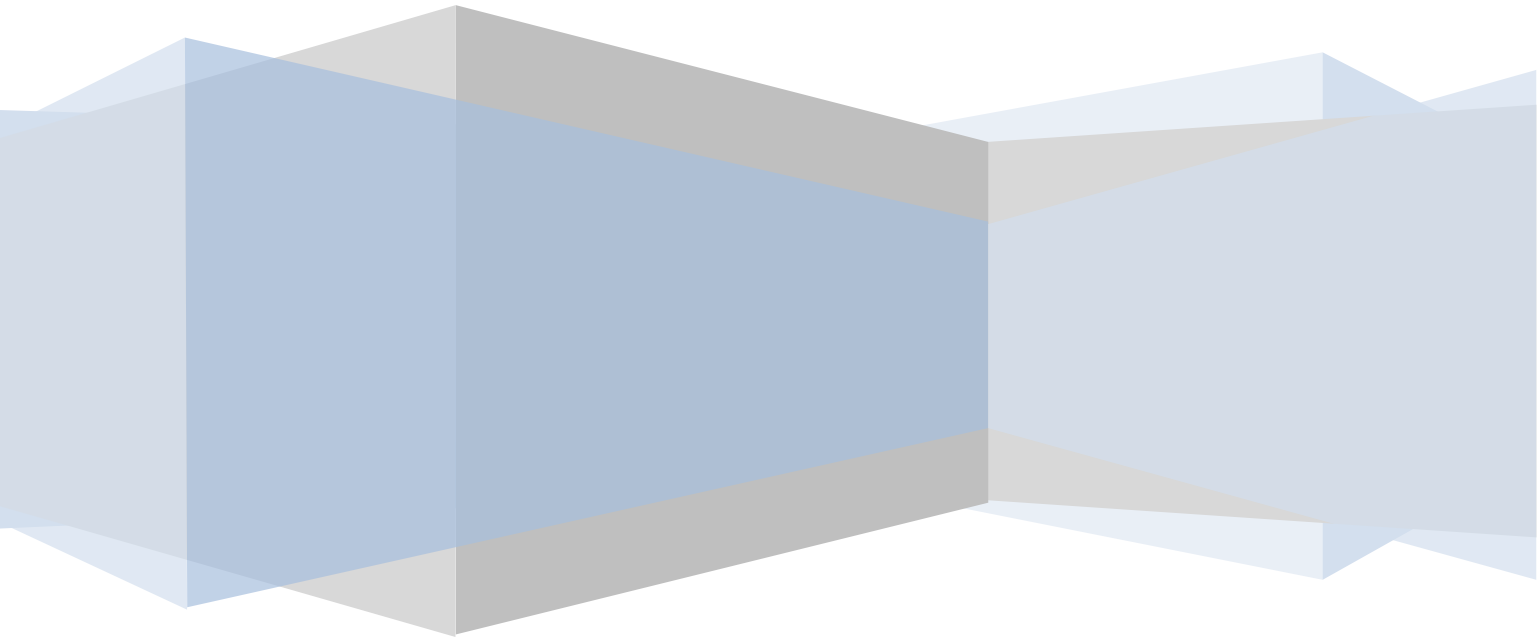

'Python' PROGRAMMING TRAINING MODULE

The Python Training module will make the reader accustomed to python language. This material will help the reader in understanding the basics of the python language, Python libraries and the use of python for the analytics.

DELIVERY METHOD	• 25% Self-paced Learning • 75% Instructor led Training
VERSION	• 2020
LEARNING OBJECTIVES	• Explain what Python is • Advantages and disadvantages of Python • Getting started with Python and its different versions • Explain variables, strings and functions • Use of mathematical operators and functions • Explain different statements like if, for etc. • Explain the python libraries • Explain Details of the Pandas library o Series and Data Frames o Grouping and aggregating o Merging and joining • Define error handling in Python
PREREQUISITES SKILLS	• Computer Science fundamentals • Basic knowledge of applied math, algorithms, and data modelling • Basic knowledge of statistics
DURATION	• 24 Hours
SKILL LEVEL	• Basic-Intermediate
HARDWARE REQUIREMENTS	• Processor: 2 GHZ or Higher • GB RAM: 8 GB • GB DISK: 80 GB

COURSE AGENDA: MODULE-1

Lecture I - Fundamentals of Python Programming Language

Duration: 3 Hrs.

Overview	Python programs in this section performs arithmetic operations like Addition, Multiplication and Division on the given set of input numbers.
Learning Objectives	• What is Python? • What's a variable? • Data Variables and Operators.
Lab Exercise	• Python Program to exchange the values of two numbers without using a temporary variable. • Python Program to take the temperature in Celsius and convert it to Fahrenheit. • Python Program to read two numbers and print their quotient and remainder. • Python Program to find the area of a triangle given all three sides. • Python Program to read height in centimeters and then convert the height to feet and inches • Python Program to compute simple interest given all the required values.

Lecture II – If-Else & Basic Loop

Duration: 3 Hrs.

Overview	Python programs in this section allows you to make a decision, based upon the result of a condition. & build logics to solve various problems and basic concept of loop.
Learning Objectives	• Conditional statement • If-else • Basic loop • Ranges
Lab Exercise	• Python Program to check whether a given year is a leap year or not. • Python Program to take in the marks of 5 subjects and display the grade. • Python Program to check if a number is an Armstrong number. • Python Program to find the sum of digits in a number. • Python Program to print odd numbers within a given range. • Python Program to check whether a given number is a palindrome. • Python Program to print all numbers in a range divisible by a given number. • Python Program to read a number n and print an inverted star pattern of the desired size. • Python Program to find the sum of first N Natural Numbers.

Lecture III – Loop and Function Definition

Duration: 3 Hrs.

Overview	Python programs in this section clears the concept of loop and user defined function in python.
Learning Objectives	• Understand why loops are useful. • Recall the need of indentation • Implement mix of loops and control flow to extract information from a data structure.
Lab Exercise	• Python Program to read a number n and print and compute the series "1+2+…+n=". • Python Program to find the sum of series: 1 + 1/2 + 1/3 + ….. + 1/N. • Python Program to find the sum of series: 1 + x^2/2 + x^3/3 + … x^n/n. • Python Program to find the sum of series: 1 + 1/2 + 1/3 + ….. + 1/N. • Python program to find whether a number is a power of two.

Lecture IV – List and Tuple

Duration: 3 Hrs.

Overview	Python programs in this section provides the concept of collection of elements.
Learning Objectives	● What is List? ● What is Tuple? ● Difference between List and Tuple.
Lab Exercise	● Python Program to find the second largest number in a list. ● Python Program to put the even and odd elements in a list into two different lists. ● Python Program to merge two lists and sort it. ● Python Program to find the second largest number in a list using bubble sort. ● Python Program to find the intersection of two lists. ● Python Program to sort a list of tuples in increasing order by the last element in each tuple. ● Python Program to remove the duplicate items from a list.

Lecture V – Strings

Duration: 3 Hrs.

Overview	Collection of character is known as string. Python programs in this section performs operations on string
Learning Objectives	• What is String? • Different operations on string
Lab Exercise	• Python Program to replace all occurrences of 'a' with '$' in a string. • Python Program to detect if two strings are anagrams. • Python Program to count the number of vowels in a string. • Python Program to calculate the length of a string without using library functions. • Python Program to check if a string is a palindrome or not. • Python Program to check if a substring is present in a given string.

Lecture VI – Dictionary and Set

Duration: 3 Hrs.

Overview	Python programs in this section performs different operation on dictionary and Set.
Learning Objectives	• Identify Dictionary and Sets • Recognize difference between them
Lab Exercise	• Python Program to add a key-value pair to a dictionary. • Python Program to concatenate two dictionaries into one dictionary. • Python Program to check if a given key exists in a dictionary or not. • Python Program to remove the given key from a dictionary. • Python Program to count the frequency of words appearing in a string using a dictionary. • Python Program to create a dictionary with key as first character and value as words starting with that character. • Python Program to count the number of vowels present in a string using sets • Python Program to check common letters in the two input strings. • Python Program to display which letters are present in both the strings.

Lecture VII – Recursion and File Handling

Duration: 3 Hrs.

Overview	Python programs in this section performs recursive method to perform operation and file handling tasks.
Learning Objectives	• Implementing recursive function • Understand different properties for different file formats.
Lab Exercise	• Python Program to find the fibonacci series using recursion. • Python Program to find the factorial of a number using recursion. • Python Program to find the GCD of two numbers using recursion. • Python Program to reverse a string using recursion. • Python Program to read the contents of a file. • Python Program to count the number of words in a text file. • Python Program to copy the contents of one file into another. • Python Program to append the contents of one file to another file. • Python Program to read a file and capitalize the first letter of every word in the file.

Lecture VIII – Data Structure using Python

Duration: 3 Hrs.

Overview	Python programs in this section performs different data structure program.
Learning Objectives	• Implement a data structure. • Distinguish different approaches.
Lab Exercise	• Python program to search for an element in a linked list using recursion. • Python program to implement a stack. • Python program to find the Nth node in the in-order traversal of a binary tree.

PROGRAM SOLUTION

Problem Statement	Python Program to exchange the values of two numbers without using a temporary variable.
Concept	1. Take the values of both the elements from the user. 2. Store the values in separate variables. 3. Add both the variables and store it in the first variable. 4. Subtract the second variable from the first and store it in the second variable. 5. Then, subtract the first variable from the second variable and store it in the first variable. 6. Print the swapped values. 7. Exit.
Code	a=int(input("Enter value of first variable: ")) b=int(input("Enter value of second variable: ")) a=a+b b=a-b a=a-b print("a is:",a," b is:",b)

Problem Statement	Python Program to take the temperature in Celsius and convert it to Fahrenheit.
Concept	1. Take the value of temperature in Celsius and store it in a variable. 2. Convert it to Fahrenheit. 3. Using the formula of: f=(c*1.8)+32, convert Celsius to Fahrenheit. 4. Print the final result. 5. Exit.
Code	celsius=int(input("Enter the temperature in celcius:")) f=(celsius*1.8)+32 print("Temperature in farenheit is:",f)

Problem Statement	Python Program to read two numbers and print their quotient and remainder.
Concept	1. Take in the first and second number and store it in separate variables. 2. Then obtain the quotient using division and the remainder using modulus operator. 3. Exit.
Code	a=int(input("Enter the first number: ")) b=int(input("Enter the second number: ")) quotient=a//b remainder=a%b print("Quotient is:",quotient) print("Remainder is:",remainder)

Problem Statement	Python Program to find the area of a triangle given all three sides.
Concept	1. Take in all the three sides of the triangle and store it in three separate variables. 2. Then using the Heron's formula, compute the area of the triangle. 3. Print the area of the triangle. 4. Exit.
Code	import math a=int(input("Enter first side: ")) b=int(input("Enter second side: ")) c=int(input("Enter third side: ")) s=(a+b+c)/2 area=math.sqrt(s*(s-a)*(s-b)*(s-c)) print("Area of the triangle is: ",round(area,2))

Problem Statement	Python Program to read height in centimeters and then convert the height to feet and inches
Concept	1. Take the height in centimeters and store it in a variable. 2. Convert the height in centimeters into inches and feet. 3. Print the length in inches and feet. 4. Exit.
Code	cm=int(input("Enter the height in centimeters:")) inches=0.394*cm feet=0.0328*cm print("The length in inches",round(inches,2)) print("The length in feet",round(feet,2))

Problem Statement	Python Program to compute simple interest given all the required values.
Concept	1. Take in the values for principle amount, rate and time. 2. Using the formula, compute the simple interest. 3. Print the value for the computed interest. 4. Exit.
Code	principle=float(input("Enter the principle amount:")) time=int(input("Enter the time(years):")) rate=float(input("Enter the rate:")) simple_interest=(principle*time*rate)/100 print("The simple interest is:",simple_interest)

Problem Statement	Python Program to check whether a given year is a leap year or not.
Concept	1. Take the value of the year as input 2. Using an if-statement, check whether the year is a leap year or not 3. Print the final result 4. Exit

Code	year=int(input("Enter year to be checked:"))
	if(year%4==0 and year%100!=0 or year%400==0):
	  print("The year is a leap year!)
	else:
	  print("The year isn't a leap year!)

Problem Statement	Python Program to take in the marks of 5 subjects and display the grade.
Concept	1. Take in the marks of 5 subjects from the user and store it in different variables. 2. Find the average of the marks. 3. Use an else condition to decide the grade based on the average of the marks. 4. Exit.

Code	sub1=int(input("Enter marks of the first subject: "))
	sub2=int(input("Enter marks of the second subject: "))
	sub3=int(input("Enter marks of the third subject: "))
	sub4=int(input("Enter marks of the fourth subject: "))
	sub5=int(input("Enter marks of the fifth subject: "))
	avg=(sub1+sub2+sub3+sub4+sub4)/5
	if(avg>=90):
	  print("Grade: A")
	elif(avg>=80&avg<90):
	  print("Grade: B")
	elif(avg>=70&avg<80):
	  print("Grade: C")
	elif(avg>=60&avg<70):
	  print("Grade: D")
	else:
	  print("Grade: F")

Problem Statement	Python Program to check if a number is an Armstrong number.
Concept	1. Take in an integer and store it in a variable. 2. Convert each digit of the number to a string variable and store it in a list. 3. Then cube each of the digits and store it in another list. 4. Find the sum of the cube of digits in the list and check if it is equal to the original number. 5. Print the final result. 6. Exit.

Code	n=int(input("Enter any number: "))
	a=list(map(int,str(n)))
	b=list(map(lambda x:x**3,a))
	if(sum(b)==n):
	  print("The number is an armstrong number. ")
	else:
	  print("The number isn't an arsmtrong number. ")

Problem Statement	Python Program to find the sum of digits in a number.
Concept	1. Take the value of the integer and store in a variable. 2. Using a while loop, get each digit of the number and add the digits to a variable. 3. Print the sum of the digits of the number. 4. Exit.

Code	n=int(input("Enter a number:"))
	tot=0
	while(n>0):
	  dig=n%10
	  tot=tot+dig
	  n=n//10
	print("The total sum of digits is:",tot)

Problem Statement	Python Program to print odd numbers within a given range.
Concept	1. Take in the upper range limit and the lower range limit and store it in separate variables. 2. Use a for-loop ranging from the lower range to the upper range limit. 3. Use an if statement if check whether the number is odd or not and print the number. 4. Exit.

Code	lower=int(input("Enter the lower limit for the range:"))
	upper=int(input("Enter the upper limit for the range:"))
	for i in range(lower,upper+1):
	  if(i%2!=0):
	  print(i)

Problem Statement	Python Program to check whether a given number is a palindrome.
Concept	1. Take the value of the integer and store in a variable. 2. Transfer the value of the integer into another temporary variable. 3. Using a while loop, get each digit of the number and store the reversed number in another variable. 4. Check if the reverse of the number is equal to the one in the temporary variable. 5. Print the final result. 6. Exit.

Code	n=int(input("Enter number:"))
	temp=n
	rev=0
	while(n>0):
	  dig=n%10
	  rev=rev*10+dig
	  n=n//10
	if(temp==rev):
	  print("The number is a palindrome!")
	else:
	  print("The number isn't a palindrome!")

Problem Statement	Python Program to print all numbers in a range divisible by a given number.
Concept	1. Take in the upper range and lower range limit from the user. 2. Take in the number to be divided by from the user. 3. Using a for loop, print all the factors which is divisible by the number. 4. Exit.

Code	lower=int(input("Enter lower range limit:"))
	upper=int(input("Enter upper range limit:"))
	n=int(input("Enter the number to be divided by:"))
	for i in range(lower,upper+1):
	  if(i%n==0):
	  print(i)

Problem Statement	Python Program to read a number n and print an inverted star pattern of the desired size.
Concept	1. Take a value from the user and store it in a variable n. 2. Use a for loop where the value of i ranges between the values of n-1 and 0 with a decrement of 1 with each iteration. 3. Multiply empty spaces with n-i and '*' with i and print both of them. 4. Exit.
Code	n=int(input("Enter number of rows: ")) for i in range (n,0,-1): print((n-i) * ' ' + i * '*')

Problem Statement	Python Program to find the sum of first N Natural Numbers.
Concept	1. Take in the number of natural numbers to find the sum of and store it in a separate variable. 2. Initialize the sum variable to 0. 3. Use a while loop to find the sum of natural numbers and decrement the number for each iteration. 4. The numbers are added to the sum variable and this continues until the the value of the number is greater than 0. 5. Then the sum of first N natural numbers is printed. 6. Exit.

Code	n=int(input("Enter a number: "))
	sum1 = 0
	while(n > 0):
	  sum1=sum1+n
	  n=n-1
	print("The sum of first n natural numbers is",sum1)

Problem Statement	Python Program to read a number n and print and compute the series “1+2+…+n=”.
Concept	1. Take a value from the user and store it in a variable n. 2. Use a for loop where the value of i ranges between the values of 1 and n. 3. Print the value of i and ‘+’ operator while appending the value of i to a list. 4. Then find the sum of elements in the list. 5. Print ‘=’ followed by the total sum. 6. Exit.

Code	n=int(input("Enter a number: "))
	a=[]
	for i in range(1,n+1):
	  print(i,sep=" ",end=" ")
	  if(i<n):
	  print("+",sep=" ",end=" ")
	  a.append(i)
	print("=",sum(a))
	
	print()

Problem Statement	Python Program to find the sum of series: 1 + 1/2 + 1/3 + ….. + 1/N.
Concept	1. Take in the number of terms to find the sum of the series for. 2. Initialize the sum variable to 0. 3. Use a for loop ranging from 1 to the number and find the sum of the series. 4. Print the sum of the series after rounding it off to two decimal places. 5. Exit.
Code	n=int(input("Enter the number of terms: ")) sum1=0 for i in range(1,n+1): sum1=sum1+(1/i) print("The sum of series is",round(sum1,2))

Problem Statement	Python Program to find the sum of series: 1 + x^2/2 + x^3/3 + … x^n/n.
Concept	1. Take in the number of terms to find the sum of the series for. 2. Initialize the sum variable to 0. 3. Use a for loop ranging from 1 to the number and find the sum of the series. 4. Print the sum of the series after rounding it off to two decimal places. 5. Exit.
Code	n=int(input("Enter the number of terms:")) x=int(input("Enter the value of x:")) sum1=1 for i in range(2,n+1): sum1=sum1+((x**i)/i) print("The sum of series is",round(sum1,2))

Problem Statement	Python Program to find the sum of series: 1 + 1/2 + 1/3 + ….. + 1/N.
Concept	1. Take in the number of terms to find the sum of the series for. 2. Initialize the sum variable to 1. 3. Use a for loop ranging from 1 to the number and find the sum of the series. 4. Print the sum of the series after rounding it off to two decimal places. 5. Exit.
Code	import math n=int(input("Enter the number of terms: ")) sum1=1 for i in range(1,n+1): sum1=sum1+(1/math.factorial(i)) print("The sum of series is",round(sum1,2))

Problem Statement	Python program to find whether a number is a power of two.
Concept	1. The function is_power_of_two is defined. 2. It takes a number n as argument and returns True if the number is a power of two. 3. If n is not positive, False is returned. 4. If n is positive, then n & (n – 1) is calculated. 5. The above expression equals n with its rightmost set bit cleared. That is, the rightmost 1 in the binary representation of n is made 0. 6. All powers of two have only one set bit in their binary representation and all numbers with only one set bit is a power of two. 7. Thus n & (n – 1) will equal zero iff n is a power of two. 8. This is used to determine if n is a power of two if n is positive.

Code	def is_power_of_two(n):
	"""Return True if n is a power of two."""
	if n <= 0:
	return False
	else:
	return n & (n - 1) == 0
	
	n = int(input('Enter a number: '))
	
	if is_power_of_two(n):
	print('{} is a power of two.'.format(n))
	else:
	print('{} is not a power of two.'.format(n))

Problem Statement	Python Program to find the second largest number in a list.
Concept	1. Take in the number of elements and store it in a variable. 2. Take in the elements of the list one by one. 3. Sort the list in ascending order. 4. Print the second last element of the list. 5. Exit.
Code	a=[] n=int(input("Enter number of elements:")) for i in range(1,n+1): b=int(input("Enter element:")) a.append(b) a.sort() print("Second largest element is:",a[n-2])

Problem Statement	Python Program to put the even and odd elements in a list into two different lists.
Concept	1. Take in the number of elements and store it in a variable. 2. Take in the elements of the list one by one. 3. Use a for loop to traverse through the elements of the list and an if statement to check if the element is even or odd. 4. If the element is even, append it to a separate list and if it is odd, append it to a different one. 5. Display the elements in both the lists. 6. Exit.

Code	a=[]
	n=int(input("Enter number of elements:"))
	for i in range(1,n+1):
	  b=int(input("Enter element:"))
	  a.append(b)
	even=[]
	odd=[]
	for j in a:
	  if(j%2==0):
	  even.append(j)
	  else:
	  odd.append(j)
	print("The even list",even)
	print("The odd list",odd)

Problem Statement	Python Program to merge two lists and sort it.
Concept	1. Take in the number of elements for the first list and store it in a variable. 2. Take in the elements of the list one by one. 3. Similarly, take in the elements for the second list also. 4. Merge both the lists using the '+' operator and then sort the list. 5. Display the elements in the sorted list. 6. Exit.

Code	a=[]
	c=[]
	n1=int(input("Enter number of elements:"))
	for i in range(1,n1+1):
	  b=int(input("Enter element:"))
	  a.append(b)
	n2=int(input("Enter number of elements:"))
	for i in range(1,n2+1):
	  d=int(input("Enter element:"))
	  c.append(d)
	new=a+c
	new.sort()
	print("Sorted list is:",new)

Problem Statement	Python Program to find the second largest number in a list using bubble sort.
Concept	1. Take in the number of elements for the list and store it in a variable. 2. Take in the elements of the list one by one. 3. Then sort the list using bubble sort. 4. Print the second last element of the sorted list which is the second largest number. 5. Exit.

Code	a=[]
	n=int(input("Enter number of elements:"))
	for i in range(1,n+1):
	  b=int(input("Enter element:"))
	  a.append(b)
	for i in range(0,len(a)):
	  for j in range(0,len(a)-i-1):
	  if(a[j]>a[j+1]):
	  temp=a[j]
	  a[j]=a[j+1]
	  a[j+1]=temp
	print('Second largest number is:',a[n-2])

Problem Statement	Python Program to find the intersection of two lists.
Concept	1. Define a function which accepts two lists and returns the intersection of them. 2. Declare two empty lists and initialize them to an empty list. 3. Consider a for loop to accept values for the two lists. 4. Take the number of elements in the list and store it in a variable. 5. Accept the values into the list using another for loop and insert into the list. 6. Repeat 4 and 5 for the second list also. 7. Find the intersection of the two lists. 8. Print the intersection. 9. Exit.

Code	def intersection(a, b):
	  return list(set(a) & set(b))
	
	def main():
	  alist=[]
	  blist=[]
	  n1=int(input("Enter number of elements for list1:"))
	  n2=int(input("Enter number of elements for list2:"))
	  print("For list1:")
	  for x in range(0,n1):
	  element=int(input("Enter element" + str(x+1) + ":"))
	  alist.append(element)
	  print("For list2:")
	  for x in range(0,n2):
	  element=int(input("Enter element" + str(x+1) + ":"))
	  blist.append(element)
	  print("The intersection is :")
	  print(intersection(alist, blist))
	main()

Problem Statement	Python Program to sort a list of tuples in increasing order by the last element in each tuple.
Concept	1. Take a list of tuples from the user. 2. Define a function which returns the last element of each tuple in the list of tuples. 3. Define another function with the previous function as the key and sort the list. 4. Print the sorted list. 5. Exit.

Code	def last(n):
	  return n[-1]
	
	def sort(tuples):
	  return sorted(tuples, key=last)
	
	a=input("Enter a list of tuples:")
	print("Sorted:")
	print(sort(a))

Problem Statement	Python Program to remove the duplicate items from a list.
Concept	1. Take the number of elements in the list and store it in a variable. 2. Accept the values into the list using a for loop and insert them into the list. 3. Use a for loop to traverse through the elements of the list. 4. Use an if statement to check if the element is already there in the list and if it is not there, append it to another list. 5. Print the non-duplicate items of the list. 6. Exit.

Code	a=[]
	n= int(input("Enter the number of elements in list:"))
	for x in range(0,n):
	  element=int(input("Enter element" + str(x+1) + ":"))
	  a.append(element)
	b = set()
	unique = []
	for x in a:
	  if x not in b:
	  unique.append(x)
	  b.add(x)
	print("Non-duplicate items:")
	print(unique)

Problem Statement	Python Program to replace all occurrences of ‘a’ with ‘$’ in a string.
Concept	1. Take a string and store it in a variable. 2. Using the replace function, replace all occurrences of ‘a’ and ‘A’ with ‘$’ and store it back in the variable. 3. Print the modified string. 4. Exit.
Code	string=raw_input("Enter string:") string=string.replace('a','$') string=string.replace('A','$') print("Modified string:") print(string)

Problem Statement	Python Program to detect if two strings are anagrams.
Concept	1. Take two strings from the user and store them in separate variables. 2. Then use sorted() to sort both the strings into lists. 3. Compare the sorted lists and check if they are equal. 4. Print the final result. 5. Exit.

Code	s1=raw_input("Enter first string:")
	s2=raw_input("Enter second string:")
	if(sorted(s1)==sorted(s2)):
	  print("The strings are anagrams.")
	else:
	  print("The strings aren't anagrams.")

Problem Statement	Python Program to count the number of vowels in a string.
Concept	1. Take a string from the user and store it in a variable. 2. Initialize a count variable to 0. 3. Use a for loop to traverse through the characters in the string. 4. Use an if statement to check if the character is a vowel or not and increment the count variable if it is a vowel. 5. Print the total number of vowels in the string. 6. Exit

Code	string=raw_input("Enter string:")
	vowels=0
	for i in string:
	  if(i=='a' or i=='e' or i=='i' or i=='o' or i=='u' or i=='A' or i=='E' or i=='I' or i=='O' or i=='U'):
	  vowels=vowels+1
	print("Number of vowels are:")
	print(vowels)

Problem Statement	Python Program to calculate the length of a string without using library functions.
Concept	1. Take a string from the user and store it in a variable. 2. Initialize a count variable to 0. 3. Use a for loop to traverse through the characters in the string and increment the count variable each time. 4. Print the total count of the variable. 5. Exit.
Code	string=raw_input("Enter string:") count=0 for i in string: count=count+1 print("Length of the string is:") print(count)

Problem Statement	Python Program to check if a string is a palindrome or not.
Concept	1. Take a string from the user and store it in a variable. 2. Reverse the string using string slicing and compare it back to the original string. 3. Print the final result 4. Exit.

Code	string=raw_input("Enter string:")
	if(string==string[::-1]):
	  print("The string is a palindrome")
	else:
	  print("The string isn't a palindrome")

Problem Statement	Python Program to check if a substring is present in a given string.
Concept	1. Take a string and a substring from the user and store it in separate variables. 2. Check if the substring is present in the string using find() in-built function. 3. Print the final result. 4. Exit.

Code	string=raw_input("Enter string:")
	sub_str=raw_input("Enter word:")
	if(string.find(sub_str)==-1):
	  print("Substring not found in string!")
	else:
	  print("Substring in string!")

Problem Statement	Python Program to add a key-value pair to a dictionary.
Concept	1. Take a key-value pair from the user and store it in separate variables. 2. Declare a dictionary and initialize it to an empty dictionary. 3. Use the update() function to add the key-value pair to the dictionary. 4. Print the final dictionary. 5. Exit.
Code	key=int(input("Enter the key (int) to be added:")) value=int(input("Enter the value for the key to be added:")) d={} d.update({key:value}) print("Updated dictionary is:") print(d)

Problem Statement	Python Program to concatenate two dictionaries into one dictionary.
Concept	1. Declare and initialize two dictionaries with some key-value pairs 2. Use the update() function to add the key-value pair from the second dictionary to the first dictionary. 3. Print the final dictionary. 4. Exit.
Code	d1={'A':1,'B':2} d2={'C':3} d1.update(d2) print("Concatenated dictionary is:") print(d1)

Problem Statement	Python Program to check if a given key exists in a dictionary or not.
Concept	1. Declare and initialize a dictionary to have some key-value pairs. 2. Take a key from the user and store it in a variable. 3. Using an if statement and the in operator, check if the key is present in the dictionary using the dictionary.keys() method. 4. If it is present, print the value of the key. 5. If it isn’t present, display that the key isn’t present in the dictionary. 6. Exit.

Code	d={'A':1,'B':2,'C':3}
	key=raw_input("Enter key to check:")
	if key in d.keys():
	  print("Key is present and value of the key is:")
	  print(d[key])
	else:
	  print("Key isn't present!")

Problem Statement	Python Program to remove the given key from a dictionary.
Concept	1. Declare and initialize a dictionary to have some key-value pairs. 2. Take a key from the user and store it in a variable. 3. Using an if statement and the in operator, check if the key is present in the dictionary. 4. If it is present, delete the key-value pair. 5. If it isn't present, print that the key isn't found and exit the program. 6. Exit.

Code	d = {'a':1,'b':2,'c':3,'d':4}
	print("Initial dictionary")
	print(d)
	key=raw_input("Enter the key to delete(a-d):")
	if key in d:
	  del d[key]
	else:
	  print("Key not found!")
	  exit(0)
	print("Updated dictionary")
	print(d)

Problem Statement	Python Program to count the frequency of words appearing in a string using a dictionary.
Concept	1. Enter a string and store it in a variable. 2. Declare a list variable and initialize it to an empty list. 3. Split the string into words and store it in the list. 4. Count the frequency of each word and store it in another list. 5. Using the zip() function, merge the lists containing the words and the word counts into a dictionary. 3. Print the final dictionary. 4. Exit.
Code	test_string=raw_input("Enter string:") l=[] l=test_string.split() wordfreq=[l.count(p) for p in l] print(dict(zip(l,wordfreq)))

Problem Statement	Python Program to create a dictionary with key as first character and value as words starting with that character.
Concept	1. Enter a string and store it in a variable. 2. Declare an empty dictionary. 3. Split the string into words and store it in a list. 4. Using a for loop and if statement check if the word already present as a key in the dictionary. 5. If it is not present, initialize the letter of the word as the key and the word as the value and append it to a sublist created in the list. 6. If it is present, add the word as the value to the corresponding sublist. 7. Print the final dictionary. 8. Exit.

Code	test_string=raw_input("Enter string:")
	l=test_string.split()
	d={}
	for word in l:
	  if(word[0] not in d.keys()):
	  d[word[0]]=[]
	  d[word[0]].append(word)
	  else:
	  if(word not in d[word[0]]):
	  d[word[0]].append(word)
	for k,v in d.items():
	  print(k,":",v)

Problem Statement	Python Program to count the number of vowels present in a string using sets
Concept	1. Enter a string and store it in a variable. 2. Initialize a count variable to 0. 3. Create a set containing vowels. 4. Use a for loop to traverse through the letters in the string. 5. Using an if statement, check if the letter in the string is equal to a vowel. 6. If it is equal, increment the vowel count. 7. Print the final count of the vowels. 8. Exit.

Code	s=raw_input("Enter string:")
	count = 0
	vowels = set("aeiou")
	for letter in s:
	  if letter in vowels:
	  count += 1
	print("Count of the vowels is:")
	print(count)

Problem Statement	Python Program to check common letters in the two input strings.
Concept	1. Enter two input strings and store it in separate variables. 2. Convert both of the strings into sets and find the common letters between both the sets. 3. Store the common letters in a list. 4. Use a for loop to print the letters of the list. 5. Exit.
Code	s1=raw_input("Enter first string:") s2=raw_input("Enter second string:") a=list(set(s1)&set(s2)) print("The common letters are:") for i in a: print(i)

Problem Statement	Python Program to display which letters are present in both the strings.
Concept	1. Enter two input strings and store it in separate variables. 2. Convert both of the strings into sets and find the union between both the sets. 3. Store the letters in a list. 4. Use a for loop to print the letters of the list. 5. Exit.
Code	s1=raw_input("Enter first string:") s2=raw_input("Enter second string:") a=list(set(s1)|set(s2)) print("The letters are:") for i in a: print(i)

Problem Statement	Python Program to find the fibonacci series using recursion.
Concept	1. Take the number of terms from the user and store it in a variable. 2. Pass the number as an argument to a recursive function named fibonacci. 3. Define the base condition as the number to be lesser than or equal to 1. 4. Otherwise call the function recursively with the argument as the number minus 1 added to the function called recursively with the argument as the number minus 2. 5. Use a for loop and print the returned value which is the fibonacci series. 6. Exit.

Code	def fibonacci(n):
	  if(n <= 1):
	  return n
	  else:
	  return(fibonacci(n-1) + fibonacci(n-2))
	n = int(input("Enter number of terms:"))
	print("Fibonacci sequence:")
	for i in range(n):
	  print fibonacci(i),

Problem Statement	Python Program to find the factorial of a number using recursion.
Concept	1. Take a number from the user and store it in a variable. 2. Pass the number as an argument to a recursive factorial function. 3. Define the base condition as the number to be lesser than or equal to 1 and return 1 if it is. 4. Otherwise call the function recursively with the number minus 1 multiplied by the number itself. 5. Then return the result and print the factorial of the number. 6. Exit.

Code	def factorial(n):
	  if(n <= 1):
	  return 1
	  else:
	  return(n*factorial(n-1))
	n = int(input("Enter number:"))
	print("Factorial:")
	print(factorial(n))

Problem Statement	Python Program to find the GCD of two numbers using recursion.
Concept	1. Take two numbers from the user. 2. Pass the two numbers as arguments to a recursive function. 3. When the second number becomes 0, return the first number. 4. Else recursively call the function with the arguments as the second number and the remainder when the first number is divided by the second number. 5. Return the first number which is the GCD of the two numbers. 6. Print the GCD. 7. Exit.

Code	def gcd(a,b):
	  if(b==0):
	  return a
	  else:
	  return gcd(b,a%b)
	a=int(input("Enter first number:"))
	b=int(input("Enter second number:"))
	GCD=gcd(a,b)
	print("GCD is: ")
	print(GCD)

Problem Statement	Python Program to reverse a string using recursion.
Concept	1. Take a string from the user. 2. Pass the string as an argument to a recursive function to reverse the string. 3. In the function, put the base condition that if the length of the string is equal to 0, the string is returned. 4. Otherwise recursively call the reverse function to slice the part of the string except the first character and concatenate the first character to the end of the sliced string. 5. Print the reversed string. 6. Exit.

Code	def reverse(string):
	  if len(string) == 0:
	  return string
	  else:
	  return reverse(string[1:]) + string[0]
	a = str(input("Enter the string to be reversed: "))
	print(reverse(a))

Problem Statement	Python Program to read the contents of a file.
Concept	1. Take the file name from the user. 2. Use readline() function for the first line first. 3. Use a while loop to print the first line and then read the remaining lines and print it till the end of file. 4. Exit.

Code	a=str(input("Enter the name of the file with .txt extension:"))
	file2=open(a,'r')
	line=file2.readline()
	while(line!=""):
	  print(line)
	  line=file2.readline()
	file2.close()

Problem Statement	Python Program to count the number of words in a text file.
Concept	1. Take the file name from the user. 2. Read each line from the file and split the line to form a list of words. 3. Find the length of items in the list and print it. 4. Exit.

Code	fname = input("Enter file name: ")
	
	num_words = 0
	
	with open(fname, 'r') as f:
	  for line in f:
	  words = line.split()
	  num_words += len(words)
	print("Number of words:")
	print(num_words)

Problem Statement	Python Program to copy the contents of one file into another.
Concept	1. Open one file called test.txt in read mode. 2. Open another file out.txt in write mode. 3. Read each line from the input file and write it into the output file. 4. Exit.

Code	with open("test.txt") as f:
	  with open("out.txt", "w") as f1:
	  for line in f:
	  f1.write(line)

Problem Statement	Python Program to append the contents of one file to another file.
Concept	1. Take the name of the file to be read from and the file to append into from the user. 2. Open the file to be read and read the content of the file and store it in a variable. 3. Open the file to append the data to and write the data stored in the variable into the file. 4. Close both the files. 5. Exit.
Code	name1 = input("Enter file to be read from: ") name2 = input("Enter file to be appended to: ") fin = open(name1, "r") data2 = fin.read() fin.close() fout = open(name2, "a") fout.write(data2) fout.close()

Problem Statement	Python Program to read a file and capitalize the first letter of every word in the file.
Concept	1. Take the file name from the user. 2. Read each line from the file and use the title() function to capitalize each word in the line. 3. Print the altered lines of the file. 5. Exit.

Code	fname = input("Enter file name: ")
	
	with open(fname, 'r') as f:
	  for line in f:
	  l=line.title()
	  print(l)

Problem Statement	Python program to search for an element in a linked list using recursion.
Concept	1. Create a class Node. 2. Create a class LinkedList. 3. Define methods append and display inside the class LinkedList to append data and display the linked list respectively. 4. Define methods find_index and find_index_helper. 5. find_index calls find_index_helper to search for the key recursively. 6. Create an instance of LinkedList, append data to it and display the list. 7. Prompt the user for a key to search and search for it.

Code	class Node:
	  def __init__(self, data):
	  self.data = data
	  self.next = None
	
	class LinkedList:
	  def __init__(self):
	  self.head = None
	  self.last_node = None
	
	  def append(self, data):
	  if self.last_node is None:
	  self.head = Node(data)
	  self.last_node = self.head
	  else:
	  self.last_node.next = Node(data)
	  self.last_node = self.last_node.next
	
	  def display(self):
	  current = self.head
	  while current is not None:
	  print(current.data, end = ' ')
	  current = current.next
	
	  def find_index(self, key):
	  return self.find_index_helper(key, 0, self.head)
	
	  def find_index_helper(self, key, start, node):
	  if node is None:
	  return -1
	
	  if node.data == key:
	  return start
	  else:

```
            return self.find_index_helper(key, start + 1, node.next)

a_llist = LinkedList()
for data in [3, 5, 0, 10, 7]:
   a_llist.append(data)
print('The linked list: ', end = '')
a_llist.display()
print()

key = int(input('What data item would you like to search for? '))
index = a_llist.find_index(key)
if index == -1:
   print(str(key) + ' was not found.')
else:
   print(str(key) + ' is at index ' + str(index) + '.')
```

Problem Statement	Python program to implement a stack.
Concept	1. Create a class Stack with instance variable items initialized to an empty list. 2. Define methods push, pop and is_empty inside the class Stack. 3. The method push appends data to items. 4. The method pop pops the first element in items. 5. The method is_empty returns True only if items is empty. 6. Create an instance of Stack and present a menu to the user to perform operations on the stack.

Code	class Stack:
	  def __init__(self):
	  self.items = []
	
	  def is_empty(self):
	  return self.items == []
	
	  def push(self, data):
	  self.items.append(data)
	
	  def pop(self):
	  return self.items.pop()
	
	s = Stack()
	while True:
	  print('push <value>')
	  print('pop')
	  print('quit')
	  do = input('What would you like to do? ').split()
	
	  operation = do[0].strip().lower()
	  if operation == 'push':
	  s.push(int(do[1]))
	  elif operation == 'pop':
	  if s.is_empty():
	  print('Stack is empty.')
	  else:
	  print('Popped value: ', s.pop())
	  elif operation == 'quit':
	  break

Problem Statement	Python program to find the Nth node in the in-order traversal of a binary tree.
Concept	1. Create a class BinaryTree with instance variables key, left and right. 2. Define methods set_root, insert_left, insert_right, inorder_nth, inorder_nth_helper and search. 3. The method set_root takes a key as argument and sets the variable key equal to it. 4. The methods insert_left and insert_right insert a node as the left and right child respectively. 5. The method search returns a node with a specified key. 6. The method inorder_nth displays the nth element in the in-order traversal of the tree. It calls the recursive method inorder_nth_helper.

Code	class BinaryTree:
	  def __init__(self, key=None):
	  self.key = key
	  self.left = None
	  self.right = None
	
	  def set_root(self, key):
	  self.key = key
	
	  def inorder_nth(self, n):
	  return self.inorder_nth_helper(n, [])
	
	  def inorder_nth_helper(self, n, inord):
	  if self.left is not None:
	  temp = self.left.inorder_nth_helper(n, inord)
	  if temp is not None:
	  return temp
	  inord.append(self)
	  if n == len(inord):
	  return self
	  if self.right is not None:
	  temp = self.right.inorder_nth_helper(n, inord)
	  if temp is not None:
	  return temp
	
	  def insert_left(self, new_node):
	  self.left = new_node
	
	  def insert_right(self, new_node):
	  self.right = new_node
	
	  def search(self, key):
	  if self.key == key:

```
            return self
        if self.left is not None:
            temp =  self.left.search(key)
            if temp is not None:
                return temp
        if self.right is not None:
            temp =  self.right.search(key)
            return temp
        return None


btree = None

print('Menu (this assumes no duplicate keys)')
print('insert <data> at root')
print('insert <data> left of <data>')
print('insert <data> right of <data>')
print('inorder <index>')
print('quit')

while True:
    do = input('What would you like to do? ').split()

    operation = do[0].strip().lower()
    if operation == 'insert':
        data = int(do[1])
        new_node = BinaryTree(data)
        suboperation = do[2].strip().lower()
        if suboperation == 'at':
                btree = new_node
        else:
            position = do[4].strip().lower()
            key = int(position)
            ref_node = None
            if btree is not None:
                ref_node = btree.search(key)
            if ref_node is None:
                print('No such key.')
                continue
            if suboperation == 'left':
                ref_node.insert_left(new_node)
            elif suboperation == 'right':
                ref_node.insert_right(new_node)
```

```
    elif operation == 'inorder':
        if btree is not None:
            index = int(do[1].strip().lower())
            node = btree.inorder_nth(index)
            if node is not None:
                print('nth term of inorder traversal: {}'.format(node.key))
            else:
                print('index exceeds maximum possible index.')
        else:
            print('Tree is empty.')

    elif operation == 'quit':
        break
```