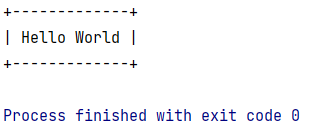
PRACTICAL SET - 01

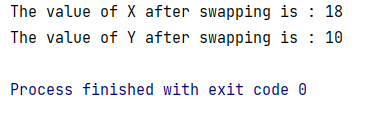
**# Set-1 Practical-1 : Write a Python program to print “Hello World”.**print ("+-------------+")  
print ("| Hello World |")  
print ("+-------------+")

OUTPUT:



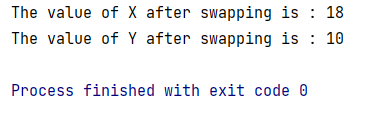
**# Set-1 Practical-2 : Write a Python program to swap two variables using third variable.**X=10  
Y=18  
  
Z=X  
X=Y  
Y=Z  
  
print (**"The value of X after swapping is:"**, X)  
print (**"The value of Y after swapping is:"**, Y)

OUTPUT:



**# Set-1 Practical-3 : Write a Python program to swap two variables without third variable.**X = 10  
Y = 18  
  
X = X + Y  
Y = X - Y  
X = X - Y  
  
print (**"The value of X after swapping is :"**, X)  
print (**"The value of Y after swapping is :"**, Y)

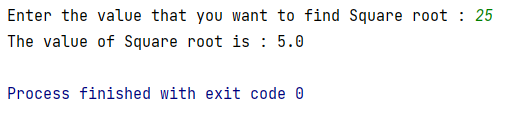
OUTPUT:



**# Set-1 Practical-4 : Write a Python program to find square root of positive number.**

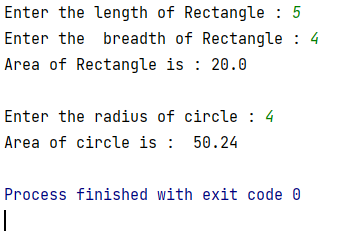
A = float(input(**"Enter the value that you want to find Square root : "**))  
sqrt = A\*\*0.5  
print (**"The value of Square root is :"**, sqrt)  
  
*# Using sqrt() function:  
# import math  
# A = float(input("Enter the value that you want to find Square root : "))  
# sqrt = math.sqrt(A)  
# print ("The value of Square root is :", sqrt)*

OUTPUT:



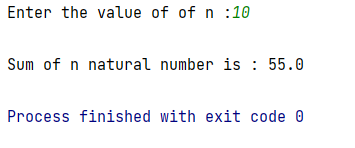
**# Set-1 Practical-5 : Write a Python program to find area of a rectangle and circle.** *# Area of Rectangle :*L = float(input(**"Enter the length of Rectangle : "**))  
B = float(input(**"Enter the breadth of Rectangle : "**))  
R = L\*B  
print(**"Area of Rectangle is :"**, R)  
  
*# Area of circle :*r = float(input(**"\nEnter the radius of circle : "**))  
C = 3.14\*r\*r  
print(**"Area of circle is : "**, C)

OUTPUT:



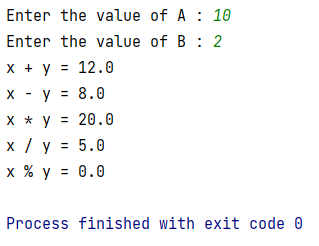
**# Set-1 Practical-6 : Write a Python program to find sum of n natural numbers without loop.**n = float(input(**"Enter the value of of n :"**))  
n = n\*(n+1)/2  
  
print (**"\nSum of n natural number is :"**, n)

OUTPUT:



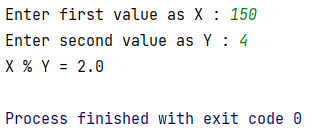
**# Set-1 Practical-7 : Check various arithmetic operators of Python.**A = float(input(**"Enter the value of A : "**))  
B = float(input(**"Enter the value of B : "**))  
  
print(**'x + y ='**, A + B)  
print(**'x - y ='**, A - B)  
print(**'x \* y ='**, A \* B)  
print(**'x / y ='**, A / B)  
print(**'x % y ='**, A % B)

OUTPUT:



**# Set-1 Practical-8 : Write a Python program to check output of modulo operator.**X = float(input(**"Enter first value as X : "**))  
Y = float(input(**"Enter second value as Y : "**))  
print(**"X % Y ="**, X % Y)

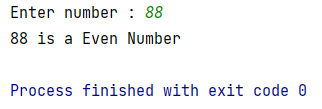
OUTPUT:



PRACTICAL SET - 02

**# Set-2 Practical-1 : Write a Python program to check whether entered number is even or odd.**a = int(input(**"Enter number : "**))  
print(a, **"is a Even Number"**) **if** a % 2 == 0 **else** print(a, **"is a Odd Number"**)

OUTPUT:

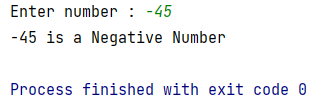


**# Set-2 Practical-2 : Write a Python program to find whether entered number is positive, negative or zero.**

a = int(input(**"Enter number : "**))

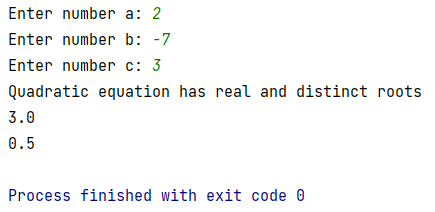
print(a, **"is a Negative Number"**) **if** a < 0 **else** print(a, **"is a Positive Number "**) **if** a > 0 **else** print(a, **"Number is Zero"**)

OUTPUT:



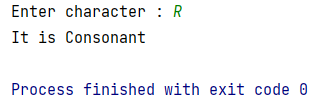
**# Set-2 Practical-3 : Write a Python program to find roots of quadratic equations if roots are real.****def** root(a, b, c):  
 d = b \*\* 2 - 4 \* a \* c  
 sqrt\_d = abs(d) \*\* 0.5  
  
 **if** d > 0:  
 print(**"Quadratic equation has real and distinct roots"**)  
 print((-b + sqrt\_d) / (2 \* a))  
 print((-b - sqrt\_d) / (2 \* a))  
  
 **if** d == 0:  
 print(**"Quadratic equation has real and one roots"**)  
 print(-b / (2 \* a))  
  
 **if** d < 0:  
 print(**"Quadratic equation has imaginary and distinct roots"**)  
 print(-b / (2 \* a), **"+ i"**, sqrt\_d / (2 \* a))  
 print(-b / (2 \* a), **"- i"**, sqrt\_d / (2 \* a))  
  
  
p = int(input(**"Enter number a: "**))  
q = int(input(**"Enter number b: "**))  
r = int(input(**"Enter number c: "**))  
  
**if** p == 0:  
 print(**"Enter correct quadratic equation."**)  
**else**:  
 root(p, q, r)

OUTPUT:



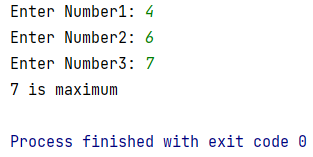
**# Set-2 Practical-4 : Write a Python program to check whether entered character is vowel or consonant.**a = input(**"Enter character : "**)  
**if** (a == **'a'**) | (a == **'e'**) | (a == **'i'**) | (a == **'o'**) | (a == **'u'**) | (a == **'A'**) | (a == **'E'**) | (a == **'I'**) | (a == **'O'**) | (a == **'U'**):  
 print(**"It is Vowel"**)  
**else**:  
 print(**"It is Consonant"**)

OUTPUT:



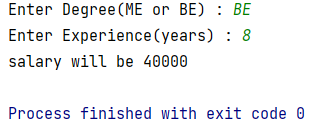
**# Set-2 Practical-5 : Write a Python program to find maximum of three numbers (nested if-else).**a = int(input(**"Enter Number1: "**))  
b = int(input(**"Enter Number2: "**))  
c = int(input(**"Enter Number3: "**))  
**if** a > b:  
 **if** a > c:  
 print(a, **"is maximum"**)  
 **elif** c > b:  
 print(c, **"is maximum"**)  
**elif** b > c:  
 print(b, **"is maximum"**)  
**else**:  
 print(c, **"is maximum"**)

OUTPUT:



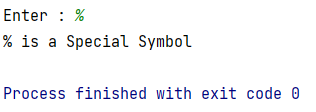
**# Set-2 Practical-6 : Write a Python program to calculate the salary of an employee based on following conditions  
# (nested if-else):  
# 1. if degree = B.E. and experience < 5 years, salary=30000  
# 2. if degree = B.E. and experience >= 5 years, salary=40000  
# 3. if degree = M.E. and experience < 5 years, salary=50000  
# 4. if degree = M.E. and experience >= 5 years, salary= 60000**degree = input(**"Enter Degree(ME or BE) : "**)  
exp = int(input(**"Enter Experience(years) : "**))  
  
**if** degree == **'BE'**:  
 **if** exp < 5:  
 print(**"salary will be 30000"**)  
 **elif** exp >= 5:  
 print(**"salary will be 40000"**)  
**elif** degree == **'ME'**:  
 **if** exp < 5:  
 print(**"salary will be 50000"**)  
 **elif** exp >= 5:  
 print(**"salary will be 60000"**)  
**else**:  
 print(**"Enter valid Degree."**)

OUTPUT:



**# Set-2 Practical-7 : Write a Python program to check whether entered input is character, digit or special symbol using ladder if-else.**a = input(**"Enter : "**)  
**if** a.isalpha():  
 print(a, **"is a Character"**)  
**elif** a.isnumeric():  
 print(a, **"is a Number"**)  
**else**:  
 print(a, **"is a Special Symbol"**)

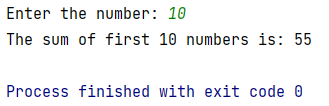
OUTPUT:



PRACTICAL SET - 03

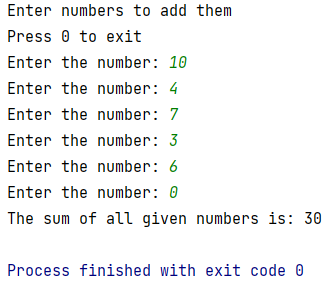
**# Set-3 Practical-1 : Write a Python program to find sum of first N numbers.**su = 0  
n = int(input(**"Enter the number: "**))  
a = n  
**while** n > 0:  
 su = su + n  
 n = n - 1  
print(**"The sum of first"**, a, **"numbers is:"**, su)

OUTPUT:



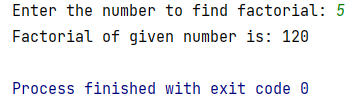
**# Set-3 Practical-2 : Write a Python program to find sum of N scanned numbers.**su = 0  
print(**"Enter numbers to add them"**)  
print(**"Press 0 to exit"**)  
**while True**:  
 n = int(input(**"Enter the number: "**))  
 **if** n == 0:  
 **break** su = su + n  
print(**"The sum of all given numbers is:"**, su)

OUTPUT:



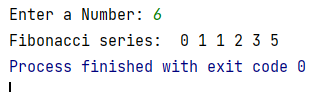
**# Set-3 Practical-3 : Write a Python program to find N!**num = int(input(**"Enter the number to find factorial: "**))  
factorial = 1  
**if** num == 0 **or** num == 1:  
 print(**"Factorial of given number is:"**, factorial)  
**else**:  
 **for** i **in** range(1, num+1):  
 factorial = factorial \* i  
 print(**"Factorial of given number is:"**, factorial)

OUTPUT:



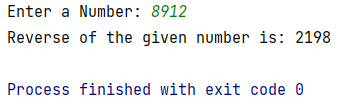
**# Set-3 Practical-4 : Write a Python program to print Fibonacci series upto n terms.****def** fibo(i):  
 **if** i <= 1:  
 **return** i  
 **else**:  
 **return** fibo(i - 1) + fibo(i - 2)  
  
  
num = int(input(**"Enter a Number: "**))  
**if** num <= 0:  
 print(**"Enter a positive number."**)  
**else**:  
 print(**"Fibonacci series: "**, end=**" "**)  
 **for** n **in** range(num):  
 print(fibo(n), end=**" "**)

OUTPUT:



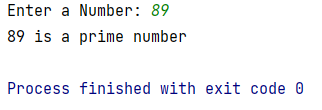
**# Set-3 Practical-5 : Write a Python program to find the reverse of given numbers (Example 2564-4652).**n = int(input(**"Enter a Number: "**))  
rev = 0  
**while** n > 0:  
 dig = n % 10  
 rev = rev \* 10 + dig  
 n = n // 10  
print(**"Reverse of the given number is:"**, rev)

OUTPUT:



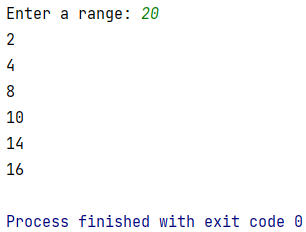
**# Set-3 Practical-6 : Write a Python program to check whether entered number is prime or not.**num = int(input(**"Enter a Number: "**))  
**if** num > 2:  
 **for** i **in** range(2, int(num/2)+1):  
 **if** num % i == 0:  
 print(num, **"is not a prime number"**)  
 **break  
 else**:  
 print(num, **"is a prime number"**)  
**else**:  
 print(num, **"is not a prime number"**)

OUTPUT:



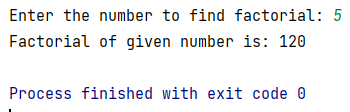
**# Set-3 Practical-7 : Write a Python program to print all even numbers between 1 to n except the numbers divisible by 6.**n = int(input(**"Enter a range: "**))  
**for** i **in** range(2, n, 2):  
 **if** i % 6 != 0:  
 print(i)

OUTPUT:



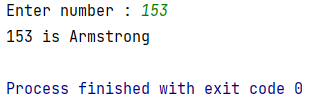
**# Set-3 Practical-8 : Write a Python program to calculate N!.**num = int(input(**"Enter the number to find factorial: "**))  
factorial = 1  
**if** num == 0 **or** num == 1:  
 print(**"Factorial of given number is:"**, factorial)  
**else**:  
 **for** i **in** range(1, num+1):  
 factorial = factorial \* i  
 print(**"Factorial of given number is:"**, factorial)

OUTPUT:



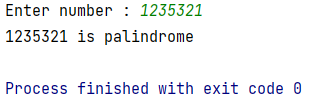
**# Set-3 Practical-9 : Write a Python program to check whether given number is Armstrong or not.**sum = 0  
count = 0  
n = int(input(**"Enter number : "**))  
temp = n  
**while** n != 0:  
 count += 1  
 n = n // 10  
n = temp  
**while** n != 0:  
 rev = n % 10  
 sum = rev \*\* count + sum  
 n = n // 10  
**if** sum == temp:  
 print(temp, **"is Armstrong"**)  
**else**:  
 print(temp, **"is not Armstrong"**)

OUTPUT:



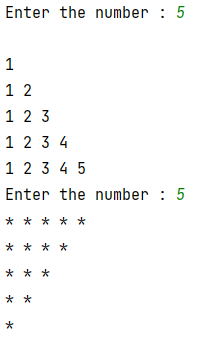
**# Set-3 Practical-10 : Write a Python program to check whether given number is Palindrome or not.**rev = 0  
n = int(input(**"Enter number : "**))  
temp = n  
**while** n != 0:  
 rev = rev \* 10 + n % 10  
 n = n // 10  
**if** rev == temp:  
 print(temp, **"is palindrome"**)  
**else**:  
 print(temp, **"is not palindrome"**)

OUTPUT:



**# Set-3 Practical-11 : Write a Python program to print the following:  
# 1) 1 2) \* \* \* \* \*  
# 1 2 \* \* \* \*  
# 1 2 3 \* \* \*  
# 1 2 3 4 \* \*  
# 1 2 3 4 5 \***n = int(input(**"Enter the number : "**))  
**for** i **in** range(n + 1):  
 **for** j **in** range(1, i + 1):  
 print(j, end=**' '**)  
 print()  
  
n = int(input(**"Enter the number : "**))  
**for** i **in** range(n + 1):  
 **for** j **in** range(i, n):  
 print(**"\*"**, end=**' '**)  
 print()

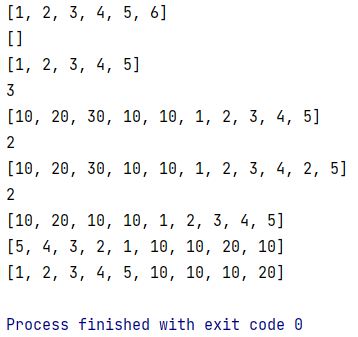
OUTPUT:



PRACTICAL SET – 04

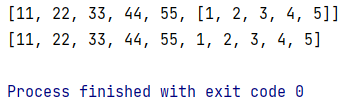
**# Set-4 Practical-1 : Write a Python program which covers all the methods (functions) of list.**L1 = [1, 2, 3, 4, 5]  
  
*# Adds an element at the end of the list*L1.append(6)  
print(L1)  
  
*# Removes all the elements from the list*L1.clear()  
print(L1)  
  
*# Returns a copy of the list*L1 = [1, 2, 3, 4, 5]  
L2 = L1.copy()  
print(L2)  
  
*# Returns the number of elements with the specified value*L3 = [10, 20, 30, 10, 10]  
n = L3.count(10)  
print(n)  
  
*# Add the elements of a list (or any iterable), to the end of the current list*L3.extend(L1)  
print(L3)  
  
*# Returns the index of the first element with the specified value*print(L3.index(30))  
  
*# Adds an element at the specified position insert(pos,value)*L3.insert(9, 2)  
print(L3)  
  
*# Removes the element at the specified position*print(L3.pop(9))  
  
*# Removes the item with the specified value*L3.remove(30)  
print(L3)  
  
*# Reverses the order of the list*L3.reverse()  
print(L3)  
  
*# sorted list*L3.sort()  
print(L3)

OUTPUT:



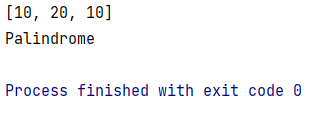
**# Set-4 Practical-2 : Write a Python program to append a list to the second list.**L1 = [1, 2, 3, 4, 5]  
L2 = [11, 22, 33, 44, 55]  
  
L2.append(L1)  
print(L2)  
  
L1 = [1, 2, 3, 4, 5]  
L2 = [11, 22, 33, 44, 55]  
  
L2.extend(L1)  
print(L2)

OUTPUT:



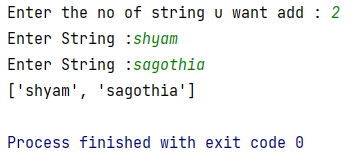
**# Set-4 Practical-3 : Write a Python program to check whether the given list is palindrome or not.**l = [10, 20, 10]  
r = l[::-1]  
print(r)  
**if** l == r:  
 print(**"Palindrome"**)  
**else**:  
 print(**"Not palindrome"**)

OUTPUT:



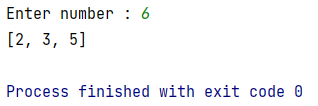
**# Set-4 Practical-4 : Write a Python program to store strings in list and then print them.**n = int(input(**"Enter the no of string u want add : "**))  
L1 = []  
  
**for** i **in** range(n):  
 str = input(**"Enter String :"**)  
 L1.append(str)  
  
print(L1)

OUTPUT:



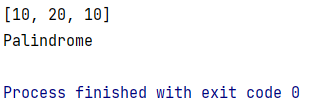
**# Set-4 Practical-5 : Write a Python program to print list of prime numbers upto N using loop and else clause.**l = []  
n = int(input(**"Enter number : "**))  
  
**for** i **in** range(2, n + 1):  
 **for** j **in** range(2, (i // 2) + 1):  
 **if** i % j == 0:  
 **break  
 else**:  
 l.append(i)  
print(l)

OUTPUT:



**# Set-4 Practical-6 : Write a Python program to check whether the given list is palindrome or not.**l = [10, 20, 10]  
r = l[::-1]  
print(r)  
**if** l == r:  
 print(**"Palindrome"**)  
**else**:  
 print(**"Not palindrome"**)

OUTPUT:



**# Set-4 Practical-7 : Write a Python program to multiply all the items in a list.**L1 = [1, 2, 3, 4, 5]  
ans = 1  
**for** i **in** L1:  
 ans \*= i  
  
print(ans)

OUTPUT:

120

**# Set-4 Practical-8 : Write a Python program to get the largest number from a list.**l = [99, 50, 30, 75, 22]  
l.sort()  
m = l[-1]  
print(m)

OUTPUT:

99

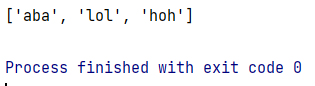
**# Set-4 Practical-9 : Write a Python program to find the second-smallest number in a list.**l = [99, 50, 30, 75, 22]  
l.sort()  
m = l[1]  
print(m)

OUTPUT:

30

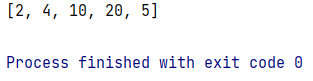
**# Set-4 Practical-10 : Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are same from a given list of strings.**l = [**'aba'**, **'lol'**, **'hello'**, **'val'**, **'hoh'**]  
l1 = []  
n = len(l)  
**for** j **in** range(n):  
 a = l[j]  
 i = len(a)  
 i -= 1  
  
 **if** a[0] == a[i]:  
 l1.append(a)  
print(l1)

OUTPUT:



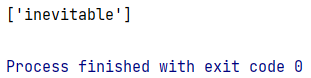
**# Set-4 Practical-11 : Write a Python program to remove duplicates from a list.****def** Remove(duplicate):  
 final\_list = []  
 **for** num **in** duplicate:  
 **if** num **not in** final\_list:  
 final\_list.append(num)  
 **return** final\_list  
duplicate = [2, 4, 10, 20, 5, 2, 20, 4]  
print(Remove(duplicate))

OUTPUT:



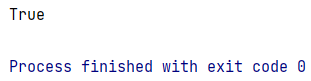
**# Set-4 Practical-12 : Write a Python program to find the list of words that are longer than n from a given string.**L1 = []  
**def** str\_find(n, str):  
 str1 = str.split(**" "**)  
 **for** i **in** str1:  
 **if** n < len(i):  
 L1.append(i)  
str = **"I am inevitable"**str\_find(2, str)  
print(L1)

OUTPUT:



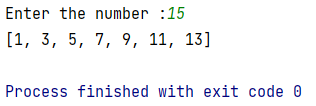
**# Set-4 Practical-13 : Write a Python function that takes two lists and returns True if they have at least one common member.****def** common\_inlist(L1, L2):  
 **for** i **in** L1:  
 **for** j **in** L2:  
 **if** i **is** j:  
 **return True  
 else**:  
 **return False**L1 = [1, 2, 3, 4, 5]  
L2 = [6, 7, 8, 9, 5]  
  
print(common\_inlist(L1, L2))

OUTPUT:



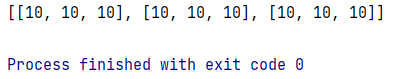
**# Set-4 Practical-14 : Write a Python program to print the numbers of a specified list after removing even numbers from it.**n = int(input(**"Enter the number :"**))  
L1 = [i **for** i **in** range(n) **if** i % 2 != 0]  
print(L1)

OUTPUT:



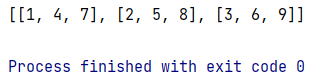
**# Set-4 Practical-15 : Write a Python program to add two matrices.**L3 = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]  
  
  
**def** add\_matrix(L1, L2):  
 **for** i **in** range(len(L1)):  
 **for** j **in** range(len(L1[i])):  
 L3[i][j] = L1[i][j] + L2[i][j]  
 **return** L3  
L1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
L2 = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]  
print(add\_matrix(L1, L2))

OUTPUT:



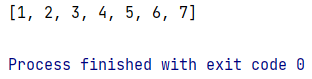
**# Set-4 Practical-16 : Write a Python program to transpose a given matrix.**L2 = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]  
  
**def** transpose\_matrix(L):  
 **for** i **in** range(len(L)):  
 **for** j **in** range(len(L[i])):  
 **if** i == j:  
 L2[i][j] = L1[i][j]  
 **else**:  
 L2[i][j] = L1[j][i]  
 **return** L2  
  
L1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
print(transpose\_matrix(L1))

OUTPUT:



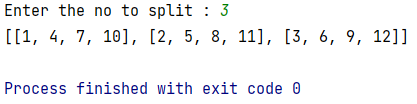
**# Set-4 Practical-17 : Flatten a nested list structure.  
# Example: if list1 = [1, [2, 3], [4, 5, [6, 7] ] ] then try to convert it in 1-dimensional  
# [1, 2, 3, 4, 5, 6, 7]**L2 = []  
  
**def** convert\_1\_dimension(L):  
 **for** i **in** L:  
 **if** type(i) == list:  
 convert\_1\_dimension(i)  
 **else**:  
 L2.append(i)  
 **return** L2  
  
L1 = [1, [2, 3], [4, 5, [6, 7]]]  
print(convert\_1\_dimension(L1))

OUTPUT:



**# Set-4 Practical-18 : Write a Python program to split a list every Nth element.****def** split\_step(L, n):  
 **return** (L[i::n] **for** i **in** range(n))  
  
L1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]  
n = int(input(**"Enter the no to split : "**))  
print(list(split\_step(L1, n)))

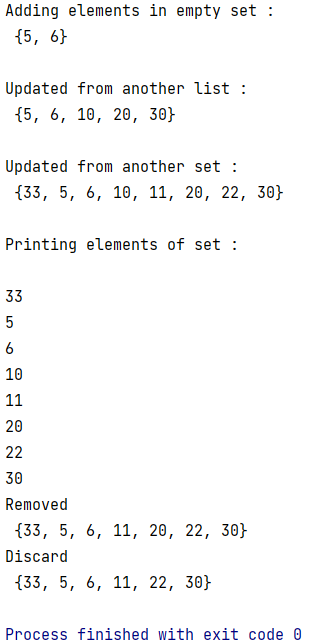
OUTPUT:



PRACTICAL SET – 05

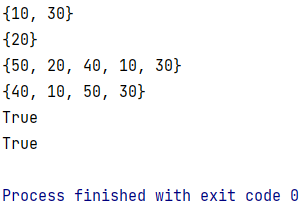
**# Set-5 Practical-1 : Create a set of integers as follows:  
# • initialize the set directly  
# • initialize empty set and then add values  
# • from a list  
# • from another set  
# • using range  
# • update an existing set using another set  
# • print the elements of set iteratively  
# • check the functionality of remove and discard**set1 = {1, 2, 3}  
set2 = set()  
  
**'''using add func'''**set2.add(5)  
set2.add(6)  
  
print(**"Adding elements in empty set :\n"**, set2)  
  
list1 = [10, 20, 30]  
  
**'''using update func to add value from list'''**set2.update(list1)  
print(**"\nUpdated from another list : \n"**, set2)  
  
**'''using update func to add value from another set'''**set3 = {11, 22, 33}  
set2.update(set3)  
print(**"\nUpdated from another set : \n"**, set2)  
  
**''' print the elements of set iteratively'''**print(**"\nPrinting elements of set :\n"**)  
**for** i **in** set2:  
 print(i)  
  
**'''using remove func'''**set2.remove(10)  
print(**"Removed\n"**, set2)  
  
**'''using discard func'''**set2.discard(20)  
print(**"Discard\n"**, set2)

OUTPUT:



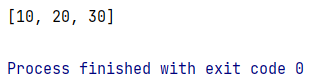
**# Set-5 Practical-2 : Create two sets of integers and find their difference, intersection, union and symmetric difference. Also find subset and superset from these two. Apply methods as well as operators for all operations.**set1 = {10, 20, 30}  
set2 = {20, 40, 50}  
  
**'''difference between two sets'''**print(set1 - set2)  
  
**'''intersection between two set'''**print(set1.intersection(set2))  
  
**''' union '''**print(set1.union(set2))  
  
**''' symmetric difference '''**print(set1.symmetric\_difference(set2))  
  
**''' subset '''**set3 = {40, 50, 10, 20, 30}  
print(set1.issubset(set3))  
  
**''' super set'''**print(set3.issuperset(set1))

OUTPUT:



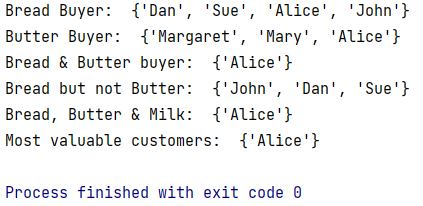
**# Set-5 Practical-3 : Write a function called find\_dups that takes a list of integers as its input argument and returns a set of those integers that occur two or more times in the list.****def** find\_dups(list1):  
 list2 = []  
 **for** i **in** list1:  
 n = list1.count(i)  
 **if** n > 1:  
 **if** list2.count(i) == 0:  
 list2.append(i)  
 **return** list2  
list1 = [10, 20, 10, 30, 20, 30, 40]  
print(find\_dups(list1))

OUTPUT:



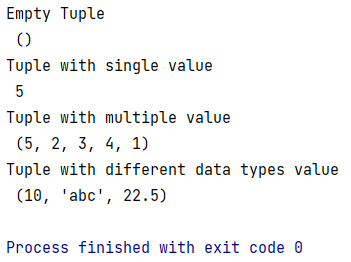
**# Set-5 Practical-4 : The following company details are given for analysis: customer acc no, customer name, purchased product no, product category, unit price. Marketing is interested in understanding customer purchase  
# patterns. Find the answers of following questions:  
# • How many customers have purchased bread?  
# • How many customers have purchased butter?  
# • How many customers have purchased bread and butter?  
# • Who has purchased bread but not butter?  
# • Which customers have purchased bread, butter and milk?  
# • Print the name of the most valuable customers who have purchased all three items.**bread\_buyer = {**'John'**, **'Alice'**, **'Dan'**, **'Sue'**}  
butter\_buyer = {**'Alice'**, **'Margaret'**, **'Mary'**}  
milk\_buyer = {**'Mary'**, **'Margaret'**, **'Dan'**, **'Alice'**}  
  
A = bread\_buyer & butter\_buyer  
B = bread\_buyer - butter\_buyer  
C = bread\_buyer & butter\_buyer & milk\_buyer  
  
print(**"Bread Buyer: "**, bread\_buyer)  
print(**"Butter Buyer: "**, butter\_buyer)  
print(**"Bread & Butter buyer: "**, A)  
print(**"Bread but not Butter: "**, B)  
print(**"Bread, Butter & Milk: "**, C)  
print(**"Most valuable customers: "**, C)

OUTPUT:



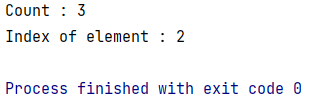
**# Set-5 Practical-5 : Write a Python program to create an empty tuple, tuple with single value, tuple with multiple values/collections and a tuple with different data types.**tuple1 = ()  
tuple2 = (5)  
tuple3 = (5, 2, 3, 4, 1)  
tuple4 = (10, **'abc'**, 22.5)  
print(**"Empty Tuple\n"**, tuple1, **"\nTuple with single value\n"**, tuple2, **"\nTuple with multiple value\n"**, tuple3,  
 **"\nTuple with different data types value\n"**, tuple4)

OUTPUT:



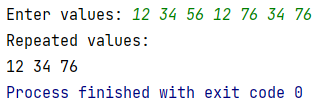
**# Set-5 Practical-6 : Check all the methods of tuple.**tuple1 = (1, 2, 5, 3, 4, 5, 7, 5)  
print(**"Count :"**, tuple1.count(5))  
print(**"Index of element :"**, tuple1.index(5))

OUTPUT:



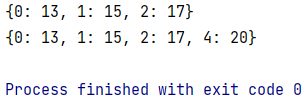
**# Set-5 Practical-7 : Write a Python program to find multiple items of a tuple.**tuple1 = [int(x) **for** x **in** input(**"Enter values: "**).split()]  
tuple1 = tuple(tuple1)  
t = tuple1  
print(**"Repeated values: "**)  
**for** i **in** range(0, len(t)):  
 **for** j **in** range(i+1, len(t)):  
 **if** t[i] == t[j]:  
 print(t[j], end=**" "**)

OUTPUT:



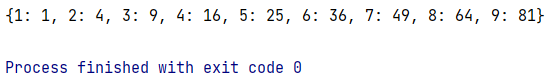
**# Set-5 Practical-8 : Write a Python program to add a key to a dictionary.**dic = {0: 13, 1: 15, 2: 17}  
print(dic)  
dic.update({4: 20})  
print(dic)

OUTPUT:



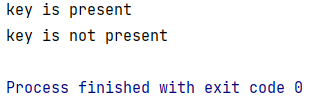
**# Set-5 Practical-9 : Write a Python program to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.**dic = dict()  
**for** x **in** range(1, 10):  
 dic[x] = x\*\*2  
print(dic)

OUTPUT:



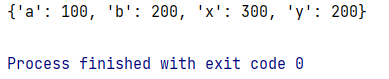
**# Set-5 Practical-10 : Write a Python program to check if a given key already exists in a dictionary.**dic = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}  
  
  
**def** key\_present(x):  
 **if** x **in** dic:  
 print(**"key is present"**)  
 **else**:  
 print(**"key is not present"**)  
key\_present(5)  
key\_present(9)

OUTPUT:



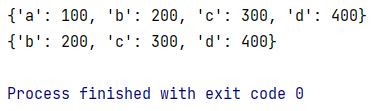
**# Set-5 Practical-11 : Write a Python program to merge two Python dictionaries.**dic1 = {**'a'**: 100, **'b'**: 200}  
dic2 = {**'x'**: 300, **'y'**: 200}  
d = dic1.copy()  
d.update(dic2)  
print(d)

OUTPUT:



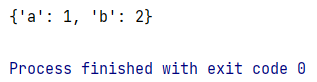
**# Set-5 Practical-12 : Write a Python program to remove a key from a dictionary.**dic = {**'a'**: 100, **'b'**: 200, **'c'**: 300, **'d'**: 400}  
print(dic)  
**if 'a' in** dic:  
 **del** dic[**'a'**]  
print(dic)

OUTPUT:



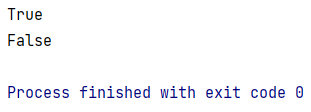
**# Set-5 Practical-13 : Write a Python program to create a dictionary from two lists.**key\_val = [**'a'**, **'b'**]  
val\_list = [1, 2]  
zip\_list = zip(key\_val, val\_list)  
dic = dict(zip\_list)  
print(dic)

OUTPUT:



**# Set-5 Practical-14 : Write a Python program to check if all dictionaries in a list are empty or not.**my\_list = [{}, {}, {}]  
my\_list1 = [{1, 2}, {}, {}]  
  
print(all(**not** d **for** d **in** my\_list))  
print(all(**not** d **for** d **in** my\_list1))

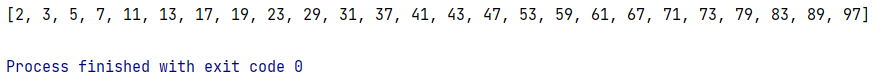
OUTPUT:



PRACTICAL SET – 06

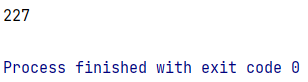
**# Set-6 Practical-1 : Write a Python program to find the prime numbers in a specific range using filter.****def** is\_prime(t):  
 **if** t <= 1:  
 **return False  
 for** i **in** range(2, t):  
 **if** t % i == 0:  
 **return False  
 return True**lst = [i **for** i **in** range(100)]  
lst = list(filter(is\_prime, lst))  
print(lst)

OUTPUT:



**# Set-6 Practical-2 : Write a Python program to make sum of particular range using reduce.****from** functools **import** reduce  
  
list1 = [2, 3, 5, 6, 7, 9, 11, 12, 15, 17, 19, 23, 55, 43]  
sum = reduce(**lambda** x, y: x + y, list1)  
print(sum)

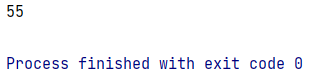
OUTPUT:



**# Set-6 Practical-3 : Write a Python program to find maximum from a list using reduce.**

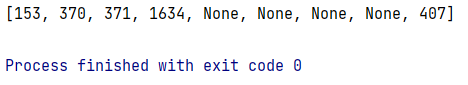
**from** functools **import** reduce  
  
list1 = [2, 3, 5, 6, 7, 9, 11, 12, 15, 17, 19, 23, 55, 43]  
print(reduce(max, list1))

OUTPUT:



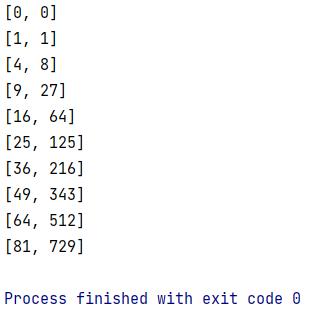
**# Set-6 Practical-4 : Write a Python program to find Armstrong number in a specific range using map.**list1 = [153, 370, 371, 1634, 125, 207, 100, 310, 407]  
  
**def** armstrong(a):  
 temp = a  
 count = 0  
 sum = 0  
 **while** a != 0:  
 count += 1  
 a = a // 10  
 a = temp  
 **while** a != 0:  
 rev = a % 10  
 sum = rev \*\* count + sum  
 a = a // 10  
 **if** sum == temp:  
 **return** temp  
  
list2 = list(map(armstrong, list1))  
print(list2)

OUTPUT:



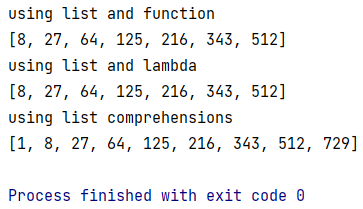
**# Set-6 Practical-5 : Write a Python program to apply two functions (square and cube) simultaneously on a specific range using map.****def** square(num):  
 **return** num \* num  
**def** cube(num):  
 **return** num \* num \* num  
func = [square, cube]  
**for** i **in** range(10):  
 ans = list(map(**lambda** x: x(i), func))  
 print(ans)

OUTPUT:



**# Set-6 Practical-6 : Write python programs using (i) map/filter and function (ii) map/filter and lambda  
# (iii) list comprehension  
# • Create a list to store the cube of all the elements in a given list.  
# • Create a list of equivalent Celsius degree from Fahrenheit.  
# • Create a list that stores only positive numbers from given list.  
# • Create a list that stores only alphabets from given list.**print(**"using list and function"**)  
**def** cu(a):  
 **return** a \* a \* a  
list1 = [2, 3, 4, 5, 6, 7, 8]  
cube = list(map(cu, list1))  
print(cube)  
print(**"using list and lambda"**)  
list1 = [2, 3, 4, 5, 6, 7, 8]  
cube = list(map(**lambda** x: x \* x \* x, list1))  
print(cube)  
print(**"using list comprehensions"**)  
list2 = [i \*\* 3 **for** i **in** range(1, 10)]  
print(list2)

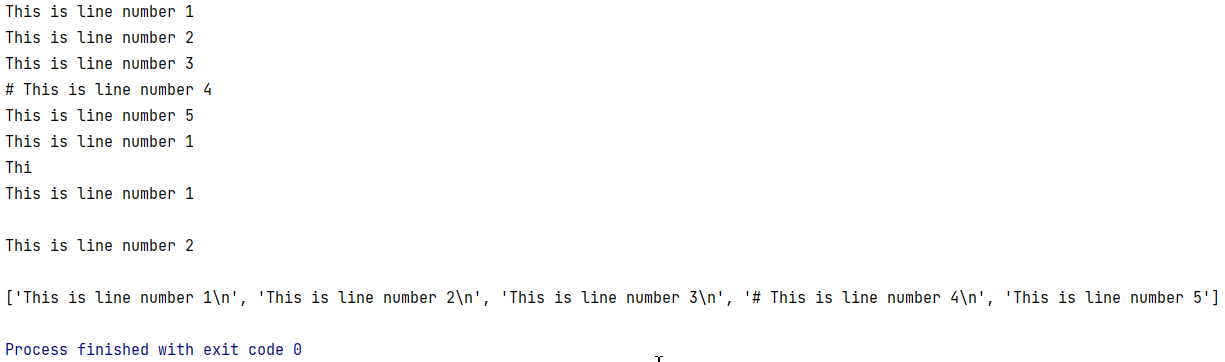
OUTPUT:



PRACTICAL SET - 07

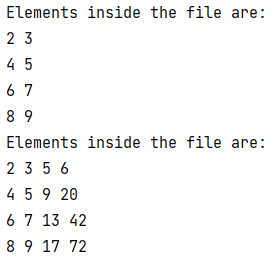
**# Set-7 Practical-1 : Write a Python program to read the text file using read (), readlines() and readline() methods.**f = open(**'try.txt'**, **'r'**)  
print(f.read())  
f.close()  
  
f = open(**'try.txt'**, **'r'**)  
print(f.read(25))  
f.close()  
  
f = open(**'try.txt'**, **'r'**)  
print(f.readline())  
print(f.readline())  
f.close()  
  
f = open(**'try.txt'**, **'r'**)  
print(f.readlines())  
f.close()

OUTPUT:



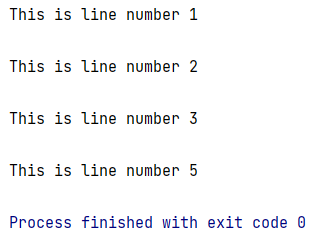
**# Set-7 Practical-2 : Write a Python program to read a file containing pairs of numbers in a file. Create a file that contains the pairs of numbers as well as addition and multiplication of the two numbers in the same line.**file1 = open(**'number.txt'**, **'r'**)  
print(**"Elements inside the file are: "**)  
print(file1.read())  
file1.close()  
  
file1 = open(**'number.txt'**, **'r'**)  
data\_list = file1.readlines()  
file1.close()  
  
file1 = open(**'number.txt'**, **'w'**)  
result\_list = []  
**for** data **in** data\_list:  
 data = data.replace(**'\n'**, **''**)  
 num1, num2 = data.split(**' '**)  
 result = **'{} {} {} {} \n'**.format(num1, num2, int(num1)+int(num2), int(num1)\*int(num2))  
 result\_list.append(result)  
file1.writelines(result\_list)  
file1.close()  
  
file1 = open(**'number.txt'**, **'r'**)  
print(**"Elements inside the file are: "**)  
print(file1.read())  
file1.close()

OUTPUT:



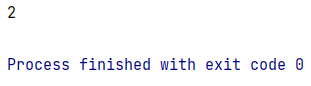
**# Set-7 Practical-3 : A text file contains a header line, few comments lines followed by actual lines of data. Write a python program to create a function skip\_header() that skips the header and all the comment lines and prints only actual lines of data.****def** skip\_header():  
 **if** line.startswith(**'#'**):  
 **pass  
 else**:  
 print(line)  
**with** open(**'try.txt'**) **as** f:  
 **while** line := f.readline():  
 skip\_header()

OUTPUT:



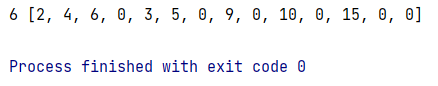
**# Set-7 Practical-4 : Write a Python program to create a function that returns smallest value from the given text file.**f = open(**'number1.txt'**, **'r'**)  
data\_list = f.readlines()  
f.close()  
result\_list = []  
  
**for** data **in** data\_list:  
 data = data.replace(**'\n'**, **''**)  
 result\_list.append(int(data))  
print(min(result\_list))

OUTPUT:



**# Set-7 Practical-5 : Write the program-4 for a text file with missing values (missing values are represented as hyphen (-)).**f = open(**'number2.txt'**, **'r'**)  
data\_list = f.readlines()  
f.close()  
result\_list = []  
count = 0  
  
**for** data **in** data\_list:  
 data = data.replace(**'\n'**, **''**)  
 **if** data == **'-'**:  
 count = count + 1  
 data = data.replace(**'-'**, **'0'**)  
 result\_list.append(int(data))  
print(count, result\_list)

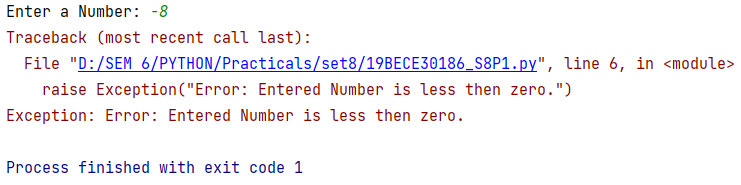
OUTPUT:



PRACTICAL SET - 08

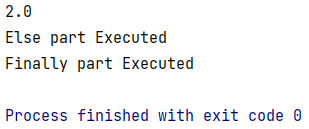
**# Set-8 Practical-1 : Write a Python program which will throw exception if the value entered by user is less than zero.**n = int(input(**"Enter a Number: "**))  
  
**if** n < 0:  
 **raise** Exception(**"Error: Entered Number is less then zero."**)

OUTPUT:



**# Set-8 Practical-2 : Write a Python program to demonstrate use of finally and else keywords.****try**:  
 print(10 / 5)  
**except**:  
 print(**"ERROR"**)  
**else**:  
 print(**"Else part Executed"**)  
**finally**:  
 print(**"Finally part Executed"**)

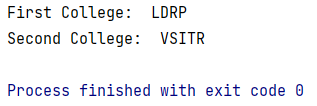
OUTPUT:



PRACTICAL SET - 09

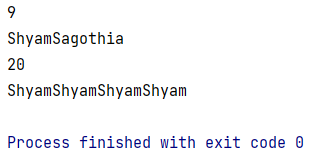
**# Set-9 Practical-1 : Write a Python program to create class KSV with attributes like class variable cnt, instance variables x and y, instance methods get\_value and print\_value.****class** KSV:  
 **def** \_\_init\_\_(self, clg1, clg2):  
 self.clg1 = clg1  
 self.clg2 = clg2  
  
 **def** get\_value(self):  
 print(**"First College: "**, self.clg1, **"\nSecond College: "**, self.clg2)  
  
  
x = KSV(**"LDRP"**, **"VSITR"**)  
x.get\_value()

OUTPUT:



**# Set-9 Practical-2 : Write a Python program to demonstrate overloading of add (+) operator.**print(4 + 5)  
print(**"Shyam"** + **"Sagothia"**)  
print(4 \* 5)  
print(**"Shyam"** \* 4)

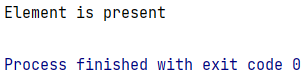
OUTPUT:



PRACTICAL SET - 10

**# Set-10 Practical-1 : Write a Python program to search a specific value from a given list of values using binary search method.****def** binary\_search(arr, low, high, x):  
 **if** high >= low:  
 mid = (high + low) // 2  
 **if** arr[mid] == x:  
 **return** mid  
 **elif** arr[mid] > x:  
 **return** binary\_search(arr, low, mid - 1, x)  
 **else**:  
 **return** binary\_search(arr, mid + 1, high, x)  
 **else**:  
 **return** -1  
  
  
arr = [2, 3, 4, 10, 60, 89]  
x = 60  
  
result = binary\_search(arr, 0, len(arr)-1, x)  
  
**if** result != -1:  
 print(**"Element is present"**)  
**else**:  
 print(**"Element is not present"**)

OUTPUT:



**# Set-10 Practical-2 : Write a Python program to sort the elements of list values using selection sort.**A = [64, 25, 12, 22, 11]  
  
**for** i **in** range(len(A)):  
 min\_idx = i  
 **for** j **in** range(i+1, len(A)):  
 **if** A[min\_idx] > A[j]:  
 min\_idx = j  
 A[i], A[min\_idx] = A[min\_idx], A[i]  
  
print(**"Sorted Array"**)  
print(A)

OUTPUT:

