**PYTHON**

1. **Implement factorial using recursion**

def recur\_factorial(n):

if n == 1:

return n

else:

return n\*recur\_factorial(n-1)

num =7

if num < 0:

print(“no”, factorial does not exist for negative numbers")

elif num == 0:

print("The factorial of 0 is 1")

else:

print("The factorial of", num, "is", recur\_factorial(num))

**2.Program using doubly linked list**

class Node:

def \_\_init\_\_(self, data):

self.item = data

self.next = None

self.prev = None

class doublyLinkedList:

def \_\_init\_\_(self):

self.start\_node = None

def InsertToEmptyList(self, data):

if self.start\_node is None

new\_node = Node(data)

self.start\_node = new\_node

else:

print("The list is empty")

def InsertToEnd(self, data):

if self.start\_node is None:

new\_node = Node(data)

self.start\_node = new\_node

return

n = self.start\_node

while n.next is not None:

n = n.next

new\_node = Node(data)

n.next = new\_node

new\_node.prev = n

def DeleteAtStart(self):

if self.start\_node is None:

print("The Linked list is empty, no element to delete")

return

if self.start\_node.next is None:

self.start\_node = None

return 0

self.start\_node = self.start\_node.next

self.start\_prev = None;

def delete\_at\_end(self):

if self.start\_node is None:

print("The Linked list is empty, no element to delete")

return

if self.start\_node.next is None:

self.start\_node = None

return

n = self.start\_node

while n.next is not None:

n = n.next

n.prev.next = None

def Display(self):

if self.start\_node is None:

print("The list is empty")

return

else:

n = self.start\_node

while n is not None:

print("Element is: ", n.item)

n = n.next

print("\n")

NewDoublyLinkedList = doublyLinkedList()

NewDoublyLinkedList.InsertToEmptyList(10)

NewDoublyLinkedList.InsertToEnd(20)

NewDoublyLinkedList.InsertToEnd(30)

NewDoublyLinkedList.InsertToEnd(40)

NewDoublyLinkedList.InsertToEnd(50)

NewDoublyLinkedList.InsertToEnd(60)

NewDoublyLinkedList.Display()

NewDoublyLinkedList.DeleteAtStart()

NewDoublyLinkedList.DeleteAtStart()

NewDoublyLinkedList.Display()

**3.Implement multiple inheritance using interface**

class Mammal:

def mammal\_info(self):

print("Mammals can give direct birth.")

class WingedAnimal:

def winged\_animal\_info(self):

print("Winged animals can flap.")

class Bat(Mammal, WingedAnimal):

pass

b1 = Bat()

b1.mammal\_info()

b1.winged\_animal\_info()

**4.Print all pronic numbers between 1 and 100.**

def isPronicNumber(num):

flag = False;

for j in range(1, num+1):

if((j\*(j+1)) == num):

flag = True;

break;

return flag;

print("Pronic numbers between 1 and 100: ");

for i in range(1, 101):

if(isPronicNumber(i)):

print(i),

print(" "),

**5.Implement method overloading & overiding in python.**

**6.Program to find duplicate values for ArrayList.**

arr = [1, 2, 3, 4, 2, 7, 8, 8, 3];

print("Duplicate elements in given array:

for i in range(0, len(arr)):

for j in range(i+1, len(arr)):

if(arr[i] == arr[j]):

print(arr[j]);

**7. Python program to print the elements of an array in reverse order.**

arr = [1, 2, 3, 4, 5];

print("Original array: ");

for i in range(0, len(arr)):

print(arr[i]),

print("Array in reverse order: ");

#Loop through the array in reverse order

for i in range(len(arr)-1, -1, -1):

print(arr[i]),

**8.Python program to determine whether the given number is a Harshad Number**

num = 156;

rem = sum = 0;

n = num;

while(num > 0):

rem = num%10;

sum = sum + rem;

num = num//10;

if(n%sum == 0):

print(str(n) + " is a harshad number");

else:

print(str(n) + " is not a harshad number");

**9.Implement a program to merge two Arrays.**

def find(array1, array2, n1, n2):

for i in array2:

array1.append(i)

array1 = list(set(sorted(array1)))

array2 = array1[len(array1) - n2:]

array1 = array1[:len(array1) - n2]

print("After")

print("Array1: ", array1, "\nArray2: ", array2)

array1 = [1, 2, 3, 5, 8, 9, 10, 13, 15, 20]

array2 = [2, 3, 8, 13]

print("Before: ")

print("Array1: ", array1)

print("Array2: ", array2)

find(array1, array2, len(array1), len(array2))

**10.Program to find duplicate values for ArrayList.**

arr = [1, 2, 3, 4, 2, 7, 8, 8, 3];

print("Duplicate elements in given array: ");

for i in range(0, len(arr)):

for j in range(i+1, len (arr)):

if(arr[i] == arr[j]):

print(arr[j]);

**11.Implement a program to sort a map by value / Key**

**12.Write a python Program for Fibonacci series**.

def Fibonacci(n):

if n < 0:

print("Incorrect input")

elif n == 0:

return 0

elif n == 1 or n == 2:

return 1

else:

return Fibonacci(n-1) + Fibonacci(n-2)

print(Fibonacci(9))

**13.Python program to print the elements of an array in reverse order.**

arr = [1, 2, 3, 4, 5];

print("Original array: ");

for i in range(0, len(arr)):

print(arr[i]),

print("Array in reverse order: ");

for i in range(len(arr)-1, -1, -1):

print(arr[i]),

**14.Write a python Program for Fibonacci series.**

def Fibonacci(n):

if n < 0:

print("Incorrect input")

elif n == 0:

return 0

elif n == 1 or n == 2:

return 1

else:

return Fibonacci(n-1) + Fibonacci(n-2)

print(Fibonacci(9))

**15.Constructor Overloading.**

The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

**16.Python program to print the elements of an array in reverse order.**

arr = [1, 2, 3, 4, 5];

print("Original array: ");

for i in range(0, len(arr)):

print(arr[i]),

print("Array in reverse order: ");

for i in range(len(arr)-1, -1, -1):

print(arr[i]),

**17.Implement Exception Handling without Catch block.**

**18.** **Python program to determine whether the given number is a Harshad Number.**

num = 156;

rem = sum = 0;

n = num;

while(num > 0):

rem = num%10;

sum = sum + rem;

num = num//10;

if(n%sum == 0):

print(str(n) + " is a harshad number");

else:

print(str(n) + " is not a harshad number");

**19.** **Compare StringBuffer with a string.**

String is an immutable class and its object can’t be modified after it is created but definitely reference other objects. They are very useful in multithreading environment because multiple threads can’t change the state of the object so immutable objects are thread safe.

String buffer is mutable classes which can be used to do operation on string object such as reverse of string, concating string and etc. We can modify string without creating new object of the string. String buffer is also thread safe.Also, string concat + operator internally uses StringBuffer or StringBuilder class.

**20.Python program to print the elements of an array in reverse order.**

arr = [1, 2, 3, 4, 5];

print("Original array: ");

for i in range(0, len(arr)):

print(arr[i]),

print("Array in reverse order: ");

for i in range(len(arr)-1, -1, -1):

print(arr[i]),

**21.remove duplicates from sorted array.**

def removeDuplicates(arr, n)

if n == 0 or n == 1:

return n

temp = list(range(n))

j = 0

for i in range(0, n-1):

if arr[i] != arr[i+1]:

temp[j] = arr[i]

j += 1

temp[j] = arr[n-1]

j += 1

for i in range(0, j):

arr[i] = temp[i]

return j

if \_\_name\_\_ == '\_\_main\_\_':

arr = [1, 2, 2, 3, 4, 4, 4, 5, 5]

n = len(arr)

n = removeDuplicates(arr, n)

for i in range(n):

print("%d" % (arr[i]), end=" ")

**22.Python program to print the elements of an array in reverse order.**

arr = [1, 2, 3, 4, 5];

print("Original array: ");

for i in range(0, len(arr)):

print(arr[i]),

print("Array in reverse order: ");

for i in range(len(arr)-1, -1, -1):

print(arr[i]),

**23.swap two numbers without using temporary variable.**

x = 10

y = 5

x = x ^ y;

y = x ^ y;

x = x ^ y;

print ("After Swapping: x = ", x, " y =", y)

**24.Python program to create a doubly linked list from a ternary tree.**

class Node:

def \_\_init\_\_(self,data):

self.data = data;

self.left = None;

self.middle = None;

self.right = None;

class TernaryTreeToDLL:

def \_\_init\_\_(self):

self.root = None;

self.head = None;

self.tail = None;

def convertTernaryToDLL(self, node):

if(node == None):

return;

left = node.left;

middle = node.middle;

right = node.right;

if(self.head == None):

self.head = self.tail = node;

node.middle = None;

self.head.left = None;

self.tail.right = None;

else:

self.tail.right = node;

node.left = self.tail;

node.middle = None;

self.tail = node;

self.tail.right = None;

self.convertTernaryToDLL(left);

self.convertTernaryToDLL(middle);

self.convertTernaryToDLL(right);

def displayDLL(self):

current = self.head;

if(self.head == None):

print("List is empty");

return;

print("Nodes of generated doubly linked list: ");

while(current != None):

print(current.data),

current = current.right;

tree = TernaryTreeToDLL();

tree.root = Node(5);

tree.root.left = Node(10);

tree.root.middle = Node(12);

tree.root.right = Node(15);

tree.root.left.left = Node(20);

tree.root.left.middle = Node(40);

tree.root.left.right = Node(50);

tree.root.middle.left = Node(24);

tree.root.middle.middle = Node(36);

tree.root.middle.right = Node(48);

tree.root.right.left = Node(30);

tree.root.right.middle = Node(45);

tree.root.right.right = Node(60);

tree.convertTernaryToDLL(tree.root);

tree.displayDLL();

**25.Find Maximum repeated charcter count in a string.**

ASCII\_SIZE = 256

def getMaxOccurringChar(str):

count = [0] \* ASCII\_SIZE

max = -1

c = ''

for i in str:

count[ord(i)] += 1

for i in str:

if max < count[ord(i)]:

max = count[ord(i)]

c = i

return c

str = "sample string"

print("Max occurring character is", getMaxOccurringChar(str))

**26.Implement quick sorting.**

def quicksort(arr):

if len(arr) <= 1:

return arr

else:

pivot = arr[0]

left = []

right = []

for i in range(1, len(arr)):

if arr[i] < pivot:

left.append(arr[i])

else:

right.append(arr[i])

return quicksort(left) + [pivot] + quicksort(right)

**27) Find duplicate elements in a string**

string = input("Enter a string: ")

char\_list = list(string)

unique\_chars = list(set(char\_list))

duplicate\_chars = []

for char in unique\_chars:

if char\_list.count(char) > 1:

duplicate\_chars.append(char)

print("Duplicate characters:", duplicate\_chars)

**28) ) Python program to determine whether the given number is a Harshad Number**

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

sum\_of\_digits = sum(int(digit) for digit in str(num))

if num % sum\_of\_digits == 0:

print(num, "is a Harshad number")

else:

print(num, "is not a Harshad number")

**29) Programs with list and tuples**

**Program for list;**

num\_list = input("Enter a list of numbers (comma-separated): ")

num\_list = list(map(int, num\_list.split(",")))

even\_sum = 0

for num in num\_list:

if num % 2 == 0:

even\_sum += num

print("Sum of even numbers:", even\_sum)

**Program for tuples;**

str\_tuple = input("Enter a tuple of strings (comma-separated): ")

str\_tuple = tuple(str\_tuple.split(","))

longest\_str = ""

for string in str\_tuple:

if len(string) > len(longest\_str):

longest\_str = string

print("Longest string:", longest\_str)

**30) Implement dictionary**

my\_dict = {"apple": 1, "banana": 2, "orange": 3}

print(my\_dict["apple"]) # Output: 1

my\_dict["banana"] = 4

print(my\_dict) # Output: {"apple": 1, "banana": 4, "orange": 3}

my\_dict["grape"] = 5

print(my\_dict)

del my\_dict["orange"]

print(my\_dict) # Output: {"apple": 1, "banana": 4, "grape": 5}

print("apple" in my\_dict) # Output: True

print(my\_dict.keys()) # Output: dict\_keys(['apple', 'banana', 'grape'])

print(my\_dict.values()) # Output: dict\_values([1, 4, 5])

**31) Python program to determine whether the given number is a Harshad Number**

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

sum\_of\_digits = sum(int(digit) for digit in str(num))

if num % sum\_of\_digits == 0:

print(num, "is a Harshad number")

else:

print(num, "is not a Harshad number")

**32) Python program to create a doubly linked list from a ternary tree**

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.left = None

self.middle = None

self.right = None

class DoublyLinkedListNode:

def \_\_init\_\_(self, data):

self.data = data

self.prev = None

self.next = None

def convert\_ternary\_tree\_to\_doubly\_linked\_list(root):

if root is None:

return None

left\_list = convert\_ternary\_tree\_to\_doubly\_linked\_list(root.left)

middle\_list = convert\_ternary\_tree\_to\_doubly\_linked\_list(root.middle)

right\_list = convert\_ternary\_tree\_to\_doubly\_linked\_list(root.right)

root\_node = DoublyLinkedListNode(root.data)

root\_node.prev = None

if left\_list:

left\_list.prev = root\_node

root\_node.next = left\_list

elif middle\_list:

middle\_list.prev = root\_node

root\_node.next = middle\_list

elif right\_list:

right\_list.prev = root\_node

root\_node.next = right\_list

else:

root\_node.next = None

if right\_list:

right\_list.next = None

return right\_list

elif middle\_list:

middle\_list.next = None

return middle\_list

elif left\_list:

left\_list.next = None

return left\_list

else:

return root\_node

root = Node(1)

root.left = Node(2)

root.middle = Node(3)

root.right = Node(4)

root.left.left = Node(5)

root.left.middle = Node(6)

root.left.right = Node(7)

root.middle.left = Node(8)

root.middle.middle = Node(9)

root.middle.right = Node(10)

root.right.left = Node(11)

root.right.middle = Node(12)

root.right.right = Node(13)

doubly\_linked\_list\_head = convert\_ternary\_tree\_to\_doubly\_linked\_list(root)

current\_node = doubly\_linked\_list\_head

while current\_node is not None:

print(current\_node.data, end=" ")

current\_node = current\_node.next

**33) compare two arrays and return the common elements**

arr1 = input("Enter the first array (comma-separated): ")

arr2 = input("Enter the second array (comma-separated): ")

arr1 = list(map(int, arr1.split(",")))

arr2 = list(map(int, arr2.split(",")))

set1 = set(arr1)

set2 = set(arr2)

common\_elements = set1.intersection(set2)

common\_elements\_list = list(common\_elements)

print("Common elements:", common\_elements\_list)

**34) Write a python Program to find whether a string or number is palindrome or not.**

def is\_palindrome(value):

value\_str = str(value)

return value\_str == value\_str[::-1]

print(is\_palindrome("racecar")) # True

print(is\_palindrome("hello")) # False

print(is\_palindrome(12321)) # True

print(is\_palindrome(12345)) # False

**35) Implement more than one interface in a single class**

from abc import ABC, abstractmethod

class Interface1(ABC):

def method1(self):

pass

class Interface2(ABC):

def method2(self):

pass

class MyClass(Interface1, Interface2):

def method1(self):

print("Implementation of method1")

def method2(self):

print("Implementation of method2")

**36) Python program to determine whether the given number is a Harshad Number**

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

sum\_of\_digits = sum(int(digit) for digit in str(num))

if num % sum\_of\_digits == 0:

print(num, "is a Harshad number")

else:

print(num, "is not a Harshad number")

**37) Implement a program for encapsulation**

class BankAccount:

def \_\_init\_\_(self):

self.\_\_balance = 0

def deposit(self, amount):

self.\_\_balance += amount

def withdraw(self, amount):

if amount <= self.\_\_balance:

self.\_\_balance -= amount

else:

print("Insufficient balance")

def get\_balance(self):

return self.\_\_balance

**38) Print all pronic numbers between 1 and 100**

for i in range(1, 101):

if i\*(i+1) <= 100:

print(i\*(i+1))

else:

break

**39) convert string to char and vice versa**

**To char;**

my\_string = "hello"

char\_list = list(my\_string)

print(char\_list)

**To string;**

char\_list = ['h', 'e', 'l', 'l', 'o']

my\_string = ''.join(char\_list)

print(my\_string)

**40) Iterate the LinkedHashMap values**

from collections import OrderedDict

my\_dict = OrderedDict()

my\_dict['key1'] = 'value1'

my\_dict['key2'] = 'value2'

my\_dict['key3'] = 'value3'

for value in my\_dict.values():

print(value)

**41) Implement a program for abstraction**

from abc import ABC, abstractmethod

class Animal(ABC):

def make\_sound(self):

pass

class Dog(Animal):

def make\_sound(self):

print("Woof!")

class Cat(Animal):

def make\_sound(self):

print("Meow!")

def animal\_sound(animal):

animal.make\_sound()

dog = Dog()

cat = Cat()

animal\_sound(dog)

animal\_sound(cat)

**42) Print all pronic numbers between 1 and 100**

for i in range(1, 101):

if i\*(i+1) <= 100:

print(i\*(i+1))

else:

break

**43) Implement a program to handle more than one exception**

try:

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

result = 10 / num

print("Result:", result)

except ZeroDivisionError:

print("Cannot divide by zero!")

except ValueError:

print("Please enter a valid integer!")

except Exception as e:

print("An error occurred:", e)

finally:

print("Program finished")

**44) Python program to create a doubly linked list from a ternary tree**

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.left = None

self.middle = None

self.right = None

class DoublyLinkedListNode:

def \_\_init\_\_(self, data):

self.data = data

self.prev = None

self.next = None

def convert\_ternary\_tree\_to\_doubly\_linked\_list(root):

if root is None:

return None

left\_list = convert\_ternary\_tree\_to\_doubly\_linked\_list(root.left)

middle\_list = convert\_ternary\_tree\_to\_doubly\_linked\_list(root.middle)

right\_list = convert\_ternary\_tree\_to\_doubly\_linked\_list(root.right)

root\_node = DoublyLinkedListNode(root.data)

root\_node.prev = None

if left\_list:

left\_list.prev = root\_node

root\_node.next = left\_list

elif middle\_list:

middle\_list.prev = root\_node

root\_node.next = middle\_list

elif right\_list:

right\_list.prev = root\_node

root\_node.next = right\_list

else:

root\_node.next = None

if right\_list:

right\_list.next = None

return right\_list

elif middle\_list:

middle\_list.next = None

return middle\_list

elif left\_list:

left\_list.next = None

return left\_list

else:

return root\_node

root = Node(1)

root.left = Node(2)

root.middle = Node(3)

root.right = Node(4)

root.left.left = Node(5)

root.left.middle = Node(6)

root.left.right = Node(7)

root.middle.left = Node(8)

root.middle.middle = Node(9)

root.middle.right = Node(10)

root.right.left = Node(11)

root.right.middle = Node(12)

root.right.right = Node(13)

doubly\_linked\_list\_head = convert\_ternary\_tree\_to\_doubly\_linked\_list(root)

current\_node = doubly\_linked\_list\_head

while current\_node is not None:

print(current\_node.data, end=" ")

current\_node = current\_node.next

**45) Convert arraylist into string**

my\_list = ['apple', 'banana', 'orange', 'pear']

delimiter = ', '

result = delimiter.join(my\_list)

print(result)

**46) Python program to determine whether the given number is a Harshad Number**

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

sum\_of\_digits = sum(int(digit) for digit in str(num))

if num % sum\_of\_digits == 0:

print(num, "is a Harshad number")

else:

print(num, "is not a Harshad number")

**47) Convert a set to stream**

my\_set = {'apple', 'banana', 'orange', 'pear'}

stream = (item for item in my\_set)

for item in stream:

print(item)

**48) Python program to create a doubly linked list from a ternary tree**

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.left = None

self.middle = None

self.right = None

class DoublyLinkedListNode:

def \_\_init\_\_(self, data):

self.data = data

self.prev = None

self.next = None

def convert\_ternary\_tree\_to\_doubly\_linked\_list(root):

if root is None:

return None

left\_list = convert\_ternary\_tree\_to\_doubly\_linked\_list(root.left)

middle\_list = convert\_ternary\_tree\_to\_doubly\_linked\_list(root.middle)

right\_list = convert\_ternary\_tree\_to\_doubly\_linked\_list(root.right)

root\_node = DoublyLinkedListNode(root.data)

root\_node.prev = None

if left\_list:

left\_list.prev = root\_node

root\_node.next = left\_list

elif middle\_list:

middle\_list.prev = root\_node

root\_node.next = middle\_list

elif right\_list:

right\_list.prev = root\_node

root\_node.next = right\_list

else:

root\_node.next = None

if right\_list:

right\_list.next = None

return right\_list

elif middle\_list:

middle\_list.next = None

return middle\_list

elif left\_list:

left\_list.next = None

return left\_list

else:

return root\_node

root = Node(1)

root.left = Node(2)

root.middle = Node(3)

root.right = Node(4)

root.left.left = Node(5)

root.left.middle = Node(6)

root.left.right = Node(7)

root.middle.left = Node(8)

root.middle.middle = Node(9)

root.middle.right = Node(10)

root.right.left = Node(11)

root.right.middle = Node(12)

root.right.right = Node(13)

doubly\_linked\_list\_head = convert\_ternary\_tree\_to\_doubly\_linked\_list(root)

current\_node = doubly\_linked\_list\_head

while current\_node is not None:

print(current\_node.data, end=" ")

current\_node = current\_node.next

**49) Write a program in python to check whether number is palindrom or not using recursive method?**

def is\_palindrome(num):

if num // 10 == 0:

return True

else:

first\_digit = num % 10

last\_digit = num // (10 \*\* (len(str(num)) - 1))

if first\_digit == last\_digit:

remaining\_num = (num - (last\_digit \* (10 \*\* (len(str(num)) - 1)))) // 10

return is\_palindrome(remaining\_num)

else:

return False

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

if is\_palindrome(num):

print(num, "is a palindrome")

else:

print(num, "is not a palindrome")

**50) Swap two numbers without using third variable**

a = int(input("Enter the first number: "))

b = int(input("Enter the second number: "))

print("Before swapping: a =", a, ", b =",b)

a = a + b

b = a - b

a = a - b

print("After swapping: a =", a, ", b =", b)

**52) Write a program to print all the prime numbers between two numbers**

**Between 1 and 10 between 20 to 30**

def is\_prime(num):

if num <= 1:

return False

for i in range(2, int(num \*\* 0.5) + 1):

if num % i == 0:

return False

return True

def print\_primes(start, end):

for num in range(start, end+1):

if is\_prime(num):

print(num)

print("Prime numbers between 1 and 10:")

print\_primes(1, 10)

print("Prime numbers between 20 and 30:")

print\_primes(20, 30)

**53) Write a program to check the string is palindrome or not**

strings = ["Madam", "wow", "cycle"]

for string in strings:

string = string.lower().replace(" ", "")

if string == string[::-1]:

print(string, "is a palindrome")

else:

print(string, "is not a palindrome")

**54)Write a program to print pattern?**

**Input= 4:**

n = 4

for i in range(n):

for j in range(n-i):

print("\*", end="")

print()

**Input=5:**

n = 5

for i in range(n):

for j in range(n-i):

print("\*", end="")

print()

**55) Write a program to check the vowels in the string.**

string = "Codoid innovations"

vowels = 'aeiouAEIOU'

for char in string:

if char in vowels:

print(char, "is a vowel")

**56) Remove the duplicate elements in the array without using builtin function  
[ 5,4,10,20,4,6,10,39,4,39]**

arr = [5, 4, 10, 20, 4, 6, 10, 39, 4, 39]

new\_arr = []

for num in arr:

if num not in new\_arr:

new\_arr.append(num)

print("Original array:", arr)

print("Array with duplicates removed:", new\_arr)

**57) Find the largest number in the array (without using pre define functions)**

n = int(input("Enter the number of elements in the array: "))

arr = []

for i in range(n):

element = int(input("Enter element {}: ".format(i+1)))

arr.append(element)

max\_num = arr[0]

for num in arr:

if num > max\_num:

max\_num = num

print("The largest number in the array is:", max\_num)

PYHON

**58) Change the vowel characters to ”S”**

string = input("Enter a string: ")

vowels = "aeiouAEIOU"

new\_string = ""

for char in string:

if char in vowels:

new\_string += "S"

else:

new\_string += char

print("New string:", new\_string)