INSTITUTE OF SCIENCE & TECHNOLOGY FOR ADVANCED STUDIES & RESEARCH

ISTAR-CONSTITUENT COLLEGE OF CVM UNIVERSITY

DEPARTMENT MASTER OF COMPUTER APPLICATION Lab

MANUAL

FOR

PAPER CODE :101550125

PAPER TITLE : PRACTICAL BASED ON PYTHON – BEGINNER TO PROFESSIONAL

2024-25

Prepared By:

Monani Charmi
22401550301007
MCA
ISTAR

Reviewed By:

Dr. Suchita patel
Asst. Professor
Computer Science Department
ISTAR

	Unit – 1
Prpgramme _01	**'''Open the hello_world.py file you just created.make a typo somewhere in the line and run the program again.Can you make a typo that generates an error? Can you make sense of the error message? Can you make a typo that doesn't generate an error? Why do you think it didn't make an error?'''**
CODE:-	print("charmi") print ("Monani Charmi") #This generates no error, because syntax is correct Print ("Monani Charmi") # This generates error because the syntax is incorrect(type error)
Output:-	= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/hello_world.py charmi Monani Charmi Traceback (most recent call last): File "C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/hello_world.py", line 5, in <module> Print ("Monani Charmi") NameError: name 'Print' is not defined. Did you mean: 'print'?
Program 2_01	**Simple Message: Store a message in a variable, and then print that message**
CODE:-	print("CHARMI") msg = "I love learning to use Python." print(msg)
Output:-	= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/simple_message.py CHARMI I love learning to use Python.
Program 2_02	**Simple message _2: -Store a message in a variable and then print the message .**
CODE:-	'''Simple message_2 Store a message in a variable and print the message, then change the value of your variable to new message and print to new message.''' print("CHARMI") p = "GM" print("stmt1:",p) p = "GOOD MOARNING" print("stmt2:",p)
Output:-	= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/simple_message2.py CHARMI stmt1: GM stmt2: GOOD MOARNING
Program 2-3	Personal message Store a person's name in a variable and print a message to that person . Your message should be simple, such as "Hello Student, would you like to learn some python today ?

CODE:-	print("CHARMI") name = "eric" msg = f"Hello {name.title()}, would you like to learn some Python today?" print(msg)
Output:-	========== RESTART: C:/Users/HP/Documents/MCA/python/program 2-3.py ======== CHARMI Hello Eric, would you like to learn some Python today?
Program 2-4	String manipulation Store a person's name in a variable and then print that person's name in lowercase, uppercase and title case
Code:-	print("CHARMI") name = "python" print(name.lower()) print(name.upper()) print(name.title())
Output:-	========== RESTART: C:/Users/HP/Documents/MCA/python/program 2-3.py ======== CHARMI Hello Eric, would you like to learn some Python today?
Program 2-5	Find quote from a famous person admire .Print the quote and the name of its author. Your output should look something like the following including the quotation marks: Albert Einstein once said ,"A person who never made a mistake never tried anything new"
Code:-	**print("CHARMI")** **a = "'Abdul Kalam'"** **Quote=" 'The best brains of the nation may be found on the last benches of the classroom '"** **print(a,"Once Said",Quote)**
Output:-	========== RESTART: C:/Users/HP/Documents/MCA/python/program 2-5.py ========== CHARMI 'Abdul Kalam' Once Said 'The best brains of the nation may be found on the last benche f the classroom '
Program 2-6	Repeat Exercise 2-5 but this time store the famous persons name in a variable called famous_person.Then compose your message and store it in a new variable called message.Print your message

Code:-	print("CHARMI") print('Albert Einstein once said, "A person who never made a mistake') print('never tried anything new."') print("\tCharmi Monani\n" print("MCA DEPARTMENT:") print(name) print("\nUsing lstrip():") print(name.lstrip()) print("\nUsing rstrip():") print(name.rstrip()) print("\nUsing strip():") print(name.strip())
Output:-	========== RESTART: C:/Users/HP/Documents/MCA/python/program 2-6.py ======= CHARMI Albert Einstein once said, "A person who never made a mistake never tried anything new."
Program 2-7	'''Store a person's name,and include some whitespaces character at beginning and end of the name.make sure you use each character combinbation,"\t" and "\n",at least once.print the name once, so the whitespace around the Name is displayed. Then the name using each of the three stripping function, strip (), strip (), and strip ().'''
Code:-	print("CHARMI") name = "\tCharmi Monani\n" print("MCA DEPARTMENT:") print(name) print("\nUsing lstrip():") print(name.lstrip()) print("\nUsing rstrip():") print(name.rstrip()) print("\nUsing strip():") print(name.strip())
Output:-	========== RESTART: C:/Users/HP/Documents/MCA/python/program 2-7.py ====== CHARMI MCA DEPARTMENT: Charmi Monani Using lstrip(): Charmi Monani Using rstrip(): Charmi Monani Using strip(): Charmi Monani
Program 2-8	Write addition,subtraction,division and mulltiplicaton operations that eachresult in the nimber 8.Be sure to enclose your operations in print statements to see the results you should create four lines that looks like this:print(5+3) Your output should simply be four lines with the number 8 appearing once on each
Code:-	print("CHARMI")

	print(5+3) print(2*4) print(16-8) print(16/2)
Output:-	========== RESTART: C:/Users/HP/Documents/MCA/python/program 2-8.py ====== CHARMI 8 8 8 8.0
Program 2-9	'''store your favorite number in a variable .then using that variable ,create a message that reveals your favourite number.print the message'''
Code:-	**print("CHARMI")** **c=168203** **print("MY FAVORITE NUMBER IS:",c)**
Output:-	========== RESTART: C:/Users/HP/Documents/MCA/python/program 2-9.py ========= CHARMI MY FAVORITE NUMBER IS: 168203
Program 2-10	Adding comments: Choose two of the programs you have written and add at least one comment each.if you don't have anything specific to write because your programs are too simple at this point just add yoyur name and the current date at the top of each program file.then write one sentence describing what the program does
Code:-	**# Monani Charmi** **# Date:22-09-2024** **# This program stores and prints my favorite number.** **print("CHARMI")** **fav_Num =16** **message = ("My favourite number is ",fav_Num)** **print(message)** **# Date: 22-09-2024** **# This program stores a quote from a movie character and prints it.** **print("CHARMI")** **name = "python"** **print(name.lower())** **print(name.upper())** **print(name.title())**

	========== RESTART: C:/Users/HP/Documents/MCA/python/program 2-10.py ===== CHARMI ('My favourite number is ', 16) CHARMI python PYTHON Python
Programme _03	'''Write A program to use python string'''
Code:-	**print("CHARMI")** **String=['abcd',786,2.23,'John',70.2]** **tinystring=[123,'john']** **print(string)#print complete string** **print(string[0])#print first element of the string** **print(string[1:3])#print element starting from 2nd till 3rd** **print(string[2:])#print element starting from 3rd** **print(tinystring*2)#print string two times** **print(tuple+tinystring)#print concatenated string**
	CHARMI ['abcd', 786, 2.23, 'John', 70.2] abcd [786, 2.23] [2.23, 'John', 70.2] [123, 'john', 123, 'john'] ['abcd', 786, 2.23, 'John', 70.2, 123, 'john']
Programme _04	'''Write A program to use python list'''
Code:-	**print("CHARMI")** **list=['abcd',786,2.23,'John',70.2]** **tinylist=[123,'john']** **print(list)#print complete list** **print (list [0]) #print first element of the list** **print(list[1:3])#print element starting from 2nd till 3rd** **print(list[2:])#print element starting from 3rd** **print(tinylist*2)#print list two times** **print(list+tinylist)#print concatenated lists**

	CHARMI ['abcd', 786, 2.23, 'John', 70.2] abcd [786, 2.23] [2.23, 'John', 70.2] [123, 'john', 123, 'john'] ['abcd', 786, 2.23, 'John', 70.2, 123, 'john']
Programme _05	'''Write A program to use python tuple'''
Code:-	**print("CHARMI")** **tuple=['abcd',786,2.23,'John',70.2]** **tinytuple=[123,'john']** **print(tuple)#print complete tuple** **print(tuple[0])#print first element of the tuple** **print(tuple[1:3])#print element starting from 2nd till 3rd** **print(tuple[2:])#print element starting from 3rd** **print(tinytuple*2)#print tuple two times** **print(tuple+tinytuple)#print concatenated tuple**
	CHARMI ['abcd', 786, 2.23, 'John', 70.2] abcd [786, 2.23] [2.23, 'John', 70.2] [123, 'john', 123, 'john'] ['abcd', 786, 2.23, 'John', 70.2, 123, 'john']
Programme _06	Write a program to use Python Dictionary.
Code:-	print("CHARMI") dict = {} dict['one'] = "This is one" dict['two'] = "This is two" tinydict = {'name':'charmi','code':1111,'dept':'graphics'} print(dict['one']) #Print value for one key print(dict['two']) #Print value for two key print(tinydict) #Prints complete dictionary print(tinydict.keys()) #Prints all the keys

	print(tinydict.values()) #Prints all the values
	CHARMI This is one This is two {'name': 'charmi', 'code': 1111, 'dept': 'graphics'} dict_keys(['name', 'code', 'dept']) dict_values(['charmi', 1111, 'graphics'])
Programme _3_1	Names:Store the names of a few of your friends in a list called names.print each person's name by accessing each element in the list one at a time
Code:-	**print("CHARMI")** **name=["Kruti","Preeti","Harsh","Pinki","radhu"]** **print (name[0])** **print(name[1])** **print (name[2])** **print (name[3])** **print (name[4])** **print (name[-2])** **print (name[-4])**
	CHARMI Kruti Preeti Harsh Pinki radhu Pinki Preeti
Programme _3_2	Greetings: Start with the list you used in Exercise 3-1, but instead of just printing each person's name, print a message to them. The text of each message should be the same, but each message should be personalized with the person's name.
Code:-	**print("CHARMI")** **names=["Kriti","Geeta","Meeta"]** **for names in names:** **print("You look like a princess "+names)**

	CHARMI You look like a princess Kriti You look like a princess Geeta You look like a princess Meeta
Programme _3_3	Your Own List: Think of your favorite mode of transportation, such as a motorcycle or a car, and make a list that stores several examples. Use your list to print a series of statements about these items, such as “I would like to own a Honda motorcycle.”
Code:-	print("CHARMI") vehical = ["Honda","Ferrari La Ferrari","BMW","Tesla","Mercedes"] for car in vehical: print("The type of cars I weant to own"+car[0:]+"and wanna ride in it")
	CHARMI The type of cars I weant to ownHondaand wanna ride in it The type of cars I weant to ownFerrari La Ferrariand wanna ride in it The type of cars I weant to ownBMWand wanna ride in it The type of cars I weant to ownTeslaand wanna ride in it The type of cars I weant to ownMercedesand wanna ride in it
Programme _07	#Write a program to use python Arithmetic operations
Code:-	print("CHARMI") a=21 b=10 c=0 c=a+b print("Line 1-value of c is",c) c=a-b print("Line 2-value of c is",c) c=a*b print("Line 3-value of c is",c) c=a/b print("Line 4-value of c is",c) c=a%b print("Line 5-value of c is",c) a=2 b=3 c=a**b print("Line 6-value of c is",c) a=10 b=5 c=a//b print("Line 7-value of c is",c

	CHARMI Line 1-value of c is 31 Line 2-value of c is 11 Line 3-value of c is 210 Line 4-value of c is 2.1 Line 5-value of c is 1 Line 6-value of c is 8 Line 7-value of c is 2
Programme _08	'''Write a program to perform Assignment operators'''
Code:-	print("CHARMI") a = 21 b = 10 c = 0 c = a + b print("Line1- The value of c is",c) c+=a print("Line2- The value of c is",c) c-=a print("Line3- The value of c is",c) c*=a print("Line4- The value of c is",c) c/=a print("Line5- The value of c is",c) c%=a print("Line6- The value of c is",c) c**a print("Line7- The value of c is",c) c//a print("Line8- The value of c is",c)

	CHARMI Line1- The value of c is 31 Line2- The value of c is 52 Line3- The value of c is 31 Line4- The value of c is 651 Line5- The value of c is 31.0 Line6- The value of c is 10.0 Line7- The value of c is 10.0 Line8- The value of c is 10.0
Programme _09	#Write a python program to use Logical operator

Code:-	print("CHARMI")
	a=10
	b=20
	list=[1,2,3,4,5]
	if(a in list):
	  print("Line-1 is available in the given list")
	else:
	  print("Line-1 is not available in the given list")
	if(b not in list):
	  print("Line-2 is not available in the given list")
	else:
	  print("Line-2 is available in the given list")
	c=a/b
	if(c in list):
	  print("Line-3 is available in the given list")
	else:
	  print("Line-3is not available in the given list")

	CHARMI Line-1 is not available in the given list Line-2 is not available in the given list Line-3is not available in the given list
Programme _10	'''Write a python program -discount is calculated on the input amount. rate of discount is 5%, if the amount is less than 1000, and 10% if it is above 10,000'''

Code:-	print("CHARMI")
	amount=int(input("Enter Amount:"))
	if amount<1000:
	  discount=amount*0.05
	  print("Discount",discount)
	else:
	  discount=amount*0.10
	  print("Discount",discount)
	print("Net payable:",amount-discount)

	CHARMI Enter Amount:780 Discount 39.0 Net payable: 741.0
Programme _11	Write the same program with the help of if...elif...else
Code:-	**print("CHARMI")** **amount=int(input("Enter Amount:"))** **if amount<1000:** ** discount=amount*0.05** ** print("Discount",discount)** **elif amount<5000:** ** print("Discount",discount)** **else:** ** discount=amount*0.15** **print("Discount",discount)** **print("Net payable:",amount-discount)**

	CHARMI Enter Amount:78 Discount 3.9000000000000004 Discount 3.9000000000000004 Net payable: 74.1
Programme _12	Write a program for use of Nested if
Code:-	**print("CHARMI")** **num=int(input("Enter Number:--"))** **if num%2==0:** ** if num%3==0:** ** print("Divisible by 3 and 2")** ** else:** ** print("Divisible by 2 not divisible by 3")** **else:** ** if num%3==0:** ** print("Divisible by 3 not divisible by 2")** **else:** ** print("not divisible by 3 not divisible by 2")**
	CHARMI Enter Number:--78 Divisible by 3 and 2 ========== RESTART: G:\charmi\Python_program\program16.py ========== CHARMI Enter Number:--8 Divisible by 2 not divisible by 3
Programme _13	#Write a python program to use while loop
Code:-	**print("CHARMI")** **count=0** **while(count<9):** ** print("The Count is:",count)** ** count=count+1** ** print ("Good Bye!")**

	CHARMI The Count is: 0 Good Bye! The Count is: 1 Good Bye! The Count is: 2 Good Bye! The Count is: 3 Good Bye! The Count is: 4 Good Bye! The Count is: 5 Good Bye! The Count is: 6 Good Bye! The Count is: 7 Good Bye! The Count is: 8 Good Bye!
Programme _14	#Write a python program to use for loop
Code:-	**print("CHARMI")** **for var in list(range(5)):** **print(var)**
	CHARMI 0 1 2 3 4
Programme _15	Write a python program to use break statement

Code:-	for letter in 'Python':
	if letter == 'h':
	break
	print("Current letter:",letter)

```
var=10#Second example
while var>0:
   print("Current variable  value ",var)
   var=var-1
   if var==5:
      break
   print("Good Bye!")
```

```
CHARMI
Current letter: P
Current letter: y
Current letter: t
Current variable name 10
Good Bye!
Current variable name 9
Good Bye!
Current variable name 8
Good Bye!
Current variable name 7
Good Bye!
Current variable name 6
```

Programme _16

Write a python program to use continue statement

Code:-

```
print("CHARMI")
for letter in 'Python':#first Example
   if letter == 'h':
      continue
   print("Current letter:",letter)

var=10#Second example
while var>0:
   var=var-1
   if var==5:
      continue
```

	print("Current letter:",var) print("Good Bye!")
	CHARMI Current letter: P Current letter: y Current letter: t Current letter: o Current letter: n Current letter: 9 Good Bye! Current letter: 8 Good Bye! Current letter: 7 Good Bye! Current letter: 6 Good Bye! Current letter: 4 Good Bye! Current letter: 3 Good Bye! Current letter: 2 Good Bye! Current letter: 1 Good Bye! Current letter: 0 Good Bye!
Programme _17	Write a python program to use pass statement

Code:-	print("CHARMI")
	for letter in 'Python':#first Example
	  if letter == 'h':
	  pass
	  print("This is a pass block")
	  print("Current letter:",letter)
	  print("Good Bye!")

	CHARMI This is a pass block Current letter: P Good Bye! This is a pass block Current letter: y Good Bye! This is a pass block Current letter: t Good Bye! This is a pass block Current letter: h Good Bye! This is a pass block Current letter: o Good Bye! This is a pass block Current letter: n Good Bye!
Programme _18	Write a python program to use of break in a for loop iterating over a list .User inputs a number , which is searched in the list . if it is found , then the loop terminates With the 'found' message

Code:-	print("CHARMI")
	no = int(input("Enter the number: "))
	numbers = [11, 33, 55, 39, 55, 75, 37, 21, 23, 41, 13]
	if no in numbers:
	  print("Number found in the list")
	else:
	  print("Number not found in the list")

	CHARMI Enter the number: 2 Number not found in the list ========== RESTART: C:/U CHARMI Enter the number: 55 Number found in the list
Programme _19	Write a program to purposefully raise Indentation Error and Correct it
Code:-	**print("CHARMI")** **n1=int(input("Enter n1 value"))** **n2=int(input("Enter n2 value"))** **if n1>n2:** ** print("n1 is big")** **else:** ** print("n2 is big")**
	CHARMI Enter n1 value74 Enter n2 value25 n1 is big
Programme _20	''Write a python program to print a number is positive or negative using else'''
Code:-	**print("CHARMI")** **number=int(input("Enter a Number:-"))** **if number>0:** ** print("number is positive")** **else:** ** print("number is negetive")**

	CHARMI Enter a Number:-78 number is positive ========== RESTART: G:/charmi/Python_program/prg18.py ========== CHARMI Enter a Number:--89 number is negetive
Programme _21	'''Write a python program to calculate the square root'''
Code:-	**print("CHARMI")** **#To take the input from user** **num=float(input("Enter a Number:-"))** **num_sqrt=num**0.5** **print("The square root of %0.3f is %0.3f%",(num,num_sqrt))**
	CHARMI Enter a Number:-4 The square root of %0.3f is %0.3f% (4.0, 2.0)
Programme _22	Write a program to compute distance between two points taking input from the user (Pythagorean Theorem).
Code:-	**print("CHARMI")** **import math;** **x1=int(input("Enter x1----->"))** **y1=int(input("Enter y1----->"))** **x2=int(input("Enter x2----->"))** **y2=int(input("Enter y2----->"))** **d1=(x2-x1)*(x2-x1);** **d2=(y2-y1)*(y2-y1);** **res=math.sqrt(d1+d2)** **print("Distance between two points:",res)**
	CHARMI Enter x1----->4 Enter y1----->5 Enter x2----->8 Enter y2----->3 Distance between two points: 4.47213595499958
Programme _23	'''Python program to convert temperature in celcius to Fahrenheit'''
Code:-	**print("CHARMI")** **celsius=37.5** **fahrenheit= celsius*1.8+32**

	print('%0.1f degree Celsius is equal to %0.1f degree Fahrenhit'%(celsius,fahrenheit))
	CHARMI 37.5 degree Celsius is equal to 99.5 degree Fahrenhit
Programme _24	'''Write a python program to check whether the number is prime or not'''

Code:-	print("CHARMI")
	num = int(input("Enter the number:"))
	for i in range(2,num):
	if num%i==0:
	print(num, "is not a prime number")
	break;
	else:
	print(num,"is a prime number")

	CHARMI Enter the number:3 3 is a prime number
Programme _25	Write a Python program to display all the prime numbers within an interval.

Code:-	print("CHARMI")
	lower=int(input("Enter starting value: "))
	upper=int(input("Enter ending value: "))
	print("prime numbers between",lower,"and",upper,"are:")
	for num in range(lower,upper+1):
	if num>1:
	for i in range(2,num):
	if num%i==0:
	break
	else:
	print(num)

	CHARMI Enter starting value: 5 Enter ending value: 10 prime numbers between 5 and 10 are: 5 7
Programme _26	'''Write a program in python to find the number is palindrome or not'''

Code:-	print("CHARMI")
	num = int(input("Enter the number: "))
	temp = num # Store the original number
	rev = 0
	while num > 0:
	temp1 = num % 10 # Get the last digit
	rev = rev * 10 + temp1 # Build the reversed number
	num = num // 10 # Remove the last digit from the number
	if temp == rev:
	print(temp, "is a palindrome")
	else:
	print(temp, "is not a palindrome number")

	CHARMI Enter the number: 45 45 is not a palindrome number ========== RESTART: C:/U: CHARMI Enter the number: 44 44 is a palindrome
Programme _27	Write a python program to Find the area of the circle
Code:-	**print("CHARMI")** **def circle_area(radius):**

```
    pi = 3.14159
    area = pi * (radius ** 2)
    return area

radius = float(input("Enter the radius of the circle: "))
area = circle_area(radius)
print("The area of the circle is:", area)
```

```
CHARMI
Enter the radius of the circle: 85
The area of the circle is: 22697.98775
```

Programme _28	Write a python program to find the compound interest.

Code:-

```
print("CHARMI")
p= 1500
t= 2
r= 5.4
a=p*(1+(r/100))**t
ci=a-p # compound interest = amount - principal amount
print(ci)
```

```
CHARMI
166.37400000000002
```

Programme _29	Write a program to find largest number among three input numbers.

Code:-

```
print("CHARMI")
num1=int(input("ENTER A FIRST NUMBER: "))
num2=int(input("ENTER A SECOND NUMBER: "))
num3=int(input("ENTER A THIRD NUMBER: "))
if(num1>num2) and (num1>num3):
   print(num1,"It is a large number")
elif(num2>num1) and (num2>num3):
      print(num2,"It is a large number")
else :
      print(num3,"It is a large number ")
```

	CHARMI ENTER A FIRST NUMBER: 78 ENTER A SECOND NUMBER: 85 ENTER A THIRD NUMBER: 55 85 It is a large number
Programme _30	Print the first 10 natural numbers using loop.
Code:-	**print("CHARMI")** **i=1** **while i<=10:** **print(i)** **i+=1**
	CHARMI 1 2 3 4 5 6 7 8 9 10
Programme _31	Write a program to print multiplication table(from 1 to 10) in python.
Code:-	**print("CHARMI")** **n=int(input("Enter a number:-"))** **for i in range(1,11):** **print(n,"x",i,"=",n*i)**

	CHARMI Enter a number:-74 74 x 1 = 74 74 x 2 = 148 74 x 3 = 222 74 x 4 = 296 74 x 5 = 370 74 x 6 = 444 74 x 7 = 518 74 x 8 = 592 74 x 9 = 666 74 x 10 = 740
Programme _32	Python program to print all the even numbers within the given range.

Code:-	print("CHARMI")
	n=int(input("Enter number: "))
	print("Even numbers from 0 to",n,":")
	for i in range(n):
	  if i%2==0:
	  print(i)

	CHARMI Enter number: 15 Even numbers from 0 to 15 : 0 2 4 6 8 10 12 14
Programme _33	Python program to count total numbers of digits in a number.
Code:-	print("CHARMI") n=int(input("Enter a number:")) n=str(n) count=0 for i in n: count+=1 print(count)
	CHARMI Enter a number:1234568523 10
Programme _34	Python program that acepts a word from user and reverses it.
Code:-	print("CHARMI") enter_string = input() reverse_string = ""

	for i in enter_string: reverse_string = i + reverse_string print(reverse_string)
	CHARMI Input a word to reverse: mishri irhsim
Programme _35	Write a program to count the number of vowels present in a text file.

Code:-	print("CHARMI")
	def find_vowels(text):
	  vowels = "aeiouAEIOU" # List of vowels (both uppercase and lowercase)
	  found_vowels = [] # List to store the found vowels
	  for char in text:
	  if char in vowels:
	  found_vowels.append(char)
	  return found_vowels
	# Example usage
	input_text = input("Enter a string: ")
	vowels_in_text = find_vowels(input_text)
	print(f"Vowels found in the text: {vowels_in_text}")

Output:-	CHARMI Input a word to reverse: mishri irhsim
Programme _36	Write a python program to display the Fibonacci sequence up to n-th term

Code:-	print("CHARMI")
	n = int(input("Enter the number"))
	n1,n2 = 0,1
	count = 0
	if (n<=0):
	  print("Please enter a positive number:")
	else:
	  for i in range(n):
	  print(n1,end = " ")
	  count= n1 +n2
	  n1 = n2
	  n2 = count

	CHARMI Enter the number 10 0 1 1 2 3 5 8 13 21 34
Programme _37	Write a program using a for loop that loops over a sequence

Code:-	print("CHARMI")
	List1 = []
	num = int(input("Enter the number of items you wanna enter:"))
	for i in range(num):
	  List1.insert(i,int(input("Enter the number:")))
	print(List1)
	for i in List1:
	  print(i)

	CHARMI Enter the number of items you wanna enter:5 Enter the number:45 Enter the number:85 Enter the number:96 Enter the number:16 Enter the number:55 [45, 85, 96, 16, 55] 45 85 96 16 55
Programme _38	Write a program using a while loop that asks the user for a number and prints a countdown from that number to zero.

Code:-	print("CHARMI")
	num = int(input("Enter the number to start countdown:"))
	while num >0:
	  print(num,end = ">")
	  num = num - 1

```
CHARMI
Enter the number to start countdown:8
8>7>6>5>4>3>2>1>
```

Programme _39	Python program to find the factorial of a given number.

Code:-	print("CHARMI")
	num = int(input("Enter the number to find the factorial: "))
	if num < 0:
	  print("Please enter a positive digit; don't you know maths?")
	else:
	  fact = 1
	  for i in range(2, num + 1):
	  fact *= i
	  print("The factorial of", num, "is", fact)

```
CHARMI
Enter the number to find the factorial: 0
The factorial of 0 is 1

================ RESTART: C:/Users/HP/Documents/MCA/py
CHARMI
Enter the number to find the factorial: -8
Please enter a positive digit; don't you know maths?

================ RESTART: C:/Users/HP/Documents/MCA/py
CHARMI
Enter the number to find the factorial: 5
The factorial of 5 is 120
```

Programme _40	Python program to check the validity of password input by users

Code:-	print("CHARMI")
	password = input("Please enter your password:")
	if len(password)<8:
	  print("Your password must contain atleast 8 charachters")
	else:
	  has_upper = False
	  has_lower = False
	  has_digits = False
	  has_special =False
	  special_char = "!@#$%^&*(),.?\":{}|<>"

```
for i in password:
    if i.isupper():
        has_upper = True
    elif i.islower():
        has_lower = True
    elif i.isdigit():
        has_digits = True
    elif i in special_char:
        has_special = True

if not has_upper:
    print("Your password should have atleast one upper charchter")
elif not has_lower:
    print("Your password should have atleast one lower charchter")
elif not has_digits:
    print("Your password should have atleast one digit charchter")
elif not has_special:
    print("Your password should have atleast one special charchter")
else:
    print("A valid password")
```

```
================ RESTART: C:/Users/HP/Documents/MCA/python/43.py =====
CHARMI
Please enter your password:charmi2076
Your password should have atleast one upper charchter

================ RESTART: C:/Users/HP/Documents/MCA/python/43.py =====
CHARMI
Please enter your password:Charmi2076
Your password should have atleast one special charchter

================ RESTART: C:/Users/HP/Documents/MCA/python/43.py =====
CHARMI
Please enter your password:Charmi_2076
Your password should have atleast one special charchter

================ RESTART: C:/Users/HP/Documents/MCA/python/43.py =====
CHARMI
Please enter your password:Charmi**2076
A valid password
```

Programme _41

Python program to convert the month name to a number of days.

Code:-

```
print("CHARMI")
month = ["January","February","March","April","May","June","July","August",
         "September","October","November","December"]
dict={month[0]:"31",month[1]:"28",month[2]:"31",month[3]:"30",month[4
]:"31",
```

```
month[5]:"30",month[6]:"31",month[7]:"31",month[8]:"30",month[9]:"31"
,
    month[10]:"30",month[11]:"31"}
dict2={month[0]:"31",month[1]:"29",month[2]:"31",month[3]:"30",month[
4]:"31",

month[5]:"30",month[6]:"31",month[7]:"31",month[8]:"30",month[9]:"31"
,
    month[10]:"30",month[11]:"31"}
num = int(input("Enter the Year:"))
num1 = input("Enter the month:")
if num%4==0:
   print(dict2[num1])
else:
   print(dict[num1])
```

Output :-

```
CHARMI
Enter the Year:2000
Enter the month:July
31
```

Programme _42

Write a python program to count the numbers of characters in the string and store them in a dictionary data structure

Code:-

```
print("CHARMI")
num = input("Enter the word:")
dict = {}
for i in range(len(num)):
   dict[i] = num[i]
print(dict)
```

```
CHARMI
Enter the word:MEENA
{0: 'M', 1: 'E', 2: 'E', 3: 'N', 4: 'A'}
```

Programme _43

Write a program to use split and join methods in the string and trace a birthday with a dictionary data structure.

Code:-

```
# Define a dictionary to store birthday data
birthdays = {
   "John": "1990-05-12",
```

```
    "Alice": "1985-07-23",
    "Bob": "2000-01-15"
}
# Function to display all birthdays
def display_birthdays():
    print("\nCurrent Birthdays:")
    for name, date in birthdays.items():
        print(f"{name}: {date}")

# Function to add a new birthday
def add_birthday(name, date):
    birthdays[name] = date
    print(f"Added birthday for {name} on {date}.")

# Function to trace a birthday using split and join
def trace_birthday(name):
    if name in birthdays:
        date = birthdays[name]
        year, month, day = date.split("-")  # Split the date string
        formatted_date = "/".join([day, month, year])  # Join parts in a different
format
        print(f"{name}'s birthday is on {formatted_date}.")
    else:
        print(f"No birthday found for {name}.")

# Main program
while True:
    print("\n--- Birthday Tracker ---")
    print("1. Display Birthdays")
    print("2. Add Birthday")
    print("3. Trace Birthday")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")

    if choice == "1":
        display_birthdays()
    elif choice == "2":
        name = input("Enter the name: ")
        date = input("Enter the birthday (YYYY-MM-DD): ")
        add_birthday(name, date)
    elif choice == "3":
        name = input("Enter the name to trace: ")
        trace_birthday(name)
    elif choice == "4":
```

	print("Exiting the program. Goodbye!") break else: print("Invalid choice. Please try again.")

```
--- Birthday Tracker ---
1. Display Birthdays
2. Add Birthday
3. Trace Birthday
4. Exit
Enter your choice (1-4): 1

Current Birthdays:
John: 1990-05-12
Alice: 1985-07-23
Bob: 2000-01-15

--- Birthday Tracker ---
1. Display Birthdays
2. Add Birthday
3. Trace Birthday
4. Exit
Enter your choice (1-4): 3
Enter the name to trace: NO
No birthday found for NO.

--- Birthday Tracker ---
1. Display Birthdays
2. Add Birthday
3. Trace Birthday
4. Exit
Enter your choice (1-4): 2
Enter the name: CHARMI
Enter the birthday (YYYY-MM-DD): 16/8/2003
Added birthday for CHARMI on 16/8/2003.
```

Programme _44	Write a program to discount is calculated on the input amount. Rate of discount is 5%, if the amount is less than 1000, and 10% if it is above 10000.

Code:-	# Program to calculate discount based on input amount
	def calculate_discount(amount):
	  if amount < 1000:
	  discount_rate = 0.05 # 5% discount
	  elif amount > 10000:
	  discount_rate = 0.10 # 10% discount
	  else:
	  discount_rate = 0.00 # No discount
	  discount = amount * discount_rate
	  final_amount = amount - discount
	  return discount, final_amount
	# Main program
	while True:
	  try:
	  print("\n--- Discount Calculator ---")
	  amount = float(input("Enter the amount: "))
	  if amount < 0:
	  print("Amount cannot be negative. Please try again.")
	  continue
	  discount, final_amount = calculate_discount(amount)
	  print(f"Original Amount: ${amount:.2f}")
	  print(f"Discount: ${discount:.2f}")
	  print(f"Final Amount after Discount: ${final_amount:.2f}")
	  another = input("Do you want to calculate another discount? (yes/no): ").strip().lower()
	  if another != "yes":
	  print("Goodbye!")
	  break
	  except ValueError:
	  print("Invalid input. Please enter a valid amount.")

	= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI **--- Discount Calculator ---** **Enter the amount: 45** **Original Amount: $45.00** **Discount: $2.25** **Final Amount after Discount: $42.75** **Do you want to calculate another discount? (yes/no):**
Programme _45	Write a program the combination of an else statement with a for statement that searches for even number in given list
Code:-	print("CHARMI")

```
numList = [] #User input list
num = int(input("Enter the number of numbers you wanna enter:"))
for i in range(num):
    numList.insert(i,int(input("Enter the number:")))
print(numList)
for p in numList:
    if p%2==0:
        print("Wow, I see there's an Even number in the list")
        break
else:
    print("Nothing's wrong, all are Odd numbers here")
```

```
CHARMI
Enter the number of numbers you wanna enter:8
Enter the number:5
Enter the number:2
Enter the number:2
Enter the number:5
Enter the number:4
Enter the number:3
Enter the number:2
Enter the number:7
[5, 2, 2, 5, 4, 3, 2, 7]
Wow, I see there's an Even number in the list
```

Programme _46

Write a program to use break statement.

Code:-

```
# Program to demonstrate the use of the break statement in Python
while True:
    print("\n--- Number Checker ---")
    user_input = input("Enter a number (or type 'exit' to quit): ").strip()
    if user_input.lower() == "exit":
        print("Exiting the program. Goodbye!")
        break  # Exit the loop when 'exit' is entered
    try:
        number = int(user_input)
        if number < 0:
            print("You entered a negative number.")
        elif number == 0:
            print("You entered zero.")
        else:
            print("You entered a positive number.")
    except ValueError:
        print("Invalid input. Please enter a valid number or type 'exit' to quit.")
```

Output:-	CHARMI Current letter: P Current letter: y Current letter: t Current variable name 10 Good Bye! Current variable name 9 Good Bye! Current variable name 8 Good Bye! Current variable name 7 Good Bye! Current variable name 6
Programme _47	Write a program demonstrates the use of break in a for loop iterating over a list. User inputs a number, which is searched in the list. If it is found, then the loop terminates with the 'found' message.

Code:-	# Program to demonstrate the use of the break statement in a for loop
	
	# Sample list of numbers
	numbers = [10, 20, 30, 40, 50, 60, 70, 80, 90]
	
	# User input
	print("\n--- Number Search in List ---")
	try:
	  search_number = int(input("Enter a number to search in the list: "))
	
	  # Iterating over the list
	  for num in numbers:
	  if num == search_number:
	  print(f"Number {search_number} found in the list.")
	  break # Exit the loop when the number is found
	  else:
	  print(f"Number {search_number} is not in the list.")

	except ValueError: print("Invalid input. Please enter a valid number.")
Output:-	CHARMI Enter the number to check the number if is it in list or not:4 The number 4 not found in the list ========== RESTART: C:/Users/HP/Documents/MCA/python/47.p CHARMI Enter the number to check the number if is it in list or not:23 The number 23 found in the list
	Unit- 2
Programme _01	Write a python program to learn Pyhton list methods

Code:-	print("CHARMI")
	#Create an empty list using square brackets.
	numbers=[]
	print(numbers)
	#Create an empty list using list().
	num=list()
	print(num)
	#Create list of numbers.
	num=[1,2,3]
	print(num)
	#Create list of numbers in a range.
	num=list(range(1,4))
	print(num)
	#Create a list of tuples.
	tuples_list=[(1,2),(2,4),(3,6)]
	print(tuples_list)
	#Create a list of lists
	list_list=[[1,2],[2,4],[3,6]]
	print(list_list)
	#create list with items of different datatypes
	random_list=[1,"hey",[1,2]]
	print(random_list)
	#Get length of list using len() method.
	num=[5,8,8]
	print(len(num))
	#Access elements of a list by indexing.
	str_list=["hey","there!","how","are","you?"]
	print(str_list[0])
	print(str_list[len(str_list)-1])
	print(str_list[-1])
	#Slicing a list

```
str_list=["hey","there!","how","are","you?"]
print(str_list[2:])
print(str_list[:2])
print(str_list[-3:])
print(str_list[:-3])
print(str_list[1:4])
#Get a copy list by slicing.
print(str_list[:])
#Append to a list.
num=[1,2]
print(num)
num.append(3)
print(num)
#Concatenate lists.
num=[1,2]
str=["hey","there"]
print(num+str)
#Mutate a list, that is, change its contents.
num=[1,2,3]
num[0]=100
print(num)
num[0:2]=[300,400]
print(num)
num[1:3]=[]
print(num)
num[:]=[]
print(num)
#Insert item to a list.
greeting=["how","you"]
greeting.insert(1,"are")
print(greeting)
print()
#Append item to list.
namelist=['wub_wub','RubyPinch','goldfish','Nitori']
namelist.append('pb122!')
if 'pb122!' in namelist:
    print("Now I know pb122!")
print()
#Extend item to list.
namelist=['wub_wub','RubyPinch','goldfish','Nitori']
namelist.extend('theelous3')
if input("Enter your name:") in namelist:
    print("I know you.")
else:
```

```
    print("I don't know you.")
print()
#Check item available in list.
name1='wub_wub'
name2='theelous3'
name3='RubyPinch'
name4='goldfish'
name5='Nitori'
name=input("Enter your name:")
if (name==name1) or (name==name2) or (name==name3) or (name==name4) or (name==name5):
    print("I know you")
else:
    print("Sorry,I don't know who are you?")
```

Output :-	CHARMI [] [] [1, 2, 3] [1, 2, 3] [(1, 2), (2, 4), (3, 6)] [[1, 2], [2, 4], [3, 6]] [1, 'hey', [1, 2]] 3 hey you? you? ['how', 'are', 'you?'] ['hey', 'there!'] ['how', 'are', 'you?'] ['hey', 'there!'] ['there!', 'how', 'are'] ['hey', 'there!', 'how', 'are', 'you?'] [1, 2] [1, 2, 3] [1, 2, 'hey', 'there'] [100, 2, 3] [300, 400, 3] [300] [] ['how', 'are', 'you'] Now I know pb122! Enter your name:CHARMI I don't know you. Enter your name:CHARMI Sorry,I don't know who are you?
Programme _02	Exercise 3.4 Guest list: if you could invite anyone, living or deceased, to dinner, who would you invite? Make a list that includes at least three people you'd like to invite to dinner. Then use your list to

	print a message to each person, inviting them to dinner.

Code:-	print("CHARMI")
	myGuest = []
	beLovedguest = int(input("Enter the number of guest who will deligtfully join the evening:"))
	for i in range(beLovedguest):
	  myGuest.insert(i,input("Enter our dear guest's name:"))
	print("The evening is going to be special with",myGuest,"\n")
	print("----------Greetings sent----------")
	for i in myGuest:
	  print("Dear",i,"you are heartfully invited in our evening party!")

Output :-	**CHARMI** **Enter the number of guest who will deligtfully join the evening:5** **Enter our dear guest's name:REETA** **Enter our dear guest's name:MEENA** **Enter our dear guest's name:TINA** **Enter our dear guest's name:DINA** **Enter our dear guest's name:PINA** **The evening is going to be special with ['REETA', 'MEENA', 'TINA', 'DINA', 'PINA']** **----------Greetings sent----------** **Dear REETA you are heartfully invited in our evening party!** **Dear MEENA you are heartfully invited in our evening party!** **Dear TINA you are heartfully invited in our evening party!** **Dear DINA you are heartfully invited in our evening party!** **Dear PINA you are heartfully invited in our evening party!**

	Exercise 3.5
	Changing Guest List:
	You just heard that one of your guest's can't make the dinner, so you need to send out a new set of invitations.
	You''ll have to think of someone else to invite.
	
	- Start with your program from Exercise 3.4.Add a print statement of the end of your program stating the of
	  the guest who can't make it.
	
	- Modify your list,replacing the name of the guest who can't make it with the name of the new person you
	  are inviting.
	
	- Print a second set of invitation messages, one for each person who is still in your list.

	print("CHARMI") myGuest = []

```
beLovedguest = int(input("Enter the number of guest who will deligtfully join the evening:"))
for i in range(beLovedguest):
   myGuest.insert(i,input("Enter our dear guest's name:"))
print("The evening is going to be special with",myGuest,"\n")
doubt = input("by the way, are there any guest who cannot make it to the dinner?")
if doubt.lower() == 'yes':
    Denied_Guest = int(input("Enter the number of our guest who cannot join:"))
   for i in range(Denied_Guest):
      denied_name = input("Enter the name of our guest who cannot join:")
      myGuest.remove(denied_name)
   print("Okay so only",myGuest,"will be joining?, that's disappointing....")
else:
   print("Alright, then let's plan for the evening dinner!!")
   print("And recalling Once again,our Guest's are",myGuest)

choice = input("Just to be sure,would you like to call new guest?(yes/no)")
if choice == 'yes' or choice == 'YES':
    newGuest = int(input("Enter the number of our new guest  who will join instead:"))
   for i in range(newGuest):
      new_Guestname = input("Enter the new Guest name:")
      myGuest.append(new_Guestname)
   print("Alright so the new Guest(s) which are joining are",myGuest)
else:
      print("Alright, no worries the final Guest list is here:",myGuest)
print("=================SENDING INVITATIONS===============")
for i in myGuest:
    print("Dear",i,"i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M")
```

Output:-

```
CHARMI
Enter the number of guest who will deligtfully join the evening:5
Enter our dear guest's name:ayush
Enter our dear guest's name:sujal
Enter our dear guest's name:meetish
Enter our dear guest's name:ashu
Enter our dear guest's name:kunal
The evening is going to be special with ['ayush', 'sujal', 'meetish', 'ashu', 'kunal']

by the way, are there any guest who cannot make it to the dinner?meetish
Alright, then let's plan for the evening dinner!!
And recalling Once again,our Guest's are ['ayush', 'sujal', 'meetish', 'ashu', 'kunal']
Just to be sure,would you like to call new guest?(yes/no)yes
Enter the number of our new guest  who will join instead:2
Enter the new Guest name:prince
Enter the new Guest name:dinesh
Alright so the new Guest(s) which are joining are ['ayush', 'sujal', 'meetish', 'ashu', 'kunal', 'prince', '
esh']
=================SENDING INVITATIONS===============
Dear ayush i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M
Dear sujal i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M
Dear meetish i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M
Dear ashu i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M
Dear kunal i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M
Dear prince i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M
Dear dinesh i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M
```

3.6-More_Guest:

You just found a bigger dinner table, so now more space is available.Think of three more guests to invite to dinner.

- start with your program from Exercise 3.4 or Exercise 3.5. Add a print Statement to the end of your program informing people that you found a bigger dinner table.

- Use insert() to add one new guest to the beginning of the list.
- Use insert() to add one new guest to the middle of the list.
- Use append() to add one new guest to the end of the list.
- Print a new set of invitation message,one for each person in your list.

```
print("CHARMI")
myGuest = []
beLovedguest = int(input("Enter the number of guests who will delightfully join the evening: "))
for i in range(beLovedguest):
    myGuest.append(input("Enter our dear guest's name: "))

print("The evening is going to be special with", myGuest, "\n")

doubt = input("By the way, are there any guests who cannot make it to the dinner? (yes/no) ")
if doubt.lower() == 'yes':
```

```
    Denied_Guest = int(input("Enter the number of guests who cannot join: "))
   for i in range(Denied_Guest):
      denied_name = input("Enter the name of our guest who cannot join: ")
      if denied_name in myGuest:
         myGuest.remove(denied_name)
      else:
         print(denied_name, "is not in the guest list.")
   print("Okay, so only", myGuest, "will be joining? That's disappointing....")
else:
   print("Alright, then let's plan for the evening dinner!!")
   print("And recalling once again, our guests are", myGuest)

choice = input("Just to be sure, would you like to call new guests? (yes/no) ")
if choice.lower() == 'yes':
     newGuest = int(input("Enter the number of new guests who will join instead: "))
   for i in range(newGuest):
      new_Guestname = input("Enter the new guest's name: ")
      myGuest.append(new_Guestname)
   print("Alright, so the new guest(s) joining are", myGuest)
else:
   print("Alright, no worries; the final guest list is here:", myGuest)
print("\nSir we got an update,actually we found a bigger dinner table!")
choice1 = input("Do you want to specify the VIP member ranks in the list?(yes/no)")
if choice1.lower() == 'yes':
    myGuest.insert(0, input("Enter the name of a new guest to invite at the beginning: "))  # Beginning
    myGuest.insert(len(myGuest) // 2, input("Enter the name of a new guest to invite in the middle: "))  # Middle
     myGuest.append(input("Enter the name of a new guest to invite at the end: "))  # End
   print(myGuest)
else:
   print("No worries, we should treat everyone equally!")
print("=================SENDING INVITATIONS===============")
for i in myGuest:
     print("Dear",i,"i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M")
```

Output:-	CHARMI Enter the number of guests who will delightfully join the evening: 3 Enter our dear guest's name: priya Enter our dear guest's name: riya Enter our dear guest's name: tina The evening is going to be special with ['priya', 'riya', 'tina'] By the way, are there any guests who cannot make it to the dinner? (yes/no) no Alright, then let's plan for the evening dinner!! And recalling once again, our guests are ['priya', 'riya', 'tina'] Just to be sure, would you like to call new guests? (yes/no) no Alright, no worries; the final guest list is here: ['priya', 'riya', 'tina'] Sir we got an update,actually we found a bigger dinner table! Do you want to specify the VIP member ranks in the list?(yes/no)yes Enter the name of a new guest to invite at the beginning: riya Enter the name of a new guest to invite in the middle: mahesh Enter the name of a new guest to invite at the end: chokshi ['riya', 'priya', 'mahesh', 'riya', 'tina', 'chokshi'] ==========SENDING INVITATIONS========== Dear riya i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M Dear priya i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M Dear mahesh i Charmi,want to invite you for the grand dinner in the evening at 9:00P Dear riya i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M Dear tina i Charmi,want to invite you for the grand dinner in the evening at 9:00P.M Dear chokshi i Charmi,want to invite you for the grand dinner in the evening at 9:00P
Programme _03	3-7_Shrinking_Guest_list: You just found out that your new dinner table wont arrive in time for the dinner, and you have space for only two guest -Start with your program from Exercise 3-6. Add a new line that prints a message saying that you can invite only two people for dinner -Use pop() to remove guests from your list one at a time until only two names remain in your list.Each time you pop a name from your list , print a message to that person letting them know you are sorry you can't invite them to dinner -Print a message to each of the two people still on your list ,letting them know they are still invited - Use "del" to remove the last two names from your list , so you have an empty list at the end of your program
Code:-	print("Charmi") myGuest = [] beLovedguest = int(input("Enter the number of guests who will delightfully join the evening: ")) for i in range(beLovedguest): myGuest.append(input("Enter our dear guest's name: ")) print("The evening is going to be special with", myGuest, "\n")

```
doubt = input("By the way, are there any guests who cannot make it to the dinner? (yes/no) ")
if doubt.lower() == 'yes':
    Denied_Guest = int(input("Enter the number of guests who cannot join: "))
   for i in range(Denied_Guest):
      denied_name = input("Enter the name of our guest who cannot join: ")
      if denied_name in myGuest:
         myGuest.remove(denied_name)
      else:
         print(denied_name, "is not in the guest list.")
   print("Okay, so only", myGuest, "will be joining? That's disappointing....")
else:
   print("Alright, then let's plan for the evening dinner!!")
   print("And recalling once again, our guests are", myGuest)

choice = input("Just to be sure, would you like to call new guests? (yes/no) ")
if choice.lower() == 'yes':
      newGuest = int(input("Enter the number of new guests who will join instead: "))
   for i in range(newGuest):
      new_Guestname = input("Enter the new guest's name: ")
      myGuest.append(new_Guestname)
   print("Alright, so the new guest(s) joining are", myGuest)
else:
   print("Alright, no worries; the final guest list is here:", myGuest)
print("Sir, we have a bad news,table's not gonna make in time,we can only invite two people for dinner.")
while len(myGuest) > 2:
   removed_guest = myGuest.pop()
   print("Sorry" ,removed_guest, "you can't be invited to dinner.")
for guest in myGuest:
   print("my dear",myGuest,"you are still invited to the dinner")
del myGuest[:]
print("The guest list is now empty.")
```

	Charmi Enter the number of guests who will delightfully join the evening: 3 Enter our dear guest's name: reena Enter our dear guest's name: meena Enter our dear guest's name: seema The evening is going to be special with ['reena', 'meena', 'seema'] By the way, are there any guests who cannot make it to the dinner? (yes/no) no Alright, then let's plan for the evening dinner!! And recalling once again, our guests are ['reena', 'meena', 'seema'] Just to be sure, would you like to call new guests? (yes/no) no Alright, no worries; the final guest list is here: ['reena', 'meena', 'seema'] Sir, we have a bad news,table's not gonna make in time,we can only invite two people for dinner. Sorry seema you can't be invited to dinner. my dear ['reena', 'meena'] you are still invited to the dinner my dear ['reena', 'meena'] you are still invited to the dinner The guest list is now empty.
Programme _04	3.8 - Seeing the World: Think of at least five places in the world you'd like to visit. -Store the locations in the list . Make sure the list is not in the alphabetical order. -Print your list in its original order.Dont worry about priting the list neatly Just print it as a raw python list -Use sorted() to print your list in the alphabetical order without modifying the actual list -Show that your list is still in the original order by printing it -Use sorted() to print your list in reverse alphabetical order without changing the order of the original list Show that your list is still in its original order by printing it again -Use reverse() to change the order of your list .Print the list top show that its order has changed -Use reverse() to change the order of your list again .Print the list to show it's back to its original order.
Code:-	print("CHARMI") Bucket_travelList = [] places = int(input("Enter the number of the places you wanna visit:")) for i in range(places): Bucket_travelList.insert(i,input("Enter the name of your desination ;)")) print("The raw List:",Bucket_travelList) print("=================SORTING=================") print("The List before sorting:",Bucket_travelList) print("The List after sorting:",sorted(Bucket_travelList)) print("=================REVERSING================") print("The original List before reversing:",Bucket_travelList) print("The original List after reversing in alphabetical order:",sorted(Bucket_travelList,reverse = True)) print("=================REVERSE FUNCTION===========")

	print("Before reversing:",Bucket_travelList) Bucket_travelList.reverse() print("Now after one used the reverse function",Bucket_travelList) Bucket_travelList.reverse() print("Again using the reverse function",Bucket_travelList)
	CHARMI Enter the number of the places you wanna visit:5 Enter the name of your desination ;)JAMNAGAR Enter the name of your desination ;)VADODARA Enter the name of your desination ;)SURAT Enter the name of your desination ;)RAJKOT Enter the name of your desination ;)AHEMDABAD The raw List: ['JAMNAGAR', 'VADODARA', 'SURAT', 'RAJKOT', 'AHEMDABAD'] ==========SORTING========== The List before sorting: ['JAMNAGAR', 'VADODARA', 'SURAT', 'RAJKOT', 'AHEMDABAD'] The List after sorting: ['AHEMDABAD', 'JAMNAGAR', 'RAJKOT', 'SURAT', 'VADODARA'] ==========REVERSING========== The original List before reversing: ['JAMNAGAR', 'VADODARA', 'SURAT', 'RAJKOT', 'AHEMDABAD'] The original List after reversing in alphabetical order: ['VADODARA', 'SURAT', 'RAJKOT', 'JAMNAGAR', 'A MDABAD'] ==========REVERSE FUNCTION========== Before reversing: ['JAMNAGAR', 'VADODARA', 'SURAT', 'RAJKOT', 'AHEMDABAD'] Now after one used the reverse function ['AHEMDABAD', 'RAJKOT', 'SURAT', 'VADODARA', 'JAMNAGAR'] Again using the reverse function ['JAMNAGAR', 'VADODARA', 'SURAT', 'RAJKOT', 'AHEMDABAD']
Programme _05	Pizza:- '''4-1-Pizzas:-Think of atleast three kinds of your favourite pizzas .Store this pizza name in the list, and then use a for loop to print name of each pizza *Modify your for loop to print a sentence using the name of pizza instaed just printing the name of pizza For each pizza you should have one line of output containing a simple statement like "I like pepperoni pizza *Add a line at the end of your program, outside the for loop that states how many pizzas do you like *The output should consist of three or more lines about the kind of pizza you like and then an additional sentence , such as i really love pizza'''

Code:-	print("CHARMI")
	favorite_pizzas = ['pepperoni', 'hawaiian', 'veggie']
	# Print the names of all the pizzas.
	for pizza in favorite_pizzas:
	  print(pizza)
	print("\n")
	# Print a sentence about each pizza.
	for pizza in favorite_pizzas:
	  print(f"I really love {pizza} pizza!")
	print("\nI really love pizza!")

	CHARMI pepperoni hawaiian veggie I really love pepperoni pizza! I really love hawaiian pizza! I really love veggie pizza! I really love pizza!
	4-2:Animals: Think of atleast three different animals that have a common chrachterstics .Store the names of this animal in the list and then use a for loop to print the name of each animal Modify the program to print a statement about each animal such as "a dog would make a great pet Add a line at the end of your program stating what these animals have common in . you could print a sentence stating all these animals could be a great pet'''

Code:-	print("CHARMI")
	animals = ["spider monkey", "lemur", "giraffe"]
	
	# Print each animal.
	for animal in animals:
	  print(animal)
	
	print("\n")
	
	# Print a statement about each animal.
	for animal in animals:
	  print(f"A {animal} has a long tail.")
	
	print("\nAll of these animals have long tails.")

Output:-	CHARMI spider monkey lemur giraffe A spider monkey has a long tail. A lemur has a long tail. A giraffe has a long tail. All of these animals have long tails.
Programme _06	'''4-3:Counting to Twenty:- Use a for loop to print the numbers from 1 to 20'''
Code:-	print("CHARMI") numbers = list(range(1, 21)) for number in numbers: print(number)

Output:-

```
CHARMI
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

'''4-4:One Million:
Make a list of numbers from one to one million , and then use a for loop to print the numbers.
(If output is taking so long , stop it by pressing Ctrl+C or by closing the output window'''

Code:-

```
print("CHARMI")
for i in range(1,1000000):
    print(i,end=" ")
```

Output:-

```
CHARMI
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 3
8 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103
104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 12
7 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 1
51 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174
175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 19
8 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 2
22 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245
246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 26
9 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 2
93 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316
317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 34
0 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 3
64 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387
388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 41
1 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 4
35 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458
459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 48
2 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 5
06 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529
530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 55
3 554 555 556
```

4-5:Summing a million:

	Make a list of the numbers from one to one million, and then use min(),and max() to make sure your list actually starts at one and ends at one million. Also, use the sum() functions to see how quickly python can add a million numbers.
Code:-	print("CHARMI") numbers = list(range(1, 1_000_001)) print(min(numbers)) print(max(numbers)) print(sum(numbers))
Output:-	CHARMI 1 1000000 500000500000
	'''4-6:Odd-Numbers: Use the third argument of the range() function to make a list of an odd numbers from 1 to 20 . use a for loop to print each number'''
Code:-	print("CHARMI") odd_numbers = list(range(1, 20, 2)) for number in odd_numbers: print(number)
Output:-	CHARMI 1 3 5 7 9 11 13 15 17 19
	'''4-7:Threes: Make the list of multiples of 3 from 3 to 30. Use the for loop to print the numbers in your list'''
Code:-	print("CHARMI")

	threes = list(range(3, 31, 3)) for number in threes: print(number)
Output:-	CHARMI 3 6 9 12 15 18 21 24 27 30
	'''4-8:Cubes: a number raised to the third power is called a cube . For eaxmple the cube of 2 is written as 2**3 in python . Make a list of the first 10 cubes print the cube of integer and use the for loop to print the value of each cube'''

Code:-	print("CHARMI")
	cubes = []
	for number in range(1, 11):
	  cube = number**3
	  cubes.append(cube)
	
	for cube in cubes:
	  print(cube)

Output:-	CHARMI 1 8 27 64 125 216 343 512 729 1000
	'''4-9:List-comprehension:- Use a list comprehension to generate a list of the first 10 cubes'''
Code:-	print("CHARMI") numbers = [1,2,3,4,5,6,7,8,9,10] double = [i **3 for i in numbers] print(double)
Output:-	CHARMI [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
Programme _07	'''4-10:Slicing :Using one of the program you wrote in this chapter , add several lines at the end of the program -Print the message , the first three items in the list are: Then use a slice to print the first three items from that program's list -Print the message , the first three items in the list are: Then use a slice to print the middle three items from that program's list -Print the message , the first three items in the list are: Then use a slice to print the last three items from that program's list '''
Code:-	print("CHARMI") pizza_list = [] name = int(input("Prepare your own menus of Pizza flavours,how much pizza's you'll add?\n:"))

```
for i in range(name):
    pizza_list.append(input("Enter the flavour of pizza:"))
print("The Menu is ready",pizza_list)
choice = input("n1)Beggining three\n 2)Mid three\n 3)Ending three\n Enter the list of choice you wanna print:-")
if choice == 'first' or choice == '1':
    print("The first three pizzas in the menu are:",pizza_list[0:3])
elif choice == 'second' or choice == '2':
    middle_items = len(pizza_list) // 2
    if len(pizza_list) >= 3:
        mid_index = len(pizza_list) // 2
        middle_three = pizza_list[mid_index - 1: mid_index + 2]
        print("The middle three pizzas in the menu are:",middle_three)
    else:
        print("Not enough items to display middle three.")
elif choice == 'third' or choice == '3':
    print("The last three pizzas in the menu are:",pizza_list[-3:])
else:
    print("Sir,please select an appropriate option")
```

Output:-

```
CHARMI
Prepare your own menus of Pizza flavours,how much pizza's you'll add?
:5
Enter the flavour of pizza:margreta
Enter the flavour of pizza:7 cheeze
Enter the flavour of pizza:corn capsicum
Enter the flavour of pizza:veg
Enter the flavour of pizza:mushroom
The Menu is ready ['margreta', '7 cheeze', 'corn capsicum', 'veg', 'mushroom ']
n1)Beggining three
 2)Mid three
 3)Ending three
 Enter the list of choice you wanna print:-1
The first three pizzas in the menu are: ['margreta', '7 cheeze', 'corn capsicum']
```

```
'''4-11:My pizzas, your pizzas:
Start with your program from Exercise 4.1 . make a copy of the list of the pizzas, and call it friends_pizza, and
do the following:
Add a new pizza to the original list
Add a different pizza to the list friends_pizzas.
Prove that you have to seperate lists. Print the message "my favourite pizzas are:"and then use a for loop to print the first list.
Print the message "My friends favourite pizza are:" and then use a for loop to print the second list .
Make sure each pizza is stored in the appropriate list'''
```

Code:-

```
print("CHARMI")
my_pizzaList = []
```

```
names = int(input("Enter the number of pizza flavours you add in your list:"))
for i in range(names):
    my_pizzaList.insert(i,input("Enter the name of your favourite flavour:"))
print("\n")
print("So the original pizza list of mine is:-",my_pizzaList)
friends_pizza = my_pizzaList.copy()
print("My Friend is a copy cat he has the same list",friends_pizza,"\n")
#Adding a new pizza to the orginal list
my_pizzaList.append(input("Enter the new flavour of pizza for Original List:"))
#Adding a pizza in the list of copy cat
friends_pizza.append(input("Enter the flavour of the pizza for Friends List:"))
print("\nSo the Pizza Flavours I like are:-")
for i in my_pizzaList:
    print(i,end = " | ")
print("\nAnd the Pizza Flavours my Friend like are:-")
for f in friends_pizza:
    print(f,end = " | ")
```

Output :-

```
CHARMI
Enter the number of pizza flavours you add in your list:3
Enter the name of your favourite flavour:VOLCANO
Enter the name of your favourite flavour:mushroom
Enter the name of your favourite flavour:corn


So the original pizza list of mine is:- ['VOLCANO', 'mushroom', 'corn']
My Friend is a copy cat he has the same list ['VOLCANO', 'mushroom', 'corn']

Enter the new flavour of pizza for Original List:margereta
Enter the flavour of the pizza for Friends List:corn capsicum

So the Pizza Flavours I like are:-
VOLCANO | mushroom | corn | margereta |
And the Pizza Flavours my Friend like are:-
VOLCANO | mushroom | corn | corn capsicum |
```

```
'''4-12:More-loops:All the versions of foods.py in this section have avoided using for loops when printing to save space.Choose a version of foods.py and write two for loops to print each list of food'''
```

```
print("CHARMI")
my_foods = ['pizza', 'falafel', 'carrot cake']
friend_foods = my_foods[:]

my_foods.append('cannoli')
```

	friend_foods.append('ice cream')
	
	print("My favorite foods are:")
	for food in my_foods:
	  print(f"- {food}")
	print("\nMy friend's favorite foods are:")
	for food in friend_foods:
	  print(f"- {food}")

Output:-	**CHARMI** **My favorite foods are:** **- pizza** **- falafel** **- carrot cake** **- cannoli** **My friend's favorite foods are:** **- pizza** **- falafel** **- carrot cake** **- ice cream**
Programme _08	'''1.1Write a python program to learn use of Python Tuple methods'''

Code:-	print("CHARMI")
	#Using the Tuple count()method
	#Creating tuples
	Tuple1 = (0,1,2,3,2,3,1,3,2)
	Tuple2 = ('java','java','python','geek','python','for','java','python')
	#Count the appearence of 3
	res = Tuple1.count(3)
	print("Count of 3 in Tuple 1 is:",res,"times.")
	#Count the appearence of python
	res1 = Tuple2.count('python')
	print("The Count of Python is:",res1,"times.")

	CHARMI **Count of 3 in Tuple 1 is: 3 times.** **The Count of Python is: 3 times.**
	'''1-2:Counting Tuples and lists as Elements in Tuple'

	print("CHARMI")
	#Using the Tuple count()method
	#Creating tuples
	Tuple1 = (0,1,2,3,2,3,1,3,2)
	Tuple2 = ('java','java','python','geek','python','for','java','python')
	#Count the appearence of 3
	res = Tuple1.count(3)
	print("Count of 3 in Tuple 1 is:",res,"times.")
	#Count the appearence of python
	res1 = Tuple2.count('python')
	print("The Count of Python is:",res1,"times.")

	= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/t2.py CHARMI Count of 3 in Tuple 1 is: 3 times. The Count of Python is: 3 times.
	'''1-3:Indexes:Using Tuple Index()method''''

	#Creating Tuple
	print("CHARMI")
	Tuple = (0,1,2,3,2,3,1,3,2)
	#Getting the index of 3
	res = Tuple.index(3)
	print("First Occurence of 3 is",res)
	#Getting the index of 3 after 4th index
	res = Tuple.index(3,4)
	print("First Occurence of 3 after 4th index is",res)

Output:-	CHARMI First Occurence of 3 is 3 First Occurence of 3 after 4th index is 5
Programme _09	'''4-13:Buffet- A buffet stye restraunt offers only five basic foods . Think of 5 simple foods and store them in a tuple. Use the for loop to print each food the restraunt offers Try to modify one of the items , and make sure python rejects the change The Restraunt chnages it Menu , relacing two of the items with different foods . Add the block of code that rewrites the tuple , and then use a for loop to print each of the items on the revised menu.'''
Code:-	print("CHARMI") menu_items = ('rockfish sandwich', 'halibut nuggets', 'smoked salmon chowder',

```
    'salmon burger', 'crab cakes',
    )

print("You can choose from the following menu items:")
for item in menu_items:
    print(f"- {item}")

menu_items = (
    'rockfish sandwich', 'halibut nuggets', 'smoked salmon chowder',
    'black cod tips', 'king crab legs',
    )

print("\nOur menu has been updated.")
print("You can now choose from the following items:")
for item in menu_items:
    print(f"- {item}")
```

Output:-

```
CHARMI
You can choose from the following menu items:
- rockfish sandwich
- halibut nuggets
- smoked salmon chowder
- salmon burger
- crab cakes

Our menu has been updated.
You can now choose from the following items:
- rockfish sandwich
- halibut nuggets
- smoked salmon chowder
- black cod tips
- king crab legs
```

Programme_10

```
'''2-1:Write a python program to learn use of Python Dictionary Methods'''
```

Code:-

```
print("CHARMI")
#Creating Dictionary
dictionary = {}
dictionary = {'Name':'charmi','Sex':'FeMale','Age':22}
print(dictionary)
```

```
print(type(dictionary))
#Accesing the elements in the dictionary
dictionary = {'Name':'charmi','Sex':'FeMale','Age':22}
print(dictionary['Name'])
print(dictionary['Sex'])
print(dict.keys(dictionary))
#dictionary['Height']
print(dictionary.get('Age'))
#Removing the elements from the dictionary
dictionary1 = {'Name':'Din Djarin','Age':'28','Sex':'Male','Height':5.11,'Occupation':'Bounty Hunter'}
print(dictionary1)
#Deleting an Element
del dictionary1['Name']
print(dictionary1)
#Popping an item
print(dictionary1.popitem())
#Popping the value
print(dictionary1.pop('Sex'))
print(dictionary1)
#Clearing the entire dictionary
dictionary1.clear()
print(dictionary1)
#Using a for loop we can iterate through each key in the dictionary
dictionary2 = {'Name':'Obi Wan','Age':'28','Sex':'Male','Height':5.11,'Occupation':'Rescue'}
#Iterating through the values
for i in dictionary2:
   print(dictionary2[i])
#Iterating through the keys
for i in dictionary2:
   print(i)
#The all () in the dictionary returns "True" if all keys of the dictionary are true(or if the dictionary is empty)
#If the key in the dictionary are true or if the dictionary all method retuerns true or else it will return false
dictionary3 = {0:'Sukuna',1:'Kyojiro Rengoku'}
print(dictionary3 )
print(all(dictionary3 ))
dictionary4 = {1:'Sukuna',1:'Kyojiro Rengoku'}
print(all(dictionary4 ))
dictionary5 = {}
print(all(dictionary5 ))
```

```
#The any()in dictionary return "True" if any key of the dictionary is true. If the dictionary is empty it will return "False"
dictionary6 = {0:'Din Djarn',1:'Obi-wan:Kenobi'}
print(dictionary6)
print(any(dictionary6))
dictionary7 = {0:'Din Djarn',0:'Obi-wan:Kenobi'}
print(any(dictionary6))
#0 is False
#1 is True
dictionary8 = {0:'Steve'}
print(any(dictionary8))
#The length() method in the dictionary returns the length of the dictionary (obviosuly)
#It returns the number of the items in the dictionary
dictionary9 = {'Name':'Stephen Grant','Age':27,'Sex':'Male','Height':5.12,'Occupation':'Moon Knight'}
print(dictionary9)
print(len(dictionary9))
#The sorted method in the dictionary returns a new sorted list of keys in the dictionary
dictionary10 = {'Name':'Albert','Age':28,'Sex':'Male','Heigth':5.12,'Ocuupation':'WerewolfByNight'}
print(dictionary10)
#Sorting in acending order
print(sorted(dictionary10))
#Sorting in decending order
print(sorted(dictionary10,reverse = True))
#Copy(As the name suggests the copy method in the dictionary returns a copy of the dictionary)
dictionary11 = {'Name':'Stephen Grant','Age':27,'Sex':'Male','Height':5.12,'Occupation':'Moon Knight'}
print(dictionary11)
dictionary12 = dictionary11.copy()
print(dictionary12)
#Python dictonary method key() returns a list of all the available keys in the dictionary
d13 = {'Name':'Albert','Age':28,'Sex':'Male','Heigth':5.12,'Ocuupation':'WerewolfByNight'}
print(d13)
print(d13.keys())
#Python dictonary method values() returns a list of all the available vaues in the given dictionary
```

	d14 = {'Name':'Din Djarin','Age':'28','Sex':'Male','Height':5.11,'Occupation':'Bounty Hunter'} print(d14) print(d14.values())
	CHARMI {'Name': 'charmi', 'Sex': 'FeMale', 'Age': 22} <class 'dict'> charmi FeMale dict_keys(['Name', 'Sex', 'Age']) 22 {'Name': 'Din Djarin', 'Age': '28', 'Sex': 'Male', 'Height': 5.11, 'Occupation': 'Bounty Hunter'} {'Age': '28', 'Sex': 'Male', 'Height': 5.11, 'Occupation': 'Bounty Hunter'} ('Occupation', 'Bounty Hunter') Male {'Age': '28', 'Height': 5.11} {} Obi Wan 28 Male 5.11 Rescue Name Age Sex Height Occupation {0: 'Sukuna', 1: 'Kyojiro Rengoku'} False True True {0: 'Din Djarn', 1: 'Obi-wan:Kenobi'} True True False {'Name': 'Stephen Grant', 'Age': 27, 'Sex': 'Male', 'Height': 5.12, 'Occupation': 'Moon Knight'} 5 {'Name': 'Albert', 'Age': 28, 'Sex': 'Male', 'Heigth': 5.12, 'Ocuupation': 'WerewolfByNight'} ['Age', 'Heigth', 'Name', 'Ocuupation', 'Sex'] ['Sex', 'Ocuupation', 'Name', 'Heigth', 'Age'] {'Name': 'Stephen Grant', 'Age': 27, 'Sex': 'Male', 'Height': 5.12, 'Occupation': 'Moon Knight'} {'Name': 'Stephen Grant', 'Age': 27, 'Sex': 'Male', 'Height': 5.12, 'Occupation': 'Moon Knight'} {'Name': 'Albert', 'Age': 28, 'Sex': 'Male', 'Heigth': 5.12, 'Ocuupation': 'WerewolfByNight'} dict_keys(['Name', 'Age', 'Sex', 'Heigth', 'Ocuupation']) {'Name': 'Din Djarin', 'Age': '28', 'Sex': 'Male', 'Height': 5.11, 'Occupation': 'Bounty Hunter'} dict_values(['Din Djarin', '28', 'Male', 5.11, 'Bounty Hunter'])
Programme _11	'''6-1:-Persons: Use a dictionary to store information about a person you know. Store their first name,Last name, age and the city they live . You should have keys such as first_name,last_name, age and the city.Print each piece of information stored in your dictionary'''
Code:-	print("CHARMI") MoonKnight = {'first_name':'Marc','last_name':'Spectre','Age':28,'City':'Mirror Dimension'} print(MoonKnight) for i in MoonKnight: print(i,":",MoonKnight[i])

	CHARMI {'first_name': 'Marc', 'last_name': 'Spectre', 'Age': 28, 'City': 'Mirror Dimension'} first_name : Marc last_name : Spectre Age : 28 City : Mirror Dimension
	'''6-2:Favourite-Numbers: Use a dictionary to store people;s favourite numbers. Think of five names and use them as keys in your dictionary . Think of a favourite number for each person, and store each as value in your dictionary . Print each person's name and their favourite number . For even more fun , poll a few friends and get some actual data for your program'''

	print("CHARMI")
	Favourite_dict = {}
	while True:
	  key = input("Enter the key value or (type 'exit' to end):")
	  if key.lower() == 'exit':
	  break
	  value = input("Enter the value for the key:")
	  Favourite_dict[key] = value
	print("So the Favourite numbers dictionary are:",Favourite_dict)

	CHARMI Enter the key value or (type 'exit' to end):charmi Enter the value for the key:168 Enter the key value or (type 'exit' to end):bhumika Enter the value for the key:256 Enter the key value or (type 'exit' to end):priya Enter the value for the key:856 Enter the key value or (type 'exit' to end):end Enter the value for the key:456 Enter the key value or (type 'exit' to end):exit So the Favourite numbers dictionary are: {'charmi': '168', 'bhumika': '256', 'priya': '856', 'end': '456'}
	'''6-3:Glossary- A python dictionary can be used to model an actual dictionary , However to avoid confusion ,let's call it glossary. Think of five programming words you've learned about in the previous chapters . Use these words as the key in your glossary , and store their meanings as values .

	Print each words and it meanings as neatly formatted output . You might print the word followed by a colon and then it's meaning , or print the word on one line and then print it's meaning indented on a second line . Use the new line chrachter (\n)to insert a blank line between each-word meaning pair in your output'''

	print("CHARMI")
	Glossary = {}
	while True:
	word = input("Enter the Programming related word or(type exit to end):")
	if word.lower() == 'exit':
	break
	meaning = input("Enter that word's defination:")
	Glossary[word] = meaning
	for i in Glossary:
	print(i,":",Glossary[i],"\t")
	print("Your own lab made dictionary is:",Glossary)

	CHARMI Enter the Programming related word or(type exit to end):PYTHON Enter that word's defination:it is easy to understand language Enter the Programming related word or(type exit to end):tkinter Enter that word's defination:it is used in gui in python Enter the Programming related word or(type exit to end):exit PYTHON : it is easy to understand language tkinter : it is used in gui in python Your own lab made dictionary is: {'PYTHON': 'it is easy to understand language', 'tkinter': 'it is used i ui in python'}
Programme _12	'''6-4:Glossary2: Now that you know how to loop through a dictionary , clean up the code from Exercise6-3(page 102)by replacing your series of print statements with a loop that runs through the dictionary keys and values .When you're sure that your loop works , add five more python terms to your glossary . when you run your program again , these new words and meanings should automatically be included in the output'''
Code:-	glossary = { 'string': 'A series of characters.', 'comment': 'A note in a program that the Python interpreter ignores.', 'list': 'A collection of items in a particular order.', 'loop': 'Work through a collection of items, one at a time.', 'dictionary': "A collection of key-value pairs.", 'key': 'The first item in a key-value pair in a dictionary.', 'value': 'An item associated with a key in a dictionary.',

```
    'conditional test': 'A comparison between two values.',
    'float': 'A numerical value with a decimal component.',
    'boolean expression': 'An expression that evaluates to True or False.',
    }
for word, definition in glossary.items():
    print(f"\n{word.title()}: {definition}")
```

CHARMI

String: A series of characters.

Comment: A note in a program that the Python interpreter ignores.

List: A collection of items in a particular order.

Loop: Work through a collection of items, one at a time.

Dictionary: A collection of key-value pairs.

Key: The first item in a key-value pair in a dictionary.

Value: An item associated with a key in a dictionary.

Conditional Test: A comparison between two values.

Float: A numerical value with a decimal component.

Boolean Expression: An expression that evaluates to True or False.

```
'''6-5:Rivers:
Make a dictionary containing three major rivers and the country each river runs through.One key-value pair might be 'Nile':'Egypt'.
-Use a loop to print a sentence about each river , such as a Nuile runs through Egypt
-Use a loop to print the name of each river included in the dictionary
-Use a loop to print the name of each country included in the dictionary '''
```

```
print("CHARMI")
rivers = {
    'nile': 'egypt',
    'mississippi': 'united states',
    'fraser': 'canada',
    'kuskokwim': 'alaska',
    'yangtze': 'china',
    }
for river, country in rivers.items():
    print(f"The {river.title()} flows through {country.title()}.")
print("\nThe following rivers are included in this data set:")
for river in rivers.keys():
    print(f"- {river.title()}")
print("\nThe following countries are included in this data set:")
```

```
for country in rivers.values():
    print(f"- {country.title()}")
```

```
CHARMI
The Nile flows through Egypt.
The Mississippi flows through United States.
The Fraser flows through Canada.
The Kuskokwim flows through Alaska.
The Yangtze flows through China.

The following rivers are included in this data set:
- Nile
- Mississippi
- Fraser
- Kuskokwim
- Yangtze

The following countries are included in this data set:
- Egypt
- United States
- Canada
- Alaska
- China
```

```
'''6-6:Polling:
Use the code in favourite_languages.py(Page 104)
-Make a list of people who should take the favourite languages poll. Include some names that are already in the dictionary and some that are not
-Loop through the list of people who should take the poll. If they have already taken the poll, print a message thanking them for responding.
If they have not yet taken the poll , Print  a message inviting them to take the poll.'''
```

```
print("CHARMI")
favorite_languages = {
    'jen': 'python',
    'sarah': 'c',
    'edward': 'ruby',
    'phil': 'python',
    }
for name, language in favorite_languages.items():
    print(f"{name.title()}'s favorite language is {language.title()}.")
print("\n")
coders = ['phil', 'josh', 'david', 'becca', 'sarah', 'matt', 'danielle']
```

	for coder in coders:
	  if coder in favorite_languages.keys():
	  print(f"Thank you for taking the poll, {coder.title()}!")
	  else:
	  print(f"{coder.title()}, what's your favorite programming language?")

	CHARMI **Jen's favorite language is Python.** **Sarah's favorite language is C.** **Edward's favorite language is Ruby.** **Phil's favorite language is Python.** **Thank you for taking the poll, Phil!** **Josh, what's your favorite programming language?** **David, what's your favorite programming language?** **Becca, what's your favorite programming language?** **Thank you for taking the poll, Sarah!** **Matt, what's your favorite programming language?** **Danielle, what's your favorite programming language?** |
Programme _13	'''Pizza-Toppings: Write a loop that prompts the user to enter the series of pizza toppings until they enter a 'quit' value. As they enter each toppings print a message saying you'll add that toppings to their pizza '''

Code:-	print("CHARMI")
	toppings = []
	while True:
	  topping = input("Enter a pizza topping(or type ' quit' to finish):")
	  if topping.lower()=='quit':
	  print("Exiting the topping selection")
	  break
	  toppings.append(topping)
	for topping in toppings:
	  print("i will add",topping,"to your pizza as toppings")

	CHARMI Enter a pizza topping(or type ' quit' to finish):panner Enter a pizza topping(or type ' quit' to finish):mushroom Enter a pizza topping(or type ' quit' to finish):olive Enter a pizza topping(or type ' quit' to finish):cheeze Enter a pizza topping(or type ' quit' to finish):corn Enter a pizza topping(or type ' quit' to finish):quit Exiting the topping selection i will add panner to your pizza as toppings i will add mushroom to your pizza as toppings i will add olive to your pizza as toppings i will add cheeze to your pizza as toppings i will add corn to your pizza as toppings
Programme _14	'''Movie-Tickets: A movie theatre charges diffrent ticket prices depending on a person's age. If a person is under the age of 3, the ticket is free; if they are between 3 to 12 the ticket is 10$; and if they are over age 12, the ticket is 15$.Write a loop in which you ask users their age ,and tell them the cost of their movie ticket'''

Code:-	print("CHARMI")
	while True:
	age = int(input("Enter the age of the person(or type 0 to exit):"))
	if age<0:
	print("Age cannot be neagtive, please enter a valid age!")
	else:
	if age==0:
	print("Exiting the ticket price calculator")
	break
	elif age <=3:
	print("Ohh, for the small child the ticket is free of cost,Enjoy!!")
	elif age >=3 and age <=12:
	print("Your ticket cost is 10$")
	else:
	print("Your ticket cost is 15$")

	CHARMI Enter the age of the person(or type 0 to exit):18 Your ticket cost is 15$ Enter the age of the person(or type 0 to exit):25 Your ticket cost is 15$ Enter the age of the person(or type 0 to exit):9 Your ticket cost is 10$ Enter the age of the person(or type 0 to exit):65 Your ticket cost is 15$ Enter the age of the person(or type 0 to exit):0 Exiting the ticket price calculator
Programme _15	Write a Python Program to learn use of Python regular functions.

Code:-	import re
	def main():
	  # Input string for demonstration
	  text = "The rain in Spain stays mainly in the plain."
	  # 1. Search for a word in a string
	  search_pattern = r"rain"
	  search_result = re.search(search_pattern, text)
	  if search_result:
	  print(f"'{search_pattern}' found at position {search_result.start()} in the text.")
	  else:
	  print(f"'{search_pattern}' not found in the text.")
	  # 2. Find all matches of a pattern
	  findall_pattern = r"in"
	  findall_results = re.findall(findall_pattern, text)
	  print(f"'{findall_pattern}' appears {len(findall_results)} times: {findall_results}")
	  # 3. Match a pattern at the beginning of the string
	  match_pattern = r"The"
	  match_result = re.match(match_pattern, text)
	  if match_result:
	  print(f"'{match_pattern}' matches the beginning of the text.")
	  else:
	  print(f"'{match_pattern}' does not match the beginning of the text.")
	  # 4. Replace parts of the string using a pattern
	  replace_pattern = r"Spain"

```
    replacement = "France"
    replaced_text = re.sub(replace_pattern, replacement, text)
    print(f"Replaced text: {replaced_text}")
    # 5. Split a string using a pattern
    split_pattern = r"\s"  # Split by whitespace
    split_result = re.split(split_pattern, text)
    print(f"Splitting the text into words: {split_result}")
    # 6. Demonstrate usage of groups
    group_pattern = r"(rain) (in)"
    group_result = re.search(group_pattern, text)
    if group_result:
        print(f"Found groups: {group_result.groups()}")
    else:
        print(f"No groups found using the pattern '{group_pattern}'.")
if __name__ == "__main__":
    main()
```

```
= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/refunc.py
'rain' found at position 4 in the text.
'in' appears 6 times: ['in', 'in', 'in', 'in', 'in', 'in']
'The' matches the beginning of the text.
Replaced text: The rain in France stays mainly in the plain.
Splitting the text into words: ['The', 'rain', 'in', 'Spain', 'stays', 'mainly', 'in', 'the', 'plain.']
Found groups: ('rain', 'in')
```

Programme _16

Using the Lambda function

Code:-

```
print("CHHARMI")
'''-------#Lambda function----------'''
greet = lambda:print("Hello World")
greet()
'''Python lambda function with an argument lambda that accepts one argument'''
greet_user = lambda name :print("Hey there",name)
#lambda call
greet_user("charmi")
numbers = [1,3,5,7,9]
double_result = map(lambda x : x * 2,numbers)
print(list(double_result))

#Function defination that takes one argunment , and that argument can be multiplied with an unknown number
def myfunc(n):
    return lambda a,b: a*b * n #Here n was taken statically as two
num = int(input("Enter the number:"))
mydoubler = myfunc(num)
print(mydoubler(11,22))
```

	= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/lamdafunc.py CHHARMI Hello World Hey there charmi soni [2, 6, 10, 14, 18] Enter the number:
Programme _17	Functions:- 1)Find the factorial of a number by defining a function

	print("CHARMI")
	num = int(input("Enter the number to find the factorial of the number:"))
	def factorial(num):
	  if num==0 or num==1:
	  return 1
	  else:
	  return num * factorial(num-1)
	print("The Factorial of",num,"is",factorial(num))

Output:-	CHARMI Enter the number to find the factorial of the number:7 The Factorial of 7 is 5040
	2)Find the Fibonacci series by defining a function using recursive function

	print("CHARMI")
	def Fibonnaci(num):
	  if num<=1:
	  return num
	  else:g
	  return Fibonnaci(num-1) + Fibonnaci(num-2)
	num = int(input("Enter the number to print the Fibonnaci Series:"))
	def Fibo_series(num):
	  print("The Fibonnaci series printed using recursive function is:")
	  for i in range(num):
	  print(Fibonnaci(i),end = " ")
	Fibo_series(num)

	CHARMI Enter the number to print the Fibonnaci Series:15 The Fibonnaci series printed using recursive function is: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
Programme _18	Write a python program to print the fibonacci series upto n_terms using recursion.

Code:-	def fibonacci(n):
	  """Recursive function to calculate the nth Fibonacci number."""
	  if n <= 0:
	  return 0
	  elif n == 1:
	  return 1
	  else:
	  return fibonacci(n - 1) + fibonacci(n - 2)
	
	def print_fibonacci_series(n_terms):
	  """Prints the Fibonacci series up to n_terms."""
	  if n_terms <= 0:
	  print("Please enter a positive integer.")
	  else:
	  print(f"Fibonacci series up to {n_terms} terms:")
	  for i in range(n_terms):
	  print(fibonacci(i), end=" ")
	
	# Driver code
	if __name__ == "__main__":
	  # Input from user
	  n_terms = int(input("Enter the number of terms: "))
	  print_fibonacci_series(n_terms)

	CHARMI Enter the number to find the factorial of the number:7 The Factorial of 7 is 5040
Programme _19	The factorial of 6 is denoted as 6! = 1*2*3*4*5*6 = 720
Code:-	def factorial_iterative(n):

```
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

def factorial_recursive(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial_recursive(n - 1)

if __name__ == "__main__":
    number = int(input("Enter a number to calculate its factorial: "))

    if number < 0:
        print("Factorial is not defined for negative numbers.")
    else:
        print(f"Factorial of {number} using iteration: {factorial_iterative(number)}")
        print(f"Factorial of {number} using recursion: {factorial_recursive(number)}")
```

```
CHARMI
Enter the number to print the Fibonnaci Series:15
The Fibonnaci series printed using recursive function is:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Programme _20

Find the length of the list and simply swap the first element with (n-1)th element.

Code:-

```
def swap_first_last(lst):
    # Find the length of the list
    n = len(lst)
    # Check if the list has at least two elements to swap
    if n < 2:
        print("List is too short to swap.")
        return lst
    # Swap the first and (n-1)th elements
    lst[0], lst[n - 1] = lst[n - 1], lst[0]
    return lst
if __name__ == "__main__":
    # Input list
    my_list = [1, 2, 3, 4, 5]
    print("Original list:", my_list)
    # Perform the swap
    swapped_list = swap_first_last(my_list)
```

	print("List after swapping first and last elements:", swapped_list)
	CHARMI Enter the number of element you wanna enter:9 Enter the number:8 Enter the number:7 Enter the number:6 Enter the number:5 Enter the number:4 Enter the number:3 Enter the number:2 Enter the number:1 Enter the number:5 The Original before Swapping is: ['8', '7', '6', '5', '4', '3', '2', '1', '5'] and the lenght of the list is: 9 meanwhile the Original list after swapping is ['5', '7', '6', '5', '4', '3', '2', '1', '8']
Programme _21	8-1. Message: Write a function called display_message() that prints one sen tence telling everyone what you are learning about in this chapter . Call the function, and make sure the message displays correctly
Code:-	
	8-2. Favorite Book: Write a function called favorite_book() that accepts one parameter, title . The function should print a message, such as One of my favorite books is Alice in Wonderland . Call the function, making sure to include a book title as an argument in the function call
	8-3. T-Shirt: Write a function called make_shirt() that accepts a size and the text of a message that should be printed on the shirt . The function should print a sentence summarizing the size of the shirt and the message printed on it . Call the function once using positional arguments to make a shirt . Call the function a second time using keyword arguments
	8-4. Large Shirts: Modify the make_shirt() function so that shirts are large by default with a message that reads I love Python . Make a large shirt and a medium shirt with the default message, and a shirt of any size with a different message
	8-5. Cities: Write a function called describe_city() that accepts the name of a city and its country . The function should print a simple sentence, such as Reykjavik is in Iceland . Give the parameter for the country a default value . Call your function for three different cities, at least one of which is not in the default country

	8-6. City Names: Write a function called city_country() that takes in the name of a city and its country . The function should return a string formatted like this: "Santiago, Chile" Call your function with at least three city-country pairs, and print the value that's returned
	8-7. Album: Write a function called make_album() that builds a dictionary describing a music album . The function should take in an artist name and an album title, and it should return a dictionary containing these two pieces of information . Use the function to make three dictionaries representing different albums . Print each return value to show that the dictionaries are storing the album information correctly . Add an optional parameter to make_album() that allows you to store the number of tracks on an album . If the calling line includes a value for the num ber of tracks, add that value to the album's dictionary . Make at least one new function call that includes the number of tracks on an album
	8-8. User Albums: Start with your program from Exercise 8-7 . Write a while loop that allows users to enter an album's artist and title . Once you have that information, call make_album() with the user's input and print the dictionary that's created . Be sure to include a quit value in the while loop
Programme _22	8-9. Magicians: Make a list of magician's names . Pass the list to a function called show_magicians(), which prints the name of each magician in the list
Code:-	
	8-10. Great Magicians: Start with a copy of your program from Exercise 8-9 . Write a function called make_great() that modifies the list of magicians by add ing the phrase the Great to each magician's name . Call show_magicians() to see that the list has actually been modified
	8-11. Unchanged Magicians: Start with your work from Exercise 8-10 . Call the function make_great() with a copy of the list of magicians' names . Because the original list will be unchanged, return the new list and store it in a separate list . Call show_magicians() with each list to show that you have one list of the origi nal names and one list with the Great added to each magician's name.

1	Python program to write data to file

Code:	F=open("drinks.dat","w")
	while(True):
	  v=input("Enter Drink Name : ")
	  if(v==""):
	  break
	  F.write(v+"\n")
	F.close()

Output:	Enter Drink Name : pepsi Enter Drink Name : coc Enter Drink Name :
2	Python program to create a new file in another directory

Code:	import os
	
	def create_file_in_another_directory():
	  # Path to the new directory where you want to create the file
	  directory_path = "otherDirectory"
	
	  # Check if the directory exists; if not, create it
	  if not os.path.exists(directory_path):
	  os.mkdir(directory_path)
	
	  # Path to the new file in the specified directory
	  file_path = os.path.join(directory_path, "newfile.txt")
	
	  # Create the new file and write some content to it
	  with open(file_path, "w") as file:
	  file.write("This is a new file created in another directory.")
	
	  print(f"File created at: {file_path}")
	
	if __name__ == "__main__":
	  create_file_in_another_directory()

Output:	File created at: otherDirectory\newfile.txt
3	Append content to a file in Python

Code:	F=open("data.dat","a")
	while(True):
	  id=input("Enter Id:")
	  name=input("Enter Name:")
	  salary=input("Enter Salary:")

	data="{0},{1},{2}\n".format(id,name,salary) F.write(data) ch=input("Continue y/n?") if(ch=="n"):break F.close()
Output:	Enter Id:1 Enter Name:cm Enter Salary:450 Continue y/n?y Enter Id:2 Enter Name:ck Enter Salary:45863 Continue y/n?n
4	Read contents of the file using readline() method in Python

Code:	def read_file_using_readline():
	  # File path (change to your file's path if needed)
	  file_path = "example.txt"
	
	  # Create and write sample content to the file (for demonstration)
	  with open(file_path, "w") as file:
	  file.write("This is the first line.\n")
	  file.write("This is the second line.\n")
	  file.write("This is the third line.\n")
	
	  # Open the file for reading
	  with open(file_path, "r") as file:
	  print("Reading file contents using readline():")
	
	  # Read lines one by one
	  while True:
	  line = file.readline()
	  if not line: # Stop when no more lines are left
	  break
	  print(line.strip()) # Remove newline characters for cleaner output
	
	if __name__ == "__main__":
	  read_file_using_readline()

Output:	Reading file contents using readline(): This is the first line. This is the second line. This is the third line.

5	Read contents of a file using readline() method and manipulating it in Python

Code:	F=open("data.dat","r")
	while(True):
	  data=F.readline()
	  if(data==""):break
	  DL=data.split(",")
	  DL[2]=DL[2].rstrip("\n")
	  DL.append(int(DL[2])*20/100)
	  print(DL)
	F.close()

Output:	['1', 'cm', '1456', 291.2] ['1', 'cm', '450', 90.0] ['2', 'ck', '45863', 9172.6]
6	Read contents of the file using readlines() method in Python

Code:	def read_file_using_readlines():
	  # File path (change to your file's path if needed)
	  file_path = "example.txt"
	
	  # Create and write sample content to the file (for demonstration)
	  with open(file_path, "w") as file:
	  file.write("This is the first line.\n")
	  file.write("This is the second line.\n")
	  file.write("This is the third line.\n")
	
	  # Open the file for reading
	  with open(file_path, "r") as file:
	  print("Reading file contents using readlines():")
	
	  # Read all lines into a list
	  lines = file.readlines()
	
	  # Iterate over the list and print each line
	  for line in lines:
	  print(line.strip()) # Remove newline characters for cleaner output
	
	if __name__ == "__main__":
	  read_file_using_readlines()

Output:	Reading file contents using readlines(): This is the first line. This is the second line. This is the third line.
7	Check if the record is present in the file using its id in Python

Code:	F=open("data.dat","r")
	
	id=input("Enter Id:")
	
	found=False
	
	while(True):
	  data=F.readline()
	  if(data==""):
	  break
	
	  DL=data.split(",")
	  if(DL[0]==id):
	  DL[2]=DL[2].rstrip("\n")
	  DL.append(int(DL[2])*20/100)
	  print(DL)
	  found=True
	  break
	if(not found):
	  print("Record Not Found")
	
	F.close()

Output:	Enter Id:12 Record Not Found
8	Copy contents from one file to another file in Python

Code:	sfile=input("Enter Source File:")
	try:
	  sf=open(sfile,"rb")
	  tfile = input("Enter Target File:")
	  tf=open(tfile,"wb")
	  tf.write(sf.read())
	  sf.close()
	  tf.close()
	  print("File Copied...")
	except FileNotFoundError as e:

	print(e)
Output:	Enter Source File:source.txt [Errno 2] No such file or directory: 'source.txt'
9	Copy odd lines of one file to another file in Python

Code:	# opening the file
	file1 = open('file1.txt', 'r')
	
	# creating another file to store odd lines
	file2 = open('file2.txt', 'w')
	
	# reading content of the files
	# and writing odd lines to another file
	lines = file1.readlines()
	type(lines)
	for i in range(0, len(lines)):
	  if(i % 2 != 0):
	  file2.write(lines[i])
	
	# closing the files
	file1.close()
	file2.close()
	
	# opening the files and printing their content
	file1 = open('file1.txt', 'r')
	file2 = open('file2.txt', 'r')
	
	# reading and printing the files content
	str1 = file1.read()
	str2 = file2.read()
	
	print("file1 content...")
	print(str1)
	
	print() # to print new line
	
	print("file2 content...")
	print(str2)
	
	# closing the files
	file1.close()
	file2.close()

Output:	Enter the source file name: source.txt Enter the target file name: target.txt Error: [Errno 2] No such file or directory: 'source.txt'
10	Count the total number of uppercase characters in a file in Python

Code:	try:
	  upperCount =0
	  F=open("names.dat","r")
	  while(True):
	  data=F.read(1)
	
	  if(data==""):
	  break
	  if (ord(data) >= 65 and ord(data) <= 90):
	  upperCount = upperCount +1
	  print(data,end='')
	
	  print("Total Upper Case:",upperCount)
	except FileNotFoundError as e:
	  print(e)
	finally:
	  F.close()

Output:	[Errno 2] No such file or directory: 'names.dat'
11	Python program to count total number of uppercase and lowercase characters in file

Code:	# Program to count total number of
	# uppercase and lowercase characters in file
	# Start of try block
	try:
	  #Counter for characters...
	  upperCount = 0
	  lowerCount = 0
	  F=open("file.dat","r")
	  while(True):
	  data=F.read(1)
	  if(data==""):
	  break
	  if (data.isupper()):
	  upperCount = upperCount + 1
	  elif (data.islower()):
	  lowerCount = lowerCount + 1

	print(data,end='') print("Total Upper Case:",upperCount) print("Total lower Case:",lowerCount) except FileNotFoundError as e: print(e) finally: F.close()
Output:	Error: [Errno 2] No such file or directory: 'file.dat'
12	Setting file offsets in Python

Code:	# creating a file
	f = open('file1.txt', 'w')
	
	# writing content to the file
	# first line
	f.write('This is line1.\n')
	# second line
	f.write('This is line2.\n')
	#third line
	f.write('This is line3.\n')
	
	# closing the file
	f.close()
	
	# now, reading operations ....
	# openingthe file
	f = open('file1.txt', 'r')
	# reading 10 characters
	str = f.read(10);
	print('str: ', str)
	
	# Check current offset/position
	offset = f.tell();
	print('Current file offset: ', offset)
	
	# Reposition pointer at the beginning once again
	offset = f.seek(0, 0);
	# reading again 10 characters
	str = f.read(10);
	print('Again the str: ', str)
	
	# closing the file

	f.close()
Output:	str: This is li Current file offset: 10 Again the str: This is li
13	Read a program from another file in Python

Code:	def read_program_file(source_file):
	  try:
	  # Open the source file in read mode
	  with open(source_file, "r") as file:
	  print("Reading contents of the file:")
	  # Read and print the contents of the file
	  for line in file:
	  print(line, end="") # Use end="" to avoid double newlines
	  except FileNotFoundError:
	  print(f"Error: The file '{source_file}' was not found.")
	  except Exception as e:
	  print(f"An unexpected error occurred: {e}")
	# Example usage
	file_name = input("Enter the file name to read the program from: ")
	read_program_file(file_name)

Output:	Enter the file name to read the program from: example_program.py Error: The file 'example_program.py' was not found.
14	Python program to delay printing of lines from a file using sleep function

Code:	import time
	def delayed_print(file_name, delay):
	  try:
	  # Open the file in read mode
	  with open(file_name, "r") as file:
	  print(f"Reading and printing lines from '{file_name}' with a delay of {delay} seconds:\n")
	  # Read and print each line with a delay
	  for line in file:
	  print(line.strip()) # Remove extra newlines for cleaner output
	  time.sleep(delay) # Introduce delay between lines
	  except FileNotFoundError:
	  print(f"Error: The file '{file_name}' was not found.")
	  except Exception as e:
	  print(f"An unexpected error occurred: {e}")
	# Example usage
	file_name = input("Enter the file name to read: ")
	delay = float(input("Enter the delay between lines (in seconds): "))

	delayed_print(file_name, delay)
Output:	Enter the file name to read: sorce Enter the delay between lines (in seconds): 2 Error: The file 'sorce' was not found.
15	Python program to count the number of lines in a file

Code:	F=open("drinks.dat","r")
	
	count=0
	
	while(True):
	  b=F.read(1)
	  if(b=='\n'):
	  count+=1
	  if(b==""):
	  break
	  print(b,end="")
	print("Line Count " , count)
	
	F.close()

Output:	pepsi coc Line Count 2
16	Python program to read first N character from each line.

Code:	def read_first_n_characters(file_name, n):
	  try:
	  # Open the file in read mode
	  with open(file_name, "r") as file:
	  print(f"Reading the first {n} characters from each line of '{file_name}':\n")
	
	  # Read and print the first N characters from each line
	  for line in file:
	  print(line[:n]) # Slicing the first N characters
	
	  except FileNotFoundError:
	  print(f"Error: The file '{file_name}' was not found.")
	  except Exception as e:
	  print(f"An unexpected error occurred: {e}")
	
	# Example usage

```
file_name = input("Enter the file name to read: ")
n = int(input("Enter the number of characters to read from each line: "))
read_first_n_characters(file_name, n)
```

Output:

```
Enter the file name to read: example.txt
Enter the number of characters to read from each line: 5
Reading the first 5 characters from each line of 'example.txt'

This
This
This
```

17 Python program to read data from file and extract record data from it

Code:

```
def extract_records(file_name, key, value):
    try:
        # Open the file in read mode
        with open(file_name, "r") as file:
            print(f"Extracting records where {key} is '{value}':\n")

            # Iterate through each line in the file
            for line in file:
                # Check if the key-value pair exists in the line
                if f"{key}: {value}" in line:
                    print(line.strip())  # Print the matching record

    except FileNotFoundError:
        print(f"Error: The file '{file_name}' was not found.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

# Example usage
file_name = input("Enter the file name to read: ")
key = input("Enter the key to search for (e.g., Department): ")
value = input("Enter the value to search for (e.g., IT): ")
extract_records(file_name, key, value)
```

Output:

```
Enter the file name to read: records.txt
Enter the key to search for (e.g., Department): department
Enter the value to search for (e.g., IT): IT
Error: The file 'records.txt' was not found.
```

18 Python program to check a file's status in file Handling

Code:

```
def check_file_status(file_name):
    try:
```

```
        # Open the file in read mode
        with open(file_name, "r") as file_object:
            # Display file information
            print("File Status Information:")
            print("------------------------")
            print(f"Name of the File: {file_object.name}")
            print(f"Closed or Not   : {file_object.closed}")
            print(f"Opening Mode    : {file_object.mode}")

        # After exiting the 'with' block, the file is automatically closed.
        print("\nAfter exiting the 'with' block:")
        print(f"Closed or Not   : {file_object.closed}")

    except FileNotFoundError:
        print(f"Error: The file '{file_name}' does not exist.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}"
# Example usage
file_name = input("Enter the file name to check status: ")
check_file_status(file_name)
```

Output:

```
Enter the file name to check status: example.txt
File Status Information:
------------------------
Name of the File: example.txt
Closed or Not   : False
Opening Mode    : r

After exiting the 'with' block:
Closed or Not   : True
```

19 Python program to read character till a count .

Code:

```
def read_characters(file_name, count):
    try:
        # Open the file in read mode
        with open(file_name, "r") as file:
            print(f"Reading the first {count} characters from '{file_name}':\n")
            # Read the specified number of characters
            data = file.read(count)
            print(data)
    except FileNotFoundError:
        print(f"Error: The file '{file_name}' was not found.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
# Example usage
```

	file_name = input("Enter the file name to read: ") count = int(input("Enter the number of characters to read: ")) read_characters(file_name, count)
Output:	Enter the file name to read: example.txt Enter the number of characters to read: 15 Reading the first 15 characters from 'example.txt': This is the fir
20	Python program to delete a file.

Code:	import os
	
	def delete_file(file_name):
	  try:
	  # Check if the file exists
	  if os.path.exists(file_name):
	  # Delete the file
	  os.remove(file_name)
	  print(f"The file '{file_name}' has been deleted successfully.")
	  else:
	  print(f"Error: The file '{file_name}' does not exist.")
	  except Exception as e:
	  print(f"An unexpected error occurred: {e}")
	# Example usage
	file_name = input("Enter the name of the file to delete: ")
	delete_file(file_name)

Output:	Enter the name of the file to delete: example.txt The file 'example.txt' has been deleted successfully.
	Unit – 3
Programme _01	Write a program to create python class and objects. Create multiple objects of Python Class Python Methods Python Constructor using Employee Class.

Code:-	class Employee:
	  # Constructor to initialize employee details
	  def __init__(self, name, emp_id, department, salary):
	  self.name = name
	  self.emp_id = emp_id
	  self.department = department
	  self.salary = salary
	  # Method to display employee details
	  def display_details(self):
	  print(f"Employee ID: {self.emp_id}")
	  print(f"Name: {self.name}")
	  print(f"Department: {self.department}")
	  print(f"Salary: {self.salary}")
	  print("-" * 30)
	  # Method to update the salary
	  def update_salary(self, new_salary):
	  self.salary = new_salary
	  print(f"Updated salary for {self.name} to {self.salary}\n")

	# Create multiple objects of the Employee class
	employee1 = Employee("Alice Johnson", 101, "Engineering", 80000)
	employee2 = Employee("Bob Smith", 102, "Marketing", 60000)
	employee3 = Employee("Charlie Brown", 103, "Human Resources", 50000)
	# Use methods to display and manipulate employee details
	employee1.display_details()
	employee2.display_details()
	employee3.display_details()
	# Update salary for an employee
	employee2.update_salary(65000)
	# Display details again to reflect updated salary
	employee2.display_details()

Output:-	Employee ID: 101 Name: Alice Johnson Department: Engineering Salary: 80000 ---------- Employee ID: 102 Name: Bob Smith Department: Marketing Salary: 60000 ---------- Employee ID: 103 Name: Charlie Brown Department: Human Resources Salary: 50000 ---------- Updated salary for Bob Smith to 65000 Employee ID: 102 Name: Bob Smith Department: Marketing Salary: 65000 ----------
Programme _02	Write a Python program to build flashcard using class in Python Approach – 1

Code:-	class Flashcard:
	  def __init__(self, question, answer):
	  self.question = question
	  self.answer = answer
	  def check_answer(self, user_answer):
	  """Check if the user's answer is correct."""
	  return user_answer.strip().lower() == self.answer.strip().lower()
	  def show_flashcard(self):
	  """Display the flashcard's question."""
	  print(f"Question: {self.question}")

```
class FlashcardApp:
    def __init__(self):
        self.flashcards = []
    def add_flashcard(self, question, answer):
        """Add a new flashcard."""
        new_flashcard = Flashcard(question, answer)
        self.flashcards.append(new_flashcard)
    def test_user(self):
        """Test the user on their flashcards."""
        correct_answers = 0
        for flashcard in self.flashcards:
            flashcard.show_flashcard()
            user_answer = input("Your Answer: ")
            if flashcard.check_answer(user_answer):
                print("Correct!\n")
                correct_answers += 1
            else:
                print(f"Incorrect. The correct answer was: {flashcard.answer}\n")
        print(f"You got {correct_answers} out of {len(self.flashcards)} correct!")
    def show_flashcards(self):
        """Display all flashcards."""
        for flashcard in self.flashcards:
            flashcard.show_flashcard()
            print(f"Answer: {flashcard.answer}\n")
# Main program to run the flashcard app
if __name__ == "__main__":
    app = FlashcardApp()
    # Add flashcards (This can be expanded with more questions)
    app.add_flashcard("What is the capital of France?", "Paris")
    app.add_flashcard("What is 2 + 2?", "4")
    app.add_flashcard("Who wrote 'Romeo and Juliet'?", "William Shakespeare")
    # Show all flashcards
    print("All Flashcards:")
    app.show_flashcards()
    # Test user
    print("\nTesting Your Knowledge:")
    app.test_user()
```

Output:-	All Flashcards: Question: What is the capital of France? Answer: Paris Question: What is 2 + 2? Answer: 4 Question: Who wrote 'Romeo and Juliet'? Answer: William Shakespeare Testing Your Knowledge: Question: What is the capital of France? Your Answer: paris Correct! Question: What is 2 + 2? Your Answer: 5 Incorrect. The correct answer was: 4 Question: Who wrote 'Romeo and Juliet'? Your Answer:
Programme _03	Write a Python program to build flashcard using class in Python Approach -2

Code:-	import os
	class Flashcard:
	  def __init__(self, question, answer):
	  self.question = question
	  self.answer = answer
	  def check_answer(self, user_answer):
	  """Check if the user's answer is correct."""
	  return user_answer.strip().lower() == self.answer.strip().lower()
	  def show_flashcard(self):
	  """Display the flashcard's question."""
	  print(f"Question: {self.question}")
	class FlashcardApp:
	  def __init__(self, file_name="flashcards.txt"):
	  self.file_name = file_name
	  self.flashcards = []
	  self.load_flashcards()
	  def add_flashcard(self, question, answer):
	  """Add a new flashcard."""
	  new_flashcard = Flashcard(question, answer)
	  self.flashcards.append(new_flashcard)
	  self.save_flashcards()

```
    def save_flashcards(self):
        """Save all flashcards to a file."""
        with open(self.file_name, 'w') as file:
            for flashcard in self.flashcards:
                file.write(f"{flashcard.question}\n{flashcard.answer}\n"
    def load_flashcards(self):
        """Load flashcards from a file."""
        if os.path.exists(self.file_name):
            with open(self.file_name, 'r') as file:
                lines = file.readlines()
                for i in range(0, len(lines), 2):
                    question = lines[i].strip()
                    answer = lines[i + 1].strip()
                   self.flashcards.append(Flashcard(question, answer))
    def test_user(self):
        """Test the user on their flashcards."""
        correct_answers = 0
        for flashcard in self.flashcards:
            flashcard.show_flashcard()
            user_answer = input("Your Answer: ")
            if flashcard.check_answer(user_answer):
                print("Correct!\n")
                correct_answers += 1
            else:
                print(f"Incorrect. The correct answer was: {flashcard.answer}\n")
        print(f"You got {correct_answers} out of {len(self.flashcards)} correct!")
    def show_flashcards(self):
        """Display all flashcards."""
        for flashcard in self.flashcards:
            flashcard.show_flashcard()
            print(f"Answer: {flashcard.answer}\n")
# Main program to run the flashcard app
if __name__ == "__main__":
    app = FlashcardApp()
    # Add flashcards (This can be expanded with more questions)
    app.add_flashcard("What is the capital of France?", "Paris")
    app.add_flashcard("What is 2 + 2?", "4")
          app.add_flashcard("Who wrote 'Romeo and Juliet'?", "William
Shakespeare")
    # Show all flashcards
    print("All Flashcards:")
    app.show_flashcards()
    # Test user
    print("\nTesting Your Knowledge:")
```

	app.test_user()
Output:-	All Flashcards: Question: What is the capital of France? Answer: Paris Question: What is 2 + 2? Answer: 4 Question: Who wrote 'Romeo and Juliet'? Answer: William Shakespeare Question: What is the capital of France? Answer: Paris Question: What is 2 + 2? Answer: 4 Question: Who wrote 'Romeo and Juliet'? Answer: William Shakespeare Testing Your Knowledge: Question: What is the capital of France? Your Answer: dfcdc Incorrect. The correct answer was: Paris Question: What is 2 + 2? Your Answer: 4 Correct! Question: Who wrote 'Romeo and Juliet'? Your Answer: juliet Incorrect. The correct answer was: William Shakespeare Question: What is the capital of France? Your Answer: no3 Incorrect. The correct answer was: Paris Question: What is 2 + 2? Your Answer:
Programme _04	Write a program to build a simple Student Management System using Python which can perform the following operations: Accept Display Search Delete Update
Code:-	# This is simplest Student data management program in python

```
# Create class "Student"
class Student:
# Constructor
        def __init__(self, name, rollno, m1, m2):
                self.name = name
                self.rollno = rollno
                self.m1 = m1
                self.m2 = m2
        def accept(self, Name, Rollno, marks1, marks2):
# use ' int(input()) ' method to take input from user
                ob = Student(Name, Rollno, marks1, marks2)
                ls.append(ob)
        # Function to display student details
        def display(self, ob):
                print("Name : ", ob.name)
                print("RollNo : ", ob.rollno)
                print("Marks1 : ", ob.m1)
                print("Marks2 : ", ob.m2)
                print("\n")
        # Search Function
        def search(self, rn):
                for i in range(ls.__len__()):
                        if(ls[i].rollno == rn):
                                return i
        # Delete Function
        def delete(self, rn):
                i = obj.search(rn)
                del ls[i]
        # Update Function
        def update(self, rn, No):
                i = obj.search(rn)
                roll = No
                ls[i].rollno = roll
# Create a list to add Students
ls = []
# an object of Student class
obj = Student('', 0, 0, 0)
print("\nOperations used, ")
print("\n1.Accept Student details\n2.Display Student Details\n3.Search Details of a Student\n4.Delete Details of Student\n5.Update Student Details\n6.Exit")
if(ch == 1):
obj.accept("A", 1, 100, 100)
obj.accept("B", 2, 90, 90)
```

```
obj.accept("C", 3, 80, 80)
 elif(ch == 2):
print("\n")
print("\nList of Students\n")
for i in range(ls.__len__()):
        obj.display(ls[i])
elif(ch == 3):
print("\n Student Found, ")
s = obj.search(2)
obj.display(ls[s])
 elif(ch == 4):
obj.delete(2)
print(ls.__len__())
print("List after deletion")
for i in range(ls.__len__()):
        obj.display(ls[i])
elif(ch == 5):
obj.update(3, 2)
print(ls.__len__())
print("List after updation")
for i in range(ls.__len__()):
        obj.display(ls[i])
else:
print("Thank You !")
```

```
--- Student Management System ---
1. Accept Student
2. Display All Students
3. Search Student
4. Delete Student
5. Update Student
6. Exit
Enter your choice (1-6): 1
Enter Student ID: 1
Enter Name: charmi
Enter Age: 21
Enter Course: mca
Student added successfully!

--- Student Management System ---
1. Accept Student
2. Display All Students
3. Search Student
4. Delete Student
5. Update Student
6. Exit
Enter your choice (1-6): 1
Enter Student ID: bhumika
Enter Name: hj
Enter Age: 15
Enter Course: msc
Student added successfully!

--- Student Management System ---
1. Accept Student
2. Display All Students
3. Search Student
4. Delete Student
5. Update Student
6. Exit
Enter your choice (1-6): 4
Enter Student ID to delete: bhumika
Student deleted successfully!
```

```
--- Student Management System ---
1. Accept Student
2. Display All Students
3. Search Student
4. Delete Student
5. Update Student
6. Exit
Enter your choice (1-6): 4
Enter Student ID to delete: bhumika
Student deleted successfully!

--- Student Management System ---
1. Accept Student
2. Display All Students
3. Search Student
4. Delete Student
5. Update Student
6. Exit
Enter your choice (1-6): 1
Enter Student ID: 3
Enter Name: bhumika
Enter Age: 21
Enter Course: msc_It
Student added successfully!

--- Student Management System ---
1. Accept Student
2. Display All Students
3. Search Student
4. Delete Student
5. Update Student
6. Exit
Enter your choice (1-6): 2

--- List of Students ---
ID: 1, Name: charmi, Age: 21, Course: mca
ID: 3, Name: bhumika, Age: 21, Course: msc_It
```

	--- Student Management System --- 1. Accept Student 2. Display All Students 3. Search Student 4. Delete Student 5. Update Student 6. Exit Enter your choice (1-6): 5 Enter Student ID to update: 3 Found Student: ID: 3, Name: bhumika, Age: 21, Course: msc_It Enter new Name: prisha Enter new Age: 15 Enter new Course: none Student updated successfully! --- Student Management System --- 1. Accept Student 2. Display All Students 3. Search Student 4. Delete Student 5. Update Student 6. Exit Enter your choice (1-6):
Programme _05	9-1. Restaurant: Make a class called Restaurant . The __init__() method for Restaurant should store two attributes: a restaurant_name and a cuisine_type . Make a method called describe_restaurant() that prints these two pieces of information, and a method called open_restaurant() that prints a message indi cating that the restaurant is open . Make an instance called restaurant from your class . Print the two attri butes individually, and then call both methods

Code:-	class Restaurant():
	  def __init__(self, name, cuisine_type):
	  self.name = name.title()
	  self.cuisine_type = cuisine_type
	  def describe_restaurant(self)
	  msg = f"{self.name} serves wonderful {self.cuisine_type}."
	  print(f"\n{msg}")
	  def open_restaurant(self):
	  msg = f"{self.name} is open. Come on in!"
	  print(f"\n{msg}")
	restaurant = Restaurant('the mean queen', 'pizza')
	print(restaurant.name)
	print(restaurant.cuisine_type)
	restaurant.describe_restaurant()
	restaurant.open_restaurant()

Output:-	= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/restaurent_9.1.py The Mean Queen pizza The Mean Queen serves wonderful pizza. The Mean Queen is open. Come on in!

	9-2. Three Restaurants: Start with your class from Exercise 9-1 . Create three different instances from the class, and call describe_restaurant() for each instance

	class Restaurant():
	  def __init__(self, name, cuisine_type)
	  self.name = name.title()
	  self.cuisine_type = cuisine_type
	  def describe_restaurant(self):
	  msg = f"{self.name} serves wonderful {self.cuisine_type}."
	  print(f"\n{msg}")
	  def open_restaurant(self):
	  msg = f"{self.name} is open. Come on in!"
	  print(f"\n{msg}")
	mean_queen = Restaurant('the mean queen', 'pizza')
	mean_queen.describe_restaurant()
	ludvigs = Restaurant("ludvig's bistro", 'seafood')
	ludvigs.describe_restaurant()
	mango_thai = Restaurant('mango thai', 'thai food')
	mango_thai.describe_restaurant()

	= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/restaurent_9.2.py The Mean Queen serves wonderful pizza. Ludvig'S Bistro serves wonderful seafood. Mango Thai serves wonderful thai food.
	9-3. Users: Make a class called User . Create two attributes called first_name and last_name, and then create several other attributes that are typically stored in a user profile . Make a method called describe_user() that prints a summary of the user's information . Make another method called greet_user() that prints a personalized greeting to the user . Create several instances representing different users, and call both methods for each user

	class User():
	  """Represent a simple user profile."""
	  def __init__(self, first_name, last_name, username, email, location):
	  """Initialize the user."""
	  self.first_name = first_name.title()
	  self.last_name = last_name.title()
	  self.username = username
	  self.email = email
	  self.location = location.title()
	  def describe_user(self):
	  """Display a summary of the user's information."""
	  print(f"\n{self.first_name} {self.last_name}")
	  print(f" Username: {self.username}")

```
        print(f"  Email: {self.email}")
        print(f"  Location: {self.location}")
    def greet_user(self):
        """Display a personalized greeting to the user."""
        print(f"\nWelcome back, {self.username}!")
eric = User('eric', 'matthes', 'e_matthes', 'e_matthes@example.com', 'alaska')
eric.describe_user()
eric.greet_user()
willie = User('willie', 'burger', 'willieburger', 'wb@example.com', 'alaska')
willie.describe_user()
willie.greet_user()
```

```
= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/restaurent_9.3.py

Eric Matthes
  Username: e_matthes
  Email: e_matthes@example.com
  Location: Alaska

Welcome back, e_matthes!

Willie Burger
  Username: willieburger
  Email: wb@example.com
  Location: Alaska

Welcome back, willieburger!
```

Programme _06

9-4. Number Served: Start with your program from Exercise 9-1 (page 166) . Add an attribute called number_served with a default value of 0 . Create an instance called restaurant from this class . Print the number of customers the restaurant has served, and then change this value and print it again . Add a method called set_number_served() that lets you set the number of customers that have been served . Call this method with a new number and print the value again . Add a method called increment_number_served() that lets you increment the number of customers who've been served . Call this method with any num ber you like that could represent how many customers were served in, say, a day of business

Code:-

```
class Restaurant():
    def __init__(self, name, cuisine_type):
        self.name = name.title()
        self.cuisine_type = cuisine_type
        self.number_served = 0
    def describe_restaurant(self):
        msg = f"{self.name} serves wonderful {self.cuisine_type}."
        print(f"\n{msg}")
    def open_restaurant(self):
        msg = f"{self.name} is open. Come on in!"
        print(f"\n{msg}")
    def set_number_served(self, number_served):
        self.number_served = number_served
    def increment_number_served(self, additional_served):
```

```
        self.number_served += additional_served
restaurant = Restaurant('the mean queen', 'pizza')
restaurant.describe_restaurant()
print(f"\nNumber served: {restaurant.number_served}")
restaurant.number_served = 430
print(f"Number served: {restaurant.number_served}")
restaurant.set_number_served(1257)
print(f"Number served: {restaurant.number_served}")
restaurant.increment_number_served(239)
print(f"Number served: {restaurant.number_served}")
```

```
= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/restaurent_9.4.py

The Mean Queen serves wonderful pizza.

Number served: 0
Number served: 430
Number served: 1257
Number served: 1496
```

9-5. Login Attempts: Add an attribute called login_attempts to your User class from Exercise 9-3 (page 166) . Write a method called increment_ login_attempts() that increments the value of login_attempts by 1 . Write another method called reset_login_attempts() that resets the value of login_ attempts to 0 . Make an instance of the User class and call increment_login_attempts() several times . Print the value of login_attempts to make sure it was incremented properly, and then call reset_login_attempts() . Print login_attempts again to make sure it was reset to 0

```
class User():
    """Represent a simple user profile."""
    def __init__(self, first_name, last_name, username, email, location):
        """Initialize the user."""
        self.first_name = first_name.title()
        self.last_name = last_name.title()
        self.username = username
        self.email = email
        self.location = location.title()
        self.login_attempts = 0
    def describe_user(self):
        """Display a summary of the user's information."""
        print(f"\n{self.first_name} {self.last_name}")
        print(f"  Username: {self.username}")
        print(f"  Email: {self.email}")
        print(f"  Location: {self.location}")
    def greet_user(self):
        """Display a personalized greeting to the user."""
        print(f"\nWelcome back, {self.username}!")
    def increment_login_attempts(self):
```

```
        """Increment the value of login_attempts."""
        self.login_attempts += 1
    def reset_login_attempts(self):
        """Reset login_attempts to 0."""
        self.login_attempts = 0
eric = User('eric', 'matthes', 'e_matthes', 'e_matthes@example.com', 'alaska')
eric.describe_user()
eric.greet_user()
print("\nMaking 3 login attempts...")
eric.increment_login_attempts()
eric.increment_login_attempts()
eric.increment_login_attempts()
print(f"  Login attempts: {eric.login_attempts}")

print("Resetting login attempts...")
eric.reset_login_attempts()
print(f"  Login attempts: {eric.login_attempts}")
```

```
= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/restaurent_9.5.py

Eric Matthes
  Username: e_matthes
  Email: e_matthes@example.com
  Location: Alaska

Welcome back, e_matthes!

Making 3 login attempts...
  Login attempts: 3
Resetting login attempts...
  Login attempts: 0
```

Programme _07

Write a python program to demonstrate various kinds of inheritance.

Code:-

```
# Single Inheritance
class Animal:
    def sound(self):
        return "Some animal sound"
class Dog(Animal):
    def speak(self):
        return "Woof!"
# Multiple Inheritance
class Father:
    def wisdom(self):
        return "Father's wisdom"
class Mother:
    def kindness(self):
        return "Mother's kindness"
class Child(Father, Mother):
    def skills(self):
        return "Child's skills"
```

```
# Multilevel Inheritance
class Grandparent:
    def legacy(self):
        return "Grandparent's legacy"
class Parent(Grandparent):
    def advice(self):
        return "Parent's advice"
class Child2(Parent):
    def ambitions(self):
        return "Child's ambitions"
# Hierarchical Inheritance
class Bird:
    def fly(self):
        return "Bird is flying"
class Sparrow(Bird):
    def chirp(self):
        return "Chirp chirp!"
class Eagle(Bird):
    def soar(self):
        return "Eagle is soaring high"
# Hybrid Inheritance (Combination of multiple and multilevel inheritance)
class Vehicle:
    def move(self):
        return "Vehicle is moving"
class Engine:
    def start(self):
        return "Engine started"
class Car(Vehicle, Engine):
    def drive(self):
        return "Car is driving"
# Testing the classes
# Single Inheritance
dog = Dog()
print(dog.sound())  # Animal method
print(dog.speak())  # Dog method
# Multiple Inheritance
child = Child()
print(child.wisdom())  # Father's method
print(child.kindness())  # Mother's method
print(child.skills())  # Child's method
# Multilevel Inheritance
child2 = Child2()
print(child2.legacy())  # Grandparent's method
print(child2.advice())  # Parent's method
```

```
print(child2.ambitions())  # Child's method
# Hierarchical Inheritance
sparrow = Sparrow()
print(sparrow.fly())  # Bird method
print(sparrow.chirp())  # Sparrow method
eagle = Eagle()
print(eagle.fly())  # Bird method
print(eagle.soar())  # Eagle method
# Hybrid Inheritance
car = Car()
print(car.move())  # Vehicle method
print(car.start())  # Engine method
print(car.drive())  # Car method
```

Output:-

```
= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/3.prog_07.py
Some animal sound
Woof!
Father's wisdom
Mother's kindness
Child's skills
Grandparent's legacy
Parent's advice
Child's ambitions
Bird is flying
Chirp chirp!
Bird is flying
Eagle is soaring high
Vehicle is moving
Engine started
Car is driving
```

Programme _08

Write a python program to Create Student Class

Code:-

```
class Student:
    # Constructor to initialize student details
    def __init__(self, name, roll_number, age, marks):
        self.name = name
        self.roll_number = roll_number
        self.age = age
        self.marks = marks
    # Method to display student details
    def display_details(self):
        print(f"Student Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Age: {self.age}")
        print(f"Marks: {self.marks}")
    # Method to calculate the grade based on marks
    def calculate_grade(self):
        if self.marks >= 90:
            return "A+"
        elif self.marks >= 80:
```

```
            return "A"
        elif self.marks >= 70:
            return "B"
        elif self.marks >= 60:
            return "C"
        else:
            return "D"
    # Method to check if the student passed
    def is_passed(self):
        return self.marks >= 50
# Creating Student objects
student1 = Student("John Doe", 101, 20, 85)
student2 = Student("Jane Smith", 102, 22, 45)
# Displaying details of student1
print("Student 1 Details:")
student1.display_details()
print(f"Grade: {student1.calculate_grade()}")
print(f"Passed: {'Yes' if student1.is_passed() else 'No'}\n")
# Displaying details of student2
print("Student 2 Details:")
student2.display_details()
print(f"Grade: {student2.calculate_grade()}")
print(f"Passed: {'Yes' if student2.is_passed() else 'No'}")
```

```
= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/3.prog_08.py
Student 1 Details:
Student Name: John Doe
Roll Number: 101
Age: 20
Marks: 85
Grade: A
Passed: Yes

Student 2 Details:
Student Name: Jane Smith
Roll Number: 102
Age: 22
Marks: 45
Grade: D
Passed: No
```

Programme _09

Write a python program to Create Student Class with Constructor and Destructor

Code:-

```
class Student:
    # Constructor to initialize student details
    def __init__(self, name, roll_number, age, marks):
        self.name = name
        self.roll_number = roll_number
        self.age = age
        self.marks = marks
        print(f"Student object for {self.name} created!")
```

```
 # Method to display student details
  def display_details(self):
     print(f"Student Name: {self.name}")
     print(f"Roll Number: {self.roll_number}")
     print(f"Age: {self.age}")
     print(f"Marks: {self.marks}")
   # Method to calculate the grade based on marks
  def calculate_grade(self):
     if self.marks >= 90:
        return "A+"
     elif self.marks >= 80:
        return "A"
     elif self.marks >= 70:
        return "B"
     elif self.marks >= 60:
        return "C"
     else:
        return "D"
  # Method to check if the student passed
  def is_passed(self):
     return self.marks >= 50
  # Destructor to display a message when the object is destroyed
  def __del__(self):
     print(f"Student object for {self.name} is being deleted.")
# Creating Student objects
student1 = Student("John Doe", 101, 20, 85)
student2 = Student("Jane Smith", 102, 22, 45)
# Displaying details of student1
print("\nStudent 1 Details:")
student1.display_details()
print(f"Grade: {student1.calculate_grade()}")
print(f"Passed: {'Yes' if student1.is_passed() else 'No'}\n")
# Displaying details of student2
print("Student 2 Details:")
student2.display_details()
print(f"Grade: {student2.calculate_grade()}")
print(f"Passed: {'Yes' if student2.is_passed() else 'No'}\n")
# Deleting objects explicitly (calling the destructor)
del student1
del student2
```

	= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI Student object for John Doe created! Student object for Jane Smith created! Student 1 Details: Student Name: John Doe Roll Number: 101 Age: 20 Marks: 85 Grade: A Passed: Yes Student 2 Details: Student Name: Jane Smith Roll Number: 102 Age: 22 Marks: 45 Grade: D Passed: No Student object for John Doe is being deleted. Student object for Jane Smith is being deleted.
Programme _10	Write a python program to implement Getters and Setters in a class

Code:-	class Student:
	  def __init__(self, name, roll_number, age, marks):
	  self._name = name # private attribute
	  self._roll_number = roll_number # private attribute
	  self._age = age # private attribute
	  self._marks = marks # private attribute
	  # Getter for name
	  @property
	  def name(self):
	  return self._name
	  # Setter for name
	  @name.setter
	  def name(self, value):
	  if len(value) > 2: # Example validation: name should be more than 2 characters
	  self._name = value
	  else:
	  print("Name should be more than 2 characters!")
	  # Getter for roll_number
	  @property
	  def roll_number(self):
	  return self._roll_number
	  # Setter for roll_number
	  @roll_number.setter
	  def roll_number(self, value):

```
        if value > 0:  # Example validation: roll number should be positive
            self._roll_number = value
        else:
            print("Roll number should be positive!")
    # Getter for age
    @property
    def age(self):
        return self._age
    # Setter for age
    @age.setter
    def age(self, value):
        if 18 <= value <= 100:  # Example validation: age should be between 18
and 100
            self._age = value
        else:
            print("Age should be between 18 and 100!")
    # Getter for marks
    @property
    def marks(self):
        return self._marks
    # Setter for marks
    @marks.setter
    def marks(self, value):
        if 0 <= value <= 100:  # Example validation: marks should be between 0
and 100
            self._marks = value
        else:
            print("Marks should be between 0 and 100!")
    # Method to display student details
    def display_details(self):
        print(f"Student Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Age: {self.age}")
        print(f"Marks: {self.marks}")
# Creating a Student object
student = Student("John", 101, 20, 85)
# Accessing attributes using getters
print("Original Details:")
student.display_details()
# Using setters to modify attributes
student.name = "Sam"         # Valid name
student.roll_number = 102     # Valid roll number
student.age = 22             # Valid age
student.marks = 90           # Valid marks
```

```
# Trying to set invalid values
student.name = "A"          # Invalid name (too short)
student.roll_number = -5     # Invalid roll number (negative)
student.age = 15            # Invalid age (too low)
student.marks = 150         # Invalid marks (too high)
# Displaying updated details
print("\nUpdated Details:")
student.display_details()
```

```
======================= RESTART: C:\Users\HP\Desktop\charmi\MCA_07\22401550301007_CHARMI MONANI\3.prog_10.py ===============
Original Details:
Student Name: John
Roll Number: 101
Age: 20
Marks: 85
Name should be more than 2 characters!
Roll number should be positive!
Age should be between 18 and 100!
Marks should be between 0 and 100!

Updated Details:
Student Name: Sam
Roll Number: 102
Age: 22
Marks: 90
```

Programme _11

Write a python program to Implement Abstraction using Abstract class

Code:-

```
from abc import ABC, abstractmethod
class Shape(ABC):
    @abstractmethod
  def area(self):
      pass
 @abstractmethod
  def perimeter(self):
      pass
# Subclass 1: Circle
class Circle(Shape):
  def __init__(self, radius):
      self.radius = radius
  def area(self):
      return 3.14 * self.radius * self.radius
  def perimeter(self):
      return 2 * 3.14 * self.radius
# Subclass 2: Rectangle
class Rectangle(Shape):
  def __init__(self, length, width):
      self.length = length
      self.width = width
  def area(self):
      return self.length * self.width
  def perimeter(self):
      return 2 * (self.length + self.width)
# Subclass 3: Triangle
```

```
class Triangle(Shape):
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
    def area(self):
        s = (self.a + self.b + self.c) / 2
        return (s * (s - self.a) * (s - self.b) * (s - self.c)) ** 0.5
    def perimeter(self):
        return self.a + self.b + self.c
# Driver code to test the abstract class and its subclasses
# Creating objects of each subclass
circle = Circle(5)
rectangle = Rectangle(10, 4)
triangle = Triangle(3, 4, 5)
# Displaying area and perimeter of each shape
print(f"Circle - Area: {circle.area()}, Perimeter: {circle.perimeter()}")
print(f"Rectangle - Area: {rectangle.area()}, Perimeter: {rectangle.perimeter()}")
print(f"Triangle - Area: {triangle.area()}, Perimeter: {triangle.perimeter()}")
```

```
= RESTART: C:/Users/HP/Desktop/charmi/MCA_07/22401550301007_CHARMI MONANI/3.prog_11.py
Circle - Area: 78.5, Perimeter: 31.400000000000002
Rectangle - Area: 40, Perimeter: 28
Triangle - Area: 6.0, Perimeter: 12
```

Programme _12

Write a program to single inheritance in Python.

Code:-

```
# Parent Class (Superclass)
class Animal:
    def __init__(self, name, species):
        self.name = name
        self.species = species
    def speak(self):
        return "Animal makes a sound"
# Child Class (Subclass) inheriting from Animal
class Dog(Animal):
    def __init__(self, name, species, breed):
        # Calling the parent class constructor
        super().__init__(name, species)
        self.breed = breed
    # Overriding the speak method to give specific behavior for Dog
    def speak(self):
        return "Woof! Woof!"
# Creating an object of the Dog class
dog = Dog("Buddy", "Dog", "Golden Retriever")
```

	# Accessing methods and attributes from both the parent and child classes
	print(f"Name: {dog.name}")
	print(f"Species: {dog.species}")
	print(f"Breed: {dog.breed}")
	print(f"Sound: {dog.speak()}")

Output :-	Name: Buddy Species: Dog Breed: Golden Retriever Sound: Woof! Woof!
Programme _13	Write a program to inheritance with two child (derived) classes in Python

Code:-	# Parent Class (Superclass)
	class Animal:
	  def __init__(self, name, species):
	  self.name = name
	  self.species = species
	  def speak(self):
	  return "Animal makes a sound"
	# Child Class 1: Dog
	class Dog(Animal):
	  def __init__(self, name, species, breed):
	  # Calling the parent class constructor
	  super().__init__(name, species)
	  self.breed = breed
	  def speak(self):
	  return "Woof! Woof!" # Overriding the speak method
	  def fetch(self):
	  return f"{self.name} is fetching the ball!"
	# Child Class 2: Cat
	class Cat(Animal):
	  def __init__(self, name, species, color):
	  # Calling the parent class constructor
	  super().__init__(name, species)
	  self.color = color
	  def speak(self):
	  return "Meow! Meow!" # Overriding the speak method
	  def scratch(self):
	  return f"{self.name} is scratching the furniture!"
	# Creating objects of the Dog and Cat classes
	dog = Dog("Buddy", "Dog", "Golden Retriever")
	cat = Cat("Whiskers", "Cat", "Black")
	# Displaying details and calling methods from the parent and child classes
	print(f"Dog's Name: {dog.name}")
	print(f"Dog's Breed: {dog.breed}")

	print(f"Dog's Sound: {dog.speak()}") print(dog.fetch()) print("\n") print(f"Cat's Name: {cat.name}") print(f"Cat's Color: {cat.color}") print(f"Cat's Sound: {cat.speak()}") print(cat.scratch())
Output:-	Dog's Name: Buddy Dog's Breed: Golden Retriever Dog's Sound: Woof! Woof! Buddy is fetching the ball! Cat's Name: Whiskers Cat's Color: Black Cat's Sound: Meow! Meow! Whiskers is scratching the furniture!
Programme _14	Write a program to multiple inheritance in Python

Code:-	# Parent Class 1: Animal
	class Animal:
	  def __init__(self, name):
	  self.name = name
	  def speak(self):
	  return "Animal makes a sound"
	# Parent Class 2: Color
	class Color:
	  def __init__(self, color):
	  self.color = color
	  def describe_color(self):
	  return f"The color of the animal is {self.color}"
	# Child Class: Dog (inherits from both Animal and Color)
	class Dog(Animal, Color):
	  def __init__(self, name, color, breed):
	  # Calling constructors of both parent classes
	  Animal.__init__(self, name)
	  Color.__init__(self, color)
	  self.breed = breed
	  def speak(self):
	  return "Woof! Woof!" # Overriding the speak method from Animal class
	  def display_info(self):
	  return f"{self.name} is a {self.breed} dog. {self.describe_color()}"
	# Creating an object of the Dog class

```
dog = Dog("Buddy", "Golden", "Golden Retriever")
# Accessing methods from both parent classes and the child class
print(f"Dog's Name: {dog.name}")
print(f"Dog's Breed: {dog.breed}")
print(dog.speak())  # Calling the overridden method from Dog class
print(dog.display_info())  # Calling the method from the Dog class
```

Output:-

```
Dog's Name: Buddy
Dog's Breed: Golden Retriever
Woof! Woof!
Buddy is a Golden Retriever dog. The color of the animal is Gol
```

Programme _15

Write a python program to check prime number using object oriented approach.

Code:-

```
class PrimeChecker:
    def __init__(self, number):
        self.number = number
    def is_prime(self):
        # Handle edge cases
        if self.number <= 1:
            return False
                # Check if the number is divisible by any number from 2 to sqrt(number)
        for i in range(2, int(self.number ** 0.5) + 1):
            if self.number % i == 0:
                return False
        return True
# Creating an object of the PrimeChecker class
num = int(input("Enter a number to check if it's prime: "))
prime_checker = PrimeChecker(num)
# Checking if the number is prime
if prime_checker.is_prime():
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
```

```
Enter a number to check if it's prime: 17
17 is a prime number.
```

Programme _16

Write a python program to check Armstrong number using object oriented approach.

Code:-

```
class ArmstrongChecker:
    def __init__(self, number):
        self.number = number
    def is_armstrong(self):
        # Convert the number to string to easily count the digits
```

```
        num_str = str(self.number)
        num_digits = len(num_str)
            # Calculate the sum of the digits each raised to the power of num_digits
        sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
        # Check if the sum of powers is equal to the original number
        return sum_of_powers == self.number
# Creating an object of ArmstrongChecker class
num = int(input("Enter a number to check if it's an Armstrong number: "))
armstrong_checker = ArmstrongChecker(num)
# Checking if the number is an Armstrong number
if armstrong_checker.is_armstrong():
    print(f"{num} is an Armstrong number.")
else:
    print(f"{num} is not an Armstrong number.")
```

```
Enter a number to check if it's an Armstrong number: 153
153 is an Armstrong number.
```

Programme_17

Write a python program to Multilevel inheritance.

Code:-

```
# Grandparent Class
class Animal:
    def __init__(self, name, species):
        self.name = name
        self.species = species
    def speak(self):
        return f"{self.name} makes a sound."
# Parent Class (inherits from Animal)
class Dog(Animal):
    def __init__(self, name, species, breed):
        # Calling the constructor of the Animal class (grandparent)
        super().__init__(name, species)
        self.breed = breed
    def speak(self):
        return f"{self.name} barks!"
# Child Class (inherits from Dog)
class Puppy(Dog):
    def __init__(self, name, species, breed, age):
        # Calling the constructor of the Dog class (parent)
        super().__init__(name, species, breed)
        self.age = age
    def display_info(self):
        return f"{self.name} is a {self.age} year old {self.breed}."
# Creating an object of the Puppy class
```

	puppy = Puppy("Buddy", "Dog", "Golden Retriever", 1)
	# Accessing methods and attributes from all the classes in the inheritance chain
	print(f"Puppy's Name: {puppy.name}")
	print(f"Puppy's Species: {puppy.species}")
	print(f"Puppy's Breed: {puppy.breed}")
	print(puppy.speak()) # Calling the speak method from the Dog class (overridden)
	print(puppy.display_info()) # Calling the method from the Puppy class

Output:-	Puppy's Name: Buddy Puppy's Species: Dog Puppy's Breed: Golden Retriever Buddy barks! Buddy is a 1 year old Golden Retriever.
Programme _18	Write a python program to check Palindrome number using object oriented approach.

Code:-	class PalindromeChecker:
	  def __init__(self, number):
	  self.number = number
	  def is_palindrome(self):
	  original_number = str(self.number)
	  reversed_number = original_number[::-1]
	  return original_number == reversed_number
	# Example usage
	if __name__ == "__main__":
	  number = 12321
	  checker = PalindromeChecker(number)
	  if checker.is_palindrome():
	  print(f"{number} is a palindrome.")
	  else:
	  print(f"{number} is not a palindrome.")

	12321 is a palindrome.
Programme _19	Write a python program to manage a phone store (mobile shop) record using class.

Code:-	class Phone:
	  def __init__(self, model, brand, price, stock):
	  self.model = model
	  self.brand = brand
	  self.price = price
	  self.stock = stock
	  def update_stock(self, quantity):
	  self.stock += quantity

```
    def sell_phone(self, quantity):
        if quantity <= self.stock:
            self.stock -= quantity
            print(f"Sold {quantity} {self.model}(s). Remaining stock: {self.stock}")
        else:
                print(f"Not enough stock for {self.model}. Only {self.stock}
available.")
    def __str__(self):
        return f"Model: {self.model}, Brand: {self.brand}, Price: ${self.price},
Stock: {self.stock}"
class MobileStore:
    def __init__(self):
        self.inventory = []
    def add_phone(self, model, brand, price, stock):
        phone = Phone(model, brand, price, stock)
        self.inventory.append(phone)
        print(f"{model} by {brand} added to the store.")
    def view_inventory(self):
        if not self.inventory:
            print("No phones in the inventory.")
        else:
            print("Current Phone Inventory:")
            for phone in self.inventory:
                print(phone)
    def search_phone(self, model=None, brand=None):
        found = False
        for phone in self.inventory:
            if (model and phone.model == model) or (brand and phone.brand ==
brand):
                print(phone)
                found = True
        if not found:
            print("Phone not found.")
    def update_stock(self, model, quantity):
        for phone in self.inventory:
            if phone.model == model:
                phone.update_stock(quantity)
                print(f"Stock for {model} updated. New stock: {phone.stock}")
                return
        print(f"{model} not found in the inventory.")
    def sell_phone(self, model, quantity):
        for phone in self.inventory:
            if phone.model == model:
                phone.sell_phone(quantity)
```

```
            return
        print(f"{model} not found in the inventory.")
# Create an instance of the MobileStore class
store = MobileStore()
# Adding some phones to the inventory
store.add_phone("iPhone 14", "Apple", 999, 50)
store.add_phone("Galaxy S23", "Samsung", 799, 30)
store.add_phone("Pixel 8", "Google", 699, 20)
# Viewing all phones in the inventory
store.view_inventory()
# Searching for a specific phone by model
store.search_phone(model="iPhone 14")
# Searching for phones by brand
store.search_phone(brand="Samsung")
# Updating stock
store.update_stock("Pixel 8", 10)
# Selling phones
store.sell_phone("iPhone 14", 5)
# Viewing the updated inventory
store.view_inventory()
```

```
iPhone 14 by Apple added to the store.
Galaxy S23 by Samsung added to the store.
Pixel 8 by Google added to the store.
Current Phone Inventory:
Model: iPhone 14, Brand: Apple, Price: $999, Stock: 50
Model: Galaxy S23, Brand: Samsung, Price: $799, Stock: 30
Model: Pixel 8, Brand: Google, Price: $699, Stock: 20
Model: iPhone 14, Brand: Apple, Price: $999, Stock: 50
Model: Galaxy S23, Brand: Samsung, Price: $799, Stock: 30
Stock for Pixel 8 updated. New stock: 30
Sold 5 iPhone 14(s). Remaining stock: 45
Current Phone Inventory:
Model: iPhone 14, Brand: Apple, Price: $999, Stock: 45
Model: Galaxy S23, Brand: Samsung, Price: $799, Stock: 30
Model: Pixel 8, Brand: Google, Price: $699, Stock: 30
```

Programme_20

Write a python program to bank management system.

Code:-

```
class BankAccount:
    def __init__(self, account_holder, account_number, balance=0):
        self.account_holder = account_holder
        self.account_number = account_number
        self.balance = balance
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
```

```
            print(f"Deposited {amount}. New balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")
    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance: {self.balance}")
        else:
            print("Insufficient balance.")
    def check_balance(self):
        print(f"Account balance: {self.balance}")
    def display_account_info(self):
        print(f"Account Holder: {self.account_holder}")
        print(f"Account Number: {self.account_number}")
        print(f"Balance: {self.balance}")
class Bank:
    def __init__(self):
        self.accounts = {}
    def create_account(self, account_holder, account_number):
        if account_number in self.accounts:
            print("Account number already exists!")
        else:
            new_account = BankAccount(account_holder, account_number)
            self.accounts[account_number] = new_account
                print(f"Account created for {account_holder} with account number
{account_number}.")
    def get_account(self, account_number):
        return self.accounts.get(account_number)
    def display_all_accounts(self):
        if self.accounts:
            for account in self.accounts.values():
                account.display_account_info()
        else:
            print("No accounts found.")
def main():
    bank = Bank()
    while True:
        print("\n--- Bank Management System ---")
        print("1. Create Account")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Check Balance")
        print("5. Display Account Info")
        print("6. Display All Accounts")
```

```
print("7. Exit")

choice = input("Enter your choice: ")
if choice == '1':
    account_holder = input("Enter account holder's name: ")
    account_number = input("Enter account number: ")
    bank.create_account(account_holder, account_number)
elif choice == '2':
    account_number = input("Enter account number: ")
    account = bank.get_account(account_number)
    if account:
        amount = float(input("Enter amount to deposit: "))
        account.deposit(amount)
    else:
        print("Account not found.")
elif choice == '3':
    account_number = input("Enter account number: ")
    account = bank.get_account(account_number)
    if account:
        amount = float(input("Enter amount to withdraw: "))
        account.withdraw(amount)
    else:
        print("Account not found.")
elif choice == '4':
    account_number = input("Enter account number: ")
    account = bank.get_account(account_number)
    if account:
        account.check_balance()
    else:
        print("Account not found.")
elif choice == '5':
    account_number = input("Enter account number: ")
    account = bank.get_account(account_number)
    if account:
        account.display_account_info()
    else:
        print("Account not found.")
elif choice == '6':
    bank.display_all_accounts()
elif choice == '7':
    print("Exiting the Bank Management System. Goodbye!")
    break
else:
    print("Invalid choice. Please try again.")
```

	if __name__ == "__main__": main()
	``` ****** Bank Management System ****** 1. Create Account 2. Deposit Money 3. Withdraw Money 4. Check Balance 5. Exit Enter your Choice (1 to 5): 1 Enter Account Number : 101 Enter Account Holder's Name : prit Enter Initial Balance : 500 Account created successfully Enter your Choice (1 to 5): 1 Enter Account Number : 205 Enter Account Holder's Name : shivi Enter Initial Balance : 650 Account created successfully Enter your Choice (1 to 5): 2 Enter Account Number : 101 Enter Amount to Deposit : 250 Deposited 250.0 successfully. New balance : 750.0 Enter your Choice (1 to 5): 3 Enter Account Number : 205 Enter Amount to Withdraw : 100 Withdrew 100.0 successfully. New balance : 550.0 Enter your Choice (1 to 5): 4 Enter Account Number : 101 Account Holder : prit Balance : 750.0 Enter your Choice (1 to 5): 5 Exiting the program. ```
Programme _21	In a real-world scenario like a payment processing system, different payment methods (Credit Card, PayPal, Bank Transfer) can all implement the same method, such as process_payment, but each one has a different implementation. Write a python program to use polymorphism to handle payments from various sources uniformly.
Code:-	from abc import ABC, abstractmethod # Base class defining the common interface for all payment methods

```
class PaymentMethod(ABC):
    @abstractmethod
    def process_payment(self, amount):
        pass
# Credit Card payment method
class CreditCardPayment(PaymentMethod):
    def __init__(self, card_number, card_holder):
        self.card_number = card_number
        self.card_holder = card_holder
    def process_payment(self, amount):
        print(f"Processing payment of {amount} using Credit Card.")
        print(f"Card Holder: {self.card_holder}, Card Number: {self.card_number[-4:]}")  # Last 4 digits only
        print(f"Payment of {amount} has been successfully processed using Credit Card.\n")
# PayPal payment method
class PayPalPayment(PaymentMethod):
    def __init__(self, email):
        self.email = email
    def process_payment(self, amount):
        print(f"Processing payment of {amount} using PayPal.")
        print(f"PayPal Account: {self.email}")
        print(f"Payment of {amount} has been successfully processed using PayPal.\n")
# Bank Transfer payment method
class BankTransferPayment(PaymentMethod):
    def __init__(self, bank_account, bank_name):
        self.bank_account = bank_account
        self.bank_name = bank_name
    def process_payment(self, amount):
        print(f"Processing payment of {amount} using Bank Transfer.")
        print(f"Bank Account: {self.bank_account}, Bank Name: {self.bank_name}")
        print(f"Payment of {amount} has been successfully processed using Bank Transfer.\n")
# Payment Processor class that accepts different payment methods
class PaymentProcessor:
    def process(self, payment_method, amount):
        payment_method.process_payment(amount)
# Main function to simulate payment processing
def main():
    # Different payment methods
    credit_card = CreditCardPayment("1234-5678-9876-5432", "John Doe")
    paypal = PayPalPayment("john.doe@example.com")
```

```
    bank_transfer = BankTransferPayment("9876543210", "Example Bank")
    # Payment Processor
    payment_processor = PaymentProcessor()
    # Process payments through different methods
    print("Processing payments with different payment methods:\n")
    payment_processor.process(credit_card, 100.50)
    payment_processor.process(paypal, 200.75)
    payment_processor.process(bank_transfer, 500.00)
if __name__ == "__main__":
    main()
```

```
Processing payments with different payment methods:

Processing payment of 100.5 using Credit Card.
Card Holder: John Doe, Card Number: 5432
Payment of 100.5 has been successfully processed using Credit Ca

Processing payment of 200.75 using PayPal.
PayPal Account: john.doe@example.com
Payment of 200.75 has been successfully processed using PayPal.

Processing payment of 500.0 using Bank Transfer.
Bank Account: 9876543210, Bank Name: Example Bank
Payment of 500.0 has been successfully processed using Bank Tran
```

Programme _22

Write a Python Program to demonstrate different types of exception handing.

Code:-

```
def divide_numbers():
    try:
        # Taking two numbers as input from the user
        num1 = int(input("Enter the first number: "))
        num2 = int(input("Enter the second number: "))
        # Trying to divide numbers
        result = num1 / num2
    except ZeroDivisionError:
        # Handle division by zero error
        print("Error: Division by zero is not allowed.")
    except ValueError:
        # Handle invalid input (non-integer) error
        print("Error: Please enter valid integers.")
    else:
        # This block executes if no exception occurred
        print(f"Result: {num1} / {num2} = {result}")
    finally:
            # This block always executes, regardless of whether an exception
occurred or not
```

```
        print("Execution completed.")
def read_file():
    try:
        # Trying to open a non-existent file
        with open("non_existent_file.txt", "r") as file:
            content = file.read()
    except FileNotFoundError:
        # Handle file not found error
        print("Error: The file does not exist.")
    except IOError:
        # Handle input-output errors
        print("Error: An I/O error occurred.")
    else:
        # This block executes if no exception occurred
        print("File read successfully.")
    finally:
        # This block always executes
        print("File operation completed.")
def main():
    print("Demonstrating exception handling for division:")
    divide_numbers()
    print("\nDemonstrating exception handling for file operations:")
    read_file()
if __name__ == "__main__":
    main()
```

```
Demonstrating exception handling for division:
Enter the first number: 1
Enter the second number: 2
Result: 1 / 2 = 0.5
Execution completed.

Demonstrating exception handling for file operations:
Error: The file does not exist.
File operation completed.
```

Programme _23	Define exception handling in python. Also write a user defined exception program in python which will except age as an input from the user and check whether the user is eligible for voting or not. If age < 18 it should raise the exception as 'Not eligible for voting'.

Code:-	(1) Python program to write data to a file
	Code:- F=open("drinks.dat","w")
	while(True):
	v=input("Enter Drink Name : ")
	if(v==""):
	break

```
    F.write(v+"\n")
F.close()
```

Output:-

```
Enter Drink Name : cocola
Enter Drink Name :
```

(2) Python program to create a new file in another directory

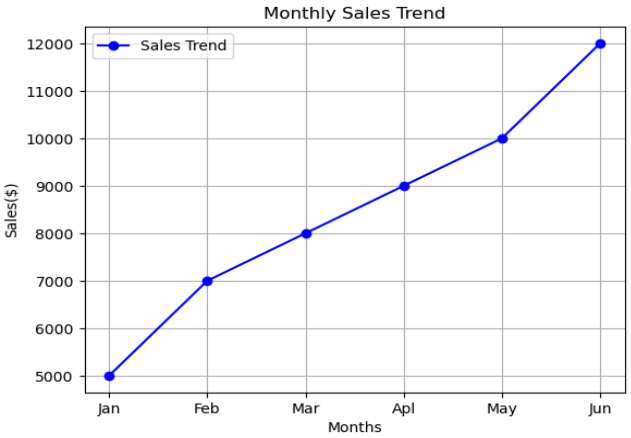
Code

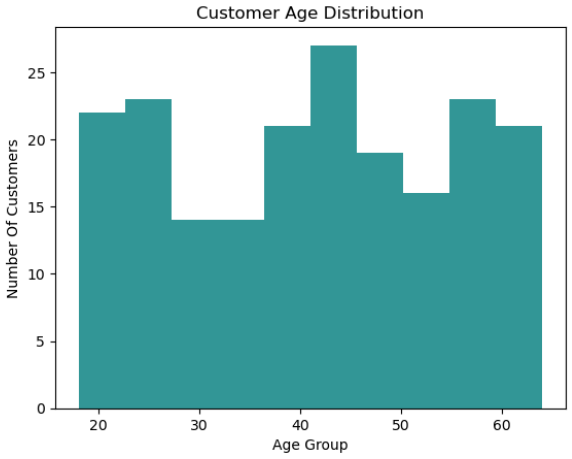
```
# importing os library
import os
def main():
    # creating a new directory
    os.mkdir("pythonFiles")
    # Changing current path to the directory
    os.chdir("pythonFiles")
    # creating a new file for writing operation
    fo = open("demo.txt","w")
    fo.write("This is demo File")
    fo.close()

    # printing the current file directory
    print("Current Directory :",os.getcwd())

if __name__=="__main__":main()
```

```
Current Directory : C:\Users\HP\pythonFiles\pythonFiles
```

Unit – 4

Programme _01	Write a list of Matplotlib programs you can use in Jupyter Notebook, focusing on real-world applications of data visualization.
(1)	Sales Trend Line Chart **Code:-** import matplotlib.pyplot as plt months=['Jan','Feb','Mar','Apl','May','Jun'] sales=[5000,7000,8000,9000,10000,12000] plt.plot(months,sales,marker='o',label='Sales Trend',color='blue') plt.title('Monthly Sales Trend') plt.xlabel('Months') plt.ylabel('Sales($)') plt.legend() plt.grid(True) plt.show()

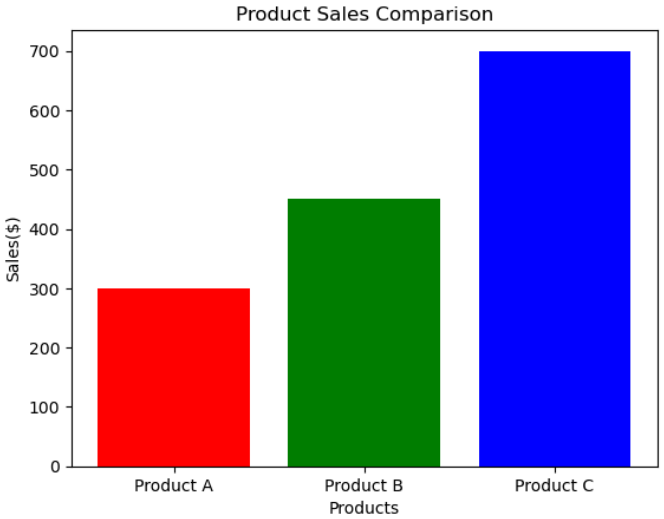
	Output: - 

(2)	Customer Age Distribution (Histogram)
	Code:-
	import matplotlib.pyplot as plt
	import numpy as np
	#data
	ages=np.random.randint(18,65,size=200)
	#plotting
	plt.hist(ages,bins=10,color='teal',alpha=0.8)
	plt.title('Customer Age Distribution')
	plt.xlabel('Age Group')
	plt.ylabel('Number Of Customers')
	plt.show()
	Output: -
	 

(3)	Product Sales Comparison (Bar Chart)
	#product sales comparison(bar chart)
	products=['Product A','Product B','Product C']
	sales=[300,450,700]
	plt.bar(products,sales,color=['red','green','blue'])
	plt.title('Product Sales Comparison')

```
plt.xlabel('Products')
plt.ylabel('Sales($)')
plt.show()
```

output:-



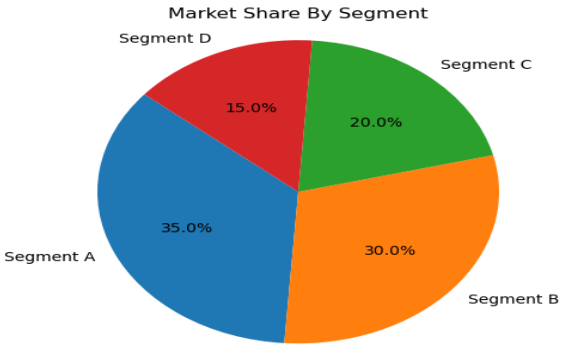
(4)

Market Share (Pie Chart)

Code:-

```
import matplotlib.pyplot as plt
segments=['Segment A','Segment B','Segment C','Segment D']
market_share=[35,30,20,15]
plt.pie(market_share,labels=segments,autopct='%1.1f%%',startangle=140)
plt.title('Market Share By Segment')
plt.axis('equal')
plt.show()
```

output:-



(5)

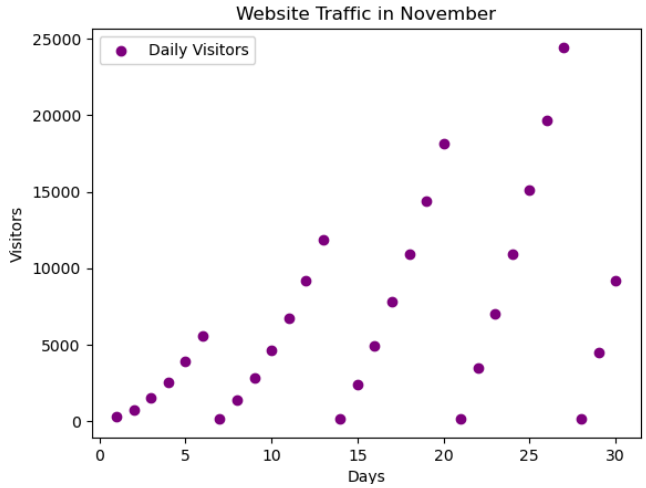
Website Traffic (Scatter Plot)

Code:-

```
import matplotlib.pyplot as plt
days = list(range(1, 31))
visitors = [150 + 10 * i * (i % 7) * 15 for i in range(1, 31)]  # Fixed the expression
plt.scatter(days, visitors, label='Daily Visitors', color='purple')
```

```
plt.title('Website Traffic in November')
plt.xlabel('Days')
plt.ylabel('Visitors')
plt.legend()
plt.show()
```

Output:-

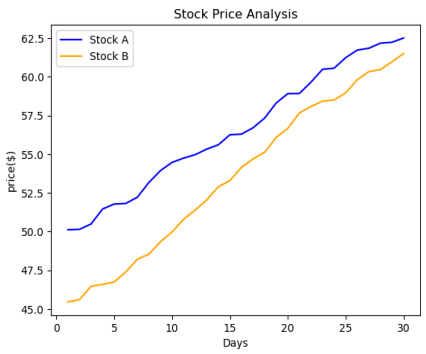


(6) Stock Price Analysis (Multiple Line Plots)

Code:-

```
#stock Price analysis
import matplotlib.pyplot as plt
import numpy as np
days=np.arange(1,31)
stock_a=50 + np.random.rand(30).cumsum()
stock_b=45 + np.random.rand(30).cumsum()
plt.plot(days,stock_a,label='Stock A',color='blue')
plt.plot(days,stock_b,label='Stock B',color='orange')
plt.title('Stock Price Analysis')
plt.xlabel('Days')
plt.ylabel('price($)')
plt.legend()
plt.show()
```

output:-



Programme _02	Write a list of NumPy programs designed to showcase its real-world applications in a Jupyter Notebook environment.

(1)	Basic Array Operations
	Code: -
	import numpy as np
	
	first_array = np.array([1, 3, 5, 7])
	second_array = np.array([2, 4, 6, 8])
	
	array1 = np.array([1, 2, 3])
	
	# using the + operator
	result1 = first_array + second_array
	print("Using the + operator:",result1)
	
	# using the - operator
	result1 = first_array - second_array
	print("Using the - operator:",result1)
	
	# using the * operator
	result1 = first_array * second_array
	print("Using the * operator:",result1)
	
	# using the / operator
	result1 = first_array / second_array
	print("Using the / operator:",result1)
	
	# using the ** operator
	result1 = array1 ** 2
	print("Using the ** operator:",result1)
	Output :-
	Using the + operator: [3 7 11 15]
	Using the - operator: [-1 -1 -1 -1]
	Using the * operator: [2 12 30 56]
	Using the / operator: [0.5 0.75 0.83333333 0.875]
	Using the ** operator: [1 4 9]

(2)	Generate a Sequence of Numbers
	Code:-
	import numpy as np
	
	# Generate a sequence of numbers from 0 to 9
	sequence = np.arange(10)
	print("Sequence:", sequence)

	# Generate a sequence from 5 to 20 with step size 2
	sequence_with_step = np.arange(5, 21, 2)
	print("Sequence with step:", sequence_with_step)
	Output:-
	Sequence: [0 1 2 3 4 5 6 7 8 9]
	Sequence with step: [5 7 9 11 13 15 17 19]

(3)	Simulate Dice Rolls
	Code:-
	import numpy as np
	
	# Simulate rolling a 6-sided die 10 times
	rolls = np.random.randint(1, 7, size=10)
	print("Rolls:", rolls)
	num_dice = 3
	rolls = [random.randint(1, 6) for _ in range(num_dice)]
	print("Rolls:", rolls)
	Output:-
	Rolls: [4 5 5 6 4 3 6 3 6 6]
	Rolls: [6, 6, 4]

(4)	Matrix Operations
	Code:-
	import numpy as np
	
	# Create a 2x2 matrix
	matrix1 = np.array([[1, 2], [3, 4]])
	
	# Create another 2x2 matrix
	matrix2 = np.array([[5, 6], [7, 8]])
	
	print("Matrix 1:\n", matrix1)
	print("Matrix 2:\n", matrix2)
	
	# Matrix addition
	sum_matrix = matrix1 + matrix2
	
	# Matrix subtraction
	diff_matrix = matrix1 - matrix2
	
	print("Matrix Addition:\n", sum_matrix)
	print("Matrix Subtraction:\n", diff_matrix)

	elementwise_mult = matrix1 * matrix2 print("Element-wise Multiplication:\n", elementwise_mult) dot_product = np.dot(matrix1, matrix2) print("Dot Product:\n", dot_product) Output:- Matrix 1: [[1 2] [3 4]] Matrix 2: [[5 6] [7 8]] Matrix Addition: [[6 8] [10 12]] Matrix Subtraction: [[-4 -4] [-4 -4]] Element-wise Multiplication: [[5 12] [21 32]] Dot Product: [[19 22] [43 50]]
(5)	Statistical Analysis
Programme _03	Write a list of Pandas programs designed to showcase real-world applications in a Jupyter Notebook environment.

(1)	Create a DataFrame
	Code:-
	#Create a DataFrame
	import pandas as pd
	#CREATE DATAFRAME
	data={
	"Name":["Charmi","Priyanka","Preet","Zarna","Riddhi"],
	"Age":[21,18,24,16,23],
	"Salary":[1500,1800,2566,7800,5500],
	"Department":["CE","ME","MCA","MSC-IC","BBA"]
	}
	df=pd.DataFrame(data)
	print(df)
	Output:-

	<pre> Name Age Salary Department 0 Charmi 21 1500 CE 1 Priyanka 18 1800 ME 2 Preet 24 2566 MCA 3 Zarna 16 7800 MSC-IC 4 Riddhi 23 5500 BBA</pre>

(2)	Read Data from a CSV File
	Code:-
	import pandas as pd
	#Display First 5 Row
	print(df.head())
	Output:-
	<pre> Name Age Salary Department
	0 Charmi 21 1500 CE
	1 Priyanka 18 1800 ME
	2 Preet 24 2566 MCA
	3 Zarna 16 7800 MSC-IC
	4 Riddhi 23 5500 BBA</pre>

(3)	Filter Data Based on Conditions
	Code:-
	import pandas as pd
	data={
	"Name":["Charmi","Priyanka","Preet","Zarna","Riddhi"],
	"Age":[21,18,24,16,23],
	"Salary":[1500,1800,2566,7800,5500],
	"Department":["CE","ME","MCA","MSC-IC","BBA"]
	}
	df=pd.DataFrame(data)
	#Filters employess with salary
	high_earners=df[df["Salary"]>50000]
	print(high_earners)
	Output:-
	<pre>Empty DataFrame
	Columns: [Name, Age, Salary, Department]
	Index: []</pre>

(4)	Group Data and Calculate Aggregates
	Code:-
	#Group data And Calculate Aggreations
	import pandas as pd
	data={
	"Department":["CE","ME","MCA","MSC-IC","BBA"],
	"Salary":[1500,1800,2566,7800,5500]
	}

df=pd.DataFrame(data)
#Group By Department And Calculate Avareage Salary
avg_salary=df.groupby("Department")["Salary"].mean()
print(avg_salary)
Output:-

```
Department
BBA       5500.0
CE        1500.0
MCA       2566.0
ME        1800.0
MSC-IC    7800.0
Name: Salary, dtype: float64
```

(5)

Add a New Column
Code:-

```
#Add A New Column
import pandas as pd
data={
  "Department":["CE","ME","MCA","MSC-IC","BBA"],
  "Salary":[1500,1800,2566,7800,5500],
  "Bonus":[5000,6000,7000,9000,1500]
  }
df=pd.DataFrame(data)
df["Total Compensation"]=df["Salary"]+df["Bonus"]
print(df)
```

Output:-

```
   Department  Salary  Bonus  Total Compensation
0          CE    1500   5000                6500
1          ME    1800   6000                7800
2         MCA    2566   7000                9566
3      MSC-IC    7800   9000               16800
4         BBA    5500   1500                7000
```

(6)

Handle Missing Data
Code:-

```
#Handle Missing Data
import pandas as pd
data={
   "Name":["Charmi","Priyanka","Preet","Zarna","Riddhi"],
  "Age":[21,18,24,16,23],
  "Salary":[1500,1800,2566,7800,5500]
  }
df=pd.DataFrame(data)
#Handle Missing data
df["Age"].fillna(df["Age"].mean(),inplace=True)
df.dropna(subset=["Name"],inplace=True)
```

	print(df)
	Output:-
	`  Name  Age  Salary`
	`0  Charmi  21  1500`
	`1  Priyanka  18  1800`
	`2  Preet  24  2566`
	`3  Zarna  16  7800`
	`4  Riddhi  23  5500`

(7)	Sort Data
	Code:-
	#Sort Data
	import pandas as pd
	data={
	"Name":["Charmi","Priyanka","Preet","Zarna","Riddhi"],
	"Age":[21,18,24,16,23],
	"Salary":[1500,1800,2566,7800,5500]
	}
	df=pd.DataFrame(data)
	#sorted by salary
	sorted_df=df.sort_values(by="Salary",ascending=False)
	print(sorted_df)
	Output:-
	`  Name  Age  Salary`
	`3  Zarna  16  7800`
	`4  Riddhi  23  5500`
	`2  Preet  24  2566`
	`1  Priyanka  18  1800`
	`0  Charmi  21  1500`

(8)	Merge Two DataFrames
	Code:-
	#merge To dataframe
	import pandas as pd
	df1=pd.DataFrame({
	"Employee Id":[1,2,3],
	"Name":["Charmi","Jinal","Bansi"]
	})
	
	df2=pd.DataFrame({
	"Employee Id":[1,12,13],
	"Name":["priyanka","lipsa","Bansari"]
	})
	
	merge_df=pd.merge(df1,df2,on="Employee Id",how="inner")

	print(merge_df)
	Output:-
	`  Employee Id  Name_x  Name_y`
	`0  1  Charmi  priyanka`

(9)	Pivot Table
	Code:-
	#pivot table
	import pandas as pd
	data = {
	"Department": ["HR", "HR", "IT", "IT", "Sales", "Sales"],
	"Month":["jan","feb","mar","apl","aug","sep"],
	"Salary":[1500,1800,2566,7800,5500,8000]
	}
	df=pd.DataFrame(data)
	pivot=df.pivot_table(values="Salary",index="Department",columns="Month",aggfunc="mean")
	print(pivot)
	`Month  apl  aug  feb  jan  mar  sep`
	`Department`
	`HR  NaN  NaN  1800.0  1500.0  NaN  NaN`
	`IT  7800.0  NaN  NaN  NaN  2566.0  NaN`
	`Sales  NaN  5500.0  NaN  NaN  NaN  8000.0`

(10)	Export Data to a CSV File
	#export data to csv file
	import pandas as pd
	data = {
	"Department": ["HR", "HR", "IT", "IT", "Sales", "Sales"],
	"Month": ["jan", "feb", "mar", "apr", "aug", "sep"],
	"Salary": [1500, 1800, 2566, 7800, 5500, 6000]
	}
	df = pd.DataFrame(data)
	df.to_csv("employee_data.csv",index=False)
	print("Data Exported to 'employee.csv'")
	`Data Exported to 'employee.csv'`

(11)	Time Series Analysis Code:- import pandas as pd import numpy as np # Create a date range

```
dates = pd.date_range(start="2023-01-01", periods=10)

# Generate random sales data
sales = np.random.randint(100, 500, size=10)

# Create a DataFrame
df = pd.DataFrame({
    "Date": dates,
    "Sales": sales
})

# Calculate the moving average with a specified window size
window = 3  # Example window size
df["Moving Average"] = df["Sales"].rolling(window=window).mean()

# Print the DataFrame
print(df)
```

```
         Date  Sales  Moving Average
0 2023-01-01    183             NaN
1 2023-01-02    299             NaN
2 2023-01-03    183      221.666667
3 2023-01-04    461      314.333333
4 2023-01-05    367      337.000000
5 2023-01-06    383      403.666667
6 2023-01-07    148      299.333333
7 2023-01-08    234      255.000000
8 2023-01-09    113      165.000000
9 2023-01-10    103      150.000000
```

(13)

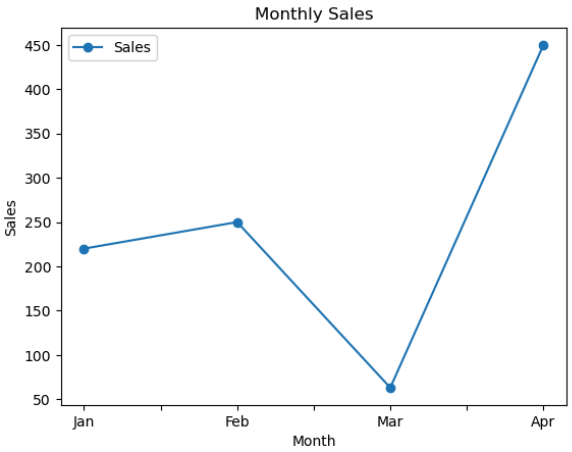
Visualization with Pandas

Code:-

```
import pandas as pd
import matplotlib.pyplot as plt

# Data
data = {
    "Month": ["Jan", "Feb", "Mar", "Apr"],
    "Sales": [220, 250, 63, 450]
}


df = pd.DataFrame(data)

# Line Chart
df.plot(x="Month", y="Sales", kind="line", marker="o")
```

```
plt.title("Monthly Sales")
plt.xlabel("Month")
plt.ylabel("Sales")
plt.show()
```

Code:-



(14)

Describe Data

```
#describe Data
import pandas as pd
data = {
   "Age":[24,45,15,20,18,21],
   "Salary":[45000,4578,45000,12500,80000,65750]
}
df = pd.DataFrame(data)
description = df.describe()
print(description)
```

Output:-

```
             Age        Salary
count   6.000000      6.000000
mean   23.833333  42138.000000
std    10.796604  29314.175138
min    15.000000   4578.000000
25%    18.500000  20625.000000
50%    20.500000  45000.000000
75%    23.250000  60562.500000
max    45.000000  80000.000000
```

(15)

Apply Functions to Columns

Code:-

```
import pandas as pd


data = {
   "Age": [24, 45, 15, 20, 18, 21],
```

```
    "Salary": [45000, 4578, 45000, 12500, 80000, 65750]
}


df = pd.DataFrame(data)


df["Updated Salary"] = df["Salary"].apply(lambda x: x * 1.10)


print(df)
```

```
   Age  Salary  Updated Salary
0   24   45000         49500.0
1   45    4578          5035.8
2   15   45000         49500.0
3   20   12500         13750.0
4   18   80000         88000.0
5   21   65750         72325.0
```

Submission Date:-	Faculty Signature:-