Assignment 5



Name – Adarsh Mishra

N1

Electrical engineering

Q1- a numpy array from a list and display its shape, size Write a program to create and data type.

solution1

-# Create a list

import numpy as np

my\_list = [1, 2, 3, 4, 5]

my\_array = np.array(my\_list)

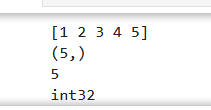
# the shape, size, and data type of the array

print(my\_array)

print(my\_array.shape)

print(my\_array.size)

print( my\_array.dtype)



2. Create a Numpy array of numbers from 1 to 20. Extract and print the first 5 elements, last 5 elements, and every second element

Soution2

import numpy as np

# Create the NumPy array

arr = np.arange(1, 21)

# Extract the first 5 elements

first\_5 = arr[:5]

# Extract the last 5 elements

last\_5 = arr[-5:]

# Extract every second element

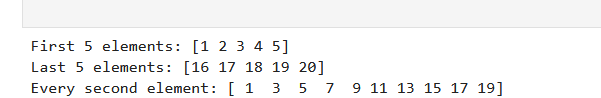
every\_second = arr[::2]

# Print the results

print("First 5 elements:", first\_5)

print("Last 5 elements:", last\_5)

print("Every second element:", every\_second)



Ques 3 Write a program to create a 1D Numpy array of 12 elements and reshape it into a 3×4

Matrix

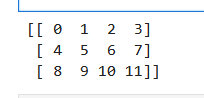
Soln3

import numpy as np

# Create a 1D Numpy array of 12 elements

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

print(arr)



Ques4

. Write a program to find sum, mean, maximum and minimum in a numpy array.

Soln4

import numpy as np

# Create a sample Numpy array

arr = np.array=([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# Calculate the sum

sum\_arr = np.sum(arr)

# Calculate the mean

mean\_arr = np.mean(arr)

# Calculate the maximum

max\_arr = np.max(arr)

# Calculate the minimum

min\_arr = np.min(arr)

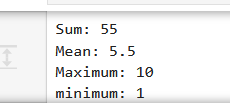
# Print the results

print("Sum:", sum\_arr)

print("Mean:", mean\_arr)

print("Maximum:", max\_arr)

print("minimum:",min\_arr)



ques05. Create two Numpy arrays and perform addition, subtraction, multiplication, and division operations

soln5

import numpy as np

# Create two sample Numpy arrays

arr1 = np.array=([10, 20, 30, 40])

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

# Perform addition

addition = np.add(arr1, arr2)

# Perform subtraction

subtraction = np.subtract(arr1, arr2)

# Perform multiplication

multiplication = np.multiply(arr1, arr2)

# Perform division

division = np.divide(arr1, arr2)

# Print the results

print("Array 1:", arr1)

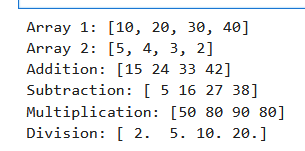
print("Array 2:", arr2)

print("Addition:", addition)

print("Subtraction:", subtraction)

print("Multiplication:", multiplication)

print("Division:", division)



Ques6

import numpy as np

# Create a NumPy array

arr = np.array([3, 1, 4, 1, 5, 9])

# Append

new\_arr = np.append(arr, 6)

print("Appended:", new\_arr)

# Insert

new\_arr = np.insert(arr, 2, 8)

print("Inserted:", new\_arr)

# Find index

index = np.where(arr == 4)[0]

if len(index) > 0:

    print("Index of 4:", index[0])

# Sort

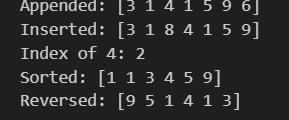
sorted\_arr = np.sort(arr)

print("Sorted:", sorted\_arr)

# Reverse

reversed\_arr = arr[::-1]

print("Reversed:", reversed\_arr)



Ques 7

.Write a program to copy array with assignment operator, shallow copy method and deep copy method

import numpy as np

import copy

# Original array

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

# Assignment

assigned = original

assigned[0, 0] = 10

print("Original (assigned change):", original)

# Shallow copy

shallow = original.view()

shallow[0, 1] = 20

print("Original (shallow change):", original)

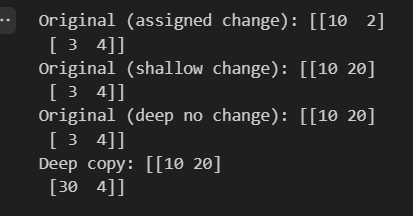
# Deep copy

deep = copy.deepcopy(original)

deep[1, 0] = 30

print("Original (deep no change):", original)

print("Deep copy:", deep)



09. Write a program to calculate and display square root, factorial, log (base10), and power of a number using math module.

import math

import cmath

x = 2.0

y = 3 + 4j

print("Math:")

print("acos:", math.acos(0.5))

print("ceil:", math.ceil(2.3))

print("exp:", math.exp(x))

print("gcd:", math.gcd(12, 18))

print("log10:", math.log10(100))

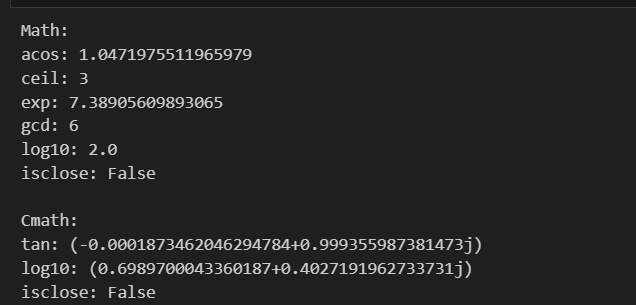
print("isclose:", math.isclose(1.0, 1.0000001))

print("\nCmath:")

print("tan:", cmath.tan(y))

print("log10:", cmath.log10(y))

print("isclose:", cmath.isclose(y, 3 + 4.0000001j))



Import maths

# Input a number

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

# 1. Calculate square root

if num >= 0:

square\_root = math.sqrt(num)

print("Square root of", num, "is", square\_root)

else:

print("Square root is not defined for negative numbers.")

# 2. Calculate factorial ( non-negative integers)

if num.is\_integer() and num >= 0:

factorial = math.factorial(int(num))

print("Factorial of", int(num), "is", factorial)

else:

print("Factorial is only defined for non-negative integers.")

# 3. Calculate log (base 10)

if num > 0:

log\_value = math.log10(num)

print("Log (base 10) of", num, "is", log\_value)

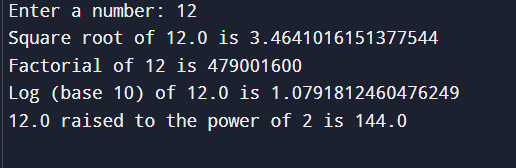
else:

print("Logarithm is not defined for zero or negative numbers.")

# 4. Calculate power (e.g., num raised to the power of 2)

power = math.pow(num, 2)

print(num, "raised to the power of 2 is", power)



ques10. Write a program to solve a quadratic equation using quadratic formula, use math.sqrt() for square root calculation.

Soln10

import math

# Input coefficients a, b, and c

a = float(input("Enter coefficient a: "))

b = float(input("Enter coefficient b: "))

c = float(input("Enter coefficient c: "))

# Calculate the discriminant (D)

discriminant = b\*\*2 - 4\*a\*c

# Check if the discriminant is positive, zero, or negative

if discriminant > 0:

# Two real and distinct roots

root1 = (-b + math.sqrt(discriminant)) / (2\*a)

root2 = (-b - math.sqrt(discriminant)) / (2\*a)

print(f"The roots are real and distinct: root1 = {root1}, root2 = {root2}")

elif discriminant == 0:

# One real root (both roots are the same)

root = -b / (2\*a)

print(f"There is one real root: root = {root}")

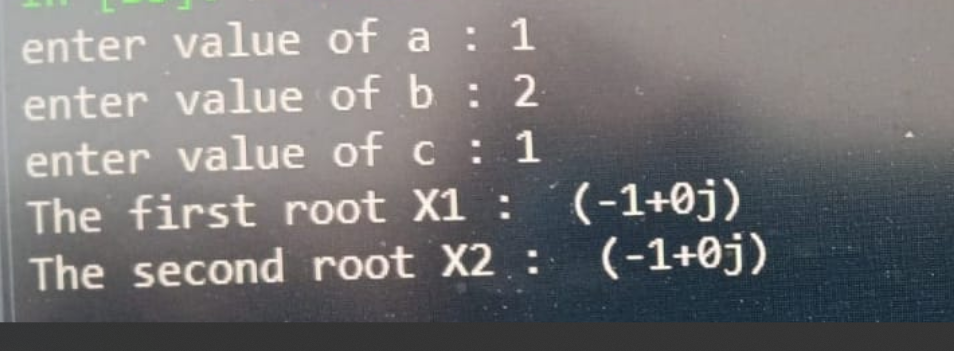
else:

# Two complex roots

real\_part = -b / (2\*a)

imaginary\_part = math.sqrt(-discriminant) / (2\*a)

print(f"The roots are complex: root1 = {real\_part} + {imaginary\_part}i, root2 = {real\_part} - {imaginary\_part}i")



Q11 mid sem

Q2(a) discuss the different scope of various variable in python including local , non local nd global variables. Provide python functions demonstrating each type

Ans

A **local variable** is defined inside a function and can only be accessed within that function. A **global variable** is defined outside any function and is accessible throughout the entire program. A **nonlocal variable** is used in nested functions and refers to a variable in the nearest enclosing function's scope, but not global. The nonlocal keyword allows modification of this variable.

Python follows the **LEGB** rule (Local, Enclosing, Global, Built-in) to search for variables in this order. Understanding these scopes helps in managing data access and avoiding unintended modifications across different parts of a program.

Q2(b) what is function explain the type of function arguments with python example

Ans

A **function** in Python is a block of reusable code designed to perform a specific task. Functions can take different types of arguments: **positional arguments**, which are passed in the order they are defined; **keyword arguments**, where values are passed by specifying the parameter names; **default arguments**, which take default values if no value is provided; and **variable-length arguments**, using \*args or \*\*kwargs for handling an arbitrary number of arguments.

def greet(name, age=18):

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

greet("Alice")

greet("Bob", 25)

Q3(a)explain the fundamental data type in python with their characterstics. Provide example demonstrating how and where they can be used in python programming

Ans

**fundamental data types** are the building blocks that define the type of data a variable can store. Python has several fundamental data types, each with its own characteristics and use cases.

**1. Integers**

Integers are whole numbers, both positive and negative, without any decimal points.They can be of arbitrary precision (i.e., their size is limited by the available memory rather than a fixed size).

**Example:**

**X=7**

**Y=9**

**Print(X+Y) = 12**

**2. Floating number**

* **Floats represent real numbers (numbers with decimal points).**
* **They are used for more precise calculations.**
* **Floating point numbers have limited precision and are subject to rounding errors in some cases.**

**M=2.2**

**N=1.1**

**Print(M+N)**

**3.3**

**Strings (str)**

* **Strings are sequences of characters enclosed in single (') or double (") .Strings are immutable (i.e., once created, they cannot be modified directly),Supports a wide range of methods for text manipulation.**
* **Example: name = “Adarsh”**

**Print(name)**

**Adarsh**

Q3(b)

def second\_largest(numbers):

    unique\_numbers = list(set(numbers))

    for i in range(len(unique\_numbers)):

        for j in range(i + 1, len(unique\_numbers)):

            if unique\_numbers[i] < unique\_numbers[j]:

                unique\_numbers[i], unique\_numbers[j] = unique\_numbers[j], unique\_numbers[i]

    return unique\_numbers[1] if len(unique\_numbers) > 1 else None

numbers = [12,45,89,76,65,60,43,9]

print(second\_largest(numbers)

Ans

76

Q4(a)

    rows, cols = len(matrix), len(matrix[0])

    sums = [0] \* cols

    for i in range(cols):

        for j in range(rows):

            sumdef column\_sums(matrix):

s[i] += matrix[j][i]

    return sums

matrix = [[1,2,3],[4,5,6],[7,8,9]]

print(column\_sums(matrix))

Ans

[12,15,18]

Q4(b)

tuples\_list = [(2,3), (4,5), (), (3,3), (), (1,)]

def remove\_empty\_tuples(lst):

    result = []

    for t in lst:

        if t:

            result.append(t)

    return result

print(remove\_empty\_tuples(tuples\_list))

Ans

(3,2),(4,5),(3,3),(1,)

Q5

def clean\_text(text):

    punctuation\_marks = """!()-[]{};:'"\,<>./?@#$%^&\*\_~"""

    stop\_words = ["the", "is", "in", "and", "to", "of", "it", "you", "that", "he", "she", "for", "on", "with", "as", "was", "at", "by", "an", "be", "this", "have"]

    text = ''.join(c for c in text if c not in punctuation\_marks)

    words = text.split()

    result = [word.lower() for word in words if word.lower() not in stop\_words]

    return result

text = "The quick brown fox jumps over the lazy dog!"

print(clean\_text(text))

Ans

['quick', 'brown', 'fox', 'jumps', 'over', 'lazy', 'dog']

<>:2: SyntaxWarning: invalid escape sequence '\,'

<>:2: SyntaxWarning: invalid escape sequence '\,'

[C:\Users\ASUS\AppData\Local\Temp\ipykernel\_22904\3860531171.py:2](file:///C:\Users\ASUS\AppData\Local\Temp\ipykernel_22904\3860531171.py:2): SyntaxWarning: invalid escape sequence '\,'

punctuation\_marks = """!()-[]{};:'"\,<>./?@#$%^&\*\_~"""