Q6) ARMSTRONG

def isarmstrong(num):

order=len(str(num))

sum=0

temp=num

while temp>0:

digit=temp%10

sum+=digit\*\*order

temp//=10

return num==sum

for num in range(1,1001):

if isarmstrong(num):

print(num)

Q7) PRIME

def prime(num):

if num<2:

return False

else:

for i in range(2,101):

if(num%i==0):

return False

else:

return True

for num in range(1,101):

if prime(num):

print(num)

Q10) LINEAR

import random

def linear(arr,x):

for i in range(len(arr)):

if arr[i]==x:

return i

return -1

arr=random.sample(range(1,100),20)

print("array:", arr)

x=int(input("enter no.:"))

result=linear(arr,x)

if result!=-1:

print("elemeny found at:",result)

else:

print("not found")

Q11) BINARY

def binary(arr,x):

low=0

high=len(arr)-1

while low<=high:

mid=(low+high)//2

if arr[mid]==x:

return mid

elif arr[mid]<x:

low=mid+1

else:

high=mid-1

return -1

arr=sorted(["banana","cherry","date","apple"])

print("sorted array:",arr)

x=input("enter element:")

result=binary(arr,x)

if result!= -1:

print("found at:",result)

else:

print("not found")

Q12). RANDOM MODULE

\*random\* :

random

randint

randrange

sample

choice

choices

seed

uniform

shuffle

gauss

Q13) MATH MODULE

\*math\* :

sqrt

cbrt

pow

ceil

floor

isclose

sin

cos

tan

factorial

log( base 10 and natural)

Q14) TYPES PF ARGUMENT

#positional argument

def greet(name, age):

print(f"Hello, {name}! You are {age} years old.")

# Calling the function with positional arguments

greet("Alice", 25)

#default argument

def greet(name, age=18): # 'age' has a default value of 18

print(f"Hello, {name}! You are {age} years old.")

# Calling the function with and without the age argument

greet("Alice")

greet("Bob", 25)

#keyword argument

def greet(name, age):

print(f"Hello, {name}! You are {age} years old.")

# Calling the function with keyword arguments

greet(age=25, name="Alice")

#arbitrary positional argument

def add(\*args):

total = sum(args)

print(f"Sum of values: {total}")

# Calling the function with a variable number of arguments

add(1, 2, 3) # Output: Sum of values: 6

add(4, 5, 6, 7, 8) #

#arbitrary keyword argument

def display\_info(\*\*kwargs):

for key, value in kwargs.items():

print(f"{key}: {value}")

# Calling the function with multiple keyword arguments

display\_info(name="Alice", age=25, city="New York")

15) FILE MANIPULATION (TEXT AND BINARY)

# Text file operations

f = open("file1.txt", "w")

f.write("hi this is me\n")

f.close()

f = open("file1.txt", "r")

print(f.read()) # Displaying the content (optional)

f.close()

f = open("file1.txt", "a")

f.write("I'm very bored\n")

f.write("Please help me out\n")

f.close()

f = open("file1.txt", "r")

f1 = open("file2.txt", "w")

for line in f:

f1.write(line)

f.close()

f1.close()

f1 = open("file2.txt", "r")

print(f1.read())

f1.close()

# Binary file operations

binary = bytes([120, 3, 255, 0, 100, 200, 150])

f = open("bfile1.txt", "bw")

f.write(binary)

f.close()

f = open("bfile1.txt", "br")

print(f.read())

f.close()

f = open("bfile1.txt", "ba")

f.write(bytes([1, 2, 3]))

f.close()

f = open("bfile1.txt", "br")

f1 = open("bfile2.txt", "bw")

f1.write(f.read())

f.close()

f1.close()

f = open("bfile2.txt", "br")

print(f.read())

f.close()

16) USER-DEFINED EXCEPTIONS

class negativevalueerror(Exception):

pass

class valuetoohigherror(Exception):

pass

def check(value):

if value<0:

raise negativevalueerror("enter a positive value")

elif value>100:

raise valuetoohigherror("enter value less than 100")

print("value",value,"is a valid number ")

try:

check(int(input("enter a positive value less than 100:")))

except negativevalueerror as nve:

print("error",nve)

except valuetoohigherror as hve:

print("error",hve)

except ValueError as e:

print("error: invalid value type")

17) EXCEPTIONAL HANDLING

def exception\_handling():

try:

numerator = float(input("Enter the numerator: "))

denominator = float(input("Enter the denominator: "))

result = numerator / denominator

print("Result of division:",result)

except ZeroDivisionError:

print("Error: Cannot divide by zero.")

except ValueError:

print("Error: Please enter valid numbers for numerator and denominator.")

try:

lst = [1, 2, 3]

index = int(input("Enter an index to access in the list [1, 2, 3]: "))

print("Value at index",index,":",lst[index])

except IndexError:

print("Error: Index out of range.")

except ValueError:

print("Error: Please enter a valid integer index.")

try:

string = input("Enter a string: ")

integer = int(input("Enter an integer: "))

print("Result of addition:", string + integer)

except TypeError:

print("Error: Cannot add string and integer.")

except ValueError:

print("Error: Please enter a valid integer to add.")

try:

file = input("Enter a filename: ")

f=open(file,"r")

print("the content of the file is:")

print(f.read())

except FileNotFoundError:

print("Error: The file was not found.")

except Exception as e:

print("Unexpected error occurred”)

exception\_handling()

18) COMMAND-LINE ARGUMENTS BINARY SEARCH

import sys

def binary\_search(arr, x):

left, right = 0, len(arr) - 1

while left <= right:

mid = (left + right) // 2

if arr[mid] == x:

return mid

elif arr[mid] < x:

left = mid + 1

else:

right = mid - 1

return -1

if \_\_name\_\_ == "\_\_main\_\_":

if len(sys.argv) < 3:

print("Usage: python binary\_search.py <sorted list of integers> <target>")

sys.exit(1)

arr = list(map(int, sys.argv[1:-1]))

x = int(sys.argv[-1])

result = binary\_search(arr, x)

if result != -1:

print(f"Element found at index {result}")

else:

print("Element not found")

19) COMMAND LINE ARGUMENTS STRING

import sys

def string(main\_s, sub\_s):

if sub\_s in main\_s:

print("the substring" ,sub\_s, "was found in the main string " ,main\_s)

else:

print("the substring" ,sub\_s, "was NOT found in the main string " ,main\_s)

if \_\_name\_\_=="\_\_main\_\_":

if len(sys.argv)!=3:

print("usage: python find.py <main\_s> <sub\_s>")

else:

main\_s= sys.argv[1]

sub\_s= sys.argv[2]

string (main\_s,sub\_s)