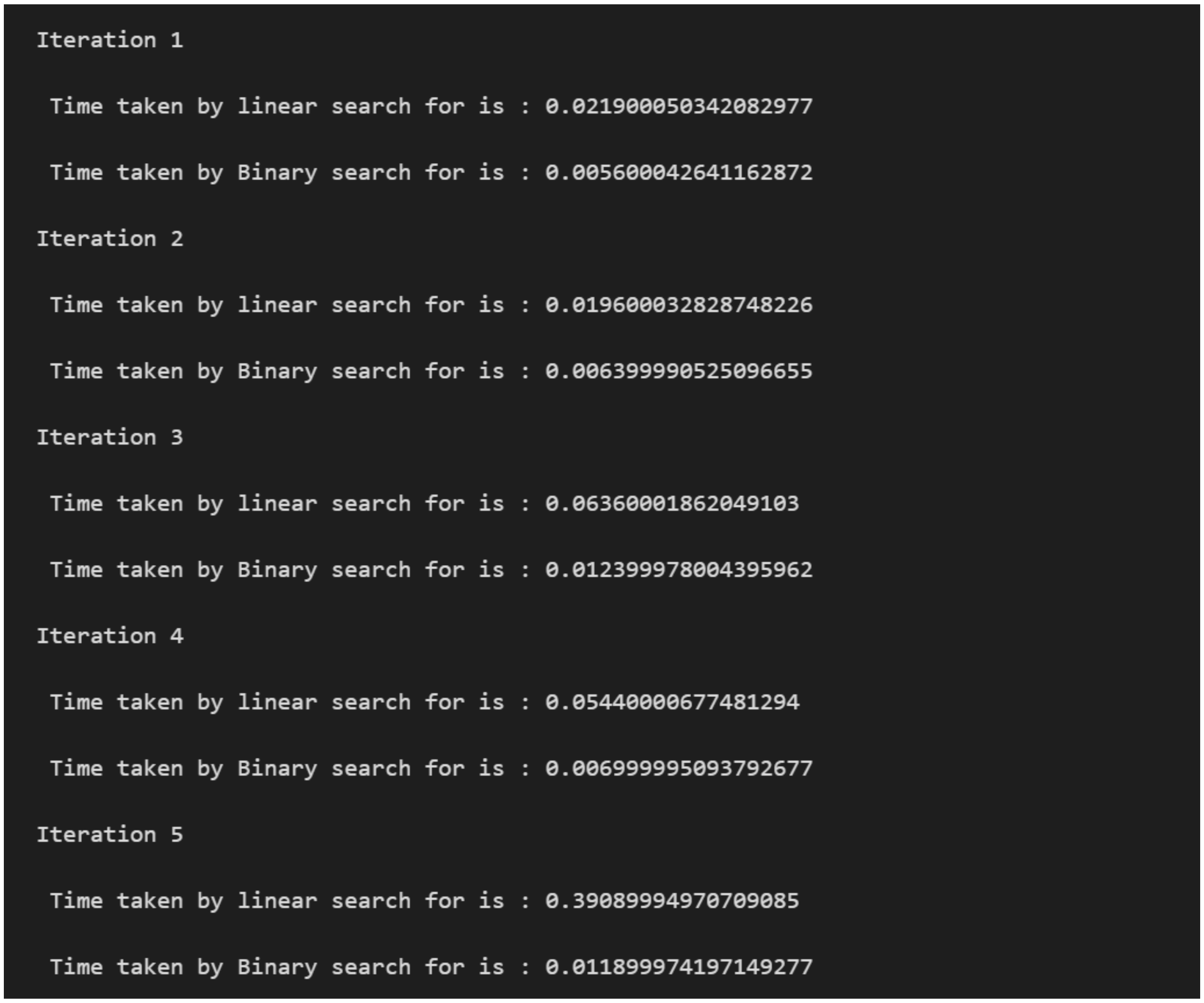



```python
# Python to generate the integer array elements in the range 1 to 1000. The search key is an input
# given by the user. Plot the time taken by the algorithm for 5 different searches when executing
# the two algorithms

# linear serach
import random
import timeit
import matplotlib.pyplot as plt



arr = []
LinearTime = []
BinaryTime = []

for i in range(0, 10000):
    arr.append(random.randint(1, 1000))

def linearSearch(arr, value):
    for i in arr:
        if i==value:
            return True
```

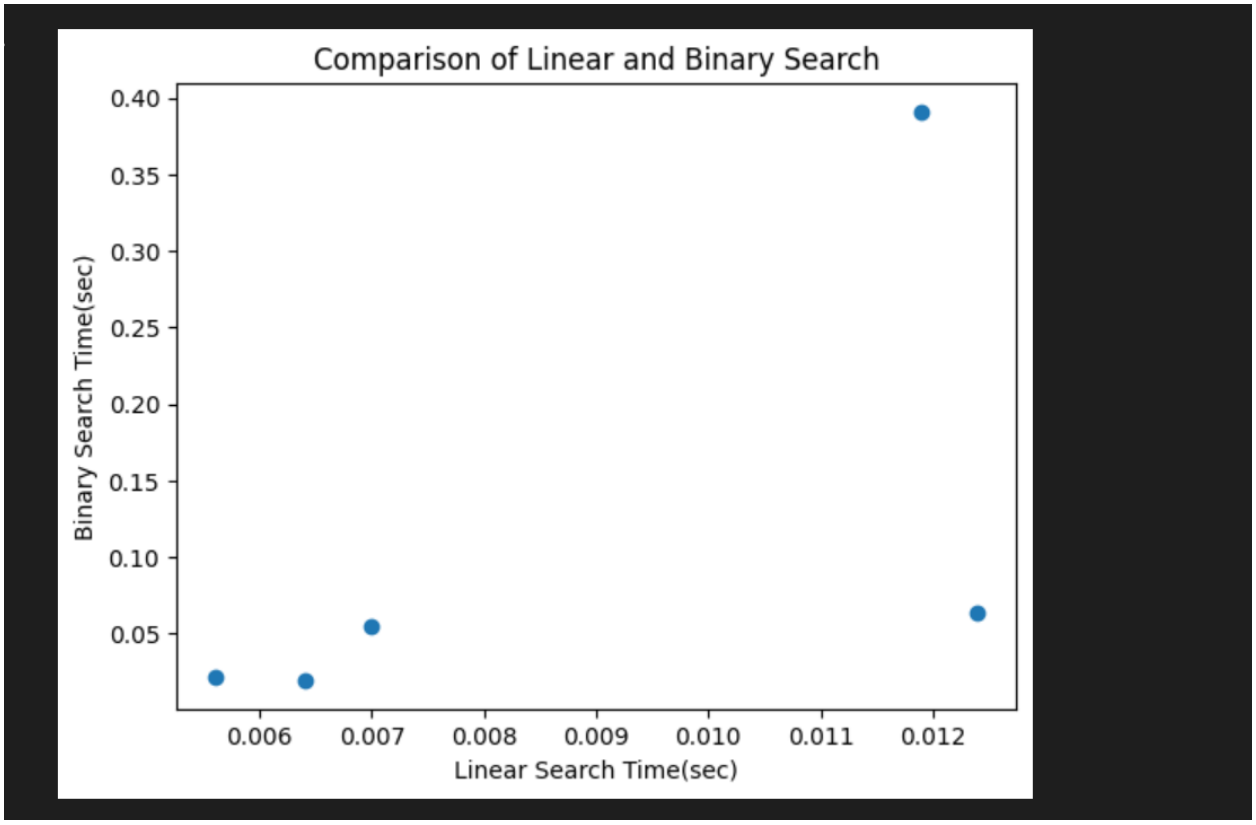

```python
def BinarySearch(arr, value):
    left, right=0, len(arr)-1
    while left<=right:
        mid = (left+right)//2
        if arr[mid]==value:
            return mid
        elif arr[mid]<value:
            left = mid+1
        else:
            right = mid-1
    return -1

count =1
for i in range(0, 5):
    value = int(input("Enter the value you want to search"))

    # for the Linear Search
    print(f'\nIteration {count}')
    LstartTime = timeit.default_timer()
    linearSearch(arr, value)
    LstopTime = timeit.default_timer()
    print(f'\n Time taken by linear search for is :',(LstopTime-LstartTime)*1e3)
    LinearTime.append((LstopTime - LstartTime)*1e3)
```

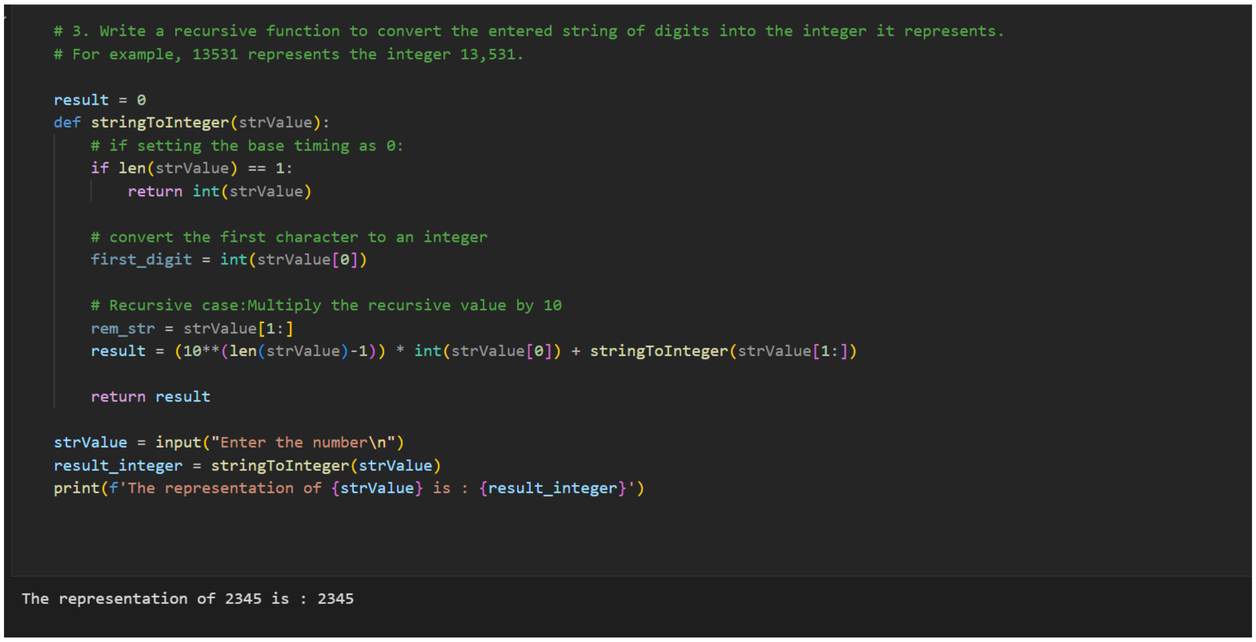


```
         # for the Binary Search
        BstartTime = timeit.default_timer()
        BinarySearch(arr, value)
        BstopTime = timeit.default_timer()
        print(f'\n Time taken by Binary search for is :',(BstopTime-BstartTime)*1e3)
        BinaryTime.append((BstopTime - BstartTime)*1e3)
        count = count+1

    plt.scatter(BinaryTime, LinearTime)
    plt.xlabel("Linear Search Time(sec)")
    plt.ylabel("Binary Search Time(sec)")
    plt.title("Comparison of Linear and Binary Search")
    plt.show()
```



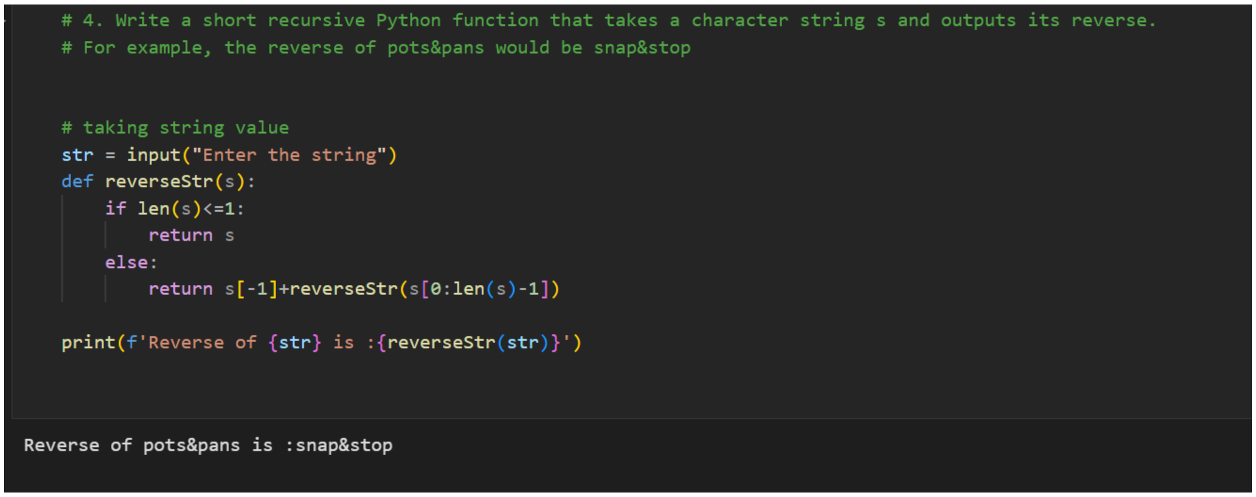
```
Iteration 1

 Time taken by linear search for is : 0.021900050342082977

 Time taken by Binary search for is : 0.005600042641162872

Iteration 2

 Time taken by linear search for is : 0.019600032828748226

 Time taken by Binary search for is : 0.006399990525096655

Iteration 3

 Time taken by linear search for is : 0.06360001862049103

 Time taken by Binary search for is : 0.012399978004395962

Iteration 4

 Time taken by linear search for is : 0.05440000677481294

 Time taken by Binary search for is : 0.006999995093792677

Iteration 5

 Time taken by linear search for is : 0.39089994970709085

 Time taken by Binary search for is : 0.011899974197149277
```





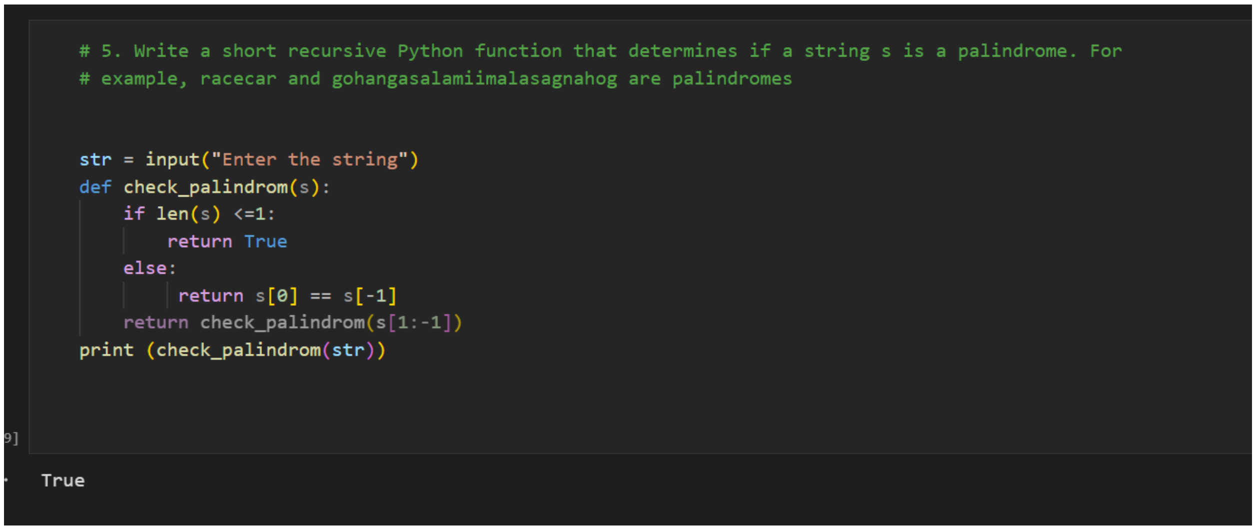
```
# 3. Write a recursive function to convert the entered string of digits into the integer it represents.
# For example, 13531 represents the integer 13,531.

result = 0
def stringToInteger(strValue):
    # if setting the base timing as 0:
    if len(strValue) == 1:
        return int(strValue)

    # convert the first character to an integer
    first_digit = int(strValue[0])

    # Recursive case:Multiply the recursive value by 10
    rem_str = strValue[1:]
    result = (10**(len(strValue)-1)) * int(strValue[0]) + stringToInteger(strValue[1:])

    return result

strValue = input("Enter the number\n")
result_integer = stringToInteger(strValue)
print(f'The representation of {strValue} is : {result_integer}')
```

```
The representation of 2345 is : 2345
```



```python
# 4. Write a short recursive Python function that takes a character string s and outputs its reverse.
# For example, the reverse of pots&pans would be snap&stop


# taking string value
str = input("Enter the string")
def reverseStr(s):
    if len(s)<=1:
        return s
    else:
        return s[-1]+reverseStr(s[0:len(s)-1])

print(f'Reverse of {str} is :{reverseStr(str)}')
```

```
Reverse of pots&pans is :snap&stop
```



```python
# 5. Write a short recursive Python function that determines if a string s is a palindrome. For
# example, racecar and gohangasalamiimalasagnahog are palindromes


str = input("Enter the string")
def check_palindrom(s):
    if len(s) <=1:
        return True
    else:
         return s[0] == s[-1]
    return check_palindrom(s[1:-1])
print (check_palindrom(str))
```

```
True
```