**Exercise no : 01**

**Date : 29.11.2021**

**GREATEST OF TWO NUMBERS**

**Aim :** To write a python program to find the greatest of two numbers.

**Algorithm:**

**Step1** : Let num1 and num2 be the variables

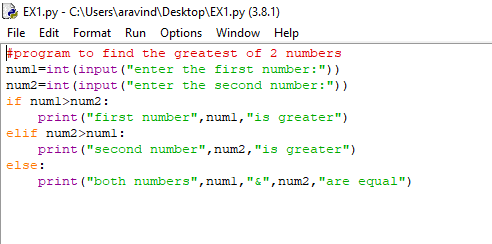
**Step2** : Assign values to num1 and num2

**Step3 :** if num1>num2 print num1 is greater than num2

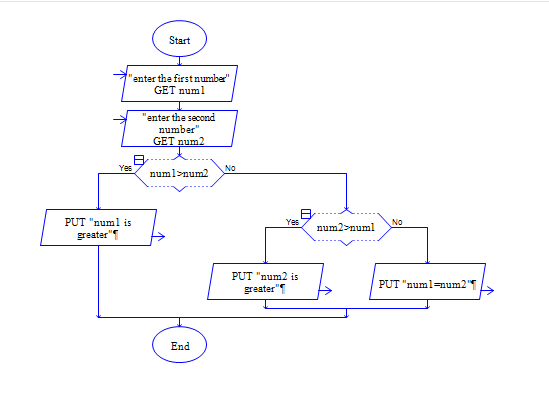
Elif num2>num1 print num2 is greater than num1

Else print num1 and num2 are equal

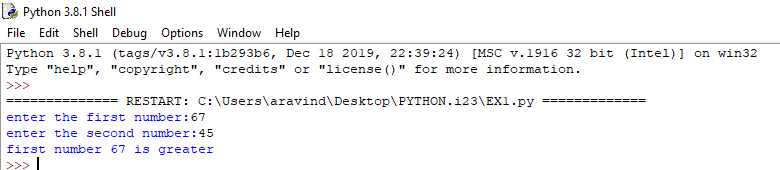
**Python code:**



**Raptor flowchart:**



**Output:**

****

**Exercise no : 02**

**Date : 15.11.2022**

**PALLINDROME**

**Aim:** to write a python programme to find whether the given number is a palindrome or not

**Algorithm:**

**Step 1**: get the number from user

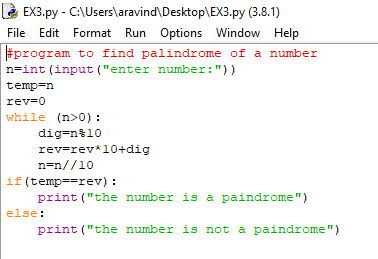
**Step 2**: declare temp=n and rev=0

**Step 3**: apply the loop condition n>0 if yes dig=n%10,rev=rev\*+dig,n=n//10

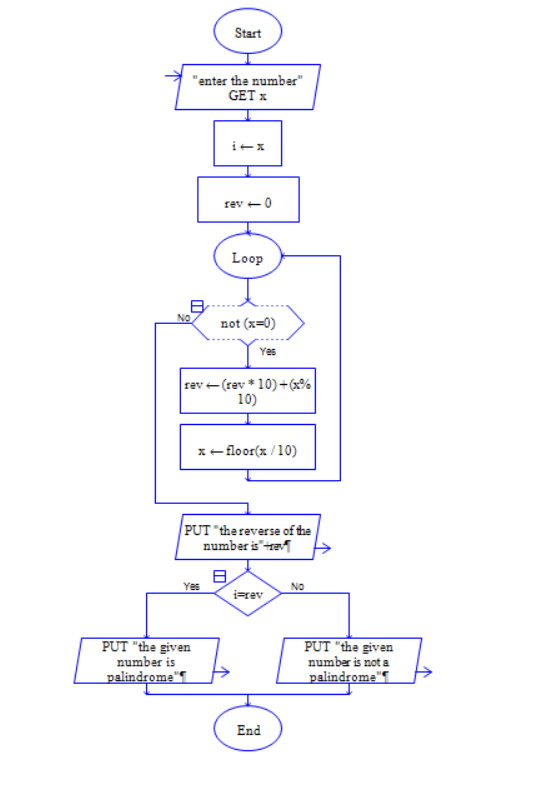
**Step 4**: if no exit from loop

**Step 5**: apply the condition (temp==rev) if yes it’s a palindrome if not its not a palindrome

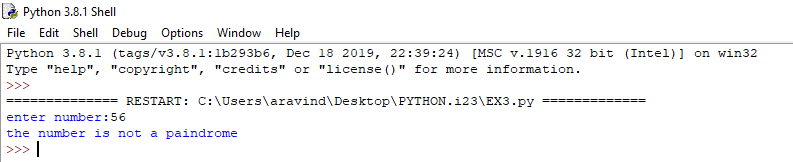
**Python code:**



**Raptor flowchart:**



**Output:**

****

**Exercise no : 03**

**Date : 15.11.2021**

**PRIME NUMBER**

**Aim:** to write a python program to find whether the given number is prime or not

**Algorithm:**

**Step 1**: get a number from user and assign to num

**Step 2**: apply the condition n=0 or n=1 if yes print the number is neither prime nor composite

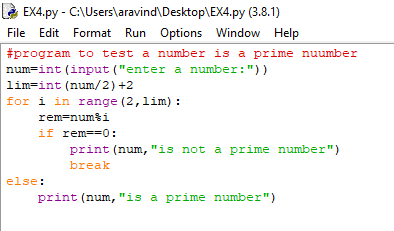
**Step 3**: if no then declare m=n-1

**Step 4**: then apply the loop with condition l=0 then declare l=n%m when the condition l=0 satisfies check for another condition m=1

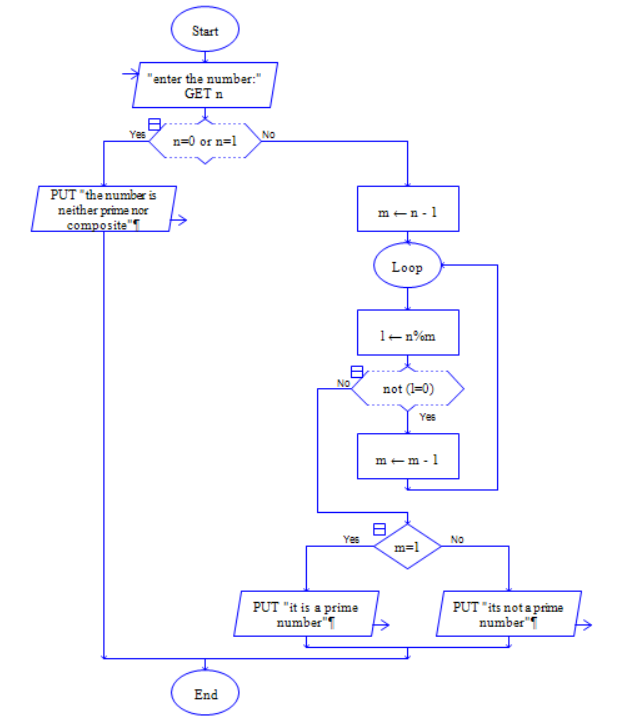
**Step 5**: when the condition l=0 is not satisfying then declare m=m-1 continue the loop until l=0 satisfes

**Step 6**: if the condition m=1 satisfies then print it is a prime number if not satisfied print it is not a prime number.

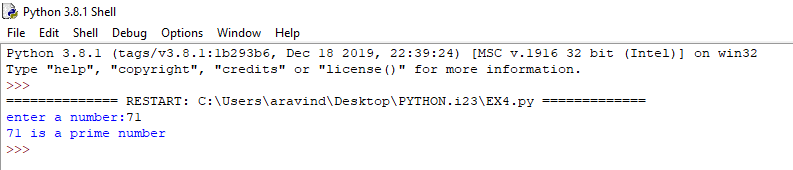
**Python code:**



**Raptor flowchart:**



**Output:**



**Exercise no : 04**

**Date : 15.11.2021**

**TO FIND THE YOUNGER AGE**

**Aim:** to write a python code to find the younger age among 2 people

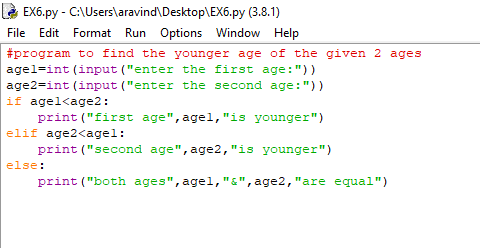
**Algorithm:**

**Step 1**: get the person 1 age as age1 and person 2 age as age2

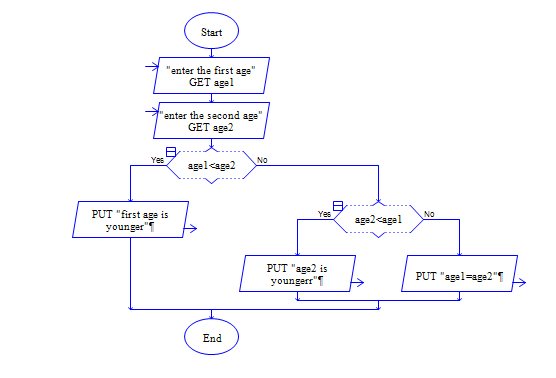
**Step 2**: if age1<age2 print person 1 is younger

Elif age2<age1 print person 2 is younger Else print both are equal

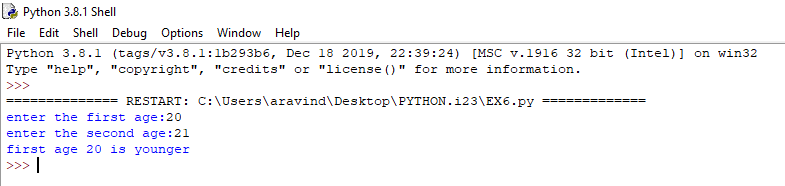
**Python code:**



**Raptor flowchart:**



**Output:**



**Exercise no : 05**

**Date : 15.11.2021**

**AVERAGE AGE OF 10 PLAYERS:**

**Aim:** to write python code to find average age of 10 players

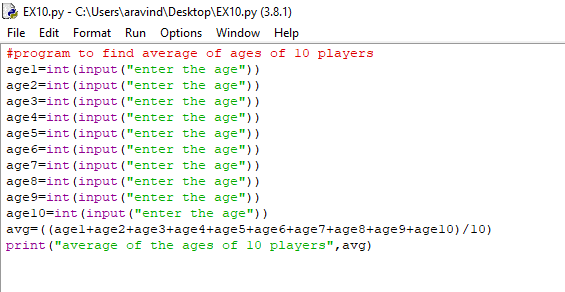
**Algorithm:**

**Step 1**: receive ten age values as age1,age2………age10 respectively

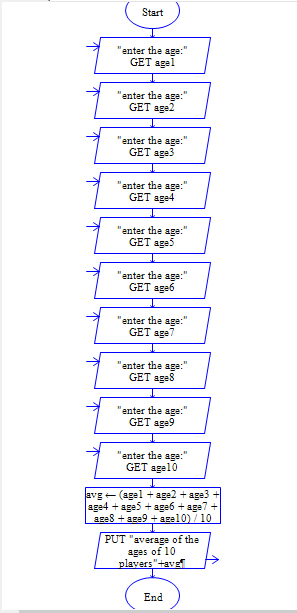
**Step 2**: assign avg= sum/10

**Step 3**: print avg

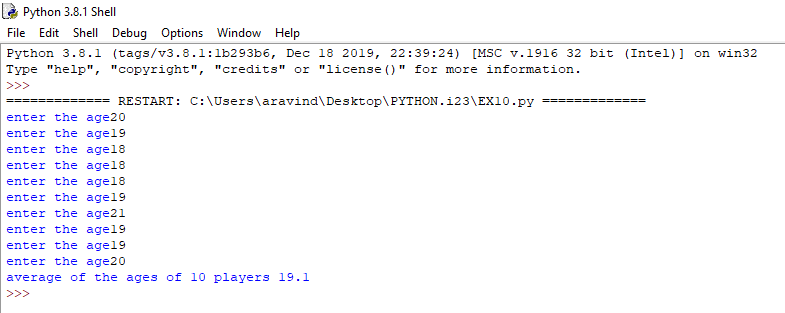
**Python code:**



**Raptor flowchart:**



**Output:**

****

**Exercise no : 06**

**Date : 15.11.2021**

**FACTORIAL PROGRAM**

**Aim:** to write a python program to find the factorial of the given number

**Algorithm:**

**Step 1** : enter the number from the user assign it as num

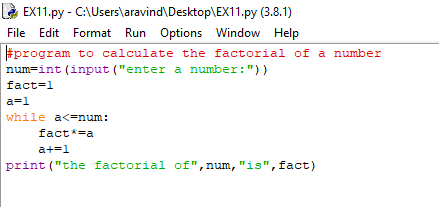
**Step 2** : assign fact=1

**Step 3** : check num=0

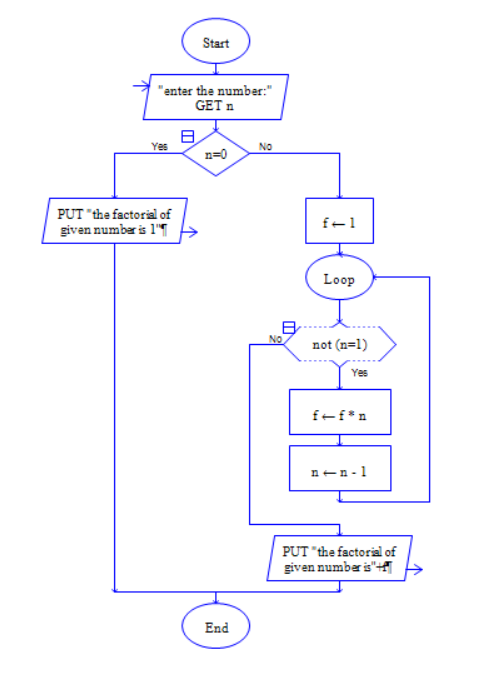
**Step 4** : if no compute fact=fact\*a+=1

**Step 5** : else put the factorial of the number is (fact)

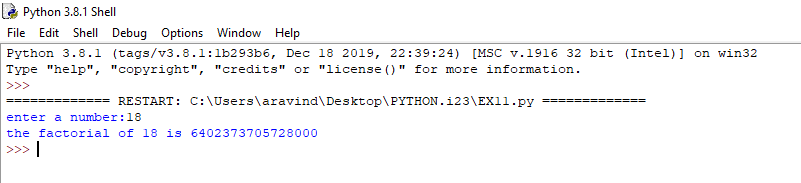
**Python code:**



**Raptor flowchart:**



**Output:**

****

**Exercise no : 07**

**Date : 4.12.2021**

**FIBONACCI SERIES**

**Aim:** to write a python code to display the Fibonacci series

**Algorithm:**

**Step 1** : enter the number of terms get num

**Step 2** : assign first=0,put first

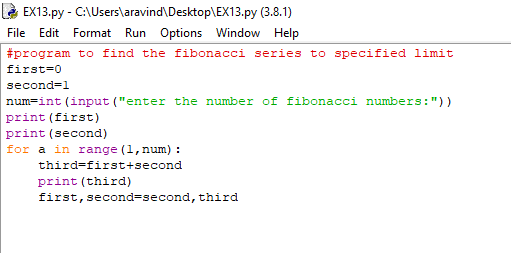
**Step 3** : assign first=0;second=1

**Step 4** : check num<=second

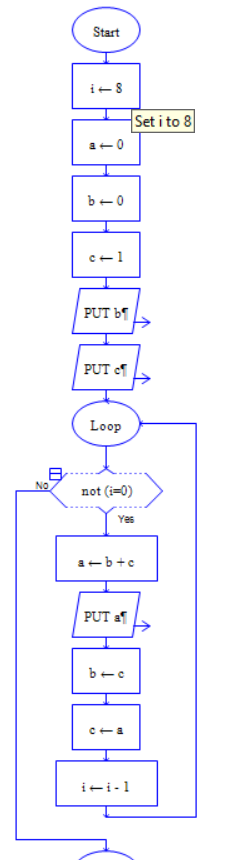
**Step 5** : third = first+second

**Step 6** : assign first,second=second,third

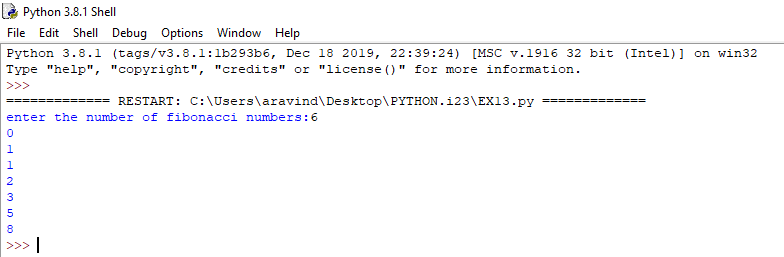
**Python code:**



**Raptor flowchart:**



**Output:**



**Exercise no : 08**

**Date : 4.12.2021**

**SUM OF SQUARE OF FIRST N NATURAL NUMBER**

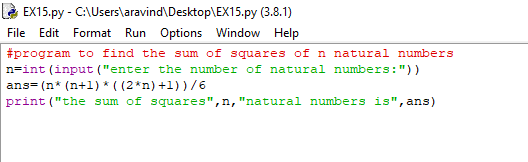
**Aim:**To write the python code to display the sum of square of first n natural number.

**Algorithm:**

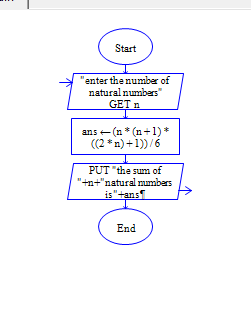
**Step 1** : enter the number of natural number as n

**Step 2** : ans = (n\*(n+1)\*((2\*n)+1))/6

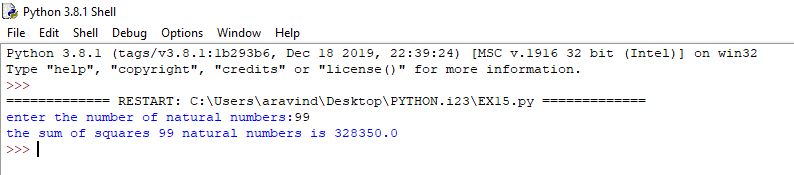
**Step 3** : print ans

**Python code:**

**Raptor flowchart:**



**Output:**



**Exercise no : 09**

**Date : 4.12.2021**

**ELECTRICITY BILL GENERATOR**

**Aim :** To create an electricity bill generator using if/else ladder.

**Algorithm:**

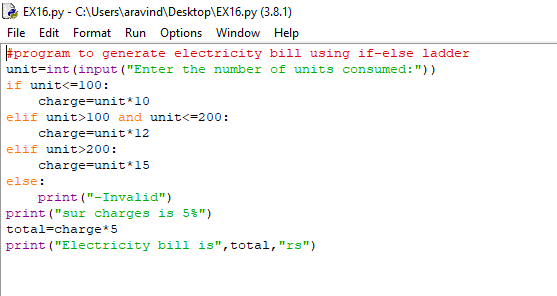
**Step 1** : Get the units in inputs as variable.

**Step 2** : If ‘unit’ is less than 100 then the price is Rs.10 per unit.

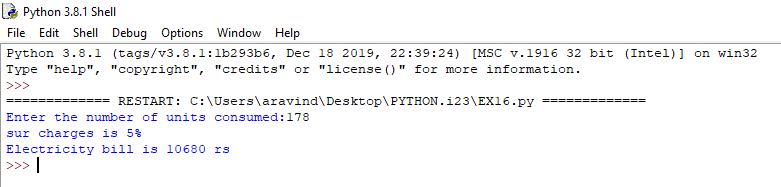
**Step 3** : If ‘unit’ is between 101 and 200 then the price is Rs.12 per unit.

**Step 4** : If ‘unit’ is more than 200 then the price is Rs.15 per unit.

**Python code:**



**Output:**



**Exercise no : 10**

**Date : 4.12.2021**

**ANAGRAMS**

**Aim :** To find the anagrams in the entered words by the user in the for loop.

**Algorithm:**

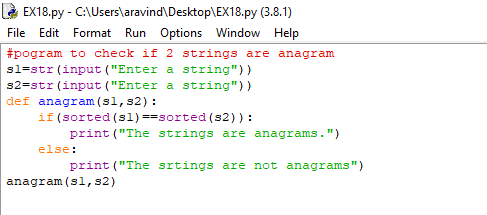
**Step 1** : we get two strings (s1,s2) from the user

**Step 2** : we define a function anagram(s1,s2)

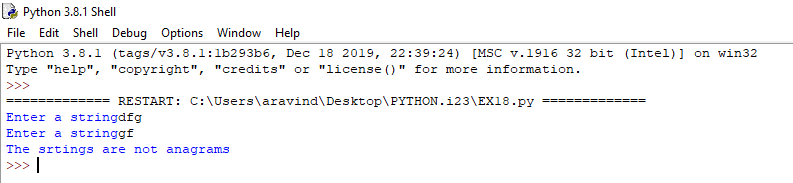
**Step 3** : if sorted value of s1 I equal to sorted value of s2 , print the strings are anagrams

**Step 4** : else print the strings are not anagrams

**Python code:**



**Output:**

****

**Exercise no : 11**

**Date : 7.12.2021**

**LENGTH OF A NUMBER**

**Aim :** To find the length of the given number.

**Algorithm:**

**Step 1** : we get the number as a

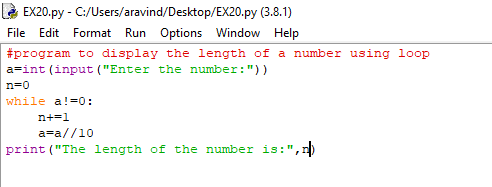
**Step 2** : Assign a variable n and equate it to 0.

**Step 3** : Put a while loop with a condition “a!=0”.

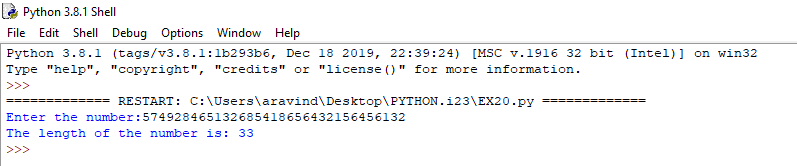
**Step 4** : n+=1

**Step 5** : print the length of number is n

**Python code:**



**Output:**



**Exercise no : 12**

**Date : 7.12.2021**

**STUDENT MARKS**

**Aim :** To create a program to input marks of 5 subjects (Physics, Chemistry, English, Maths, Computer Science) for 5 students. Display rank of each student also result of total marks, percentage, rank in a class. The rank is categorised as fail (mark<40%), pass & third grade (41% - 55%) second grade (56% to 65%), First grade (66% to 75%) Distinction (76% to 90%) and extraordinary (91% to 100%).

**Algorithm:**

**Step 1** : assingn two empty lists lst1,lst2

**Step 2** : for i in range of (0,6) obtain 5 subject marks

**Step 3** : assign total as sum of all subjects

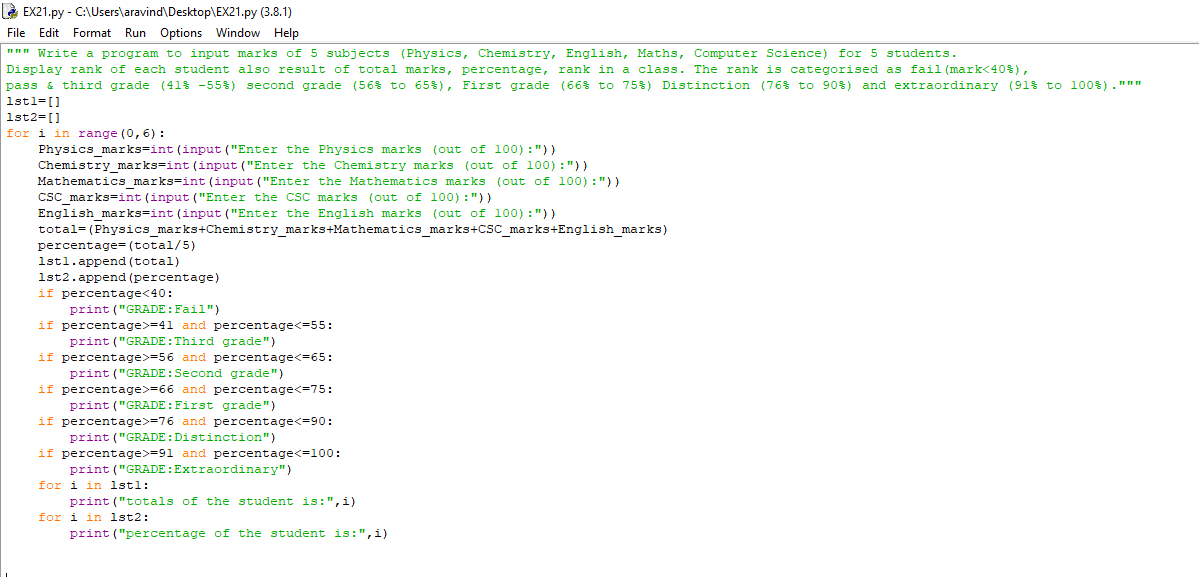
**Step 4** : assign percentage as sum/5

**Step 5** : append all total values in lst1

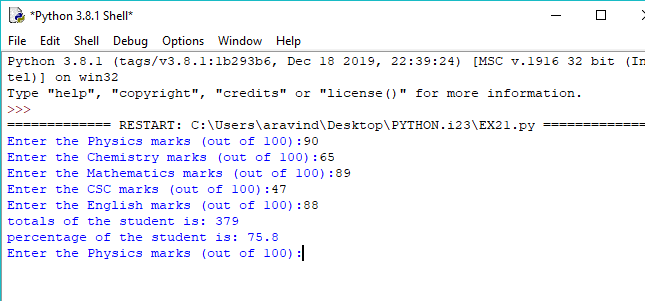
**Step 6** : append all percentage values in lst2

**Step 7** : according to the if-else statements display the grade of the students

**Python code:**



**Output:**

****

**Exercise no : 13**

**Date : 7.12.2021**

**MULTIPLICATION TABLE**

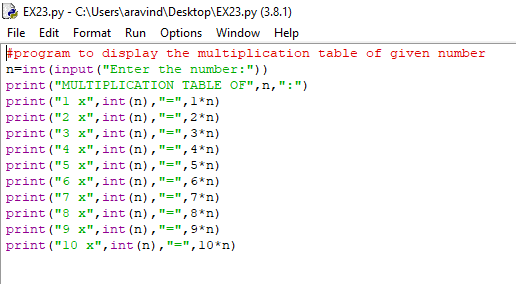
**Aim :** To create a program to get a number and print the multiplication table of it.

**Algorithm:**

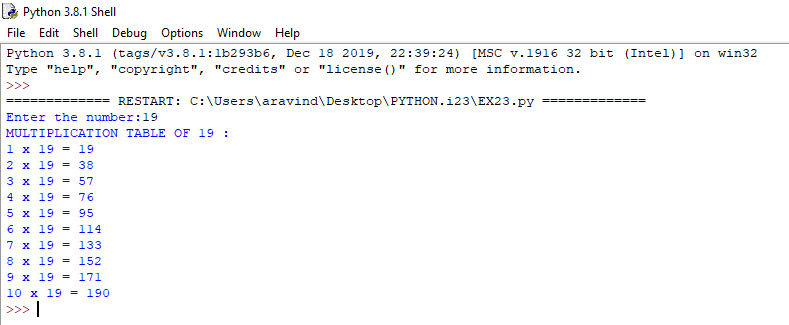
**Step 1** : obtain the number whose multiplication table is to be printed as n

**Step 2** : use series of print statements of the syntax to obtain a supposed strings of multiplication tables

**Python code:**



**Output:**

****

**Exercise no : 14**

**Date : 11.12.2021**

**SWAPPING TWO NUMBERS**

**Aim :** To create a program to swap two numbers.

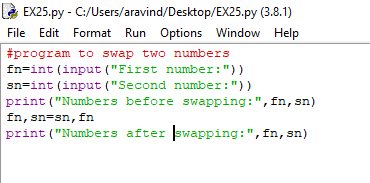
**Algorithm:**

**Step 1** : receive 2 variables fn and sn

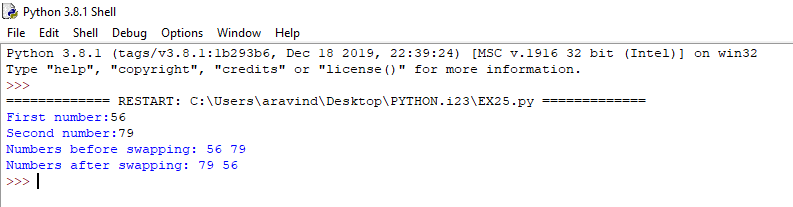
**Step 2** : assign fn,sn = sn,fn

**Step 3** : print fn , sn

**Python code:**



**Output:**



**Exercise no : 15**

**Date : 11.12.2021**

**TRANSPOSE OF A MATRIX**

**Aim :** To create a program to find the Transpose of a Matrix.

**Algorithm:**

**Step 1** : Assign a list x

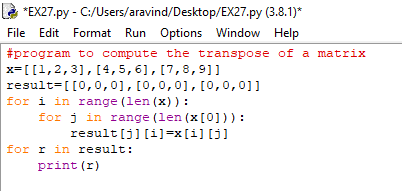
**Step 2** : Use a for loop with range length of the list ‘x’. Inside the loop put another for loop with range length of the first element in the loop.

**Step 3** : Now put the statement “x[i][j]=x[j][i]” to swap the elements.

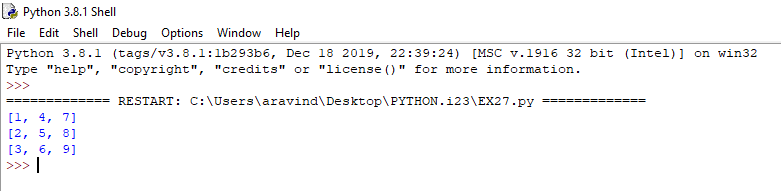
**Step 4** : Now put another for loop with range of the list “x”

**Step 5** : And inside the loop print the elements in the loop “x”

**Python code:**



**Output:**



**Exercise no : 16**

**Date : 21.12.2021**

**DIAGONAL OF A MATRIX**

**Aim :** To create a program to find the diagonal elements of a matrix.

**Algorithm:**

**Step 1**: Assign an empty list ‘lst’ and assign a nested list ‘x’

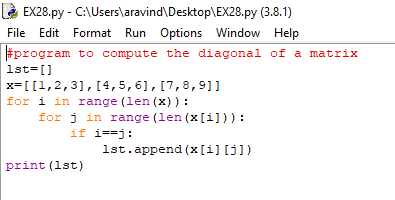
**Step 2**: Use a for loop with range as length of the list ‘x’

**Step 3**: Use another for loop with range as length of first element in ‘x’

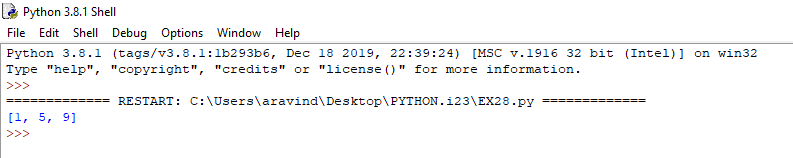
**Step 4**: Now if i=j then append first element of ‘x’ in lst

**Step 5**: Then print lst

**Python code:**



**Output:**



**Exercise no : 17**

**Date : 21.12.2021**

**LENGTH OF A STRING**

**Aim :** To create a program to find the length of the given string.

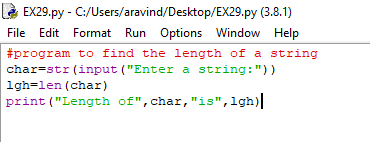
**Algorithm:**

**Step 1** : Get the input from the user in the variable ‘char'

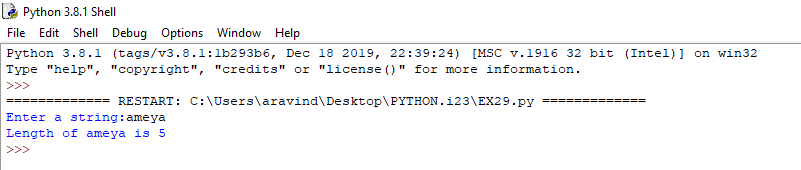
**Step 2** : Assign lgh as len(char)

**Step 3** : Print lgh as the length of char

**Python code:**



**Output:**



**Exercise no : 18**

**Date : 21.12.2021**

**COUNT OF VOWELS IN A STRING**

**Aim :** To create a program to find the count of vowels in a string and to display the vowels.

**Algorithm:**

**Step 1** : Get the input string in the variable ‘char’.

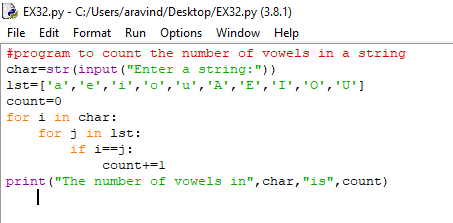
**Step 2** : Assign a list lst all the vowels as elements (upper and lower cases)

**Step 3** : Assign variable count=0

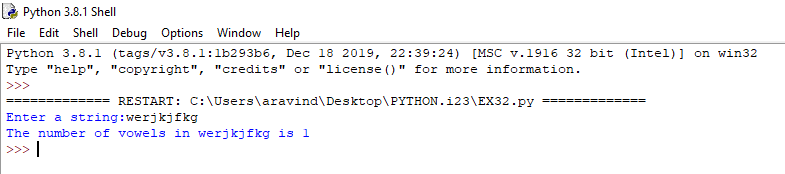
**Step 4** : Use 2 for loops one inside the other with temporary variables i and j , if i==j increment count by 1 at every equality.

**Step 5** : Now print “The no. of vowels in the string are” and display the variable ‘count’,

**Python** **code:**



**Output:**



**Exercise no : 19**

**Date : 21.12.2021**

**TO FIND ANY SPECIAL CHARACTER IN STRING**

**Aim :** To create a program to find if there is any special character in a string.

**Algorithm:**

**Step 1** : Get the input in the variable ‘char’. Assign an empty list named ‘spl\_char’.

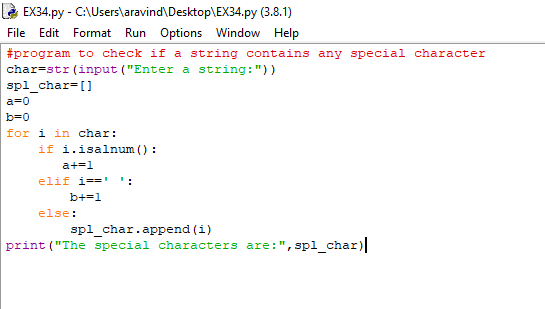
**Step 2** : Put variable ‘a’ and ‘b’ and equate them to 0. Now put a ‘for’ loop with range ‘char’.

**Step 3** : Inside the loop we use if condition and check if the input with the condition ‘isalnum’ which checks if the input has numbers or string. If it is true then variable ‘a’ is incremented or if the input is ‘space’ then variable ‘b’ is incremented.

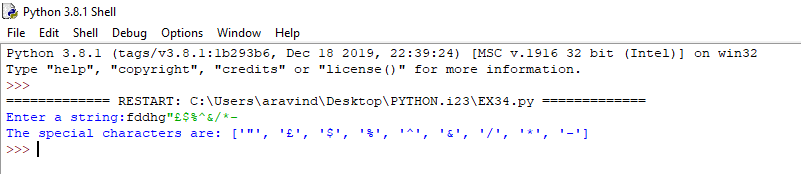
**Step 4** : If false then append the input in the list ‘spl\_char’.

**Step 5** : Now print “The special characters are:” then display the list ‘spl\_char’.

**Python code:**



**Output:**

****

**Exercise no : 20**

**Date : 27.12.2021**

**TO SPLIT AND JOIN A STRING**

**Aim :** To create a program to split and join a string.

**Algorithm:**

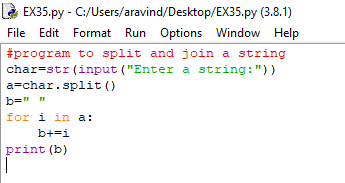
**Step 1** : Get the input in the variable ‘char’. Now split ‘char’ using “char.split()” function and assign it to a new variable ‘a’.

**Step 2** : Create an empty variable ‘b’.

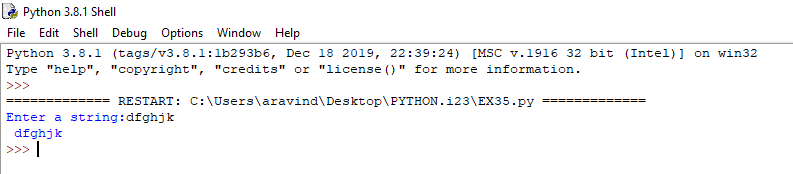
**Step 3** : Use a ‘for’ loop with range ‘a’. Inside the loop append the input to the variable ‘b’.

**Step 4** : Print the variable ‘b’.

**Python code:**

****

**Output:**

****

**Exercise no : 21**

**Date : 27.12.2021**

**COUNT OF LETTERS , DIGITS AND SPECIAL CHARACTERS**

**Aim :** To create a program to count the letters, digits and special characters from the given string.

**Algorithm:**

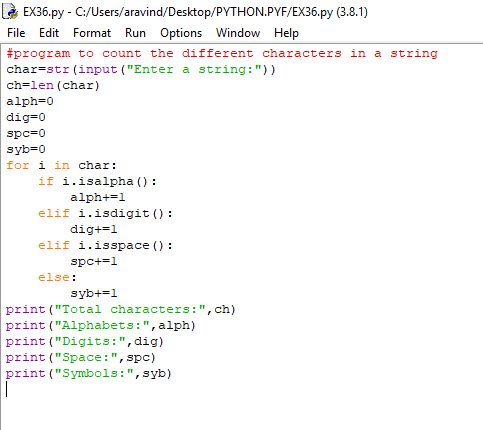
**Step 1 :** Get the input in the variable ‘char’.

**Step 2 :** Create three variables ‘alph’, ’dig’, ’spc’,’syb’ and assign all of them to 0.

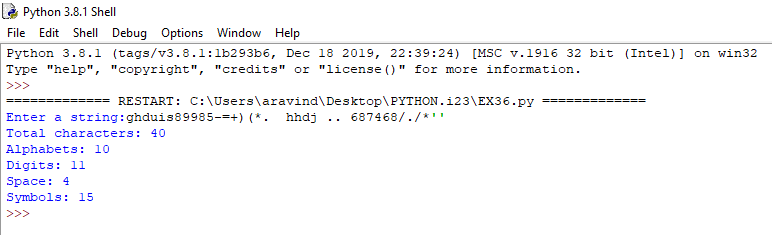
**Step 3 :** Put a for loop with range ‘len(char)’, inside the loop use the ‘if’ condition and check whether the given input are letters, digits, special character and increment the respective variables by 1.

**Step 4 :** Now print the corresponding variables holding the number of letter, digits,space and special characters.

**Python code:**

****

**Output:**

****

**Exercise no : 22**

**Date : 27.12.2021**

**REPLACE SPECIAL CHARACTERS WITH ‘#’**

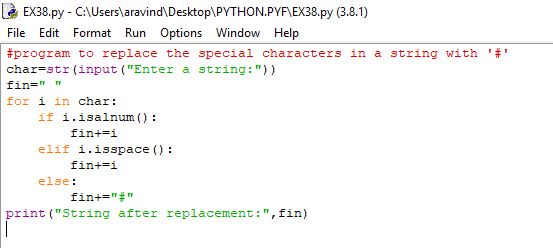
**Aim :** To create a program to replace the special characters of the given string with ‘#’ symbol.

**Algorithm:**

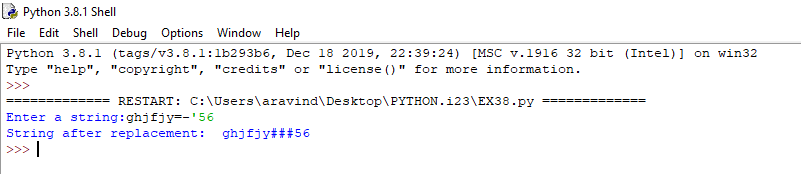
**Step 1 :** Get the input in the variable ‘char’. Assign an empty string variable named ‘fin’.

**Step2 :** Put a ‘for’ loop with the range ‘char’, inside the loop use ‘if’ condition to check whether the input contains only letters , numbers or space , if true fin+=I , if false then replace ‘i’ with ‘#’ .

**Python code:**

****

**Output:**

****

**Exercise no : 23**

**Date : 3.1.2022**

**NEW STRING WITH FIRST , LAST AND MIDDLE ELEMENT OF THE GIVEN STRING**

**Aim :** To create a program to find a word made with the 1st, middle and last letters of the given word.

**Algorithm:**

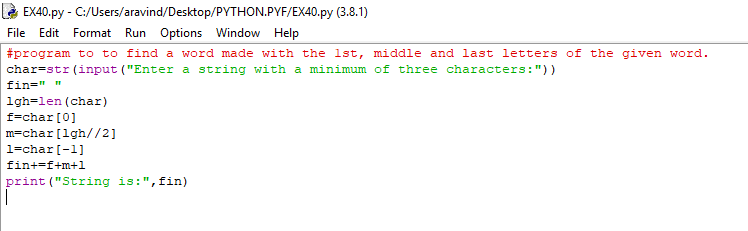
**Step 1 :** Get the input in the variable ‘char’. Assign an empty string variable ‘fin’.

**Step 2 :** Put a variable lgh holding length of char. Now put a variable f holding the fist element of char. Put a variable m holding middle element of char. Put a variable l holding the last element of char.

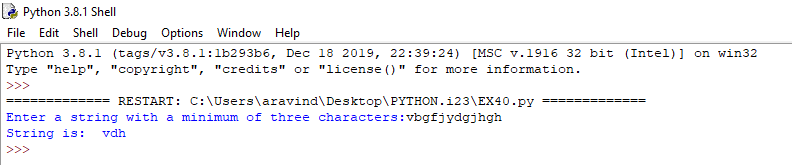
**Step 3 :** Now we add f+m+l to fin.

**Step 4 :** Then we print the fin variable which holds the new string.

**Python code:**

****

**Output:**

****

**Exercise no : 24**

**Date : 3.2.2022**

**LENGTH OF A LIST**

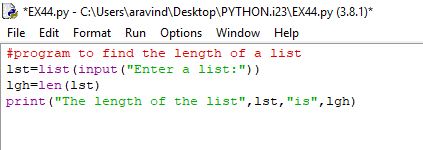
**Aim :** To create a program to count the length of the list.

**Algorithm:**

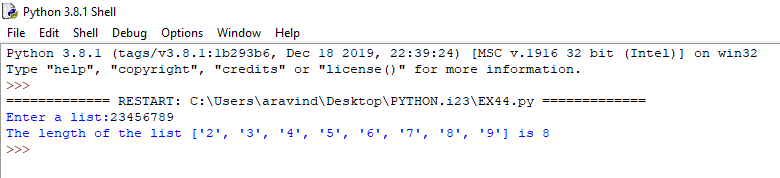
**Step 1** : Get the input in the variable ‘lst’.

**Step 2** : Assign lgh as len (lst) , print lgh

**Python code:**



**Output:**

****

**Exercise no : 25**

**Date : 3.1.2022**

**CONCATENATION OF 2 LISTS**

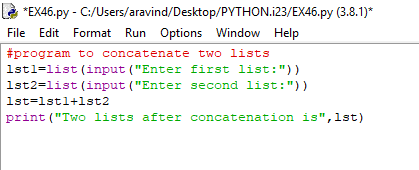
**Aim :** To create a program to concatenate 2 lists.

**Algorithm:**

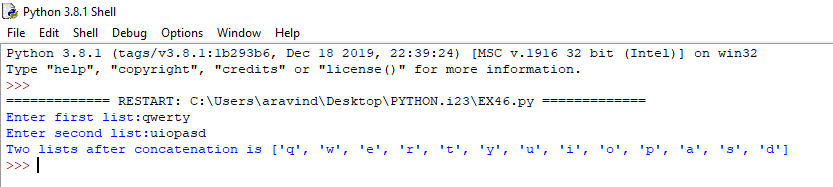
**Step 1** : Get the 2 lists as input in the variables ‘lst1’ and ‘lst2’.

**Step 2** : Equate ‘lst’ to lst1+lst2. Then print “The concatenated list is:” and print variable lst.

**Python code:**



**Output:**



**Exercise no : 26**

**Date : 3.1.2022**

**EVEN NUMBER FINDER**

**Aim :** To create a program to find the even numbers in the entered list.

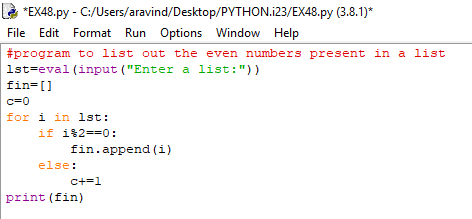
**Algorithm:**

**Step 1** : Get the input in the variable ‘lst’. Assign two variables ‘fin’ to [] and ‘c’ to 0.

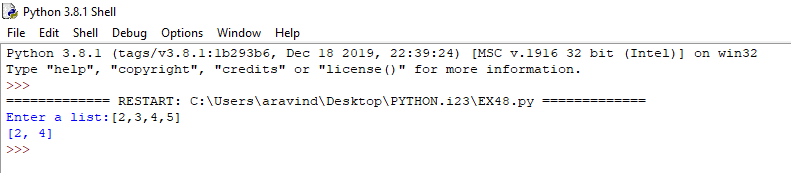
**Step 2** : Use a ‘for’ loop with range ‘lst’. Inside the loop put an ‘if’ condition to check whether the numbers in the list is even.

**Step 3** : If true the append fin with i, or else increment c by 1. Then print “The even numbers in the list are,” then display fin.

**Python code:**



**Output:**



**Exercise no : 27**

**Date : 4.1.2022**

**SECOND SMALLEST NUMBER IN THE LIST**

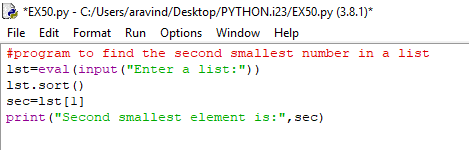
**Aim :** To create a program to find the second smallest number in the list.

**Algorithm :**

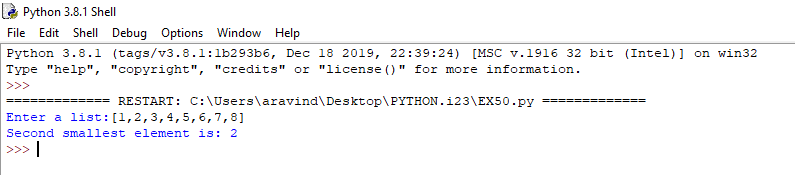
**Step 1** : Get the input in the variable ‘lst’. Use sort function to sort the list.

**Step 2** : Then print “The second smallest element is:”,‘lst[1]’.

**Python code:**



**Output:**



**Exercise no : 28**

**Date : 4.1.2022**

**JOIN NUMBERS FROM THE LIST**

**Aim :** To create a program to join the numbers in the list.

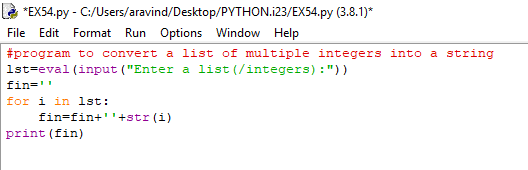
**Algorithm:**

**Step 1** : Get the input in the variable lst and assign an empty string variable ‘fin’.

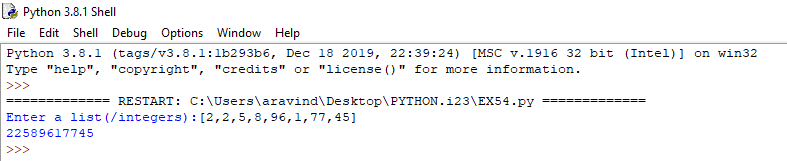
**Step 2** : Put a ‘for’ loop with range ‘lst’. Append fin with str(i).

**Step 3** : print fin

**Python code:**



**Output:**



**Exercise no : 29**

**Date : 4.1.2022**

**SECOND SMALLEST NUMBER IN A LIST**

**Aim :** To create a program to find the second smallest number in the list.

**Algorithm:**

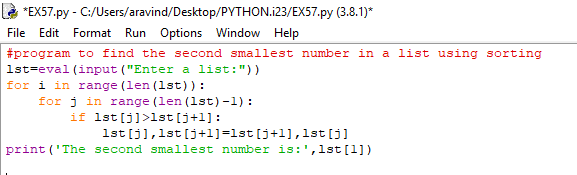
**Step 1** : Get the input in the list ‘lst’.

**Step 2** : We put a ‘for’ loop with range ‘len(lst)’.

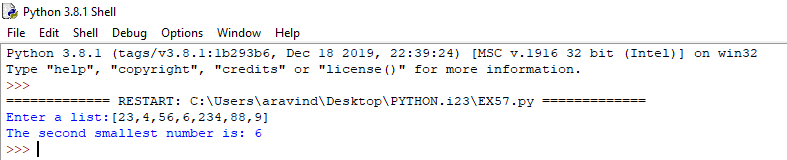
**Step 3** : Put another ‘for’ loop with range ‘len(lst)-i-1’, inside it put an ‘if’ loop checking “lst[j]>lst[j+1]”.

**Step 4** : If true the change “lst[j],lst[j+1]=lst[j+1],lst[j]”. Come out of the loop and print ”The second smallest numbers is:”,lst[1].

**Python code:**



**Output:**



**Exercise no : 30**

**Date : 25.1.2022**

**SWAPPING THE FIRST AND LAST ELEMENT OF A LIST**

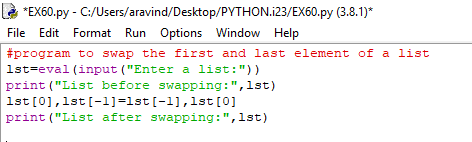
**Aim :** To write a program to swap the first and last element of the list.

**Algorithm:**

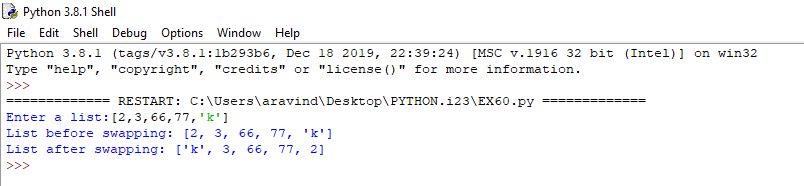
**Step 1** : Get the input in the variable ‘lst’,Now we print(lst) , Now to swap we say “lst[0],lst[-1]=lst[-1],lst[0].

**Step 2** : Now we print(lst).

**Python code:**



**Output:**



**Exercise no : 31**

**Date : 25.1.2022**

**LENGTH OF A DICTIONARY**

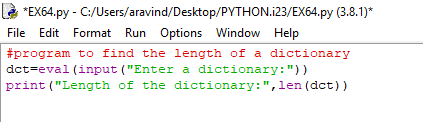
**Aim :** To create a program to find the length of the given Dictionary.

**Algorithm:**

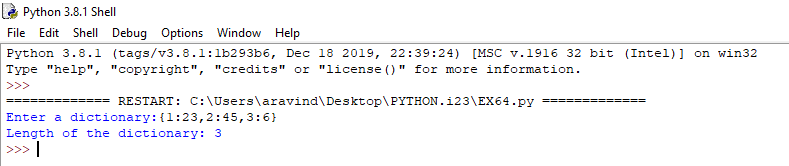
**Step 1** : Get the input in the variable ‘dct’.

**Step 2** : Then print “Length of the dictionary is:”,len(dct).

**Python code:**



**Output:**



**Exercise no : 32**

**Date : 25.1.2022**

**DICTIONARY CONCATENATION**

**Aim :** To create a program to concatenate dictionaries.

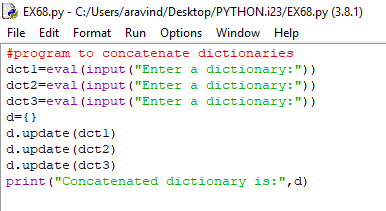
**Algorithm:**

**Step 1** : Get 3 dictionaries as input in the variable ‘dct1’, ‘dct2’ and ‘dct3’. Assign an empty dictionary named ‘d’.

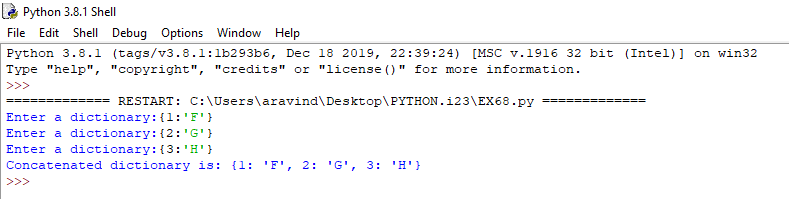
**Step 2** : Use update function to update the values of ‘dct1’, ‘dct2’, ‘dct3’ in the empty dictionary ‘d’.

**Step 3** : Now print “The concatenated dictionary is:”,d.

**Python code:**



**Output:**



**Exercise no : 33**

**Date : 14.2.2022**

**MAXIMUM AND MINIMUM VALUES IN DICTIONARY**

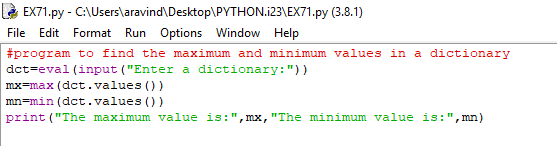
**Aim :** To create a program to find the maximum and minimum values in a dictionary.

**Algorithm:**

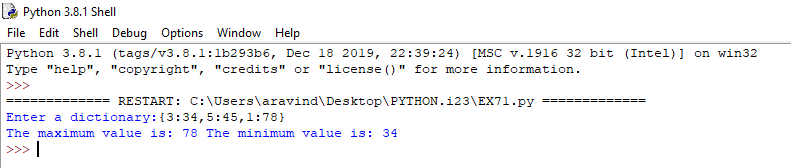
**Step 1** : Get the input in the variable ‘dct’. Assign variable ‘mx’ and ‘mn’ holding maximum and minimum values of the dictionary respectively.

**Step 2** : Then print “The maximum value is:”,mx and “The minimum value is:”,mn.

**Python code:**



**Output:**



**Exercise no : 34**

**Date : 14.2.2022**

**SUM OF NUMBERS USING FUNCTION**

**Aim :** To create a program to find the sum of the list of number using function.

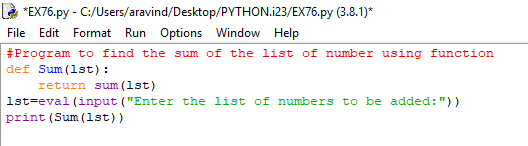
**Algorithm:**

**Step 1**: Define a function ‘Sum’ with a variable ‘lst’

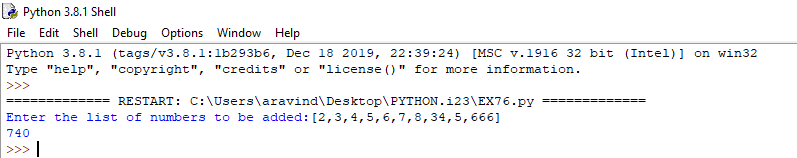
**Step 2**: Put a return statement returning the value of sum(lst)

**Step 3**: Move out of the function and get the input in the variable ‘lst’. print Sum(lst)

**Python code:**



**Output:**

****

**Exercise no : 35**

**Date : 21.2.2022**

**GEOMETRIC MEAN USING FUNCTION**

**Aim :** To create a program to find the sum of the list of number using function.

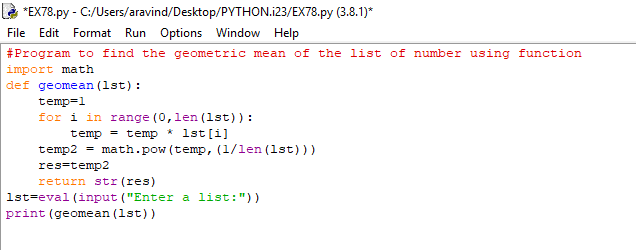
**Algorithm:**

**Step 1**: Define a function ‘geomean’,assign temp=1 ,apply a for loop in the range of the length of the list.

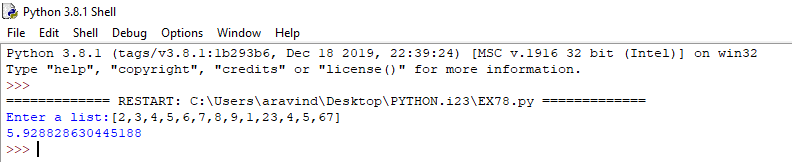
**Step 2**: Increment temp by a product of lst[i] ,temp2=math.pow(temp,(1/len(lst)))

**Step 3**: Assign res=temp2 ,return resget the input in the variable‘lst’and print(geomean(lst)).

**Python code:**



**Output:**

****

**Exercise no : 36**

**Date : 21.2.2022**

**PERMUTATION OF GIVEN STRING**

**Aim:**To create a program to find all the permutations of the given string.

**Python code:**

**Algorithm:**

**Step 1 :** Take string as the input from the user and store it in a variable str to permute the given  string

**Step 2 :** Declare a python function to find the permutation

Inside the function if the length of the string is equal to zero then call the permutation  function

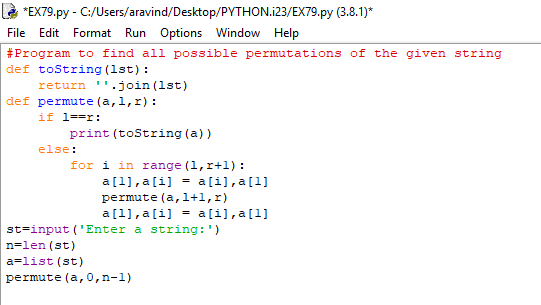
**Step 3 :**Assign the list to store the permutated string from index one to the last index and assign  a new empty list

Using for loop every value of i is traversed in the range of the string from index zero to  the length of the string stored in the list.

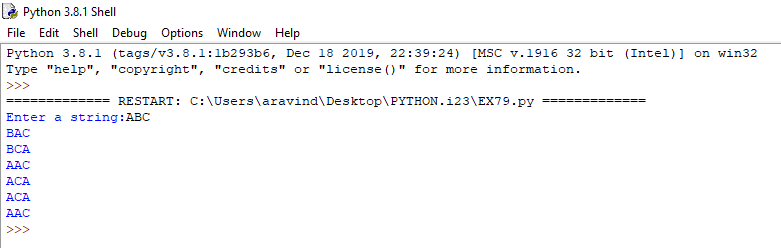
Again, using the for loop every value of j is traversed in the range of the string index zero  to the length of the actual string entered by the user

**Step 4 :**Assign a new variable and store this condition prev\_list[i] [0: j] +str [0] +prev\_list[i] [j:  len(str)-1]

Now update the string stored in the new variable to the already created empty list • Call the function recursively with the argument of the appended list and print the  permutated string.

****

**Output:**

****

**Exercise no : 37**

**Date : 21.2.2022**

**COMBINATION OF STRINGS WITH SPACES**

**Aim:**To create a program to print all the combination of strings with spaces.

**Algorithm:**

**Step 1** : Take a string input from the user and store it in a variable str for finding permutated  strings with spaces between them

**Step 2** : From math module import the power function

**Step 3** : Declare the python function to find the sequences of the string

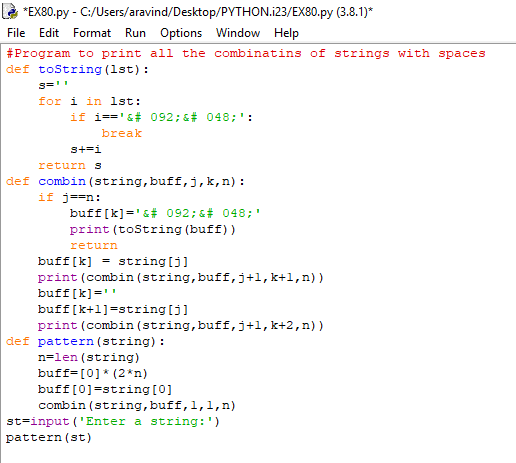
**Step 4** : Assign a variable to store the length of the string entered and assign another  variable(opsize) to store the power of (2, len(str-1))

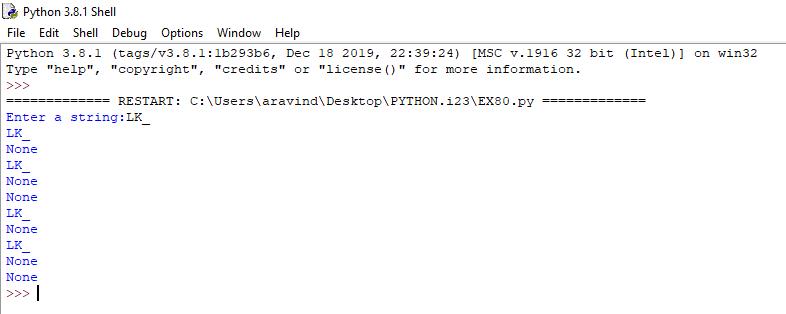
**Step 5** : using a for loop every element in the range of opsize and for every value of j in the range  of value of the length of the string print the string of j

**Step 6** : If the element of opsize and (1<<j) is true print the end statement

**Step 7** : If driver code is true then the sequences of the string is printed

**Python code:**

****

**Output:** ****

**Exercise no : 38**

**Date : 21.2.2022**

**PERMUTATION OF GIVEN STRING**

**Aim:**To create a program to find all the permutations of the given string.

**Algorithm:**

**Step 1** : Take string as the input from the user and store it in a variable str to permute the given  string

Declare a python function to find the permutation

Inside the function if the length of the string is equal to zero then call the permutation  function

**Step 2** : Assign the list to store the permutated string from index one to the last index and assign  a new empty list

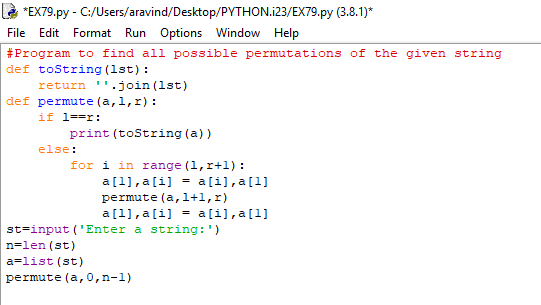
Using for loop every value of i is traversed in the range of the string from index zero to  the length of the string stored in the list.

Again, using the for loop every value of j is traversed in the range of the string index zero  to the length of the actual string entered by the user

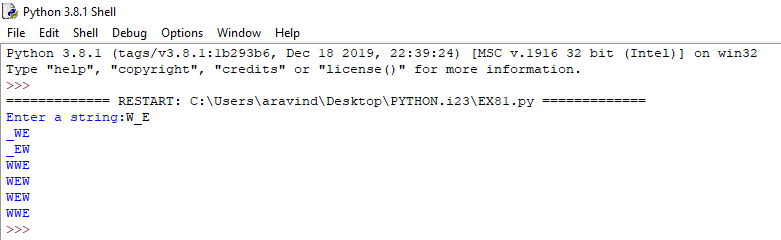
**Step 3** : Assign a new variable and store this condition prev\_list[i] [0: j] +str [0] +prev\_list[i] [j:  len(str)-1]

Now update the string stored in the new variable to the already created empty list • Call the function recursively with the argument of the appended list and print the  permutated string.

**Python code:**

****

**Output:**

****

**Exercise no : 39**

**Date : 21.2.2022**

**SUM USING RECURSION**

**Aim:**To create a program to find the sum of first 10 numbers using recursion.

**Algorithm:**

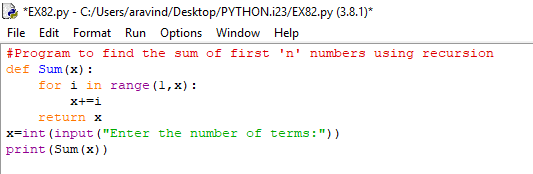
**Step 1** : Take a list as the input from the user and store it in a variable A to find the maximum  recursions of the string

Define a python function to find the sum from one to x

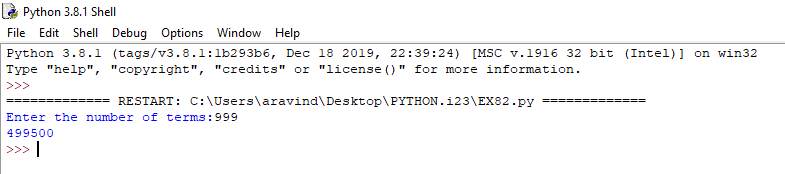
**Step 2** : If the length of the list entered by the user is equal to one call the function  recursively with the argument of A [0] then call off the function recursively with an  argument to find the maximum value of the last index of the list

If driver code is true then print the python function

**Python code:**

****

**Output:**

****

**Exercise no : 40**

**Date : 22.2.2022**

**FIBONACCI USING RECURSION**

**Aim:**To create a program to print the Fibonacci number using recursion.

**Algorithm:**

**Step 1** : Take a list as the input from the user and store it in a variable A to find the maximum  recursions of the string

Define a python function to find the sum from one to x

Take a number from the user and store it in a variable n to find the Fibonacci  numbers

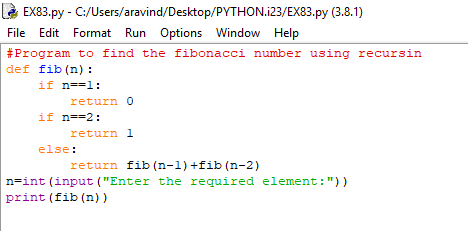
**Step 2** : Declare a python function to find the nth Fibonacci number

If n is less than or equal to zero print it is an incorrect input

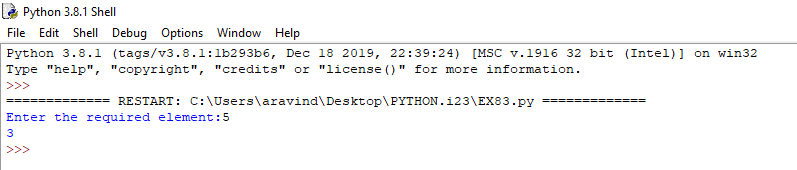
**Step 3** : Else if n is equal to one call the function recursively with an argument zero  • Else if n is equal to two call the function recursively with an argument one  • Else call the function recursively with an argument by adding the last and the second  last input by the user

Print the Fibonacci numbers

**Python code:**

****

**Output:**



**Exercise no : 41**

**Date : 22.2.2022**

**NO OF DIGITS USING RECURSION**

**Aim:**To create a program to find the no. of digits present in the number.

**Algorithm:**

**Step 1** : Take floating-point number as the input from the user and store it in the variable  num

Declare a python function to count the number of digits

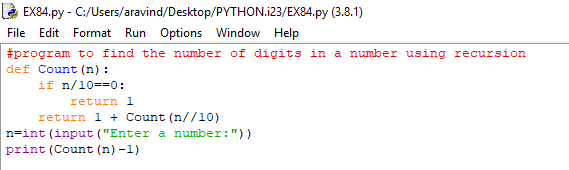
To count the number of digits entered by the user

**Step 2** : If the value entered by the user is less than ten call the function recursively with an  argument one

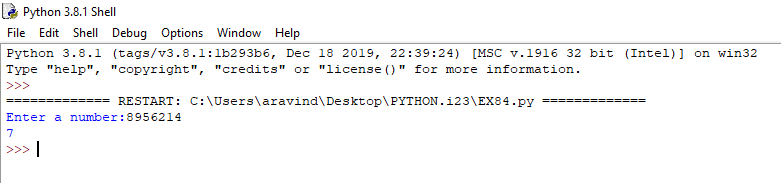
Else call off the function recursively with an argument one adding the count of n  divided by ten

Print the number of digits

**Python code:**



**Output:**



**Exercise no : 42**

**Date : 22.2.2022**

**MAXIMUM NUMBER IN THE LIST USING RECURSION**

**Aim:**To create a program to find the maximum number in the list using recursion.

**Algorithm:**

**Step 1** : Get the array for which the maximum is to be found

Recursively find the maximum according to the following:

Recursively traverse the array from the end

Base case: If the remaining array is of length 1, return the only present element i.e.  arr[0] if(n == 1)

return arr[0];

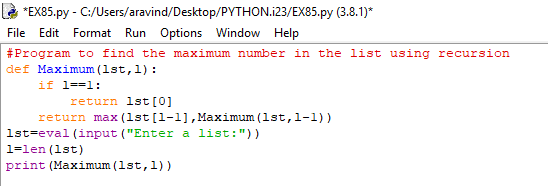
**Step 2** : Recursive call: If the base case is not met, then call the function by passing the array  of one size less from the end, i.e. from arr[0] to arr[n-1].

Return statement: At each recursive call (except for the base case), return the  maximum of the last element of the current array (i.e. arr[n-1]) and the element  returned from the previous recursive call.

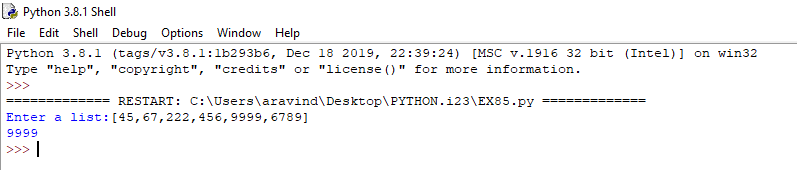
return max(arr[n-1], recursive\_function(arr, n-1));

Print the returned element from the recursive function as the maximum element

**Python code:**

****

**Output:**



**Exercise no : 43**

**Date : 22.2.2022**

**OCCURANCE OF WORDS**

**Aim:**To create a program to count the occurence of words in a text file.

**Algorithm:**

**Step 1 :** Input filename with location of the file.

Assign a variable name to read the file.

split the read file and store it in a variable.

Assign a variable and create an empty list for it.

Using for loop every element in the file is traversed and split.

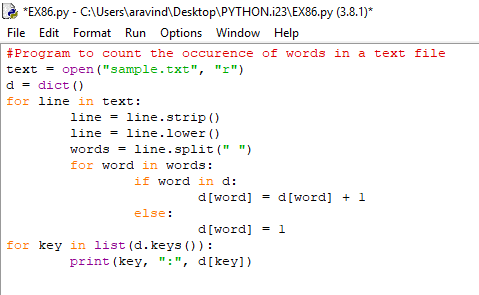
**Step 2** : If the traversed element not found in empty list update the element into the  empty list using append () function.

Using for loop every element in the list is traversed.

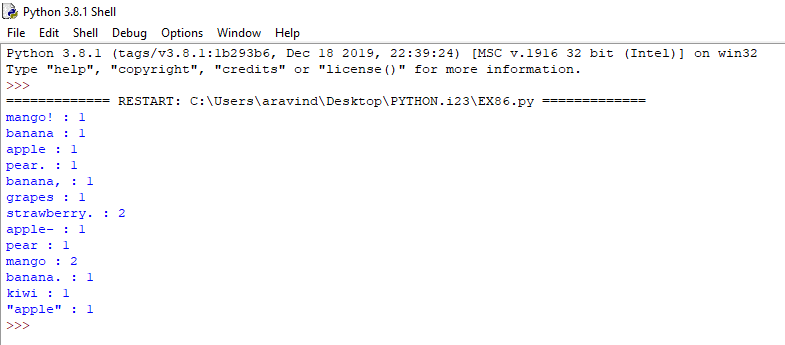
**Step 3** : If every element given is a word, the repeated words are counted using count () function and stored in temporary variable.

The element traversed and the count of the words are printed.

**Python code:**

****

**Output:**

****

**Exercise no : 44**

**Date : 22.2.2022**

**COUNT OF VOWELS**

**Aim:**To create a program to count the number of vowels in a text file.

**Algorithm:**

**Step 1** : Take the input file name from the user with the location of the file. • Open the file using open () function.

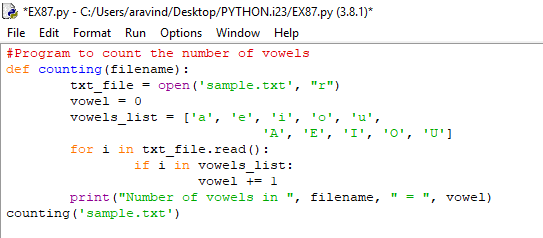
Assign the initial value of total as 0.

**Step 2** : Mention all the vowels both in lowercase and uppercase.

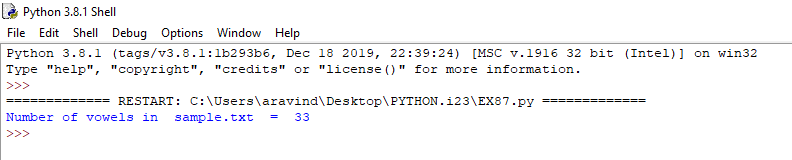
Using for loop every character in in the file opened is read by traversing it. • If the given character is a vowel, then the count of total is increased for every  vowel read in the file.

The file is closed using close () function and the total number of vowels is  printed.

**Python code:**

****

**Output:**

****

**Exercise no : 45**

**Date : 22.2.2022**

**COPYING TEXT FILE**

**Aim:**To create a program to copy a text file to another file.

**Algorithm:**

**Step 1** : Input the filename and the location of the file and open it using open () function

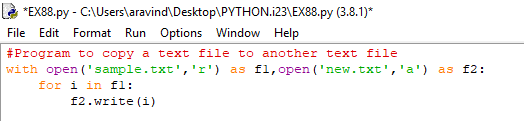
Open another file to copy the text from the first file and name the file as f1.

**Step 2** : Using for loop every line in the file is traversed and the file1 is scripted to write  mode.

Using open () function the file1 is opened and the text from file is copied to file1.

The copied text from file is printed.

**Python code:**

****

**Exercise no : 46**

**Date : 22.2.2022**

**MOST REPEATED WORD**

**Aim:**To create a program to find the most repeated word in a text file.

**Algorithm:**

**Step 1 :** Open the text file using open () function with f as file object in r+ mode. • Using the read () function read the file using reader object a and store it in a  variable.

Create an empty dictionary d.

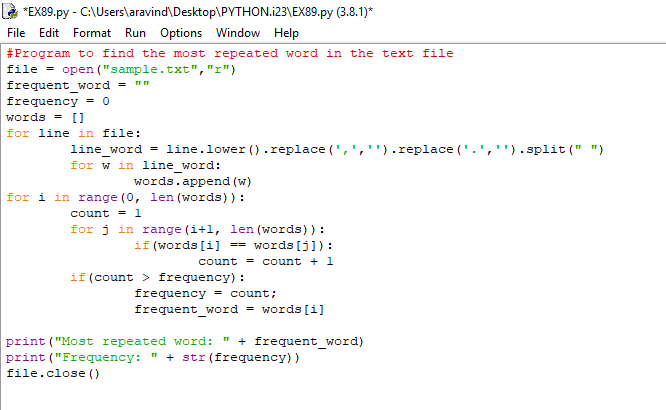
Store each word of the text file in b using a split () function.

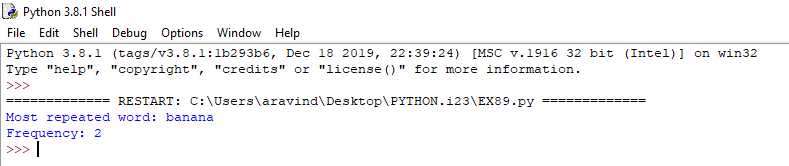
**Step 2** : Loop through b and if a word is not present in d, store it in d with value 1, else if the  word is already present in d increase its value by 1.

Loop through d and print each key, value pair if the value of the key is greater than 1 to find the most common words in the text file.

Close the file using close () function.

**Python code:**

**Output:**

****

**Exercise no : 47**

**Date : 22.2.2022**

**TOTAL VALUES**

**Aim:**To create a program to count the total lines,words,spaces and characters in a text file.

**Algorithm:**

**Step 1** : Input the filename and location of the file and open the file using open ()  function.

Initialize the no of lines, no of words and no of characters to 0.

Using for loop every line in the file is traversed.

**Step 2** : The strip () function is used to copy the string by removing both the leading and  the trailing characters to form a line.

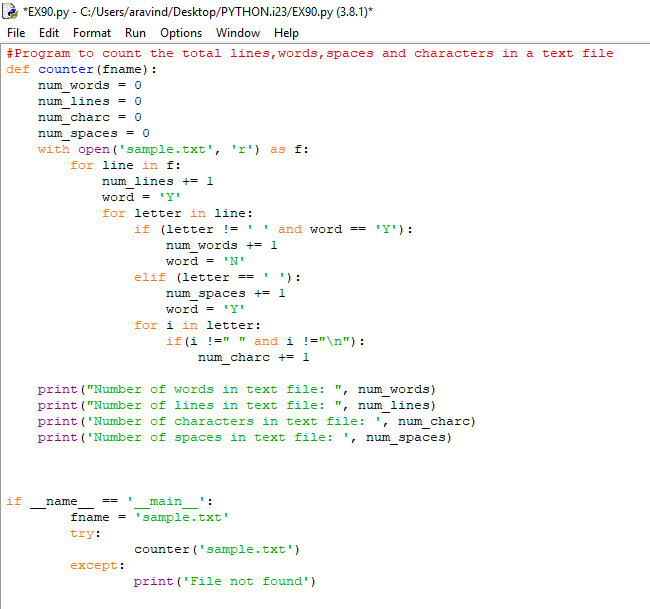
To form words the lines formed is split using split () function.

**Step 3** : For every line in the file, the count is increased and stored in a variable. • For every word in the file, the count is increased and stored in a variable added  to the length of the words.

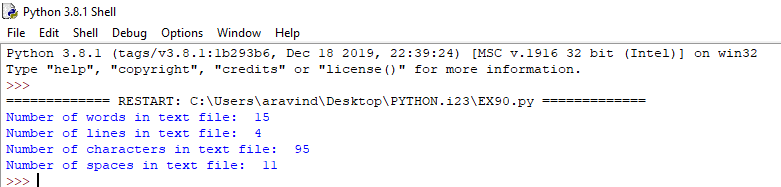
For every character in the file, the count is increased and stored in a variable added to the length of the line in the file.

The file is closed using close () function and the no of lines, no of words and no of  characters is printed for the text in the file.

**Python code:**

****

**Output:**

****

**PROJECT FILE**

**CONTENT:**

* **Introduction and synopsis**
* **System requirement**
* **Source code**
* **Sample output**

**INTRODUCTION AND SYNOPSIS:**

The Bank Account Management System is an application for maintaining a person's account in a bank.This project shows the working of a banking account system and cover the basic functionality of a Bank Account Management System.This is a prototype project which can be improved by using other programming languages like MySQL,PHP and HTML(Django/Flask). To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks. Also to enable the user’s work space to have additional functionalities which are not provided under a conventional banking project. The Bank Account Management System undertaken as a project is based on relevant technologies. The main aim of this project is to develop a prototype software for Bank Account Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcomed by this project. This project is developed using Python language Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer, while proving compliance and staying on the schedule and within budget. The impact of a poorly expressed requirement can bring a business out of compliance or even cause injury or death. Requirements definition and management is an activity that can deliver a high, fast return on investment. The system is designed as an interactive and content management system. The content management system deals with data entry, validation confirm and updating whiles the interactive system deals with system interaction with the administration and users. Thus, above features of this project will save transaction time and therefore increase the efficiency of the system.

**SYSTEM REQUIREMENT:**

**HARDAWARE REQUIREMENT:**

* **Processor –** Intel(R)core(TM)i3-3120M CPU @ 2.50GHZ
* **Manufacture –** Intel
* **Clock–** 2.5GHZ
* **Number Of Cores –** 4
* **Family – 06**
* **Model –** 3A
* **Stepping –** 9
* **RAM –** 4.0GB
* **Video card -** Intel(R)HD Graphics 4000
* **Chipset -** Intel(R)HD Graphics 4000
* **Dedicated Memory –** 32MB
* **Total Memory –** 2.0GB

**SOFTWARE REQUIREMENT:**

* **Operating System –** Windows 10
* **Service Pack –** 0
* **Size –** 32-bit
* **Version –** 10.0.18362
* **Locale –** 0409
* **Code Editor –** Visual Basic Studio Code 2019
* **Interpreter –** Python IDLE 3.8(32-bit)

**SOURCE CODE:**

from tkinter import \*

import os

from PIL import ImageTk, Image

#Main Screen

master = Tk()

master.title('Bank Management')

#Functions

def finish\_reg():

    name = temp\_name.get()

    age = temp\_age.get()

    gender = temp\_gender.get()

    password = temp\_password.get()

    all\_accounts = os.listdir()

    if name == "" or age == "" or gender == "" or password == "":

        notif.config(fg="red",text="All fields requried \* ")

        return

    for name\_check in all\_accounts:

        if name == name\_check:

            notif.config(fg="red",text="Account already exists")

            return

        else:

            new\_file = open(name,"w")

            new\_file.write(name+'\n')

            new\_file.write(password+'\n')

            new\_file.write(age+'\n')

            new\_file.write(gender+'\n')

            new\_file.write('0')

            new\_file.close()

            notif.config(fg="green", text="Account has been created")

def register():

    #Vars

    global temp\_name

    global temp\_age

    global temp\_gender

    global temp\_password

    global notif

    temp\_name = StringVar()

    temp\_age = StringVar()

    temp\_gender = StringVar()

    temp\_password = StringVar()

    #Register Screen

    register\_screen = Toplevel(master)

    register\_screen.title('Register')

    #Labels

    Label(register\_screen, text="Please enter your details below to register", font=('Algerian',12)).grid(row=0,sticky=N,pady=10)

    Label(register\_screen, text="Name", font=('Centaur',12)).grid(row=1,sticky=W)

    Label(register\_screen, text="Age", font=('Centaur',12)).grid(row=2,sticky=W)

    Label(register\_screen, text="Gender", font=('Centaur',12)).grid(row=3,sticky=W)

    Label(register\_screen, text="Password", font=('Centaur',12)).grid(row=4,sticky=W)

    notif = Label(register\_screen, font=('Centaur',12))

    notif.grid(row=6,sticky=N,pady=10)

    #Entries

    Entry(register\_screen,textvariable=temp\_name).grid(row=1,column=0)

    Entry(register\_screen,textvariable=temp\_age).grid(row=2,column=0)

    Entry(register\_screen,textvariable=temp\_gender).grid(row=3,column=0)

    Entry(register\_screen,textvariable=temp\_password,show="\*").grid(row=4,column=0)

    #Buttons

    Button(register\_screen, text="Register", command = finish\_reg, font=('Centaur',12)).grid(row=5,sticky=N,pady=10)

def login\_session():

    global login\_name

    all\_accounts = os.listdir()

    login\_name = temp\_login\_name.get()

    login\_password = temp\_login\_password.get()

    for name in all\_accounts:

        if name == login\_name:

            file = open(name,"r")

            file\_data = file.read()

            file\_data = file\_data.split('\n')

            password  = file\_data[1]

            #Account Dashboard

            if login\_password == password:

                login\_screen.destroy()

                account\_dashboard = Toplevel(master)

                account\_dashboard.title('Dashboard')

                #Labels

                Label(account\_dashboard, text="Account Dashboard", font=('Algerian',12)).grid(row=0,sticky=N,pady=10)

                Label(account\_dashboard, text="Welcome "+name, font=('Algerian',12)).grid(row=1,sticky=N,pady=5)

                #Buttons

                Button(account\_dashboard, text="Personal Details",font=('Centaur',12),width=30,command=personal\_details).grid(row=2,sticky=N,padx=10)

                Button(account\_dashboard, text="Deposit",font=('Centaur',12),width=30,command=deposit).grid(row=3,sticky=N,padx=10)

                Button(account\_dashboard, text="Withdraw",font=('Centaur',12),width=30,command=withdraw).grid(row=4,sticky=N,padx=10)

                Label(account\_dashboard).grid(row=5,sticky=N,pady=10)

                return

            else:

                login\_notif.config(fg="red", text="Password incorrect!!")

                return

    login\_notif.config(fg="red", text="No account found !!")

def deposit():

    #Vars

    global amount

    global deposit\_notif

    global current\_balance\_label

    amount = StringVar()

    file   = open(login\_name, "r")

    file\_data = file.read()

    user\_details = file\_data.split('\n')

    details\_balance = user\_details[4]

    #Deposit Screen

    deposit\_screen = Toplevel(master)

    deposit\_screen.title('Deposit')

    #Label

    Label(deposit\_screen, text="Deposit", font=('Calibri',12)).grid(row=0,sticky=N,pady=10)

    current\_balance\_label = Label(deposit\_screen, text="Current Balance : ₹"+details\_balance, font=('Centaur',12))

    current\_balance\_label.grid(row=1,sticky=W)

    Label(deposit\_screen, text="Amount : ", font=('Centaur',12)).grid(row=2,sticky=W)

    deposit\_notif = Label(deposit\_screen,font=('Centaur',12))

    deposit\_notif.grid(row=4, sticky=N,pady=5)

    #Entry

    Entry(deposit\_screen, textvariable=amount).grid(row=2,column=1)

    #Button

    Button(deposit\_screen,text="Finish",font=('Centaur',12),command=finish\_deposit).grid(row=3,sticky=W,pady=5)

def finish\_deposit():

    if amount.get() == "":

        deposit\_notif.config(text='Amount is required!',fg="red")

        return

    if float(amount.get()) <=0:

        deposit\_notif.config(text='Negative currency is not accepted', fg='red')

        return

    file = open(login\_name, 'r+')

    file\_data = file.read()

    details = file\_data.split('\n')

    current\_balance = details[4]

    updated\_balance = current\_balance

    updated\_balance = float(updated\_balance) + float(amount.get())

    file\_data       = file\_data.replace(current\_balance, str(updated\_balance))

    file.seek(0)

    file.truncate(0)

    file.write(file\_data)

    file.close()

    current\_balance\_label.config(text="Current Balance : ₹"+str(updated\_balance),fg="green")

    deposit\_notif.config(text='Balance Updated', fg='green')

def withdraw():

     #Vars

    global withdraw\_amount

    global withdraw\_notif

    global current\_balance\_label

    withdraw\_amount = StringVar()

    file   = open(login\_name, "r")

    file\_data = file.read()

    user\_details = file\_data.split('\n')

    details\_balance = user\_details[4]

    #Deposit Screen

    withdraw\_screen = Toplevel(master)

    withdraw\_screen.title('Withdraw')

    #Label

    Label(withdraw\_screen, text="Deposit", font=('Centaur',12)).grid(row=0,sticky=N,pady=10)

    current\_balance\_label = Label(withdraw\_screen, text="Current Balance : ₹"+details\_balance, font=('Centaur',12))

    current\_balance\_label.grid(row=1,sticky=W)

    Label(withdraw\_screen, text="Amount : ", font=('Centaur',12)).grid(row=2,sticky=W)

    withdraw\_notif = Label(withdraw\_screen,font=('Centaur',12))

    withdraw\_notif.grid(row=4, sticky=N,pady=5)

    #Entry

    Entry(withdraw\_screen, textvariable=withdraw\_amount).grid(row=2,column=1)

    #Button

    Button(withdraw\_screen,text="Finish",font=('Centaur',12),command=finish\_withdraw).grid(row=3,sticky=W,pady=5)

def finish\_withdraw():

    if withdraw\_amount.get() == "":

        withdraw\_notif.config(text='Amount is required!',fg="red")

        return

    if float(withdraw\_amount.get()) <=0:

        withdraw\_notif.config(text='Negative currency is not accepted', fg='red')

        return

    file = open(login\_name, 'r+')

    file\_data = file.read()

    details = file\_data.split('\n')

    current\_balance = details[4]

    if float(withdraw\_amount.get()) >float(current\_balance):

        withdraw\_notif.config(text='Insufficient Funds!', fg='red')

        return

    updated\_balance = current\_balance

    updated\_balance = float(updated\_balance) - float(withdraw\_amount.get())

    file\_data       = file\_data.replace(current\_balance, str(updated\_balance))

    file.seek(0)

    file.truncate(0)

    file.write(file\_data)

    file.close()

    current\_balance\_label.config(text="Current Balance : ₹"+str(updated\_balance),fg="green")

    withdraw\_notif.config(text='Balance Updated', fg='green')

def personal\_details():

    #Vars

    file = open(login\_name, 'r')

    file\_data = file.read()

    user\_details = file\_data.split('\n')

    details\_name = user\_details[0]

    details\_age = user\_details[2]

    details\_gender = user\_details[3]

    details\_balance = user\_details[4]

    #Personal details screen

    personal\_details\_screen = Toplevel(master)

    personal\_details\_screen.title('Personal Details')

    #Labels

    Label(personal\_details\_screen, text="Personal Details", font=('Centaur',12)).grid(row=0,sticky=N,pady=10)

    Label(personal\_details\_screen, text="Name : "+details\_name, font=('Centaur',12)).grid(row=1,sticky=W)

    Label(personal\_details\_screen, text="Age : "+details\_age, font=('Centaur',12)).grid(row=2,sticky=W)

    Label(personal\_details\_screen, text="Gender : "+details\_gender, font=('Centaur',12)).grid(row=3,sticky=W)

    Label(personal\_details\_screen, text="Balance :₹"+details\_balance, font=('Centaur',12)).grid(row=4,sticky=W)

def login():

    #Vars

    global temp\_login\_name

    global temp\_login\_password

    global login\_notif

    global login\_screen

    temp\_login\_name = StringVar()

    temp\_login\_password = StringVar()

    #Login Screen

    login\_screen = Toplevel(master)

    login\_screen.title('Login')

    #Labels

    Label(login\_screen, text="Login to your account", font=('Centaur',12)).grid(row=0,sticky=N,pady=10)

    Label(login\_screen, text="Username", font=('Centaur',12)).grid(row=1,sticky=W)

    Label(login\_screen, text="Password", font=('Centaur',12)).grid(row=2,sticky=W)

    login\_notif = Label(login\_screen, font=('Centaur',12))

    login\_notif.grid(row=4,sticky=N)

    #Entry

    Entry(login\_screen, textvariable=temp\_login\_name).grid(row=1,column=1,padx=5)

    Entry(login\_screen, textvariable=temp\_login\_password,show="\*").grid(row=2,column=1,padx=5)

    #Button

    Button(login\_screen, text="Login", command=login\_session, width=15,font=('Centaur',12)).grid(row=3,sticky=W,pady=5,padx=5)

#Image import

img = Image.open('#bank.png')

img = img.resize((150,150))

img = ImageTk.PhotoImage(img)

#Labels

Label(master, text = "BANKING", font=('Centaur',14)).grid(row=0,sticky=N,pady=10)

Label(master, text = "-BETA VERSION:)", font=('Centaur',12)).grid(row=1,sticky=N)

Label(master, image=img).grid(row=2,sticky=N,pady=15)

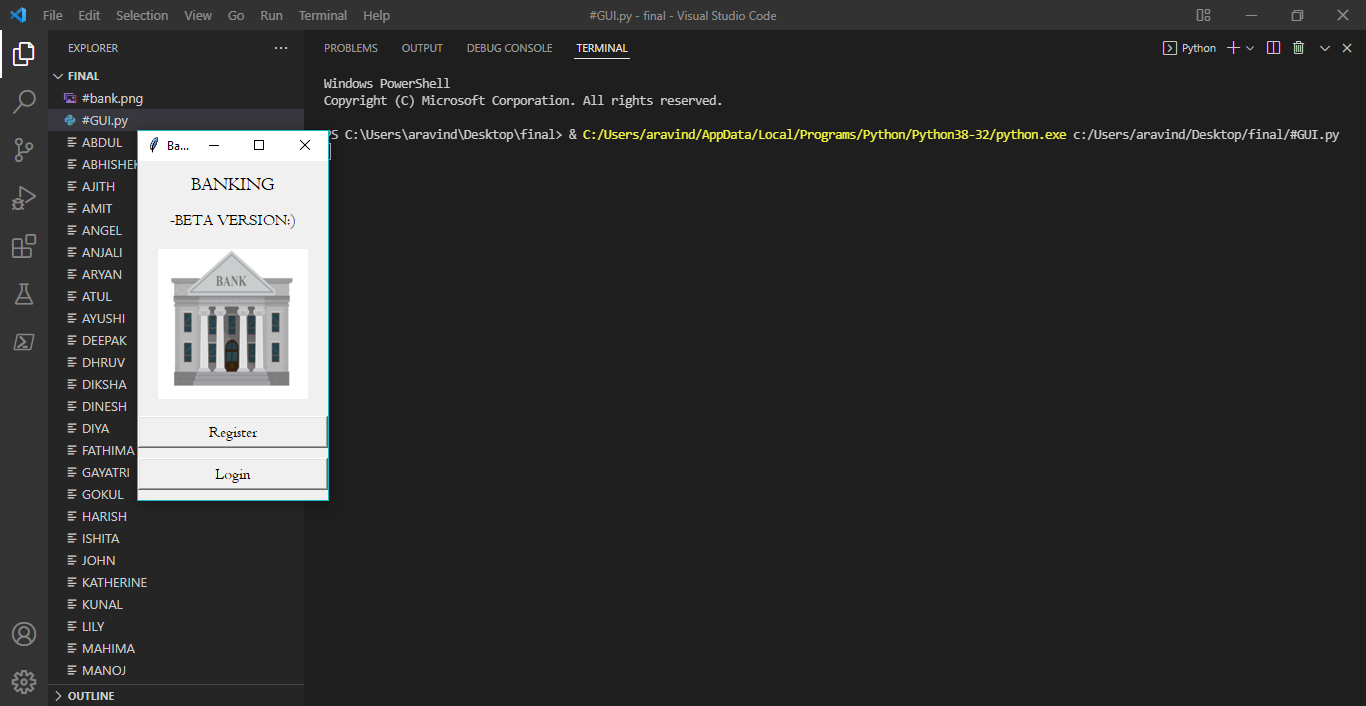
#Buttons

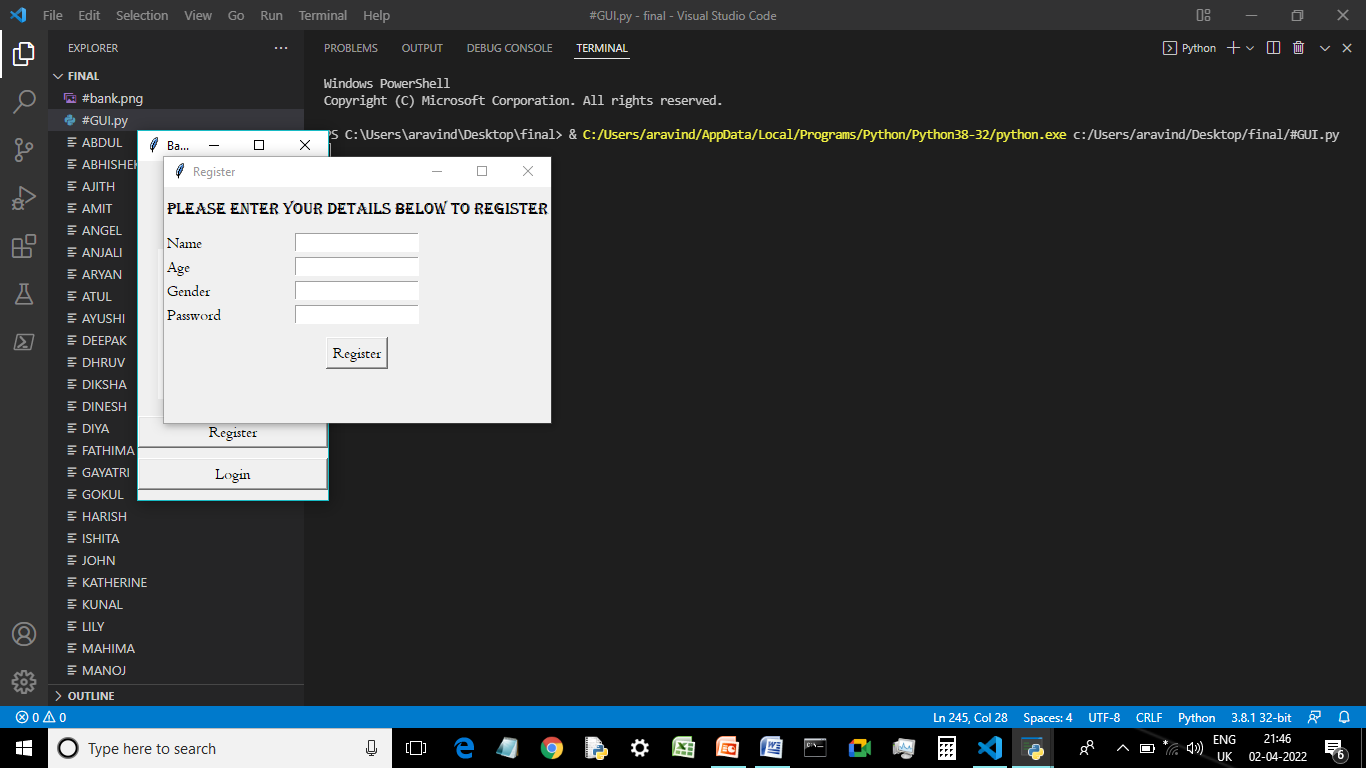
Button(master, text="Register", font=('Centaur',12),width=20,command=register).grid(row=3,sticky=N)

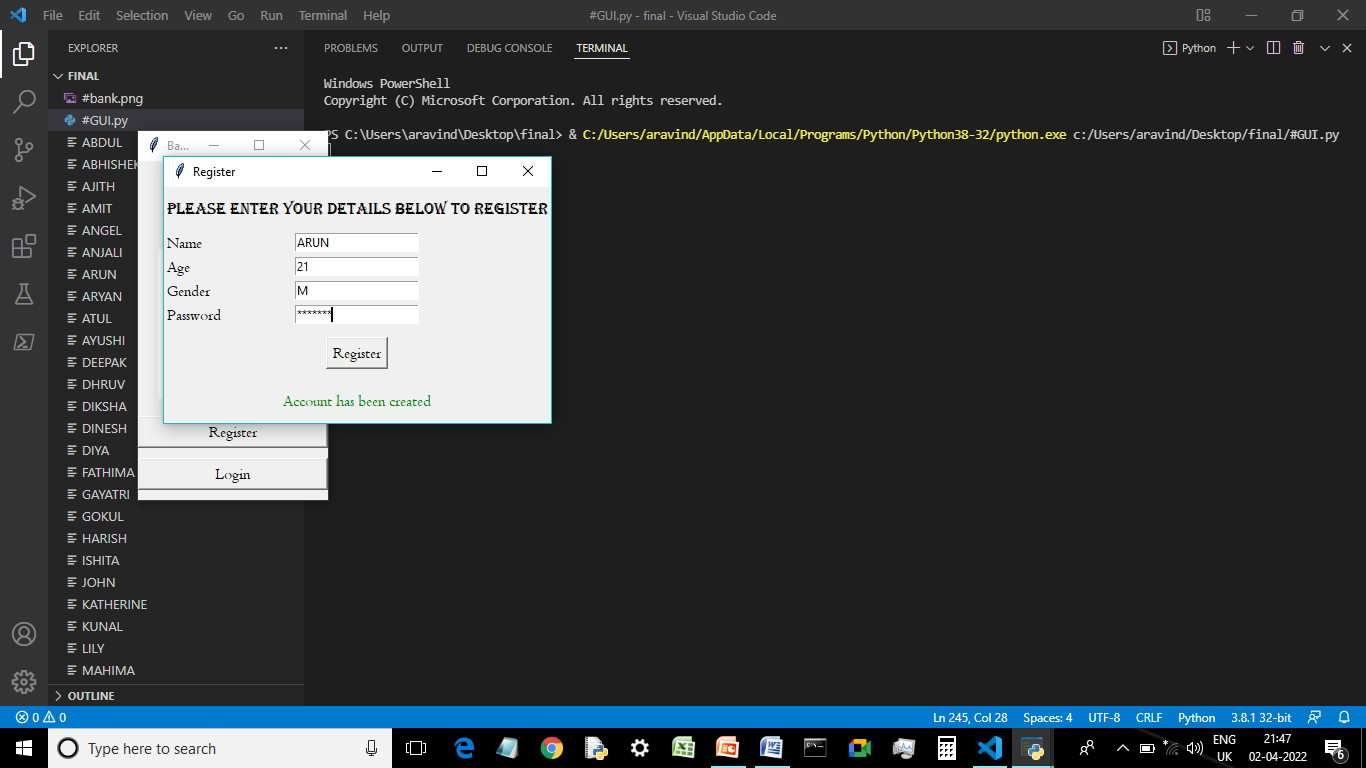
Button(master, text="Login", font=('Centaur',12),width=20,command=login).grid(row=4,sticky=N,pady=10)

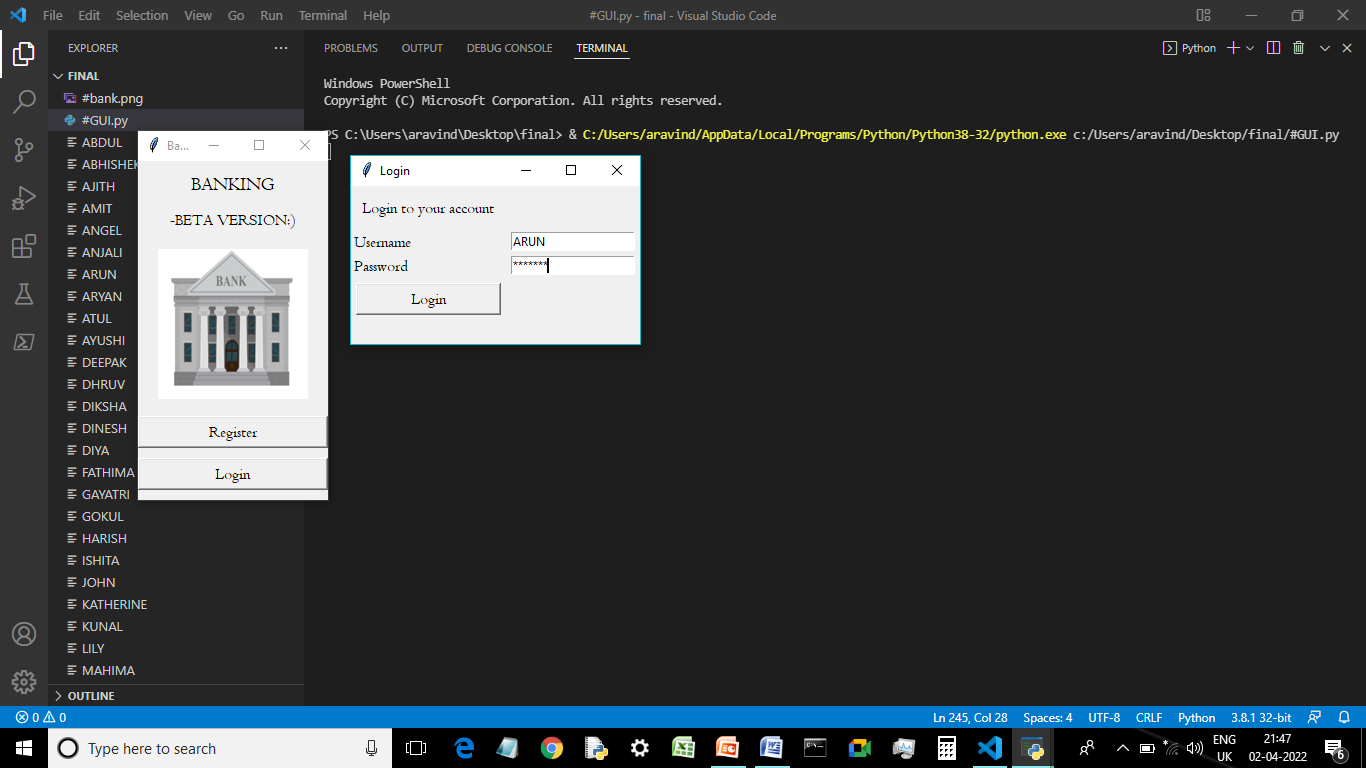
master.mainloop()

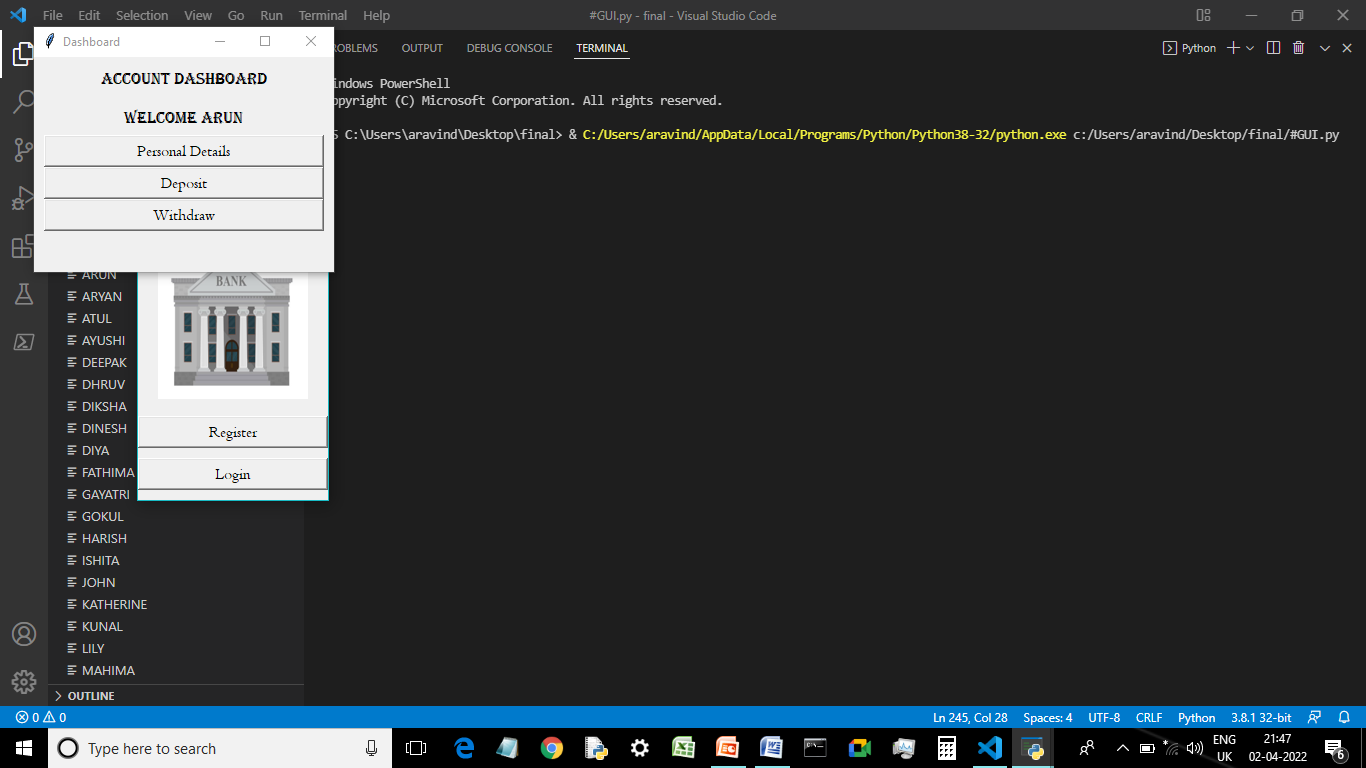
**SAMPLE OUTPUT:**

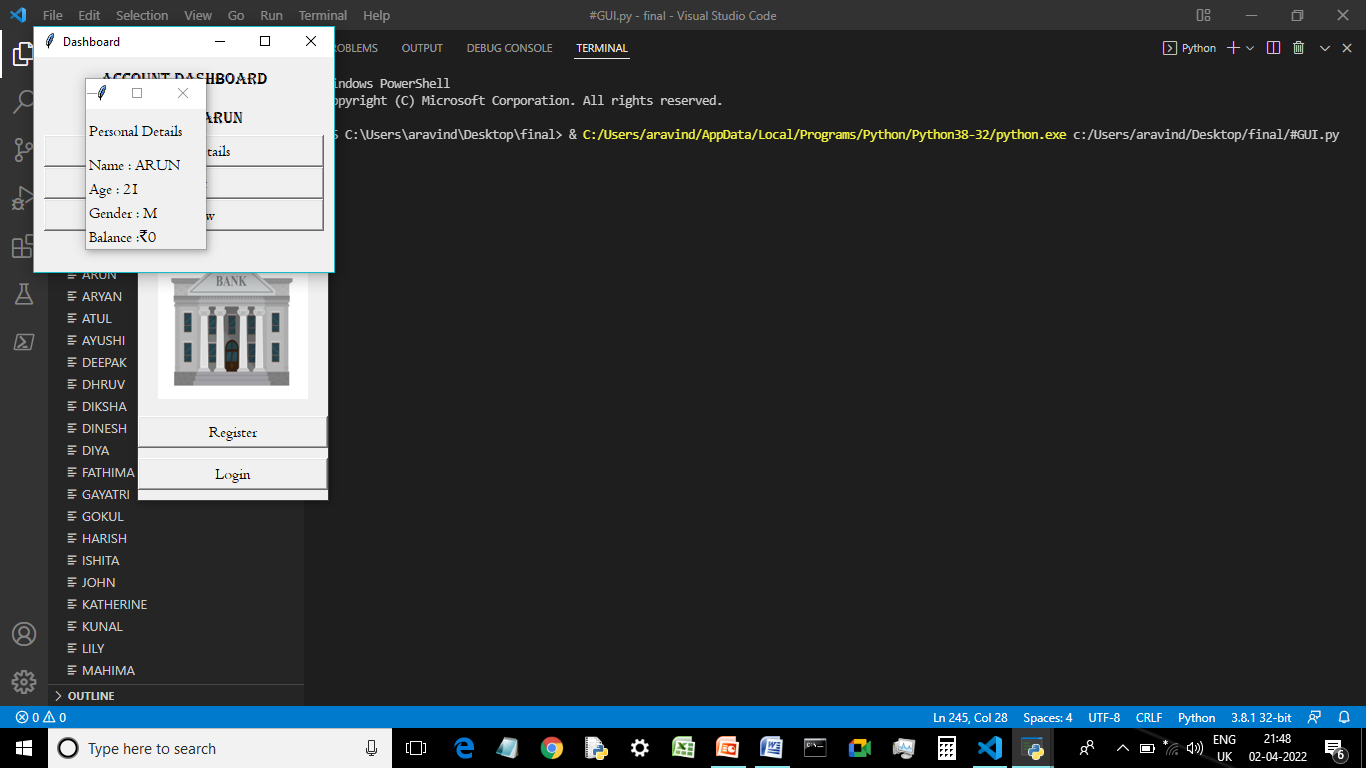
****

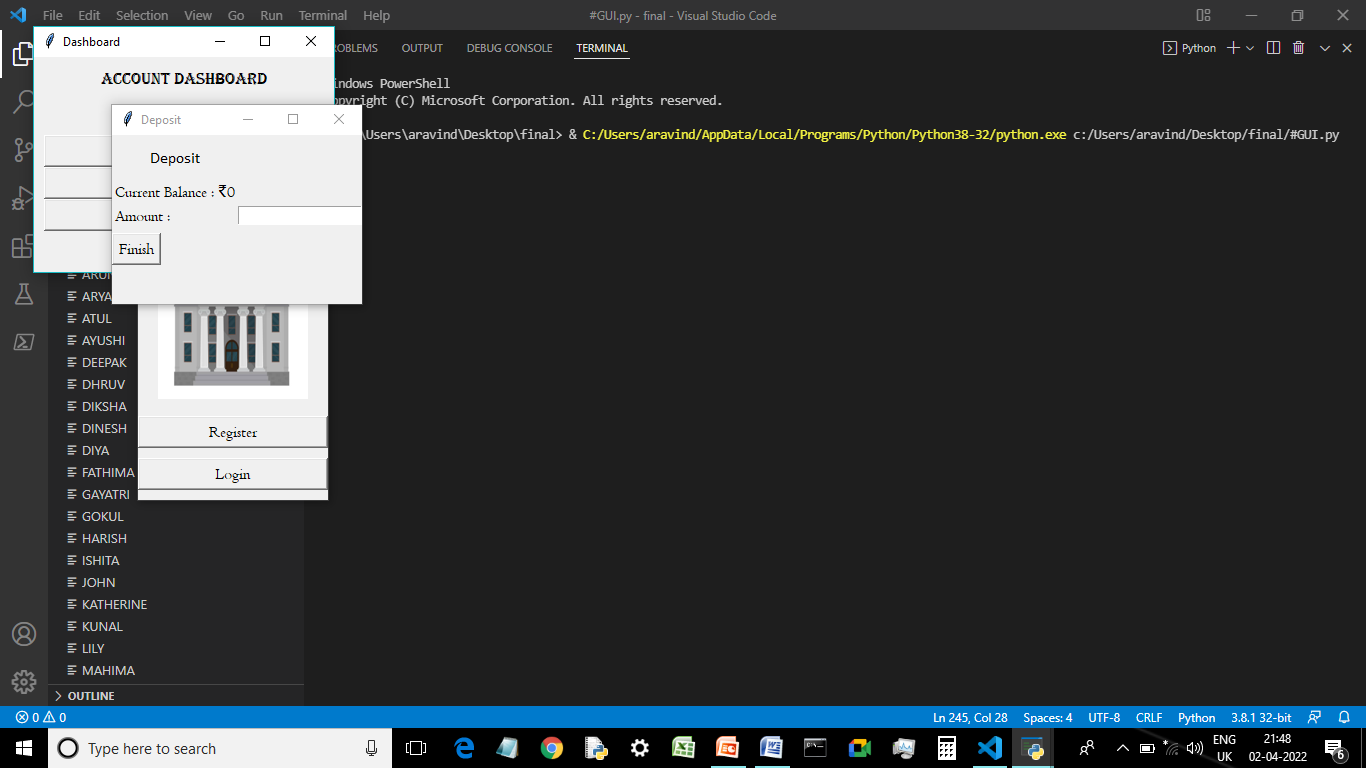
****

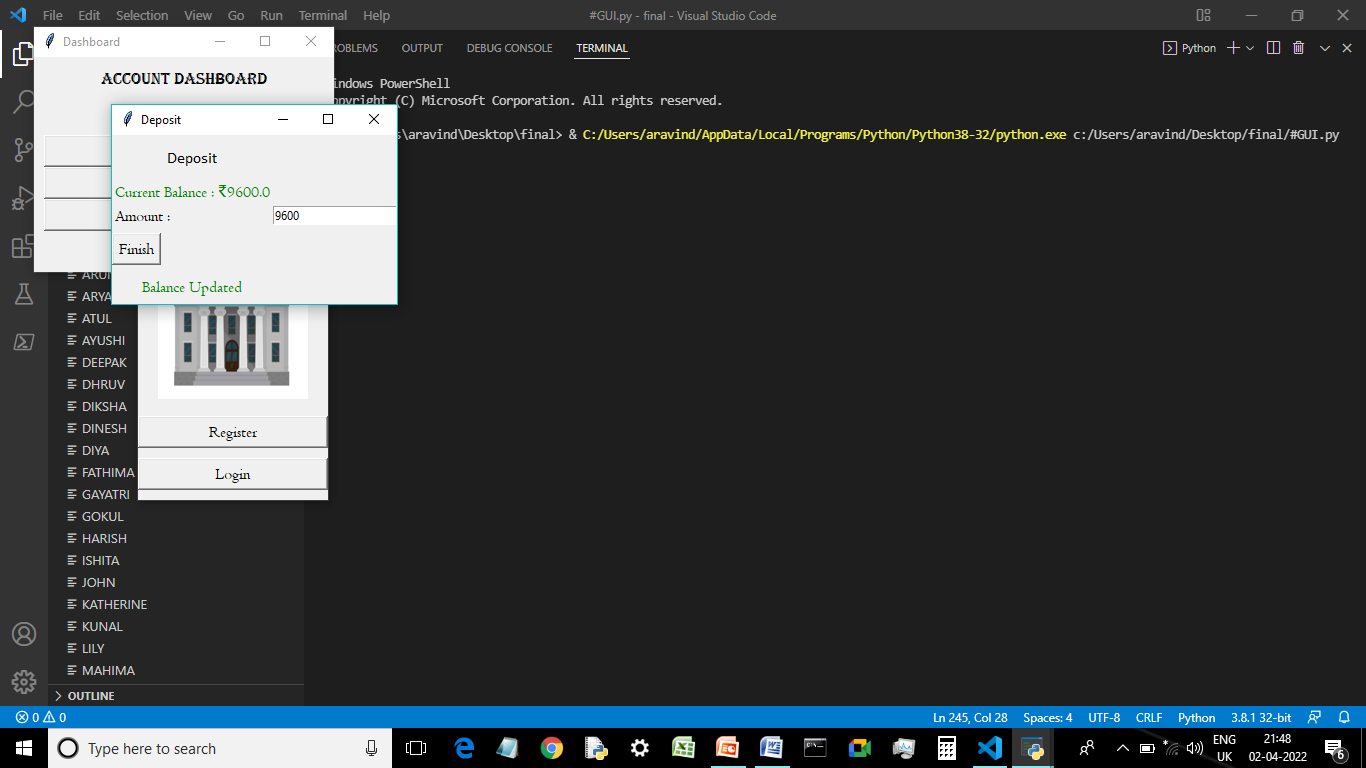
****

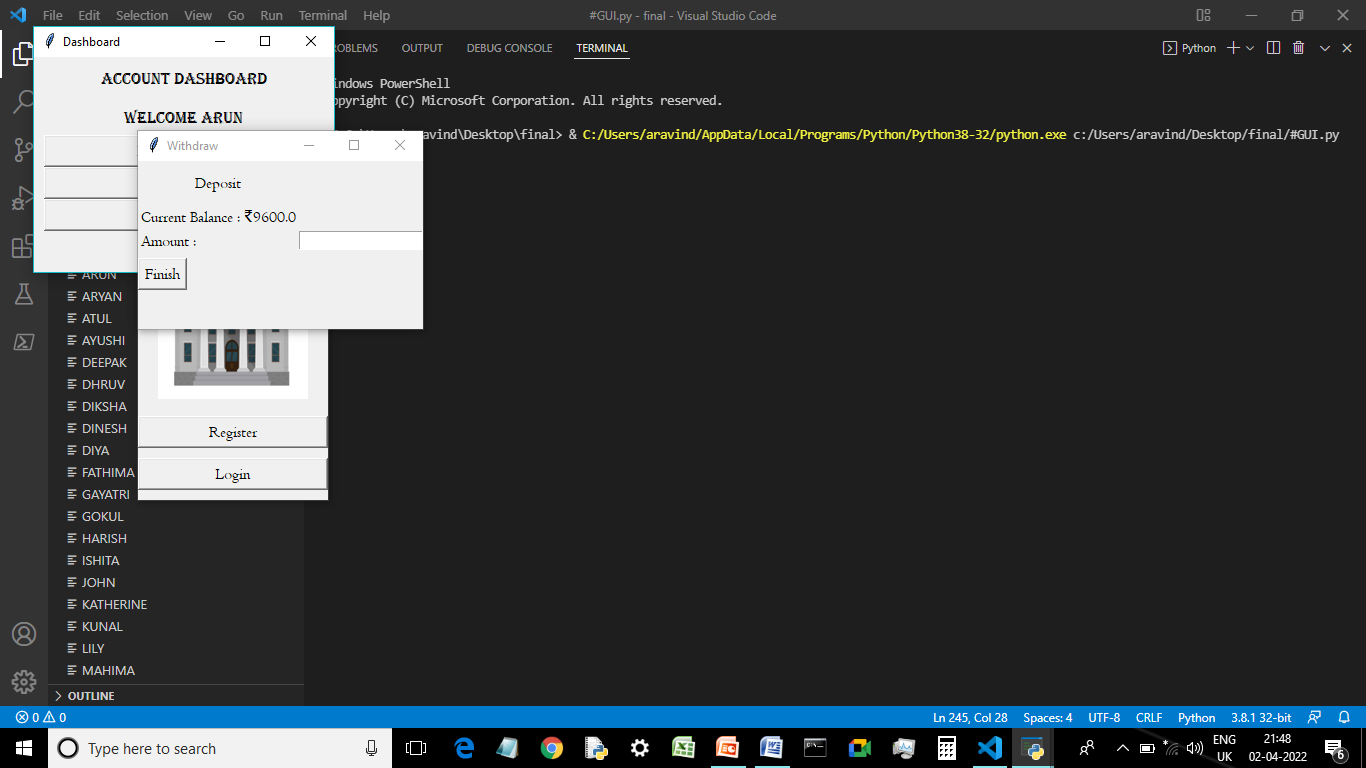
****

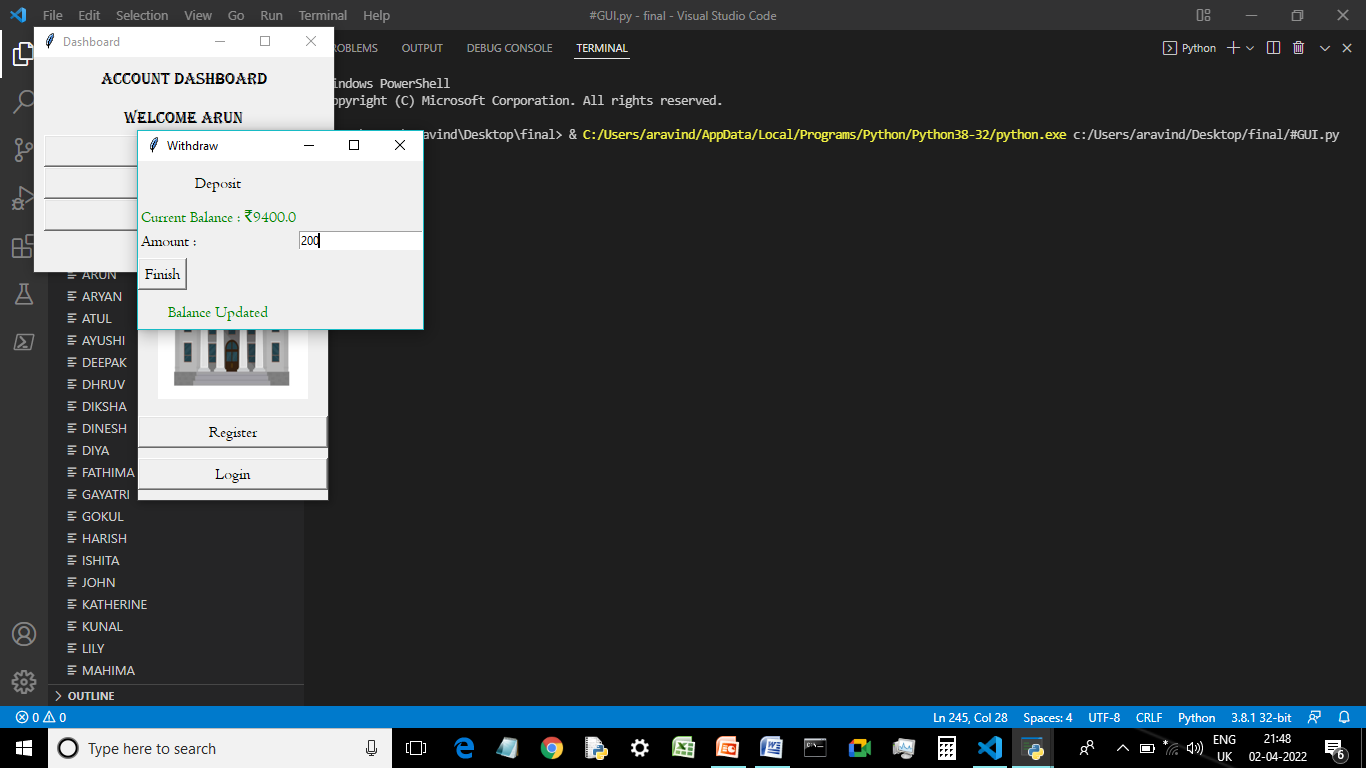
****

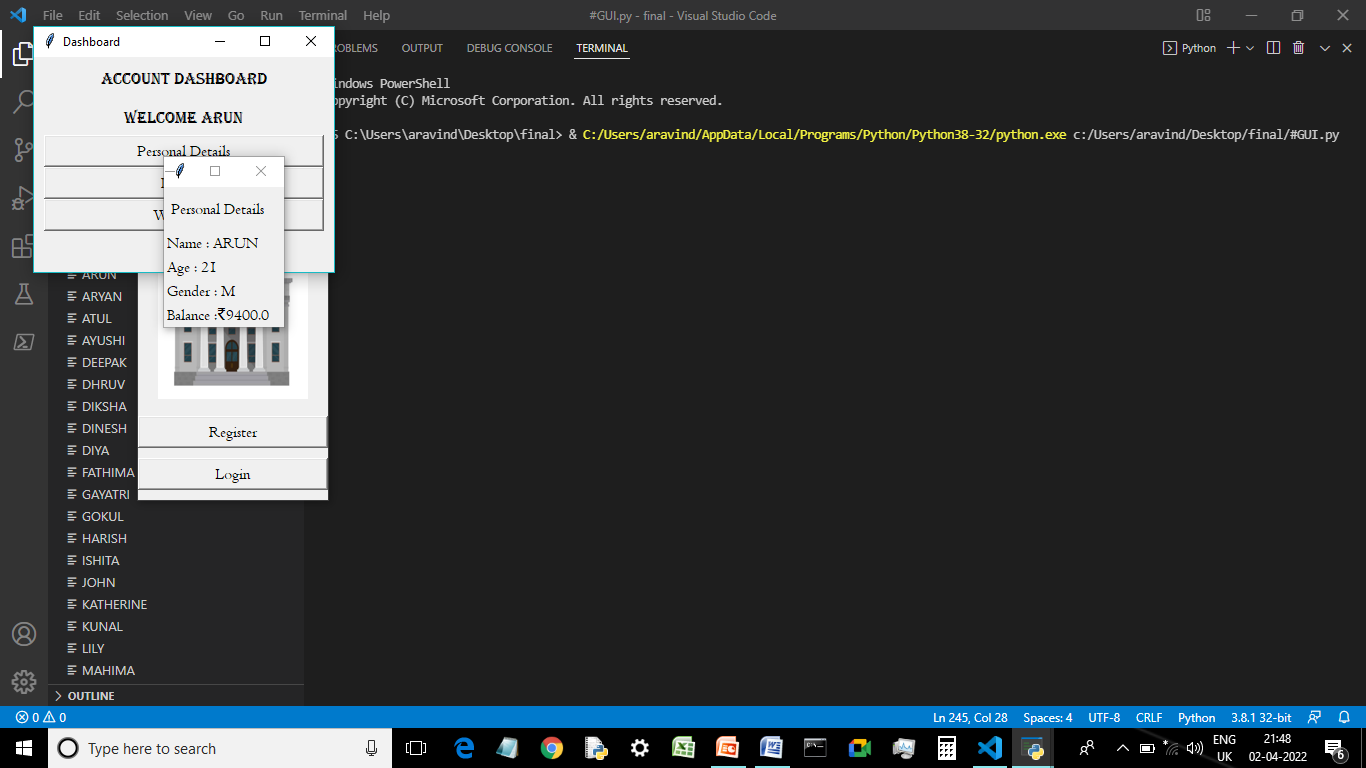
****

****

****

****

****

****