1. Write a program that generates a list of 20 random numbers between 1 and

100. Print the list.

Print the average of the elements in the list.

Print the largest and smallest values in the list.

Print the second largest and second smallest entries in the list

Print how many even numbers are in the list

**Code:**

import random

number=[random.randint(1,100) for \_ in range(20)]

print(number)

avg=sum(number)/len(number)

print("the average of numbers is :", avg)

minimum=min(number)

print("minimum of numbers is :", minimum)

maximum=max(number)

print("the maximum of numbers is:", maximum)

**output:**

[66, 86, 57, 92, 10, 26, 7, 22, 64, 35, 11, 48, 18, 26, 7, 94, 87, 51, 82, 78]

the average of numbers is : 48.35

minimum of numbers is : 7

the maximum of numbers is: 94

2. Write a program that asks the user for an integer and creates a list that consists of the

factors of that integer

**code**

num=int(input("enter an integer"))

factors=[i for i in range(1,num+1) if num%i==0]

print("the factors of", num , "are: " , factors)

**OUTPUT:**

**enter an integer 8**

**the factors of 8 are: [1, 2, 4, 8]**

3. Write a program that removes any repeated items from a list so that each item appears at most

once. For instance, the list [1,1,2,3,4,3,0,0] would become [1,2,3,4,0].

**CODE**

numbers=[1,1,2,3,4,3,0,0]

unique\_numbers=[]

for numbers in numbers:

if numbers not in unique\_numbers:

unique\_numbers.append(numbers)

print("elements after removing duplicates:", unique\_numbers)

**OUTPUT:**

Elements after removing duplicates are: [1, 2, 3, 0]

4. Write a Python program to sort a list alphabetically in a dictionary.

Code:

my\_dict = {

'fruits':['banana','peer','apple'],

'vegetables':['carrot','brinjal','anarr'],

'flowers':['sunflower','amla']

}

for keys in my\_dict:

my\_dict[keys].sort()

print("the sorted dictionary is :")

for keys,values in my\_dict.items():

print(f"{keys}:{values}")

**output:**

**the sorted dictionary is :**

**fruits:['apple', 'banana', 'peer']**

**vegetables:['carrot', 'brinjal', 'anarr']**

**flowers:['sunflower', 'amla']**

**the sorted dictionary is :**

**fruits:['apple', 'banana', 'peer']**

**vegetables:['anarr', 'brinjal', 'carrot']**

**flowers:['sunflower', 'amla']**

**the sorted dictionary is :**

**fruits:['apple', 'banana', 'peer']**

**vegetables:['anarr', 'brinjal', 'carrot']**

**flowers:['amla', 'sunflower']**

5.Write a Python program to iterate over dictionaries using for loops.

**CODE:**

sample\_dict={

'name':'spoorthi',

'rollno':45,

'section':'cse1'

}

for key in sample\_dict:

print(key)

for values in sample\_dict:

print(sample\_dict[values])

for key,values in sample\_dict.items():

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

**OUTPUT:**

**name**

**rollno**

**section**

**spoorthi**

**45**

**cse1**

**name:spoorthi**

**rollno:45**

**section:cse1**

**6. Write a  Python  script to generate and print a dictionary that contains a number (between 1**

**and n) in the form (x, x\*x).**

**code:**

**n=int(input("enter a value for n"))**

**square\_dict={}**

**for x in range(1,n+1):**

**square\_dict[x]= x\*x**

**print("the dictionary is:",square\_dict)**

**OUTPUT: enter a value for n 15**

**the dictionary is: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}**

**7. Write a Python script to merge two Python dictionaries**

**CODE:**

**my\_dict1={**

**'name':'spoorthi',**

**'class':'cse',**

**'section':1,**

**}**

**my\_dict2={**

**'rollno':45,**

**'colour':'pink'**

**}**

**merged\_dict={\*\*my\_dict1,\*\*my\_dict2}**

**print(merged\_dict)**

**OUTPUT:**

**{'name': 'spoorthi', 'class': 'cse', 'section': 1, 'rollno': 45, 'colour': 'pink'}**

**8. Write a  Python program to map two lists into a dictionary.**

**CODE:**

**keys=['name','rollno','section']**

**values=['spoorthi',45,1]**

**mapped\_dict=dict(zip(keys,values))**

**print(mapped\_dict)**

**OUTPUT:**

**{'name': 'spoorthi', 'rollno': 45, 'section': 1}**

**9. Write a Python program to combine values in a list of dictionaries.**

**CODE:**

**list\_my\_dict=[**

**{'a':1,'b':2,'c':3},**

**{'a':1,'b':2,'d':5},**

**{'a':1,'b':3},**

**]**

**combined\_dict={}**

**for d in list\_my\_dict:**

**for key,value in d.items():**

**if key in combined\_dict:**

**combined\_dict[key]+=value**

**else:**

**combined\_dict[key]=value**

**print(combined\_dict)**

**OUTPUT:**

**{'a': 1, 'b': 2, 'c': 3}**

**{'a': 2, 'b': 4, 'c': 3, 'd': 5}**

**{'a': 3, 'b': 7, 'c': 3, 'd': 5}**

**10. Write a  Python program to create a dictionary of keys x, y, and z where each key has as**

**value a list from 11-20, 21-30, and 31-40 respectively. Access the fifth value of each key**

**from the dictionary.**

**CODE:**

**my\_dict={**

**'x': list(range(11,21)),**

**'y':list(range(21,31)),**

**'z':list(range(31,41))**

**}**

**for key in my\_dict:**

**fifth\_value =my\_dict[key][4]**

**print(f"the fifth value is : key {key} :{ fifth\_value} ")**

**OUTPUT:**

**the fifth value is : key x :15**

**the fifth value is : key y :25**

**the fifth value is : key z :35**

**11. Write a Python program to drop empty items from a given dictionary.**

**Code:**

**my\_dict = {**

**'a': {},**

**'b': 1,**

**'c': 2,**

**'d': 'none'**

**}**

**cleaned\_dict = {k: v for k, v in my\_dict.items() if v}**

**print(cleaned\_dict)**

**OUTPUT:**

**{'b': 1, 'c': 2, 'd': 'none'}**

**12.Write a Python program to print a dictionary in table format.**

**# Define the dictionary**

**my\_dict = {**

**'Name': ['Alice', 'Bob', 'Charlie'],**

**'Age': [24, 27, 22],**

**'City': ['New York', 'Los Angeles', 'Chicago']**

**}**

**# Extract the headers**

**headers = my\_dict.keys()**

**# Find the longest list to determine the number of rows**

**num\_rows = max(len(values) for values in my\_dict.values())**

**# Print the headers**

**print(" | ".join(headers))**

**print("-" \* 30)**

**# Print the rows**

**for i in range(num\_rows):**

**row = []**

**for key in headers:**

**row.append(str(my\_dict[key][i] if i < len(my\_dict[key]) else ""))**

**print(" | ".join(row))**

**OUTPUT:**

**Name | Age | City**

**------------------------------**

**Alice | 24 | New York**

**Bob | 27 | Los Angeles**

**Charlie | 22 | Chicago**

**LINEAR SEARCH:**

**def linearSearch(ls1, x): #creation of function**

**for i in range(len(ls1)): #checking whether i is in the list**

**if ls1[i] == x: #if the element is found then return**

**return i**

**return -1**

**ls1 = [20, 30, 40, 68, 72, 90] #declare the list**

**x = 40 #assign the target value**

**result = linearSearch(ls1, x)**

**if result == -1:**

**print("element not found")**

**else:**

**print("element found at index", result)**

**OUTPUT:**

**Element found at index 2**

**Binary Search:**

**def binarySearch(lst, low, high, x): #creation of functions**

**if high >= low: #checking whether the low element is less than the high element**

**mid = (low + high) // 2 #if low is less than high then mid is calculated**

**if lst[mid] == x: #now the mid element is compared with the target value x**

**return mid #if both mid and x are equal return mid**

**elif lst[mid] > x: #if not equal then compare the list of mid with x**

**return binarySearch(lst, low, mid - 1, x) #if x is less than mid then do mid -1**

**else:**

**return binarySearch(lst, mid + 1, high, x) # if x > mid then do mid+1**

**else:**

**return -1 #if low is greater than high then return -1**

**lst = [2, 3, 4, 10, 40] #now give the values of list**

**x = 10 #give the targeted value**

**result = binarySearch(lst, 0, len(lst) - 1, x) #if element is found write into result**

**if result != -1: #if reult is not equal to -1 then print element is found**

**print("Element is present at index", result)**

**else: #now print element is not found**

**print("Element is not found")**

**OUTPUT:**

**Elemeent is found at index 3**

**QUICK SORT:**

**def quick\_sort(arr): #creation of function**

**if len(arr) <= 1: #if lenght of array is less than or equl to 1 then**

**return arr #we need to return the array**

**else:**

**pivot = arr[len(arr) // 2] #now we need to find the pivot element**

**left = [x for x in arr if x < pivot] #**

**middle = [x for x in arr if x == pivot]**

**right = [x for x in arr if x > pivot]**

**return quick\_sort(left) + middle + quick\_sort(right)**

**# Taking input from the user**

**user\_input = input("Enter numbers separated by spaces: ") #user will enter the array**

**arr = list(map(int, user\_input.split()))**

**# Sorting the array**

**sorted\_arr = quick\_sort(arr)**

**print("Sorted array:", sorted\_arr)**

**output:**

**ENTER elements separated by spaces: 1 , 4 , 6 , 7 , 9 , 12 , 2**

**Sorted array is 1 , 2, 4,6,7,9,12**

**MERGE SORT:**

**def merge\_sort(arr):**

**if len(arr) > 1:**

**mid = len(arr) // 2**

**left\_half = arr[:mid]**

**right\_half = arr[mid:]7**

**merge\_sort(left\_half)**

**merge\_sort(right\_half)**

**i = j = k = 0**

**while i < len(left\_half) and j < len(right\_half):**

**if left\_half[i] < right\_half[j]:**

**arr[k] = left\_half[i]**

**i += 1**

**else:**

**arr[k] = right\_half[j]**

**j += 1**

**k += 1**

**while i < len(left\_half):**

**arr[k] = left\_half[i]**

**i += 1**

**k += 1**

**while j < len(right\_half):**

**arr[k] = right\_half[j]**

**j += 1**

**k += 1**

**def main():**

**user\_input = input("Enter numbers separated by spaces: ")**

**arr = list(map(int, user\_input.split()))**

**merge\_sort(arr)**

**print("Sorted array:", arr)**

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

**main()**

**OUTPUT:**

**ENTER elements separated by spaces: 1 , 4 , 6 , 7 , 9 , 12 , 2**

**Sorted array is 1 , 2, 4,6,7,9,12**

**Insertion Sort:**

**def insertion\_sort(arr):**

**# Iterate over the array starting from the second element**

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

**# Store the current element in key**

**key = arr[i]**

**# Initialize j to the index before i**

**j = i - 1**

**# Move elements of arr[0..i-1], that are greater than key, to one position ahead of their current position**

**while j >= 0 and key < arr[j]:**

**arr[j + 1] = arr[j]**

**j -= 1**

**# Place the key in its correct position**

**arr[j + 1] = key**

**# Taking input from the user**

**user\_input = input("Enter numbers separated by spaces: ")**

**# Convert the input string into a list of integers**

**arr = list(map(int, user\_input.split()))**

**# Sorting the array using insertion sort**

**insertion\_sort(arr)**

**# Print the sorted array**

**print("Sorted array:", arr)**

**OUTPUT: ENTER elements separated by spaces: 1 , 4 , 6 , 7 , 9 , 12 , 2**

**Sorted array is 1 , 2, 4,6,7,9,12**

**Bubble sort:**

**def quick\_sort(arr):**

**if len(arr) <= 1:**

**return arr**

**else:**

**pivot = arr[len(arr) // 2]**

**left = [x for x in arr if x < pivot]**

**middle = [x for x in arr if x == pivot]**

**right = [x for x in arr if x > pivot]**

**return quick\_sort(left) + middle + quick\_sort(right)**

**# Taking input from the user**

**user\_input = input("Enter numbers separated by spaces: ")**

**arr = list(map(int, user\_input.split()))**

**# Sorting the array**

**sorted\_arr = quick\_sort(arr)**

**print("Sorted array:", sorted\_arr)**

**OUTPUT:**

**ENTER elements separated by spaces: 1 , 4 , 6 , 7 , 9 , 12 , 2**

**Sorted array is 1 , 2, 4,6,7,9,12**

**Strings**

**1. Write a program that asks the user for a large integer and inserts commas into it according to**

**the standard American convention for commas in large numbers. For instance, if the user enters**

**1000000, the output should be 1,000,000.**

**# Program to insert commas in a large integer**

**num = int(input("Enter a large integer: "))**

**formatted\_num = "{:,}".format(num)**

**print("Formatted number with commas:", formatted\_num)**

**OUTPUT:**

**Enter a large integer: 1000000**

**Formatted number with commas: 1,000,000**

**2. Write a program that asks the user to enter two strings of the same length. The program should**

**then check to see if the strings are of the same length. If they are not, the program should print**

**an appropriate message and exit. If they are of the same length, the program should alternate**

**the characters of the two strings. For example, if the user enters abcde and ABCDE the program**

**should print out AaBbCcDdEe.**

**# Program to alternate characters of two strings**

**str1 = input("Enter the first string: ")**

**str2 = input("Enter the second string: ")**

**if len(str1) != len(str2):**

**print("The strings are not of the same length.")**

**else:**

**combined = ''.join([a + b for a, b in zip(str1, str2)])**

**print("Combined string:", combined)**

**OUTPUT:**

**Enter the first string: abcde**

**Enter the second string: ABCDE**

**Combined string: AaBbCcDdEe**

**3. In algebraic expressions, the symbol for multiplication is often left out, as in 3x+4y or 3(x+5).**

**Computers prefer those expressions to include the multiplication symbol, like 3\*x+4\*y or**

**3\*(x+5). Write a program that asks the user for an algebraic expression and then inserts**

**multiplication symbols where appropriate.**

**# Program to insert multiplication symbols in algebraic expressions**

**expression = input("Enter an algebraic expression: ")**

**formatted\_expression = expression.replace("x", "\*x").replace("y", "\*y").replace("(", "\*(")**

**print("Formatted expression:", formatted\_expression)**

**OUTPUT:**

**Enter an algebraic expression: 3x+4y+3(x+5)**

**Formatted expression: 3\*x+4\*y+3\*(x+5)**

**4. Write a Python program to get a string made of the first 2 and last 2 characters of a given string.**

**If the string length is less than 2, return the empty string instead.**

**# Program to get first 2 and last 2 characters of a string**

**s = input("Enter a string: ")**

**if len(s) < 2:**

**result = ""**

**else:**

**result = s[:2] + s[-2:]**

**print("Resulting string:", result)**

**Enter a string: example**

**Resulting string: exle**

**5. Write a  Python program to remove characters that have odd index values in a given string**

**# Program to remove characters with odd index values**

**s = input("Enter a string: ")**

**result = s[::2]**

**print("String after removing characters with odd index values:", result)**

**OUTPUT:**

**Enter a string: example**

**String after removing characters with odd index values: eape**

**6. Write a Python program to count the occurrences of each word in a given sentence**

**# Program to count occurrences of each word in a sentence**

**sentence = input("Enter a sentence: ")**

**words = sentence.split()**

**word\_count = {word: words.count(word) for word in set(words)}**

**print("Word count:", word\_count)**

**OUTPUT:**

**Enter a sentence: this is a test this is only a test**

**Word count: {'only': 1, 'test': 2, 'is': 2, 'this': 2, 'a': 2}**

**7. Write a Python  script that takes input from the user and displays that input back in upper and**

**lower cases.** **# Program to display input in upper and lower cases**

**text = input("Enter some text: ")**

**print("Upper case:", text.upper())**

**print("Lower case:", text.lower())**

**OUTPUT:**

**Enter some text: Hello World**

**Upper case: HELLO WORLD**

**Lower case: hello world**

**8. Write a Python function to convert a given string to all uppercase if it contains at least 2**

**uppercase characters in the first 4 characters.**

**# Program to convert string to uppercase if it has 2 uppercase characters in the first 4 characters**

**s = input("Enter a string: ")**

**if sum(1 for c in s[:4] if c.isupper()) >= 2:**

**s = s.upper()**

**print("Resulting string:", s)**

**OUTPUT:**

**Enter a string: AbCdefgHi**

**Resulting string: ABCDEFGHI**

**9. Write a Python program to check whether a string ends with specified characters**

**# Program to check if a string ends with specified characters**

**s = input("Enter a string: ")**

**ending = input("Enter the ending sequence to check for: ")**

**result = s.endswith(ending)**

**print(f"Does the string end with '{ending}'? {result}")**

**OUTPUT:**

**Enter a string: Hello World**

**Enter the ending sequence to check for: World**

**Does the string end with 'World'? True**

**10. Write a Python program to reverse a string.**

**# Program to reverse a string**

**s = input("Enter a string: ")**

**reversed\_s = s[::-1]**

**print("Reversed string:", reversed\_s)**

**OUTPUT:**

**Enter a string: Hello World**

**Reversed string: dlroW olleH**

**NUMPY FUNCTIONS:**

**import numpy as np**

**# Create an array**

**arr = np.array([1, 2, 3, 4])**

**print(arr)**

**# Create an array of ones**

**arr1 = np.ones((3, 2), dtype=float)**

**print(arr1)**

**# Create an array of zeros**

**arr2 = np.zeros((3, 2), dtype=float)**

**print(arr2)**

**# Get the number of dimensions of an array**

**p = np.ndim(arr1)**

**print(p)**

**# Get the size of an array**

**print(arr.size)**

**# Get the length (number of rows) of a 2D array**

**print(len(arr1))**

**# Get the shape of an array**

**print(arr.shape)**

**# Create an array with a range of values**

**arr4 = np.arange(6)**

**print(arr4)**

**# Create an empty array**

**arr5 = np.empty((2, 2))**

**print(arr5)**

**# Create an array with linearly spaced values**

**arr7 = np.linspace(0.5, 5, 10)**

**print(arr7)**

**# Create an identity matrix**

**arr8 = np.identity(3)**

**print(arr8)**

**# Reshape an array**

**arr9 = np.arange(6).reshape(2, 3)**

**print(arr9)**

**# Concatenate arrays (Note: arr and arr2 have incompatible shapes, so concatenation will fail)**

**# arr10 = np.concatenate((arr, arr2))**

**# Commented out because it will raise an error. Uncomment if fixing the shapes to match.**

**# Create an identity matrix with different dimensions**

**arr11 = np.eye(3, 4)**

**print(arr11)**

**# Stack arrays vertically (row-wise)**

**arr12 = np.array([1, 2])**

**arr13 = np.array([3, 4])**

**arr\_stack = np.row\_stack((arr12, arr13))**

**print(arr\_stack)**

**# Stack arrays horizontally (column-wise)**

**arr\_stack\_col = np.column\_stack((arr12, arr13))**

**print(arr\_stack\_col)**

**# Resize an array**

**arr15 = np.array([1, 2, 3, 4])**

**arr15.resize((2, 2))**

**print("Resized array:", arr15)**

**MATHEMATICAL NUMPY FUNCTIONS :**

**# -\*- coding: utf-8 -\*-**

**"""**

**Created on Sun Nov 3 14:16:19 2024**

**@author: spoor**

**"""**

**import numpy as np**

**# Basic array operations**

**a = np.array([1, 2, 3, 4])**

**b = np.array([5, 6, 7, 8])**

**print("Addition:", a + b)**

**print("Addition using np.add:", np.add(a, b))**

**print("Subtraction using np.subtract:", np.subtract(a, b))**

**print("Multiplication using np.multiply:", np.multiply(a, b))**

**print("Division using np.divide:", np.divide(a, b))**

**print("Remainder using np.remainder:", np.remainder(a, b))**

**print("Modulus using np.mod:", np.mod(a, b))**

**print("Power using np.power:", np.power(a, b))**

**print("Reciprocal of array a using np.reciprocal:", np.reciprocal(a))**

**# Matrix operations**

**arr2 = np.array([[1, 2, 3], [4, 5, 6]])**

**arr3 = np.array([[7, 8, 9], [11, 12, 13]])**

**print("Dot product using np.dot:", np.dot(arr2, arr3.T))**

**print("Vector dot product using np.vdot:", np.vdot(arr2, arr3))**

**print("Inner product using np.inner:", np.inner(arr2, arr3))**

**print("Outer product using np.outer:", np.outer(arr2, arr3))**

**print("Matrix product using np.matmul:", np.matmul(arr2, arr3.T))**

**# Linalg functions**

**arr2\_square = np.array([[1, 2], [4, 5]])**

**print("Inverse of arr2\_square using np.linalg.inv:", np.linalg.inv(arr2\_square))**

**print("Trace of arr2\_square using np.trace:", np.trace(arr2\_square))**

**print("Matrix rank of arr2\_square using np.linalg.matrix\_rank:", np.linalg.matrix\_rank(arr2\_square))**

**arr4 = np.array([[1, 2, 3], [9, 5, 4], [1, 7, 8]])**

**val, vect = np.linalg.eig(arr4)**

**print("Eigenvalues using np.linalg.eig:", val)**

**print("Eigenvectors using np.linalg.eig:\n", vect)**

**# Creating specific types of arrays**

**print("Linspace array using np.linspace:", np.linspace(0, 10, num=4))**

**print("Logspace array using np.logspace:", np.logspace(0, 2, num=4))**

**# Additional mathematical functions**

**arr5 = np.array([0, np.pi/2, np.pi])**

**print("Sine values using np.sin:", np.sin(arr5))**

**print("Cosine values using np.cos:", np.cos(arr5))**

**print("Tangent values using np.tan:", np.tan(arr5))**

**# Statistical functions**

**arr6 = np.array([[1, 2, 3], [4, 5, 6]])**

**print("Mean of arr6 using np.mean:", np.mean(arr6))**

**print("Median of arr6 using np.median:", np.median(arr6))**

**print("Standard deviation of arr6 using np.std:", np.std(arr6))**

**print("Variance of arr6 using np.var:", np.var(arr6))**

**# Stacking arrays**

**arr12 = np.array([1, 2])**

**arr13 = np.array([3, 4])**

**arr\_stack = np.row\_stack((arr12, arr13))**

**print("Row stack using np.row\_stack:\n", arr\_stack)**

**arr\_stack\_col = np.column\_stack((arr12, arr13))**

**print("Column stack using np.column\_stack:\n", arr\_stack\_col)**

**# Resize an array**

**arr15 = np.array([1, 2, 3, 4])**

**arr15.resize((2, 2))**

**print("Resized array using np.resize:\n", arr15)**

**1.Print an array of size N x M with its main diagonal elements as 1&#39;s and 0&#39;s everywhere else.**

**import numpy as np**

**N=int(input("enter no of rows"))**

**M=int(input("enter no of columns"))**

**arr=np.eye(N,M)**

**print(arr)**

**OUTPUT:**

**enter no of rows 3**

**enter no of columns 3**

**[[1. 0. 0.]**

**[0. 1. 0.]**

**[0. 0. 1.]]**

**2. Input 2-D array with dimensions N x M. Perform the min function over axis 1 and then**

**find the max of that**

**import numpy as np**

**n=int(input("enter no of rows"))**

**m=int(input("enter no of columns"))**

**arr=np.random.randint(1,10,size=(n,m))**

**print(arr)**

**min\_values=np.min(arr,axis=1)**

**print(min\_values)**

**max\_values=np.max(min\_values)**

**print(max\_values)** **enter no of rows 3**

**enter no of columns 3**

**[[8 9 1]**

**[3 2 4]**

**[8 4 3]]**

**[1 2 3]**

**3**

**OUTPUT:**

**3. Create a null vector of size 10 but the fifth value which is 1**

**import numpy as np**

**arr=np.zeros(10)**

**arr[4]=1**

**print(arr)**

**OUTPUT:**

**[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]**

**4. Create a vector with values ranging from 10 to 49**

**import numpy as np**

**arr=np.arange(10,50)**

**print(arr)**

**OUTPUT:**

**[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 38 39 40 41 42 43 44 45 46 47 48 49]**

**5. Reverse a vector (first element becomes last)**

**import numpy as np**

**arr = np.arange(10)**

**reversed\_arr = arr[::-1]**

**print(reversed\_arr)**

**OUTPUT:**

**[9 8 7 6 5 4 3 2 1 0]**

**6.  Create a 3x3 matrix with values ranging from 0 to 8**

**import numpy as np**

**arr = np.arange(9).reshape(3, 3)**

**print(arr)**

**output:**

**7. Create a 3x3 identity matrix**

**import numpy as np**

**arr = np.eye(3)**

**print(arr)**

**8. Create a 3x3x3 array with random values**

**import numpy as np**

**arr = np.random.random((3, 3, 3))**

**print(arr)**

**9. Create a 10x10 array with random values and find the minimum and maximum values**

**import numpy as np**

**arr = np.random.random((10, 10))**

**print("Array:\n", arr)**

**print("Minimum value:", np.min(arr))**

**print("Maximum value:", np.max(arr))**

**10. Create a random vector of size 30 and find the mean value**

**import numpy as np**

**arr = np.random.random(30)**

**print("Random vector:\n", arr)**

**print("Mean value:", np.mean(arr))**

**11. Write a NumPy program to add, subtract, multiply, divide arguments element-wise.**

**import numpy as np**

**a = np.array([1, 2, 3, 4])**

**b = np.array([5, 6, 7, 8])**

**print("Addition:", np.add(a, b))**

**print("Subtraction:", np.subtract(a, b))**

**print("Multiplication:", np.multiply(a, b))**

**print("Division:", np.divide(a, b))**

**12. Write a  NumPy program to get the powers of an array values element-wise**

**import numpy as np**

**a = np.array([1, 2, 3, 4])**

**print("Power:", np.power(a, 2))**

**13. Write a NumPy program to calculate the absolute value element-wise.**

**import numpy as np**

**a = np.array([-1, -2, 3, -4])**

**print("Absolute value:", np.abs(a))**

**14. Write a NumPy program to get the floor, ceiling and truncated values of the elements of a**

**numpy array.**

**Sample Output:**

**Original array:**

**[-1.6 -1.5 -0.3 0.1 1.4 1.8 2. ]**

**Floor values of the above array elements:**

**[-2. -2. -1. 0. 1. 1. 2.]**

**Ceil values of the above array elements:**

**[-1. -1. -0. 1. 2. 2. 2.]**

**Truncated values of the above array elements:**

**[-1. -1. -0. 0. 1. 1. 2.]**

**import numpy as np**

**arr = np.array([-1.6, -1.5, -0.3, 0.1, 1.4, 1.8, 2.0])**

**print("Original array:", arr)**

**print("Floor values:", np.floor(arr))**

**print("Ceil values:", np.ceil(arr))**

**print("Truncated values:", np.trunc(arr))**

**15. Write a  NumPy program to multiply a 5x3 matrix by a 3x2 matrix and create a real**

**Matrix:**

**import numpy as np**

**a = np.random.random((5, 3))**

**b = np.random.random((3, 2))**

**print("Matrix product:\n", np.matmul(a, b))**

**16. Write a NumPy program to create an inner product of two arrays.**

**Sample Output:**

**Array x:**

**[[[ 0 1 2 3]**

**[ 4 5 6 7]**

**[ 8 9 10 11]]**

**[[12 13 14 15]**

**[16 17 18 19]**

**[20 21 22 23]]]**

**Array y:**

**[0 1 2 3]**

**Inner of x and y arrays:**

**[[ 14 38 62]**

**[ 86 110 134]]**

**import numpy as np**

**N = int(input("Enter the size of the arrays: "))**

**arr1 = np.array([list(map(int, input(f"Enter elements for row {i + 1} of the first array: ").split())) for i in range(N)])**

**arr2 = np.array(list(map(int, input("Enter the elements for the second array: ").split())))**

**print("Inner product:\n", np.inner(arr1, arr2))**

**17. Write a NumPy program to generate a matrix product of two arrays.**

**import numpy as np**

**N = int(input("Enter the size of the arrays: "))**

**arr1 = np.array([list(map(int, input(f"Enter elements for row {i + 1} of the first array: ").split())) for i in range(N)])**

**arr2 = np.array(list(map(int, input("Enter the elements for the second array: ").split())))**

**print("Inner product:\n", np.inner(arr1, arr2))**

**18. Write a  NumPy program to compute the trigonometric sine, cosine and tangent array of**

**angles given in degrees.**

**import numpy as np**

**angles = np.array(list(map(float, input("Enter the angles in degrees separated by space: ").split())))**

**radians = np.radians(angles)**

**print("Sine:", np.sin(radians))**

**print("Cosine:", np.cos(radians))**

**print("Tangent:", np.tan(radians))**

**19.Write a NumPy program to illustrate insert function(try it on both single and multi**

**dimensional arrays)**

**import numpy as np**

**arr = np.array(list(map(int, input("Enter the elements of the array separated by space: ").split())))**

**index = int(input("Enter the index at which to insert: "))**

**value = int(input("Enter the value to insert: "))**

**new\_arr = np.insert(arr, index, value)**

**print("Array after insertion:", new\_arr)**

**rows, cols = map(int, input("Enter the dimensions of the 2D array (rows columns): ").split())**

**arr2d = np.array([list(map(int, input(f"Enter elements for row {i + 1}: ").split())) for i in range(rows)])**

**axis = int(input("Enter the axis along which to insert (0 for rows, 1 for columns): "))**

**new\_arr2d = np.insert(arr2d, index, value, axis=axis)**

**print("2D Array after insertion:\n", new\_arr2d)**

**20.Write a python program to flatten two arrays**

**import numpy as np**

**rows1, cols1 = map(int, input("Enter the dimensions of the first 2D array (rows columns): ").split())**

**arr1 = np.array([list(map(int, input(f"Enter elements for row {i + 1} of the first array: ").split())) for i in range(rows1)])**

**rows2, cols2 = map(int, input("Enter the dimensions of the second 2D array**

**21.Write a python program to mereg 2 numpy arrays usning append() function**

**import numpy as np**

**arr1 = np.array([1, 2, 3])**

**arr2 = np.array([4, 5, 6])**

**merged\_arr = np.append(arr1, arr2)**

**print("Merged array:", merged\_arr)**

**22.Write a python program to remove a sub array from a numpy array**

**import numpy as np**

**arr = np.array([1, 2, 3, 4, 5, 6])**

**sub\_array = arr[1:4] # Removing elements from index 1 to 3**

**result = np.delete(arr, slice(1, 4))**

**print("Array after removing sub array:", result)**

**23.Write a python program to create a numpy array of 12 elements, reshape it to 3X4 matrix,**

**remove the second row and print the final matrix.**

**import numpy as np**

**arr = np.arange(12).reshape(3, 4)**

**print("Original matrix:\n", arr)**

**final\_matrix = np.delete(arr, 1, axis=0)**

**print("Final matrix after removing second row:\n", final\_matrix)**

**24.Write a python program to delete multiple rows and columns from a numpy array(Use any**

**method)**

**import numpy as np**

**arr = np.arange(20).reshape(4, 5)**

**print("Original matrix:\n", arr)**

**# Remove rows at indices 1 and 2, and columns at indices 0 and 3**

**result = np.delete(arr, [1, 2], axis=0)**

**result = np.delete(result, [0, 3], axis=1)**

**print("Matrix after deleting rows and columns:\n", result)**

**25.Write a Numpy Program to illustrate horizontal and vertical stacking**

**import numpy as np**

**arr1 = np.array([1, 2])**

**arr2 = np.array([3, 4])**

**h\_stack = np.hstack((arr1, arr2))**

**v\_stack = np.vstack((arr1, arr2))**

**print("Horizontal stack:\n", h\_stack)**

**print("Vertical stack:\n", v\_stack)**

**26.Write a Numpy program to sort the elements of an array.**

**import numpy as np**

**arr = np.array([5, 2, 9, 1, 5, 6])**

**sorted\_arr = np.sort(arr)**

**print("Sorted array:", sorted\_arr)**