|  |  |  |
| --- | --- | --- |
| **Sr No.** | **INDEX** | **Remark** |
| **1** | Practical sheet 1 |  |
| **2** | Practical sheet 2 |  |
| **3** | Practical sheet 3 |  |
| **4** | Practical sheet 4 |  |
| **5** | Practical sheet 5 |  |
| **6** | Practical sheet 6 |  |
| **7** | Practical sheet 7 |  |
| **8** | Practical sheet 8 |  |
| **9** | Practical sheet 9 |  |
| **10** | Practical sheet 10 |  |
| **11** | Practical sheet 11 |  |
| **12** | Practical sheet 12 |  |
| **13** | Practical sheet 13 |  |
| **14** | Practical sheet 14 |  |
| **15** | Practical sheet 15 |  |
| **16** | Practical sheet 16 |  |
| **17** | Practical sheet 17 |  |
| **18** | Practical sheet 18 |  |
| **19** | Practical sheet 19 |  |
| **20** | Practical sheet 20 |  |
| **21** | Practical sheet 21 |  |
| **22** | Practical sheet 22 |  |
| **23** | Practical sheet 23 |  |
| **24** | Practical sheet 24 |  |
| **25** | Practical sheet 25 |  |

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Shinde Tejal Gorakshanath **Batch No. :-** A **Roll No. :-** 9 **Date :-** …./…/2023

**Title :-** Python Practical **Expt.No. :-** **1 Class :-** SYBCS

**Q.1. Attempt any two of the following. [10]**

**1. Using python code sort the tuple in ascending and descending order 5, -3, 0, 1, 6,-6, 2.**

**ANS :**  original\_tuple = (5, -3, 0, 1, 6, -6, 2)

ascending\_sorted\_tuple = tuple(sorted(original\_tuple))

descending\_sorted\_tuple = tuple(sorted(original\_tuple, reverse=True))

print("Ascending Sorted Tuple:", ascending\_sorted\_tuple)

print("Descending Sorted Tuple:", descending\_sorted\_tuple)

*output :* Ascending Sorted Tuple: (-6, -3, 0, 1, 2, 5, 6)

Descending Sorted Tuple: (6, 5, 2, 1, 0, -3, -6)

**2. Write python program which deals with concatenation and repetition of lists.**

**List1 = [15, 20, 25, 30, 35, 40]**

**List2 = [7, 14, 21, 28, 35, 42]**

**(a) Find List1 + List2**

**ANS:** List1 = [15, 20, 25, 30, 35, 40]

List2 = [7, 14, 21, 28, 35, 42]

concatenated\_list = List1 + List2

print("Concatenated list:", concatenated\_list)

output : Concatenated list: [15, 20, 25, 30, 35, 40, 7, 14, 21, 28, 35, 42]

**(b) Find 9\*List1**

**ANS:** List1 = [15, 20, 25, 30, 35, 40]

List2 = [7, 14, 21, 28, 35, 42]

nine\_times\_list1 = [9 \* element for element in List1]

print("9 times List1:", nine\_times\_list1)

output: 9 times List1: [135, 180, 225, 270, 315, 360]

**(c) Find 7\*List2**

**ANS :** List1 = [15, 20, 25, 30, 35, 40]

List2 = [7, 14, 21, 28, 35, 42]

seven\_times\_list2 = [7 \* element for element in List2]

print("7 times List2:", seven\_times\_list2)

output : 7 times List2: [49, 98, 147, 196, 245, 294]

**3. Write Python code to find the square of odd numbers from 1 to 20 using while loop.**

**ANS :** number = 1

while number <= 20:

if number % 2 != 0:

square = number \*\* 2

print("Square of", number, "is", square)

number += 1

*output* :Square of 1 is 1

Square of 3 is 9

Square of 5 is 25

Square of 7 is 49

Square of 9 is 81

Square of 11 is 121

Square of 13 is 169

Square of 15 is 225

Square of 17 is 289

Square of 19 is 361

​

**Q.2. Attempt any two of the following. [10]**

**1. Using Python construct the following matrices.**

**1. An identity matrix of order 10 × 10.**

**ANS:** from sympy import \*

eye(10)

*output* : Matrix([

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 1, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 1, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 1, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]])

**2. Zero matrix of order 7 × 3.**

**ANS :**from sympy import\*

zeros(7,3)

*output :*Matrix([

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0]])

**3. Ones matrix of order 5 × 4.**

**ANS :**from sympy import\*

ones(5,4)

*output* :Matrix([

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1]])

**2. Find the data type of the following data by using Python code.**

**a. number**

**ANS :** A= "number"

print(type(A))

output :<class 'str'>

**b. 31.25**

ANS : B= 31.25

print(type(B))

output : <class 'float'>

**c. 8 + 4j**

**ANS:** C= 8 + 4j

print(type(C))

output : <class 'complex'>

**d. Mathematics**

**ANS:** D= "Mathematics"

print(type(D))

output : <class 'str'>

**e. 49**

**ANS:** E=49

print(type(E))

output : <class 'int'>

**3. Write Python program to find the determinant of matrices**

A=[1 0 5]

[2 1 6]

[3 4 0]

B=[2 5]

[-1 4]

**ANS:** import numpy as np

A = np.array([[1, 0, 5], [2, 1, 6], [3, 4, 0]])

B = np.array([[2, 5], [-1, 4]])

det\_A = np.linalg.det(A)

det\_B = np.linalg.det(B)

print("Determinant of Matrix A:", det\_A)

print("Determinant of Matrix B:", det\_B)

output : Determinant of Matrix A: 0.9999999999999989

Determinant of Matrix B: 13.0

**Q.3. a. Attempt any one of the following.**

**1. Write Python program to estimate the value of the integral R *π***

**0 *xsin*(*x*)*dx* using**

**Simpson’s (1/3 )*rd* rule (n=6).**

**ANS :**import math

def f(x):

return x \* math.sin(x)

a = 0

b = math.pi

n = 6

h = (b - a) / n

integral\_value = f(a) + f(b)

for i in range(1, n, 2):

integral\_value += 4 \* f(a + i \* h)

for i in range(2, n-1, 2):

integral\_value += 2 \* f(a + i \* h)

integral\_value \*= h / 3

print(f"The estimated value of the integral is: {integral\_value}")

output : The estimated value of the integral is: 3.1429485487583113

**2. Write Python program to estimate a root of an equation *f*(*x*) = 3*x* − *cos*(*x*) − 1**

**using Newton–Raphson method correct up to four decimal places.**

**ANS:** import math

def f(x):

return 3 \* x - math.cos(x) - 1

def f\_prime(x):

return 3 + math.sin(x)

x0 = 0.0

tolerance = 1e-4

max\_iterations = 100

iteration = 0

while iteration < max\_iterations:

x1 = x0 - f(x0) / f\_prime(x0)

if abs(x1 - x0) < tolerance:

break

x0 = x1

iteration += 1

if iteration < max\_iterations:

print(f"Estimated root: {x1:.4f}")

else:

print("Maximum iterations reached. The root may not have converged.")

*output :* Estimated root: 0.6071

**b. Attempt any one of the following.**

**1. Write Python program to find all positive prime numbers less then given number n.**

**ANS :** def sieve\_of\_eratosthenes(n):

is\_prime = [True] \* (n + 1)

is\_prime[0] = is\_prime[1] = False

for p in range(2, int(n\*\*0.5) + 1):

if is\_prime[p]:

for i in range(p \* p, n + 1, p):

is\_prime[i] = False

primes = [i for i in range(2, n + 1) if is\_prime[i]]

return primes

n = int(input("Enter a positive integer n: "))

prime\_numbers = sieve\_of\_eratosthenes(n)

print("Prime numbers less than", n, "are:", prime\_numbers)

*output :*Enter a positive integer n:

44

Prime numbers less than 44 are: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43]

**2. Write Python program to evaluate f(2.5) by forward difference formula of the given**

**data.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 0 | 1 | 2 | 3 |
| **Y=f(x)** | 2 | 1 | 2 | 10 |

**ANS:** x\_values = [0, 1, 2, 3]

y\_values = [2, 1, 2, 10]

x\_target = 2.5

h = x\_values[1] - x\_values[0]

index = (x\_target - x\_values[0]) / h

i = int(index)

f\_x\_target = y\_values[i] + (x\_target - x\_values[i]) \* (y\_values[i + 1] - y\_values[i]) / h

print("Estimated value of f(2.5) is:", f\_x\_target)

output : Estimated value of f(2.5) is: 6.0

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Shinde Tejal Gorakshanath **Batch No. :-** A **Roll No. :-** **Date :-** …./…/2023

**Title :-** Python Practical **Expt.No. :-** **2 Class :-** SYBCS

**Q.1. Attempt any two of the following. [10]**

**1. Using python code sort the tuple in ascending and descending order 5, -3, 0, 1, 6,-6, 2.**

**ANS :**  original\_tuple = (5, -3, 0, 1, 6, -6, 2)

ascending\_sorted\_tuple = tuple(sorted(original\_tuple))

descending\_sorted\_tuple = tuple(sorted(original\_tuple, reverse=True))

print("Ascending Sorted Tuple:", ascending\_sorted\_tuple)

print("Descending Sorted Tuple:", descending\_sorted\_tuple)

*output :* Ascending Sorted Tuple: (-6, -3, 0, 1, 2, 5, 6)

Descending Sorted Tuple: (6, 5, 2, 1, 0, -3, -6)

**2. Write python program which deals with concatenation and repetition of lists.**

**List1 = [15, 20, 25, 30, 35, 40]**

**List2 = [7, 14, 21, 28, 35, 42]**

**(a) Find List1 + List2**

**ANS:** List1 = [15, 20, 25, 30, 35, 40]

List2 = [7, 14, 21, 28, 35, 42]

concatenated\_list = List1 + List2

print("Concatenated list:", concatenated\_list)

output : Concatenated list: [15, 20, 25, 30, 35, 40, 7, 14, 21, 28, 35, 42]

**(b) Find 9\*List1**

**ANS:** List1 = [15, 20, 25, 30, 35, 40]

List2 = [7, 14, 21, 28, 35, 42]

nine\_times\_list1 = [9 \* element for element in List1]

print("9 times List1:", nine\_times\_list1)

output: 9 times List1: [135, 180, 225, 270, 315, 360]

**(c) Find 7\*List2**

**ANS :** List1 = [15, 20, 25, 30, 35, 40]

List2 = [7, 14, 21, 28, 35, 42]

seven\_times\_list2 = [7 \* element for element in List2]

print("7 times List2:", seven\_times\_list2)

output : 7 times List2: [49, 98, 147, 196, 245, 294]

**3. Write Python code to find the square of odd numbers from 1 to 20 using while loop.**

**ANS :** number = 1

while number <= 20:

if number % 2 != 0:

square = number \*\* 2

print("Square of", number, "is", square)

number += 1

*output* :Square of 1 is 1

Square of 3 is 9

Square of 5 is 25

Square of 7 is 49

Square of 9 is 81

Square of 11 is 121

Square of 13 is 169

Square of 15 is 225

Square of 17 is 289

Square of 19 is 361

​

**Q.2. Attempt any two of the following. [10]**

**1. Using Python construct the following matrices.**

**1. An identity matrix of order 10 × 10.**

**ANS:** from sympy import \*

eye(10)

*output* : Matrix([

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 1, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 1, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 1, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]])

**2. Zero matrix of order 7 × 3.**

**ANS :**from sympy import\*

zeros(7,3)

*output :*Matrix([

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0]])

**3. Ones matrix of order 5 × 4.**

**ANS :**from sympy import\*

ones(5,4)

*output* :Matrix([

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1]])

**2. Find the data type of the following data by using Python code.**

**a. number**

**ANS :** A= "number"

print(type(A))

output :<class 'str'>

**b. 31.25**

ANS : B= 31.25

print(type(B))

output : <class 'float'>

**c. 8 + 4j**

**ANS:** C= 8 + 4j

print(type(C))

output : <class 'complex'>

**d. Mathematics**

**ANS:** D= "Mathematics"

print(type(D))

output : <class 'str'>

**e. 49**

**ANS:** E=49

print(type(E))

output : <class 'int'>

**3. Write Python program to find the determinant of matrices**

A=[1 0 5]

[2 1 6]

[3 4 0]

B=[2 5]

[-1 4]

**ANS:** import numpy as np

A = np.array([[1, 0, 5], [2, 1, 6], [3, 4, 0]])

B = np.array([[2, 5], [-1, 4]])

det\_A = np.linalg.det(A)

det\_B = np.linalg.det(B)

print("Determinant of Matrix A:", det\_A)

print("Determinant of Matrix B:", det\_B)

output : Determinant of Matrix A: 0.9999999999999989

Determinant of Matrix B: 13.0

**Q.3. a. Attempt any one of the following.**

**1. Write Python program to estimate the value of the integral R *π***

**0 *xsin*(*x*)*dx* using**

**Simpson’s (1/3 )*rd* rule (n=6).**

**ANS :**import math

def f(x):

return x \* math.sin(x)

a = 0

b = math.pi

n = 6

h = (b - a) / n

integral\_value = f(a) + f(b)

for i in range(1, n, 2):

integral\_value += 4 \* f(a + i \* h)

for i in range(2, n-1, 2):

integral\_value += 2 \* f(a + i \* h)

integral\_value \*= h / 3

print(f"The estimated value of the integral is: {integral\_value}")

output : The estimated value of the integral is: 3.1429485487583113

**2. Write Python program to estimate a root of an equation *f*(*x*) = 3*x* − *cos*(*x*) − 1**

**using Newton–Raphson method correct up to four decimal places.**

**ANS:** import math

def f(x):

return 3 \* x - math.cos(x) - 1

def f\_prime(x):

return 3 + math.sin(x)

x0 = 0.0

tolerance = 1e-4

max\_iterations = 100

iteration = 0

while iteration < max\_iterations:

x1 = x0 - f(x0) / f\_prime(x0)

if abs(x1 - x0) < tolerance:

break

x0 = x1

iteration += 1

if iteration < max\_iterations:

print(f"Estimated root: {x1:.4f}")

else:

print("Maximum iterations reached. The root may not have converged.")

*output :* Estimated root: 0.6071

**b. Attempt any one of the following.**

**1. Write Python program to find all positive prime numbers less then given number n.**

**ANS :** def sieve\_of\_eratosthenes(n):

is\_prime = [True] \* (n + 1)

is\_prime[0] = is\_prime[1] = False

for p in range(2, int(n\*\*0.5) + 1):

if is\_prime[p]:

for i in range(p \* p, n + 1, p):

is\_prime[i] = False

primes = [i for i in range(2, n + 1) if is\_prime[i]]

return primes

n = int(input("Enter a positive integer n: "))

prime\_numbers = sieve\_of\_eratosthenes(n)

print("Prime numbers less than", n, "are:", prime\_numbers)

*output :*Enter a positive integer n:

44

Prime numbers less than 44 are: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43]

**2. Write Python program to evaluate f(2.5) by forward difference formula of the given**

**data.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 0 | 1 | 2 | 3 |
| **Y=f(x)** | 2 | 1 | 2 | 10 |

**ANS:** x\_values = [0, 1, 2, 3]

y\_values = [2, 1, 2, 10]

x\_target = 2.5

h = x\_values[1] - x\_values[0]

index = (x\_target - x\_values[0]) / h

i = int(index)

f\_x\_target = y\_values[i] + (x\_target - x\_values[i]) \* (y\_values[i + 1] - y\_values[i]) / h

print("Estimated value of f(2.5) is:", f\_x\_target)

output : Estimated value of f(2.5) is: 6.0

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Shinde Tejal Gorakshanath **Batch No. :-** A **Roll No. :-** **Date :-** …./…/2023

**Title :-** Python Practical **Expt.No. :-** **3 Class :-** SYBCS

**Q.1. Attempt any two of the following. [10]**

**1. Write python code to test whether given number is divisible by 2 or 3 or 5.**

*ANS :n*um*ber = int(input("Enter a number: "))*

if number % 2 == 0:

print(number, "is divisible by 2")

if number % 3 == 0:

print(number, "is divisible by 3")

if number % 52 != 0 and number % 3 != 0 and number % 5 != 0:

print(number, "is not divisible by 2, 3, or 5")

*output :* Enter a Number

12

12 is divisible by 2

12 is divisible by 3

**2. Repeat the following string 11 times using the string operator ‘\*’ on Python.**

**ANS :** T=’python ’ \*11

print(T)

*output*: python python python python python python python python python python python

**a. LATEX**

**ANS :** T=’LATEX ’ \*11

print(T)

*output* : LATEX LATEX LATEX LATEX LATEX LATEX LATEX LATEX LATEX LATEX LATEX

**b. MATLAB**

**ANS :** T=’MATLAB ’ \*11

print(T)

*output* : MATLAB MATLAB MATLAB MATLAB MATLAB MATLAB MATLAB MATLAB MATLAB MATLAB MATLAB

**3. Use Python code to find sum of first thirty natural numbers.**

**ANS :**sum\_of\_numbers = 0

for num in range(1, 31):

sum\_of\_numbers += num

print("The sum of the first thirty natural numbers is:", sum\_of\_numbers)

*output* :**:**The sum of the first thirty natural numbers is: 465

**Q.2. Attempt any two of the following.**

**1. Using Python construct the following matrices.**

**1. An identity matrix of order 10 × 10.**

**ANS :**from sympy import \*

eye(10)

*output* : Matrix([

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 1, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 1, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 1, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]])

**2. Zero matrix of order 7 × 3**

**ANS :.**from sympy import\*

zeros(7,3)

*output* : Matrix([

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0]])

**3. Ones matrix of order 5 × 4.**

**ANS :**from sympy import\*

ones(5,4)

*output :* Matrix([

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1]])

**2. Using python, find the eigenvalues and corresponding eigenvectors of the matrix**

**[3 −2]**

**[6 −4]**

**ANS** from sympy import\*

A=Matrix([[3,-2],[6,-4]])

A.eigenvals

*output :* <bound method MatrixEigen.eigenvals of Matrix([

[3, -2],

[6, -4]])>

2] from sympy import\*

A=Matrix([[3,-2],[6,-4]])

A.eigenvects()

*output* [(-1, 1, [Matrix([

[1/2],

[ 1]])]), (0, 1, [Matrix([

[2/3],

[ 1]])])]

**3. Generate all the prime numbers between 1 to 100 using Python code.**

**ANS:** def is\_prime(n):

if n <= 1:

return False

if n <= 3:

return True

if n % 2 == 0 or n % 3 == 0:

return False

i = 5

while i \* i <= n:

if n % i == 0 or n % (i + 2) == 0:

return False

i += 6

return True

prime\_numbers = [n for n in range(1, 101) if is\_prime(n)]

print("Prime numbers between 1 and 100 are:")

print(prime\_numbers)

*output :*Prime numbers between 1 and 100 are:

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

**Q.3. a. Attempt any one of the following. [7]**

**1. Write Python program to estimate the value of the integral R *π***

**0 *sin*(*x*)*dx* using Simpson’s**

**(1**

**3 )*rd* rule (n=6).**

**ANS :**import math

def f(x):

return math.sin(x)

a = 0

b = math.pi

n = 6

h = (b - a) / n

integral\_sum = f(a) + f(b)

odd\_sum = 0

for i in range(1, n, 2):

x\_i = a + i \* h

odd\_sum += f(x\_i)

even\_sum = 0

for i in range(2, n-1, 2):

x\_i = a + i \* h

even\_sum += f(x\_i)

integral\_estimate = (h / 3) \* (integral\_sum + 4 \* odd\_sum + 2 \* even\_sum)

print("Estimated value of the integral ∫(0 to π) sin(x) dx using Simpson's 1/3 rule (n=6):", integral\_estimate)

*output:* Estimated value of the integral ∫(0 to π) sin(x) dx using Simpson's 1/3 rule (n=6): 2.0008631896735367

**2. Write Python program to evaluate third order forward difference of the given data.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | 0 | 1 | 2 | 3 |
| **Y=f(x)** | 1 | 0 | 1 | 10 |

*Output:* x\_values = [0, 1, 2, 3]

y\_values = [1, 0, 1, 10]

h = x\_values[1] - x\_values[0]

third\_order\_forward\_difference = (1 / (6 \* h)) \* (y\_values[0] - 3 \* y\_values[1] + 3 \* y\_values[2] - y\_values[3])

print("The third-order forward difference is:", third\_order\_forward\_difference)

*output :* The third-order forward difference is: -1.0

**b. Attempt any one of the following.**

**1. Write Python program to evaluate f(3.5) of the given data.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **x** | **1** | **2** | **3** | **4** | **5** |
| **Y=f(x)** | **30** | **50** | **55** | **40** | **11** |

**ANS :**x\_values = [1, 2, 3, 4, 5]

y\_values = [30, 50, 55, 40, 11]

x\_target = 3.5

interpolated\_value = None

x1 = None

x2 = None

for i in range(len(x\_values) - 1):

if x\_values[i] <= x\_target <= x\_values[i + 1]:

x1 = x\_values[i]

x2 = x\_values[i + 1]

break

if x1 is not None and x2 is not None:

y1 = y\_values[x\_values.index(x1)]

y2 = y\_values[x\_values.index(x2)]

interpolated\_value = y1 + (x\_target - x1) \* (y2 - y1) / (x2 - x1)

if interpolated\_value is not None:

print("The interpolated value of f(3.5) is:", interpolated\_value)

else:

print("The value of x\_target is outside the range of x\_values.")

*output :*The interpolated value of f(3.5) is: 47.5

**2. Write Python program to estimate the value of the integral R 10**

**2**

**1**

**(1+*x*)*dx* using Trapezoidal**

**rule (n=5).**

**ANS :**def f(x):

return 1 + x

a = 2

b = 10

n = 5

h = (b - a) / n

integral\_sum = f(a) + f(b)

interior\_sum = 0

for i in range(1, n):

x\_i = a + i \* h

interior\_sum += f(x\_i)

integral\_estimate = (h / 2) \* (integral\_sum + 2 \* interior\_sum)

print("Estimated value of the integral ∫(2 to 10) (1+x) dx using the Trapezoidal rule (n=5):", integral\_estimate)

*output :*  Estimated value of the integral ∫(2 to 10) (1+x) dx using the Trapezoidal rule (n=5): 56.0

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Shinde Tejal Gorakshanath **Batch No. :-** A **Roll No. :-** **Date :-** …./…/2023

**Title :-** Python Practical  **Expt.No. :-** **4 Class :-** SYBCS

**Q.1. Attempt any two of the following. [10]**

**1. Using python code sort the tuple in ascending and descending order 5, -3, 0, 1, 6,-6, 2.**

**ANS :**  original\_tuple = (5, -3, 0, 1, 6, -6, 2)

ascending\_sorted\_tuple = tuple(sorted(original\_tuple))

descending\_sorted\_tuple = tuple(sorted(original\_tuple, reverse=True))

print("Ascending Sorted Tuple:", ascending\_sorted\_tuple)

print("Descending Sorted Tuple:", descending\_sorted\_tuple)

*output :* Ascending Sorted Tuple: (-6, -3, 0, 1, 2, 5, 6)

Descending Sorted Tuple: (6, 5, 2, 1, 0, -3, -6)

**2. Write python program which deals with concatenation and repetition of lists.**

**List1 = [15, 20, 25, 30, 35, 40]**

**List2 = [7, 14, 21, 28, 35, 42]**

**(a) Find List1 + List2**

**ANS:** List1 = [15, 20, 25, 30, 35, 40]

List2 = [7, 14, 21, 28, 35, 42]

concatenated\_list = List1 + List2

print("Concatenated list:", concatenated\_list)

output : Concatenated list: [15, 20, 25, 30, 35, 40, 7, 14, 21, 28, 35, 42]

**(b) Find 9\*List1**

**ANS:** List1 = [15, 20, 25, 30, 35, 40]

List2 = [7, 14, 21, 28, 35, 42]

nine\_times\_list1 = [9 \* element for element in List1]

print("9 times List1:", nine\_times\_list1)

output: 9 times List1: [135, 180, 225, 270, 315, 360]

**(c) Find 7\*List2**

**ANS :** List1 = [15, 20, 25, 30, 35, 40]

List2 = [7, 14, 21, 28, 35, 42]

seven\_times\_list2 = [7 \* element for element in List2]

print("7 times List2:", seven\_times\_list2)

output : 7 times List2: [49, 98, 147, 196, 245, 294]

**3. Write Python code to find the square of odd numbers from 1 to 20 using while loop.**

**ANS :** number = 1

while number <= 20:

if number % 2 != 0:

square = number \*\* 2

print("Square of", number, "is", square)

number += 1

*output* :Square of 1 is 1

Square of 3 is 9

Square of 5 is 25

Square of 7 is 49

Square of 9 is 81

Square of 11 is 121

Square of 13 is 169

Square of 15 is 225

Square of 17 is 289

Square of 19 is 361

​

**Q.2. Attempt any two of the following. [10]**

**1. Using Python construct the following matrices.**

**1. An identity matrix of order 10 × 10.**

**ANS:** from sympy import \*

eye(10)

*output* : Matrix([

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 1, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 1, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 1, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]])

**2. Zero matrix of order 7 × 3.**

**ANS :**from sympy import\*

zeros(7,3)

*output :*Matrix([

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0]])

**3. Ones matrix of order 5 × 4.**

**ANS :**from sympy import\*

ones(5,4)

*output* :Matrix([

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1]])

**2. Find the data type of the following data by using Python code.**

**a. number**

**ANS :** A= "number"

print(type(A))

output :<class 'str'>

**b. 31.25**

ANS : B= 31.25

print(type(B))

output : <class 'float'>

**c. 8 + 4j**

**ANS:** C= 8 + 4j

print(type(C))

output : <class 'complex'>

**d. Mathematics**

**ANS:** D= "Mathematics"

print(type(D))

output : <class 'str'>

**e. 49**

**ANS:** E=49

print(type(E))

output : <class 'int'>

**3. Write Python program to find the determinant of matrices**

A=[1 0 5]

[2 1 6]

[3 4 0]

B=[2 5]

[-1 4]

**ANS:** import numpy as np

A = np.array([[1, 0, 5], [2, 1, 6], [3, 4, 0]])

B = np.array([[2, 5], [-1, 4]])

det\_A = np.linalg.det(A)

det\_B = np.linalg.det(B)

print("Determinant of Matrix A:", det\_A)

print("Determinant of Matrix B:", det\_B)

output : Determinant of Matrix A: 0.9999999999999989

Determinant of Matrix B: 13.0

**Q.3. a. Attempt any one of the following.**

**1. Write Python program to estimate the value of the integral R *π***

**0 *xsin*(*x*)*dx* using**

**Simpson’s (1/3 )*rd* rule (n=6).**

**ANS :**import math

def f(x):

return x \* math.sin(x)

a = 0

b = math.pi

n = 6

h = (b - a) / n

integral\_value = f(a) + f(b)

for i in range(1, n, 2):

integral\_value += 4 \* f(a + i \* h)

for i in range(2, n-1, 2):

integral\_value += 2 \* f(a + i \* h)

integral\_value \*= h / 3

print(f"The estimated value of the integral is: {integral\_value}")

output : The estimated value of the integral is: 3.1429485487583113

**2. Write Python program to estimate a root of an equation *f*(*x*) = 3*x* − *cos*(*x*) − 1**

**using Newton–Raphson method correct up to four decimal places.**

**ANS:** import math

def f(x):

return 3 \* x - math.cos(x) - 1

def f\_prime(x):

return 3 + math.sin(x)

x0 = 0.0

tolerance = 1e-4

max\_iterations = 100

iteration = 0

while iteration < max\_iterations:

x1 = x0 - f(x0) / f\_prime(x0)

if abs(x1 - x0) < tolerance:

break

x0 = x1

iteration += 1

if iteration < max\_iterations:

print(f"Estimated root: {x1:.4f}")

else:

print("Maximum iterations reached. The root may not have converged.")

*output :* Estimated root: 0.6071

**b. Attempt any one of the following.**

**1. Write Python program to find all positive prime numbers less then given number n.**

**ANS :** def sieve\_of\_eratosthenes(n):

is\_prime = [True] \* (n + 1)

is\_prime[0] = is\_prime[1] = False

for p in range(2, int(n\*\*0.5) + 1):

if is\_prime[p]:

for i in range(p \* p, n + 1, p):

is\_prime[i] = False

primes = [i for i in range(2, n + 1) if is\_prime[i]]

return primes

n = int(input("Enter a positive integer n: "))

prime\_numbers = sieve\_of\_eratosthenes(n)

print("Prime numbers less than", n, "are:", prime\_numbers)

*output :*Enter a positive integer n:

44

Prime numbers less than 44 are: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43]

**2. Write Python program to evaluate f(2.5) by forward difference formula of the given**

**data.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 0 | 1 | 2 | 3 |
| **Y=f(x)** | 2 | 1 | 2 | 10 |

**ANS:** x\_values = [0, 1, 2, 3]

y\_values = [2, 1, 2, 10]

x\_target = 2.5

h = x\_values[1] - x\_values[0]

index = (x\_target - x\_values[0]) / h

i = int(index)

f\_x\_target = y\_values[i] + (x\_target - x\_values[i]) \* (y\_values[i + 1] - y\_values[i]) / h

print("Estimated value of f(2.5) is:", f\_x\_target)

output : Estimated value of f(2.5) is: 6.0

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Shinde Tejal Gorakshanath **Batch No. :-** A **Roll No. :-** **Date :-** …./…/2023

**Title :-** Python Practical **Expt.No. :-** **5 Class :-** SYBCS

**Q.1. Attempt any two of the following. [10]**

**1. Using sympy module of python find the following for the matrices**

**A = [1 1 0]**

**[8 5 2]**

**[2 −6 2]**

**and *B* =[9 0 3]**

**[1 4 1]**

**[1 0 −1]**

**ANS:**

(a) 2A + B.

from sympy import \*

A=Matrix ([[1,1,0],[8,5,2],[2,-6,2]])

B=Matrix ([[9,0,3],[1, 4, 1],[1,0,-1]])

2\*A + B

**output :** Matrix([

[11, 2, 3],

[17, 14, 5],

[ 5, -12, 3]])

(b) 3A – 5B.

ANS : from sympy import \*

A=Matrix ([[1,1,0],[8,5,2],[2,-6,2]])

B=Matrix ([[9,0,3],[1, 4, 1],[1,0,-1]])

3\*A – 5\*B

output :Matrix([

[-42, 3, -15],

[ 19, -5, 1],

[ 1, -18, 11]])

(c) *A*−1

**ANS** :from sympy import \*

A=Matrix ([[1,1,0],[8,5,2],[2,-6,2]])

B=Matrix ([[9,0,3],[1, 4, 1],[1,0,-1]])

Matrix.inv(A)

output : Matrix([

[ 11/5, -1/5, 1/5],

[ -6/5, 1/5, -1/5],

[-29/5, 4/5, -3/10]])

(d) *B*3.

**ANS**

(e) *AT* + *BT* .

**ANS :**

**2. Evaluate following expression on Python.**

**(a) M =[1,2,3,4], Find length M.**

**ANS** : M =[1,2,3,4]

len(M)

output : 4

**(b) L=“XYZ”+“pqr”, Find L.**

**ANS** : L = "XYZ" + "pqr"

print(L)

output : XYZpqr

**(c) s=‘Make In India’, Find (s[:7]) & (s[:9]).**

**ANS** : s = 'Make In India'

result\_1 = s[:7]

result\_2 = s[:9]

print(f"First Result (s[:7]): {result\_1}")

print(f"Second Result (s[:9]): {result\_2}")

output : First Result (s[:7]): Make In

Second Result (s[:9]): Make In I

**3. Use Python code to generate the square root of numbers from 21 to 49.**

**ANS :** import math

for num in range(21, 50):

square\_root = math.sqrt(num)

print(f"The square root of {num} is {square\_root:.4f}")

output :The square root of 21 is 4.5826

The square root of 22 is 4.6904

The square root of 23 is 4.7958

The square root of 24 is 4.8990

The square root of 25 is 5.0000

The square root of 26 is 5.0990

The square root of 27 is 5.1962

The square root of 28 is 5.2915

The square root of 29 is 5.3852

The square root of 30 is 5.4772

The square root of 31 is 5.5678

The square root of 32 is 5.6569

The square root of 33 is 5.7446

The square root of 34 is 5.8310

The square root of 35 is 5.9161

The square root of 36 is 6.0000

The square root of 37 is 6.0828

The square root of 38 is 6.1644

The square root of 39 is 6.2450

The square root of 40 is 6.3246

The square root of 41 is 6.4031

The square root of 42 is 6.4807

The square root of 43 is 6.5574

The square root of 44 is 6.6332

The square root of 45 is 6.7082

The square root of 46 is 6.7823

The square root of 47 is 6.8557

The square root of 48 is 6.9282

The square root of 49 is 7.0000

**Q.2. Attempt any two of the following. [10]**

**1. Using Python construct the following matrices.**

1. An identity matrix of order 10 × 10.

**ANS** : from sympy import\*

zeros(10)

output :Matrix([

[ 1,0,0,0,0,0,0,0,0,0]

[ 0,1,0,0,0,0,0,0,0,0]

[ 0,0,1,0,0,0,0,0,0,0]

[ 0,0,0,1,0,0,0,0,0,0]

[ 0,0,0,0,1,0,0,0,0,0]

[ 0,0,0,0,0,1,0,0,0,0]

[ 0,0,0,0,0,0,1,0,0,0]

[ 0,0,0,0,0,0,0,1,0,0]

[0,0,0,0,0,0,0,0,1,0]

[ 0,0,0,0,0,0,0,0,0,1]])

2. Zero matrix of order 7 × 3.

**ANS :** from sympy import \*

zeros(7,3)

*output* :Matrix([

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

[0, 0, 0]])

3. Ones matrix of order 5 × 4.

**ANS** : from sympy import \*

ones(5,4)

*output* :Matrix([

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1],

[1, 1, 1, 1]])

**2. Using linsolve command in python, solve the following system of linear equations.**

***x* − 2*y* + 3*z* = 7**

**2*x* + *y* + *z* = 4**

**−3*x* + 2*y* − 2*z* = −10**

**ANS :** from sympy import symbols, Eq, linsolve

x, y, z = symbols('x y z')

eq1 = Eq(x - 2\*y + 3\*z, 7)

eq2 = Eq(2\*x + y + z, 4)

eq3 = Eq(-3\*x + 2\*y - 2\*z, -10)

solution = linsolve([eq1, eq2, eq3], x, y, z)

print("Solution to the system of linear equations:")

print(solution)

*output* :Solution to the system of linear equations:

{(2, -1, 1)}

**3. Generate all relatively prime numbers to 111 which are less than 150 using Python code.**

**ANS:** import math

def gcd(a, b):

return math.gcd(a, b)

def is\_relatively\_prime(a, b):

return gcd(a, b) == 1

relatively\_prime\_numbers = []

for num in range(1, 150):

if is\_relatively\_prime(num, 111):

relatively\_prime\_numbers.append(num)

print("Relatively prime numbers to 111 which are less than 150:")

print(relatively\_prime\_numbers)

*output* : Relatively prime numbers to 111 which are less than 150:

[1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26, 28, 29, 31, 32, 34, 35, 38, 40, 41, 43, 44, 46, 47, 49, 50, 52, 53, 55, 56, 58, 59, 61, 62, 64, 65, 67, 68, 70, 71, 73, 76, 77, 79, 80, 82, 83, 85, 86, 88, 89, 91, 92, 94, 95, 97, 98, 100, 101, 103, 104, 106, 107, 109, 110, 112, 113, 115, 116, 118, 119, 121, 122, 124, 125, 127, 128, 130, 131, 133, 134, 136, 137, 139, 140, 142, 143, 145, 146, 149]

**Q.3. a.** Attempt **any one of the following. [7]**

1. Write Python code to find eigenvalues and corresponding eigenvectors of the matrix

*A* =1 3 3]

[2 2 3]

[4 2 1]

and hence find matrix *P* with diagonalize to *A*.

**ANS:**  import numpy as np

A = np.array([[1, 3, 3],

[2, 2, 3],

[4, 2, 1]])

eigenvalues, eigenvectors = np.linalg.eig(A)

print("Eigenvalues:")

print(eigenvalues)

print("\nEigenvectors:")

print(eigenvectors)

D = np.diag(eigenvalues)

P = np.linalg.inv(eigenvectors)

diagonalized\_A = np.dot(np.dot(P, D), eigenvectors)

print("\nMatrix P that diagonalizes A:")

print(diagonalized\_A)

*output:* Eigenvalues:

[ 7. -2. -1.]

Eigenvectors:

[[ 5.77350269e-01 4.08248290e-01 5.55111512e-17]

[ 5.77350269e-01 4.08248290e-01 -7.07106781e-01]

[ 5.77350269e-01 -8.16496581e-01 7.07106781e-01]]

Matrix P that diagonalizes A:

[[ 1.33333333 1.64991582 0.40824829]

[ 8.01387685 4.66666667 -0.57735027]

[ 7.34846923 5.19615242 -2. ]]

**2. Write Python program to estimate a root of an equation *f*(*x*) = 3*x*2+4*x*−10 using Newton–Raphson method correct up to four decimal places.**

**ANS :** def f(x):

return 3\*x\*\*2 + 4\*x - 10

def f\_prime(x):

return 6\*x + 4

def newton\_raphson(func, func\_prime, x0, tol=1e-4, max\_iter=100):

for i in range(max\_iter):

x1 = x0 - func(x0) / func\_prime(x0)

if abs(x1 - x0) < tol:

return round(x1, 4)

x0 = x1

return "No root found within the given number of iterations."

x0 = 1

root = newton\_raphson(f, f\_prime, x0)

print(f"The estimated root of the equation is: {root}")

*output*:The estimated root of the equation is: 1.277

**b. Attempt any one of the following. [8]**

**1. Write Python program to obtained the approximate real root of *x*3 −4*x*−9 = 0 by using Regula-falsi method.**

**ANS :** def f(x):

return x\*\*3 - 4\*x - 9

def regula\_falsi(func, a, b, tol=1e-6, max\_iter=100):

if func(a) \* func(b) >= 0:

print("Regula Falsi method cannot guarantee convergence.")

return None

for i in range(max\_iter):

c = (a \* func(b) - b \* func(a)) / (func(b) - func(a))

if abs(func(c)) < tol:

print(f"Root found at {c} with an error of {abs(func(c))}.")

return c

if func(c) \* func(a) < 0:

b = c

else:

a = c

print("Method did not converge within the maximum number of iterations.")

return None

b = 3

regula\_falsi(f, a, b)

*output*:Root found at 2.7065279131342495 with an error of 7.43548660864235e-07

2.7065279131342495

**2. Write Python program to evaluate f(3.5) by forward difference formula of the given**

**data**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1 | 2 | 3 | 4 | 5 |
| Y=f(X) | 41 | 62 | 65 | 50 | 17 |

**ANS** : X = [1, 2, 3, 4, 5]

Y = [41, 62, 65, 50, 17]

def forward\_difference(Y):

n = len(Y)

for i in range(1, n):

for j in range(n - i):

Y[j] = Y[j + 1] - Y[j]

return Y[0]

h = X[1] - X[0]

x\_target = 3.5

s = (x\_target - X[0]) / h

result = Y[0]

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

term = 1

for j in range(i):

term \*= (s - j)

term /= (j + 1)

result += term \* forward\_difference(Y[:len(Y) - i])

print(f"The value of f(3.5) is approximately: {result}")

*output*: The value of f(3.5) is approximately: 12.2109375

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Chaudhari Aaditya Bapusaheb **Batch No. :-** A **Roll No. :-** **Date :-** …./…/2023

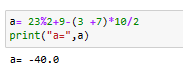
**Title :- Python Practical Expt.No. :-** **6 Class :-** SYBCS

**Q.1. Attempt any two of the following.**

**1. Using Python evaluate each of the following expression.**

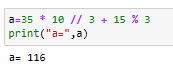
a. 23 modulus 2 + 9 - (3 +7) × 10 ÷ 2

-->



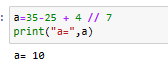
b. 35 × 10 floor division 3 + 15 modulus 3

-->

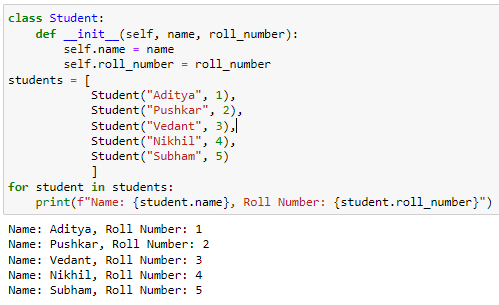


c. 35 − 25 + 4 floor division 7

-->



**2. Write Python code to list name and roll number of 5 students in B.Sc.(Computer science).**

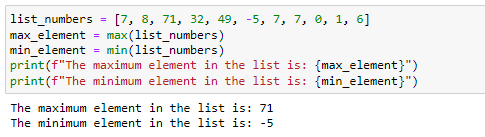
**-->**

**Code:-**

**Output:-**

**3. Write Python code to find maximum and minimum element in the given list.**

**[7*,* 8*,* 71*,* 32*,* 49*,* −5*,* 7*,* 7*,* 0*,* 1*,* 6]**

🡪**Code:-**

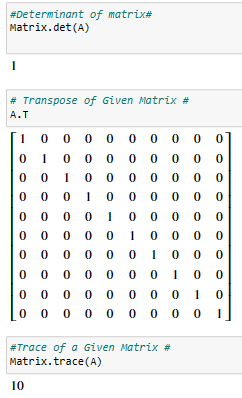
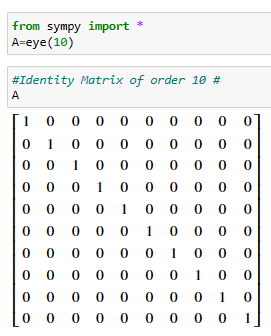
**Output:-**

**Q.2. Attempt any two of the following. [10]**

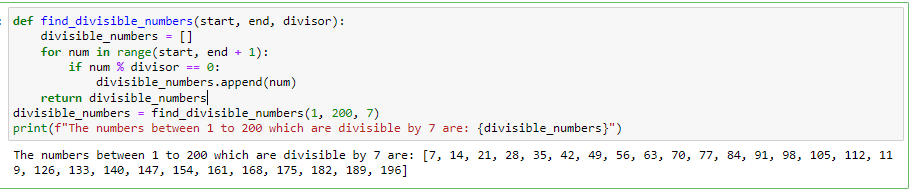
**1. Using Python code construct identity matrix of order 10 and hence find determinant,**

**trace and transpose of it.**

**🡪**



**2. Find number between 1 to 200 which are divisible by 7 using Python code**

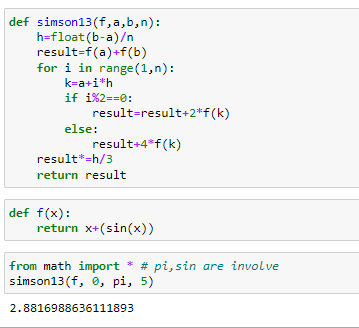
**-->** 

**Q.3. a. Attempt any one of the following. [7]**

****

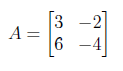
**1. Write Python program to estimate the value of the integral using**

**Simpson’s (1/3 )*rd* rule (n=5).**

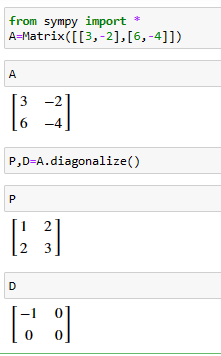
-->

**Code:-**

**Output:-**

**2. Write Python code to diagonalize matrix and find matrix *P* with diagonalize of *A* and diagonal matrix *D*.**

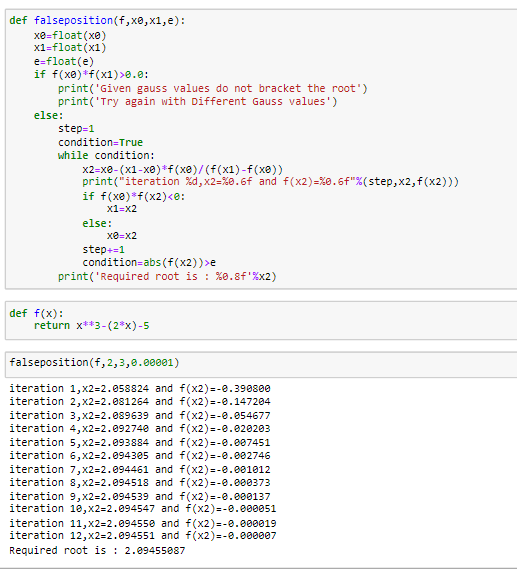
-->



**b. Attempt any one of the following. [8]**

**1. Write a Python program to obtained the approximate real root of *x3*− 2*x* − 5 = 0**

**in [2,3] using Regula-falsi method.**

**-->**

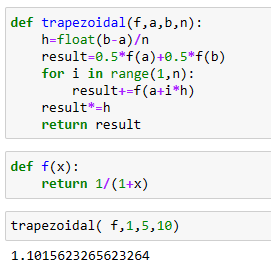
**Code:-**

****

**Output:-**

******

**2. Write a Python program to estimate the value of the integral using Trapezoidal rule (n=10).**

**-->**

**Code:-**

**Output:-**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Chaudhari Aaditya Bapusaheb **Batch No. :-** A **Roll No. :-** **Date :-** …./…/2023

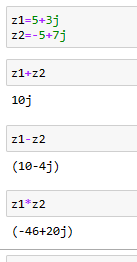
**Title :- Python Practical Expt.No. :-** **7** **Class :-** SYBCS

**Q.1. Attempt any two of the following. [10]**

**1. Using Python, evaluate the following expression of two complex number *z*1 = 5+3*j***

**and *z*2 = −5 + 7*j.***

**a. *z*1 + *z*2**

**b. *z*1 − *z*2**

**c. *z*1 ∗ *z*2**

**-->**

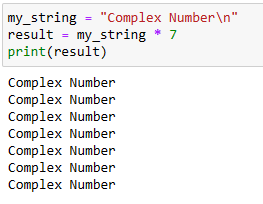
**a. *z*1 + *z*2 🡪**

**b. *z*1 − *z*2 🡪**

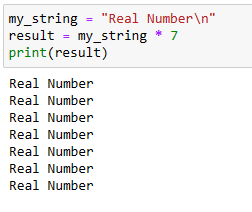
**c. *z*1 ∗ *z*2 🡪**

**2. Repeat the following string 7 times using the string operator ‘\*’ on Python.**

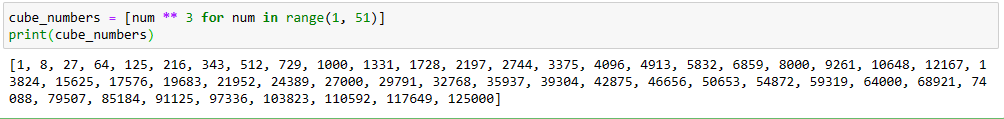
**a. Complex Number**

**-->**

**b. Real Number**

**-->**

**3. Write Python code to generate cube of numbers from 1 to 50.**

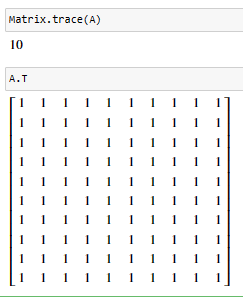
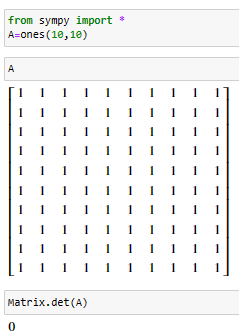
**-->**

**Q.2. Attempt any two of the following. [10]**

**1. Using Python code construct 0nes matrix of order 10 × 10 and hence find determinant,**

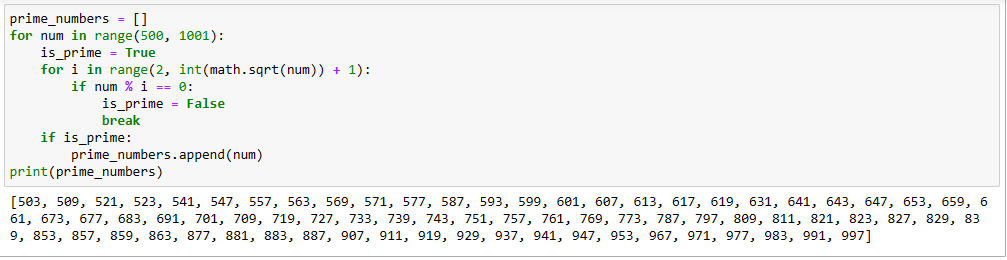
**trace and transpose of it.**

**-->**

****

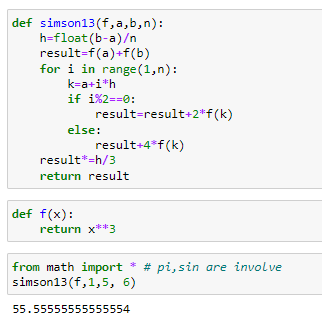
**2. Generate all the prime numbers between 500 to 1000 using Python program.**

**-->**

****

**Q.3. a. Attempt any one of the following. [7]**

**1. Write Python program to estimate the value of the integral using Simpson’s (1/3 ) *rd* rule (n=6).**

**-->**

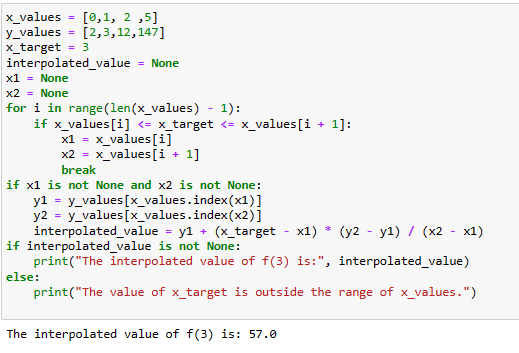
**Code:-**

**Output:-**

**2. Write Python program to evaluate interpolate value *f*(3) of the given data.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **0** | **1** | **2** | **5** |
| **Y=f(x)** | **2** | **3** | **12** | **147** |

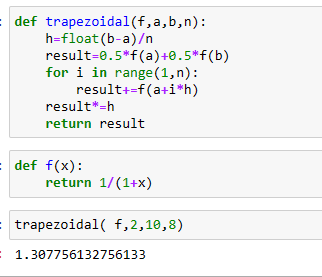
**-->**

****

**b. Attempt any one of the following. [8]**

****

**1. Write Python program to estimate the value of the integral using Trapezoidal rule (n=8).**

**-->**

**Code:-**

**Output:-**

**2. Write Python program to evaluate f(2.8) using backward difference formula of the**

**given data.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **0** | **1** | **2** | **3** |
| **Y=f(x)** | **1** | **0** | **1** | **10** |

**-->**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

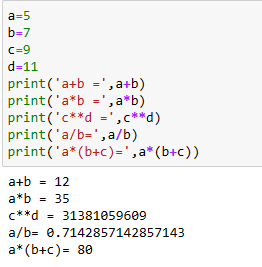
**Name :-** Chaudhari Aaditya Bapusaheb **Batch No. :-** **A** **Roll No. :- Date :- …./…/2023**

**Title :- Python Practical Expt.No. :- 8 Class :-** SYBCS

**Q.1. Attempt any two of the following. [10]**

**1. Use Python code to find *a* + *c, ab, cd, a/b* and *a*(*b* + *c*),**

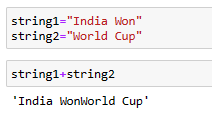
**where *a* = 5*, b* = 7*, c* = 9*, d* = 11.**

**-->**

**2. The following two statements using the ‘+’string operation on Python.**

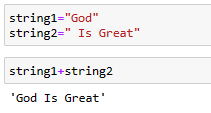
**a. string1 = India Won, string2 = World Cup**

**-->**

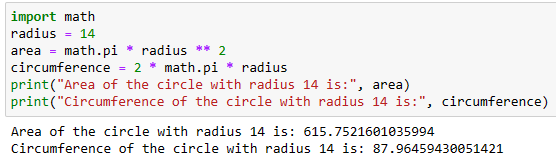
****

**b. string1 = God, string2 = is Great**

**-->**

****

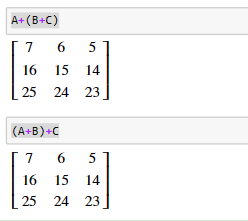
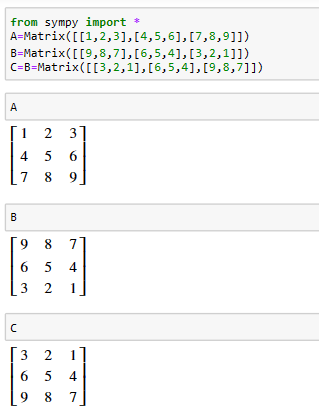
**3. Write Python code to find area and circumference of circle with radius 14.**

**-->**

**Q.2. Attempt any two of the following. [10]**

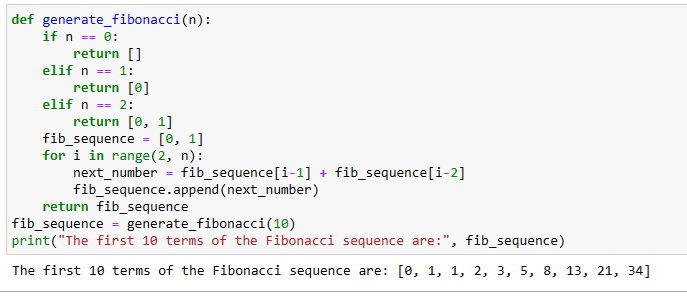
**1. Using Python code logically verify associativity of matrices with respective to matrix addition (use proper matrices).**

**-->**

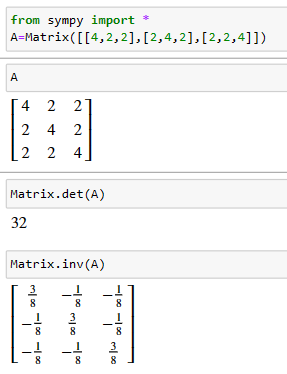
****

Associativiy Property holdes in terms of matrix.

**2. Write Python code to generate 10 terms of Fibonacci Sequence using loop.**

**-->** ****

**3. Using Python code, find determinant and inverse of the matrix if exist.**

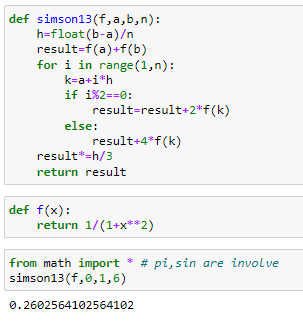


**A =**

**-->**

**Q.3. a. Attempt any one of the following. [7]**

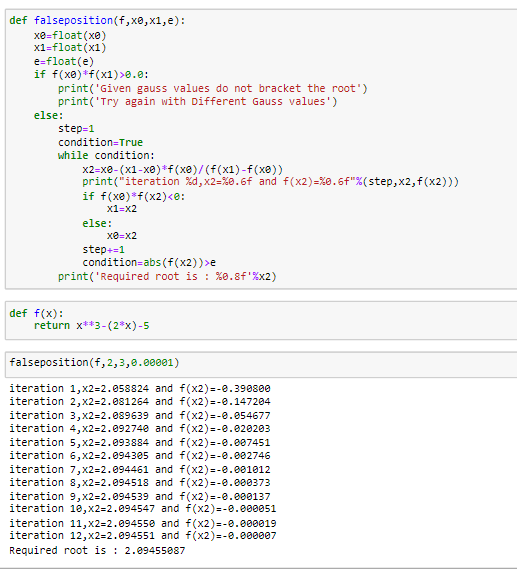
**1. Write Python program to estimate the value of the integral  using Simpson’s (1/3 )*rd* rule (n=6).**

**-->**

**Code:-**

**Output:-**

**2. Write Python program to obtained the approximate real root of *x*3 − 2*x* − 5 = 0 in**

**[2,3] using Regula-falsi method.**

**-->**

**Code:-**

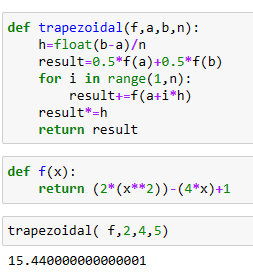
****

**Output:-**

**b. Attempt any one of the following. [8]**

**1. Write Python program to estimate the value of the integral using**

**Trapezoidal rule (n=5).**

**-->**

**Code:-**

**Output:-**

**2. Write Python program to evaluate fourth order forward difference of the given data.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **x** | **1** | **2** | **3** | **4** | **5** |
| **Y=f(x)** | **41** | **62** | **65** | **50** | **17** |

**-->**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

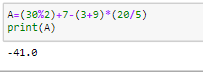
**MATHEMATICS**

**Name :-** Chaudhari Aaditya Bapusaheb **Batch No. :-** **A Roll No. :- Date :- …./…/2023**

**Title :- Python Practical Expt.No. :- 9 Class :- SYBCS**

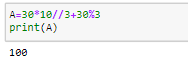
**Q.1. Attempt any two of the following. [10]**

**1. Using Python evaluate each of the following expression**.

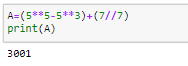
a. 30 modulus 2 + 7 - (3 +9) × 20 ÷ 5.

-->

b. 30 × 10 floor division 3 + 30 modulus 3.

-->

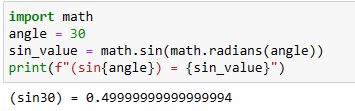
c. 55 - 53+ 7 floor division 7.

-->

**2. Use print command on Python to find**

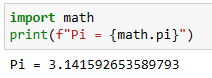
(a) sin30.

-->



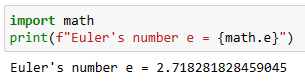
(b) Pi.

-->

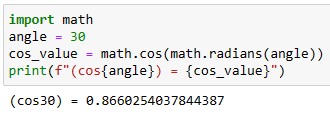


(c) e.

-->

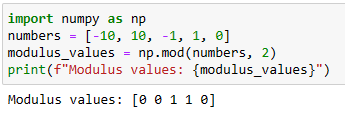


(d) cos30.

****

**3. Write Python code to generate modulus value of -10 ,10, -1,1,0**.

**-->**

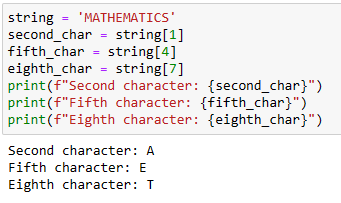
****

**Q.2. Attempt any two of the following. [10]**

**1. Use Python code to generate second, fifth, eight characters from string**

**‘MATHEMATICS ’**

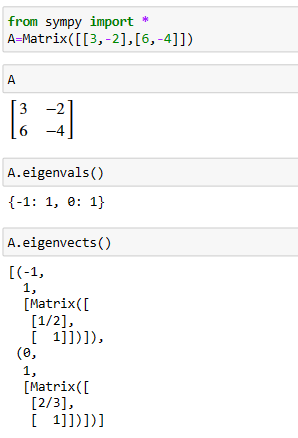
**-->**

****

**2. "Using python find the eigenvalues and corresponding eigenvectors of the matrix**

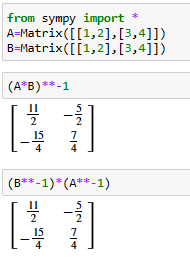
****

**-->**

****

**3. Write Python code to verify (*AB*)-1 = *B -1 A -1* (Use proper matrices *A* and *B*).**

**-->**

****

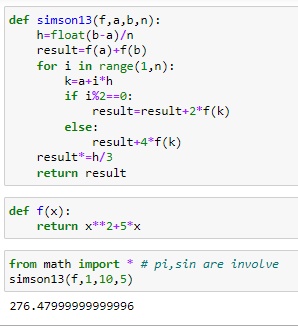
***Here*, (*AB*)-1 = *B -1 A -1 is Verified using A and B matrix.***

**Q.3. a. Attempt any one of the following. [7]**

****

**1. Write Python program to estimate the value of the integral using**

**Simpson’s (1/3 )*rd* rule (n=5).**

**-->**

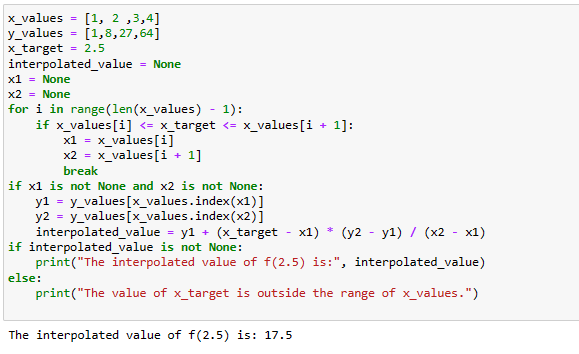
**Code:-**

**Output:-**

**2. Write Python program to evaluate interpolate value *f*(2*.*5) of the given data.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **1** | **2** | **3** | **4** |
| **Y=f(x)** | **1** | **8** | **27** | **64** |

**-->**

****

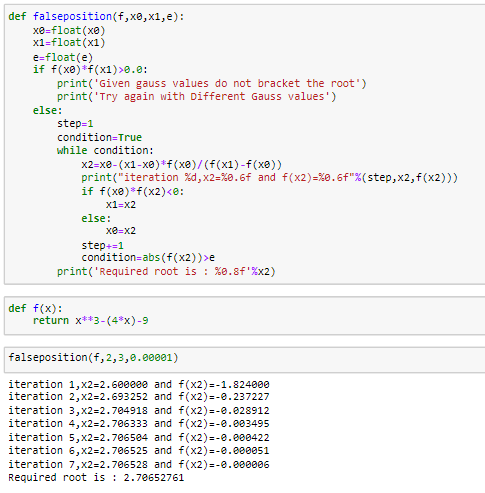
**Code:-**

**Output:-**

**b. Attempt any one of the following. [8]**

**1. Write Python program to obtained the approximate real root of *x*3 −4*x*−9 = 0 by**

**using Regula-falsi method.**

**-->**

**Code:-**

**Output:-**

**2. Write Python program to evaluate fourth order forward difference of the given data.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **x** | **x** | **2** | **3** | **4** | **5** |
| **Y=f(x)** | **40** | **60** | **65** | **50** | **18** |

**-->**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

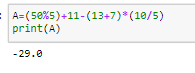
**MATHEMATICS**

**Name :-** Chaudhari Aditya Bapusaheb **Batch No. :-** **A** **Roll No. :- Date :- …./…/2023**

**Title :- Python Practical Expt.No. :- 10 Class :- SYBCS**

**Q.1. Attempt any two of the following. [10]**

**1. Using Python evaluate each of the following expression.**

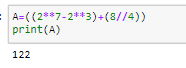
a. 50 modulus 5 + 11 - (13 +7) × 10 ÷ 5

-->

b. 60 × 20 floor division 3 + 15 modulus 3

-->

c. 27- 23 + 8 floor division 4

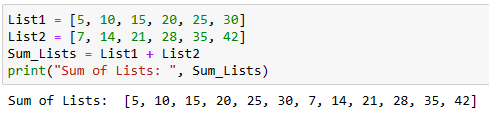
-->

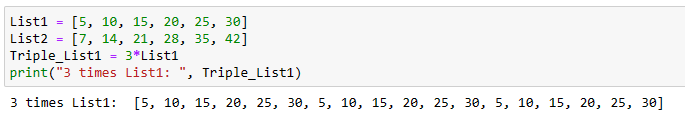
**2. Using Python code**

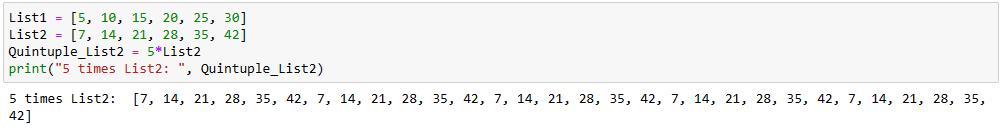
**List1 = [5, 10, 15, 20, 25, 30] and List2 = [7, 14, 21, 28, 35, 42]**

**Evaluate**

1. List1 + List2

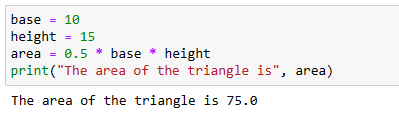


1. 3\*List1
2. 5\*List2



**3. Write Python code to find area of triangle whose base is 10 and height is 15.**

**-->**

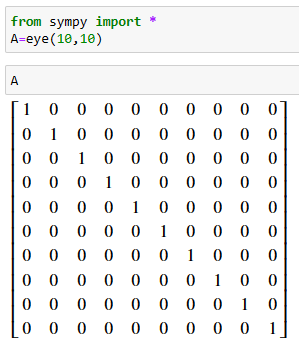
****

**Q.2. Attempt any two of the following. [10]**

**1. Using Python construct the following matrices.**

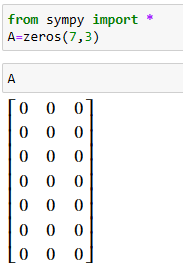
1. An identity matrix of order 10 × 10.

-->

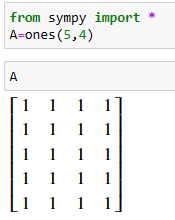


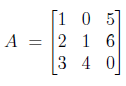
2. Zero matrix of order 7 × 3.

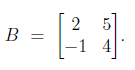
-->



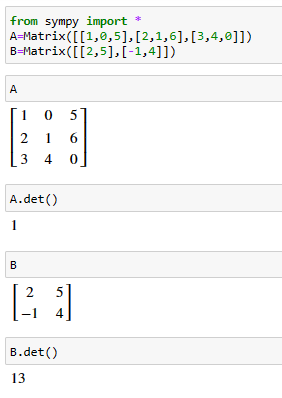
3. Ones matrix of order 5 × 4.

--> 

**2. Write Python program to find the determinant of matrix**

****

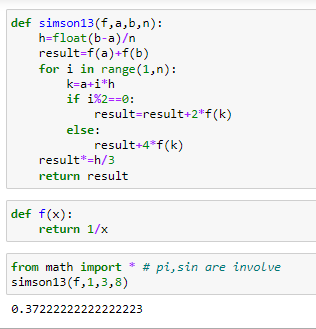
**-->**

****

**Q.3. a. Attempt any one of the following. [7]**

****

**1. Write Python program to estimate the value of the integral using Simpson’s**

**(1/3 )*rd* rule (n=8).**

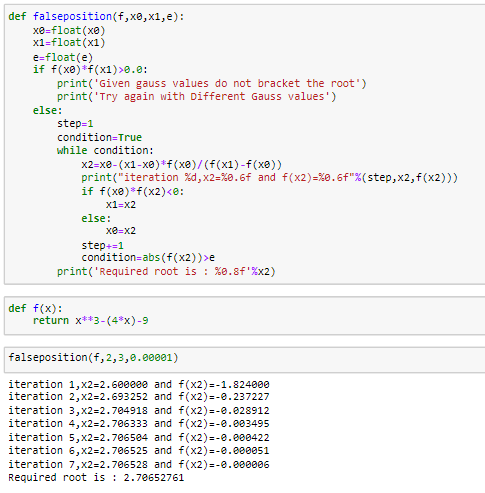
**-->**

**Code:-**

**Output:**

**b. Attempt any one of the following. [8]**

**1. Write Python program to obtained the approximate real root of *X3*−4*x*−9 = 0 by**

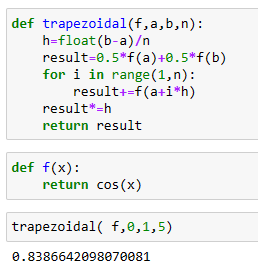
**using Regula-falsi method.**

**--> Code:-**

**Output:-**

****

**2. Write Python program to estimate the value of the integral using Trapezoidal rule (n=5).**

**-->**

**Code:-**

**Output:-**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A** Roll No. :- 3 Date :- …./…/2023

**Title :- Python Practical Expt.No. :- 11 Class :-** SYBCS

**Q.1. Attempt any two of the following. [10]**

1. Evaluate following expression on Python.

(a) M =[1,2,3,4,5,6,7], Find length M.

(b) L=“XY”+“pqr”, Find L.

(c) s=‘Make In India’, Find (s[:5]) & (s[:9]).

ANS:

1)M = [1, 2, 3, 4, 5, 6, 7]

length\_M = len(M)

print(length\_M)

2) L = "XY" + "pqr"

print(L)

3) s = 'Make In India'

substring\_1 = s[:5]

substring\_2 = s[:9]

print(substring\_1, substring\_2)

2. Use Python to evaluate expression of the following matrix.

*A=*

(a) Eigen Value of *A*.

(b) determinant of *A*.

(c) inverse of *A*.

ANS:

import numpy as np

A = np.array([[1, 1, 1], [0, 1, 1], [0, 0, 1]])

# (a) Eigenvalues of A

eigenvalues = np.linalg.eigvals(A)

print("Eigenvalues of A:")

print(eigenvalues)

# (b) Determinant of A

determinant = np.linalg.det(A)

print("Determinant of A:")

print(determinant)

# (c) Inverse of A

try:

inverse\_A = np.linalg.inv(A)

print("Inverse of A:")

print(inverse\_A)

except np.linalg.LinAlgError:

print("A is not invertible (singular matrix).")

3. Write Python code to reverse the string S=[3,4,5,6,7,8,9,10,11,12,13].

ANS:

S = [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

S\_reversed = S[::-1]

print(S\_reversed)

**Q.2. Attempt any two of the following. [10]**

1. Using Python code to list Name of 5 teacher in your college with their subject.

ANS:

teachers = [

{"name": "Teacher 1", "subject": "Mathematics"},

{"name": "Teacher 2", "subject": "Physics"},

{"name": "Teacher 3", "subject": "Computer Science"},

{"name": "Teacher 4", "subject": "History"},

{"name": "Teacher 5", "subject": "English Literature"}

]for teacher in teachers:

print(f"Name: {teacher['name']}, Subject: {teacher['subject']}")

2. Use linsolve command in python to solve the following system of linear equations.

*x* − 2*y* + 3*z* = 7

2*x* + *y* + *z* = 4

−3*x* + 2*y* − 2*z* = −10

ANS:

from sympy import symbols, Eq, linsolve

x, y, z = symbols('x y z')

eq1 = Eq(x - 2\*y + 3\*z, 7)

eq2 = Eq(2\*x + y + z, 4)

eq3 = Eq(-3\*x + 2\*y - 2\*z, -10)

solution = linsolve([eq1, eq2, eq3], (x, y, z))

print("Solution to the system of equations:")

print(solution)

3. Generate all the prime numbers between 51 to 100 using Python program.

**Q.3. a. Attempt any one of the following. [7]**

1. Write Python program to estimate the value of the integral x *dx* using Simpson’s (3/8 )*th* rule (Take h = 0.5).

def f(x, c):

return c \*\* x

def simpsons\_3\_8\_rule(f, a, b, n, c):

h = (b - a) / n

x0 = f(a, c) + f(b, c)

even\_sum = sum(3 \* f(a + i \* h, c) for i in range(1, n, 2))

odd\_sum = sum(3 \* f(a + i \* h, c) for i in range(2, n - 1, 2))

integral = (x0 + even\_sum + odd\_sum) \* 3 \* h / 8

return integral

a = 0

b = 10

n = int((b - a) / 0.5)

c = 2

integral\_value = simpsons\_3\_8\_rule(f, a, b, n, c)

print(f"The estimated integral value is: {integral\_value:.4f}")

2. Write Python program find the approximate root of the function x5 +3*x*+1, in [-2,0] using Newton Raphson Method correct upto 4 decimal places.

ANS:

def f(x):

return x\*\*5 + 3\*x + 1

def df(x):

return 5\*x\*\*4 + 3

def newton\_raphson(f, df, x0, tol=1e-4, max\_iter=100):

for i in range(max\_iter):

x1 = x0 - f(x0) / df(x0)

if abs(x1 - x0) < tol:

return round(x1, 4

x0 = x1

return None

x0 = -1

root = newton\_raphson(f, df, x0)

if root is not None:

print(f"Approximate root: {root}")

else:

print("Newton-Raphson method did not converge within the specified tolerance.")

**b. Attempt any one of the following. [8]**

1. Write Python program to obtained the approximate real root of x3 −4*x*−9 = 0 by using Regula-falsi method.

ANS:

def f(x):

return x\*\*3 - 4\*x - 9

def regula\_falsi(f, a, b, tol=1e-4, max\_iter=100):

if f(a) \* f(b) >= 0:

print("Regula Falsi method cannot guarantee convergence with the given initial points.")

return None

for i in range(max\_iter):

c = (a \* f(b) - b \* f(a)) / (f(b) - f(a))

if abs(f(c)) < tol:

return c

if f(c) \* f(a) < 0:

b = c

else:

a = c

print("Regula Falsi method did not converge within the specified number of iterations.")

return None

a = 1

b = 3

root = regula\_falsi(f, a, b)

if root is not None:

print(f"Approximate root: {root:.4f}")

2. Write Python program to evaluate interpolate value f (153) of the given data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 150 | 152 | 154 | 155 |
| Y | 12.247 | 12.329 | 12.410 | 12.490 |

ANS:

def lagrange\_interpolation(x\_values, y\_values, x):

n = len(x\_values)

result = 0.0

for i in range(n):

term = y\_values[i]

for j in range(n):

if j != i:

term \*= (x - x\_values[j]) / (x\_values[i] - x\_values[j])

result += term

return result

x\_values = [150, 152, 154, 155]

y\_values = [12.247, 12.329, 12.410, 12.490]

x\_to\_interpolate = 153

interpolated\_value = lagrange\_interpolation(x\_values, y\_values, x\_to\_interpolate)

print(f"f({x\_to\_interpolate}) is approximately {interpolated\_value:.3f}")

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A** **Roll No**. :- 3 **Date** :- …./…/2023

Title :- Python Practical **Expt.No**. :- 12 **Class** :- SYBCS

**Q.1. Attempt any two of the following. [10]**

1. Using Python evaluate each of the following expression.

a. 23 modulus 2 + 9 - (3 +7) × 10 ÷ 2

b. 35 × 10 floor division 3 + 15 modulus 3

c. 35 - 25 + 4 floor division 7

ANS:

result\_a = 23 % 2 + 9 - (3 + 7) \* 10 // 2

print("Result a:", result\_a)

result\_b = 35 \* 10 // 3 + 15 % 3

print("Result b:", result\_b)

result\_c = 35 - 25 + 4 // 7

print("Result c:", result\_c)

2. Use while command on Python to find odd positive integer between 25 to 50.

ANS:

num = 25

while num <= 50:

if num % 2 == 1:

print(num)

num += 1

3. For matrix

A=

apply the following operations by using python.

a. Delete 2*nd* row.

b. Delete 1*st* column.

c. Add column [9, 9] as 2*nd* column.

ANS:

A = [[1, 0, 5, 4],[2, 1, 6, -1],[3, 4, 0, 2]]

A.pop(1)

for row in A:

row.pop(0)

for row in A:

row.insert(1, 9)

for row in A:

print(row)

**Q.2. Attempt any two of the following. [10]**

1. Write Python find the eigenvalues and corresponding eigenvectors of the matrix

A=

import numpy as np

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

eigenvalues, eigenvectors = np.linalg.eig(A)

print("Eigenvalues:")

for eigenvalue in eigenvalues:

print(eigenvalue)

print("\nEigenvectors:")

for i, eigenvector in enumerate(eigenvectors.T):

print(f"Eigenvector {i+1}:", eigenvector)

2. Write Python program to find the product of n natural numbers using while loop.

ANS:

n = int(input("Enter a positive integer (n): "))

product = 1

count = 1

while count <= n:

product \*= count

count += 1

print(f"The product of the first {n} natural numbers is: {product}")

3. Generate all prime numbers between 1 to 200 using Python code.

ANS:

‘def is\_prime(number):

if number < 2:

return False

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

if number % i == 0:

return False

return True

for num in range(2, 201):

if is\_prime(num):

print(num, end=" ")

**Q.3. a. Attempt any one of the following. [7]**

1. Write Python program to estimate the value of the integral *dx* using Simpson’s

(1/3 )*rd* rule (n=5).

ANS:

import math

def f(x):

return math.sin(x)

def simpsons\_1\_3\_rule(f, a, b, n):

h = (b - a) / n

x = a

integral = f(a) + f(b)

for i in range(1, n):

x += h

if i % 2 == 0:

integral += 2 \* f(x)

else:

integral += 4 \* f(x)

integral \*= h / 3

return integral

a = 0

b = math.pi

n = 5 # Number of subintervals

integral\_value = simpsons\_1\_3\_rule(f, a, b, n)

print(f"The estimated integral value is: {integral\_value:.4f}")

2. Write Python program to diagonalize the matrix

A=

and find matrix P and D.

ANS:

import numpy as np

A = np.array([[3, -2],

[6, -4]])

eigenvalues, eigenvectors = np.linalg.eig(A)

P = eigenvectorsD = np.diag(eigenvalues)

# Check if P^(-1) \* A \* P equals D (should be almost equal due to floating-point precision)

result = np.allclose(np.linalg.inv(P) @ A @ P, D)

print("Matrix A:")

print(A)

print("\nMatrix P:")

print(P)

print("\nMatrix D:")

print(D)

print("\nP^(-1) \* A \* P is almost equal to D:", result)

**b. Attempt any one of the following. [8]**

1. Write a Python program to obtained the approximate real root of *x*3 − 2*x* − 5 = 0

in [2,3] using Regula-falsi method.

ANS:

def f(x):

return x\*\*3 - 2\*x - 5

def regula\_falsi(a, b, tol, max\_iter):

if f(a) \* f(b) >= 0:

print("Regula Falsi method cannot be applied because f(a) and f(b) have the same sign.")

return None

for i in range(max\_iter):

c = (a \* f(b) - b \* f(a)) / (f(b) - f(a)

if abs(f(c)) < tol:

return c

if f(a) \* f(c) < 0:

b = c

else:

a = c

print("Regula Falsi method did not converge within the specified number of iterations.")

return None

a = 2

b = 3

tolerance = 1e-6

max\_iterations = 1000

root = regula\_falsi(a, b, tolerance, max\_iterations)

if root is not None:

print("Approximate real root:", root)

2. Write a Python program to estimate the value of the integral def f(x):

return x\*\*3 - 2\*x - 5

def regula\_falsi(a, b, tol, max\_iter):

if f(a) \* f(b) >= 0:

print("Regula Falsi method cannot be applied because f(a) and f(b) have the same sign.")

return None

for i in range(max\_iter):

c = (a \* f(b) - b \* f(a)) / (f(b) - f(a)

if abs(f(c)) < tol:

return c

if f(a) \* f(c) < 0:

b = c

else:

a = c

print("Regula Falsi method did not converge within the specified number of iterations.")

return None

a = 2

b = 3

tolerance = 1e-6

max\_iterations = 1000

root = regula\_falsi(a, b, tolerance, max\_iterations)

if root is not None:

print("Approximate real root:", root)

2. Write a Python program to estimate the value of the integral *dx* using Trapezoidal

rule (n=10).

ANS:

def f(x):

return 1 / (1 + x)

def trapezoidal\_rule(a, b, n):

h = (b - a) / n

integral = 0.5 \* (f(a) + f(b)) # Initialize the integral with the endpoints

for i in range(1, n):

x\_i = a + i \* h

integral += f(x\_i)

integral \*= h

return integral

a = 1

b = 5

n = 10

result = trapezoidal\_rule(a, b, n)

print("Estimated value of the integral:", result)

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A** Roll No. :- 3 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 13 Class :- SYBCS

**Q.1. Attempt any two of the following. [10]**

1. Using Python code, evaluate the following expression of two complex number *z*1 =

3 + 2*j* and *z*2 = −4 + 1*j*

a. *z*1 + *z*2

b. *z*1 − *z*2

c. *z*1 ∗ *z*2

ANS:

z1 = 3 + 2j

z2 = -4 + 1j

result\_addition = z1 + z2

print("a. z1 + z2 =", result\_addition)

result\_subtraction = z1 - z2

print("b. z1 - z2 =", result\_subtraction)

result\_multiplication = z1 \* z2

print("c. z1 \* z2 =", result\_multiplication)

2. Use Python code to find area and circumference of square whose length is 5.

ANS:

side\_length = 5

area = side\_length \*\* 2

circumference = 4 \* side\_length

print("Area of the square:", area)

print("Circumference of the square:", circumference)

output:

Area of the square: 25

Circumference of the square: 20

3. Write Python program to generate the square number from 1 to 10.

ANS:

square\_numbers = []

for i in range(1, 11):

square = i \*\* 2

square\_numbers.append(square)

print("Square numbers from 1 to 10:", square\_numbers)

**Q.2. Attempt any two of the following. [10]**

1. Write Python code to reverse the string S=[1,2,3,4,5,6,7,8,9].

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

reversed\_S = S[::-1]

print("Reversed list:", reversed\_S)

2. Write Python program to find f(*x*) = x2 + 3*x*, Where (−1 ≤ *x* ≤ 3).

x\_values = range(-1, 4)

for x in x\_values:

fx = x\*\*2 + 3\*x

print(f"f({x}) = {fx}")

3. Write Python code to find average of number 50 to 100.

start = 50

end = 100

total = 0

count = 0

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

total += number

count += 1

average = total / count

print("Average of numbers from 50 to 100:", average)

**Q.3. a. Attempt any one of the following. [7]**

1. Write Python program to estimate the value of the integral √1 + x3dx using

Simpson’s (1/3 )*rd* rule (n=10).

ANS:

import math

def f(x):

return math.sqrt(1 + x\*\*3)

a = 0

b = 1

n = 10

h = (b - a) / n

integral = f(a) + f(b)

for i in range(1, n):

x = a + i \* h

if i % 2 == 0:

integral += 2 \* f(x)

else:

integral += 4 \* f(x)

integral \*= h / 3

print("Estimated integral:", integral)

2. Write Python program to evaluate interpolate value f(5*.*5) of the given data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 3 | 5 | 7 | 9 |
| y | 5 | 7 | 27 | 64 |

ANS:

x\_values = [3, 5, 7, 9]

y\_values = [5, 7, 27, 64]

x\_interpolate = 5.5

coefficients = []

n = len(x\_values)

for i in range(n):

coefficients.append(y\_values[i])

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

coefficients[j] = (coefficients[j] - coefficients[j - 1]) / (x\_values[i] - x\_values[i - j])

result = coefficients[0]

x\_term = 1

for i in range(1, n):

x\_term \*= (x\_interpolate - x\_values[i - 1])

result += coefficients[i] \* x\_term

print("Interpolated value at x =", x\_interpolate, "is:", result)

**b. Attempt any one of the following. [8]**

1. Write a Python program to obtained the approximate real root of x3 − 4*x* − 9 = 0 by using Regula-falsi method.

ANS;

def f(x):

return x\*\*3 - 4\*x - 9

a = 1.0

b = 3.0

tolerance = 1e-6

max\_iterations = 100

for i in range(max\_iterations):

fa = f(a)

fb = f(b)

c = (a \* fb - b \* fa) / (fb - fa)

fc = f(c)

if abs(fc)

2. Write a Python program to evaluate f(2.7) by backward difference formula of the

given data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1 | 2 | 3 | 4 | 5 |
| Y | 40 | 60 | 65 | 50 | 18 |

ANS:

x\_values = [1, 2, 3, 4, 5]

y\_values = [40, 60, 65, 50, 18]

x\_interpolate = 2.7

h = x\_values[1] - x\_values[0]

coefficients = [y\_values]

n = len(x\_values)

for i in range(1, n):

coefficients.append([])

for j in range(n - i):

coefficients[i].append(coefficients[i - 1][j + 1] - coefficients[i - 1][j])

result = y\_values[0]

x\_term = (x\_interpolate - x\_values[0]) / h

for i in range(1, n):

x\_term \*= (x\_term - i + 1) / i

result += coefficients[i][0] \* x\_term

print("Interpolated value at x =", x\_interpolate, "is:", result)

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A** Roll No. :- 3 Date :- …./…/2023

**Title :**- Python Practical **Expt.No. :-** 14 **Class** :- SYBCS

**Q.1. Attempt any two of the following. [10]**

1. Use print code on Python (a=4,b=6,c=8,d=12).

(a) print(a+c)

(b) print(a\*b)

(c) print(c\*\*d)

(d) print(a/b)

(e) Expression: 3 + ( 9 - 2) / 7 \* 2 \*\* 2

ANS:

a = 4

b = 6

c = 8

d = 12

print(a + c)

print(a \* b)

print(c \*\* d)

print(a / b)

result = 3 + (9 - 2) / 7 \* 2 \*\* 2

print(result)

2. For the following two statements use ‘+’string operation on Python.

a. string1 = Hello, string2 = World!

b. string1 = Good, string2 = Morning

ANS:

a)string1 = "Hello, "

string2 = "World!"

result = string1 + string2

print(result)

b) string1 = "Good, "

string2 = "Morning"

result = string1 + string2

print(result)

3. Use Python loop to print(‘Hallo’,i,‘You Learn Python’)

where i = [‘Saurabh’,‘Akash’,‘Sandeep’,‘Ram’,‘Sai’]

ANS:

i = ['Saurabh', 'Akash', 'Sandeep', 'Ram', 'Sai']

for name in i:

print('Hallo', name, 'You Learn Python')

**Q.2. Attempt any two of the following. [10]**

1. Using Python code construct any two matrices A and B

1. Show that A+B=B+A.

2. Find A-B.

ANS:

import numpy as np

A = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

B = np.array([[9, 8, 7],

[6, 5, 4],

[3, 2, 1]])

result1 = A + B

result2 = B + A

if np.array\_equal(result1, result2):

print("A + B is equal to B + A:")

print(result1)

result3 = A - B

print("A - B:")

print(result3)

2. Write Python program to find the sequence of function f(*x*) = *x*+5, (−5 ≤ *x* ≤5)

ANS:

x\_values = range(-5, 6)

def f(x):

return x + 5

for x in x\_values:

result = f(x)

print(f"f({x}) = {result}")

3. Using sympy module of python find the eigenvalues and corresponding eigenvectors of the matrix

A=

ANS:

import sympy as sp

A = sp.Matrix([[4, 2, 2],

[2, 4, 2],

[2, 2, 4]])

eigenvalues = A.eigenvals()

eigenvectors = A.eigenvects()

print("Eigenvalues:")

for eigenvalue, multiplicity in eigenvalues.items():

print(f"Eigenvalue: {eigenvalue}, Multiplicity: {multiplicity}")

print("\nEigenvectors:")

for eigenvector in eigenvectors:

eigenvalue = eigenvector[0]

multiplicity = eigenvector[2]

vectors = eigenvector[2]

for vector in vectors:

print(f"Eigenvalue: {eigenvalue}, Multiplicity: {multiplicity}")

print(f"Eigenvector: {vector}")

**Q.3. a. Attempt any one of the following. [7]**

1. Write a Python program to estimate the value of the integral 2)*dx* using

Simpson’s (1/3 )*rd* rule (n=4).

ANS:

def f(x):

return 1 / (1 + x)\*\*2

n = 4

h = 1 / n

integral\_sum = 0

for i in range(n + 1):

x = i \* h

if i == 0 or i == n:

integral\_sum += f(x)

elif i % 2 == 1:

integral\_sum += 4 \* f(x)

else:

integral\_sum += 2 \* f(x)

integral\_estimate = (h / 3) \* integral\_sum

print("Estimated Integral:", integral\_estimate)

2. Write a Python program to obtained a real root of *f*(*x*) = *x*3 − 8*x* − 4 = 0 using

Newton–Raphson method.

ANS:

def f(x):

return x\*\*3 - 8\*x - 4

def f\_prime(x):

return 3\*x\*\*2 - 8

x0 = 2.0

tolerance = 1e-6

max\_iterations = 100

iteration = 0

while iteration < max\_iterations:

x1 = x0 - f(x0) / f\_prime(x0)

if abs(x1 - x0) < tolerance:

print(f"Approximate Root: {x1}")

break

x0 = x1

iteration += 1

if iteration == max\_iterations:

print("Newton-Raphson did not converge within the maximum number of iterations.")

**b. Attempt any one of the following. [8]**

1. Write Python program to obtained the approximate real root of *x*3 − 2*x* − 5 = 0 in

[2,3] using Regula-falsi method.

ANS:;

def f(x):

return x\*\*3 - 2\*x - 5

a = 2

b = 3

tolerance = 1e-6

max\_iterations = 100

iteration = 0

x0 = a

x1 = b

while iteration < max\_iterations:

x2 = x1 - f(x1) \* (x1 - x0) / (f(x1) - f(x0))

if abs(x2 - x1) < tolerance:

print(f"Approximate Root: {x2}")

break

if f(x0) \* f(x2) < 0:

x1 = x2

else:

x0 = x2

iteration += 1

if iteration == max\_iterations:

print("Regula-Falsi did not converge within the maximum number of iterations.")

2. Write Python program to evaluate approximate value of f(1.5) by using forward

difference formula of the given data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1 | 2 | 3 | 4 | 5 |
| y | 30 | 50 | 65 | 40 | 18 |

ANS:

def forward\_difference(x\_values, y\_values, x\_target):

n = len(x\_values)

h = x\_values[1] - x\_values[0]

result = y\_values[0]

u = (x\_target - x\_values[0]) / h

for i in range(1, n):

term = 1

for j in range(i):

term \*= (u - j)

result += (term / math.factorial(i)) \* differences[i][0]

return result

x\_values = [1, 2, 3, 4, 5]

y\_values = [30, 50, 65, 40, 18]

x\_target = 1.5

import math

n = len(x\_values)

differences = [[0 for \_ in range(n)] for \_ in range(n)]

for i in range(n):

differences[i][0] = y\_values[i]

for j in range(1, n):

for i in range(n - j):

differences[i][j] = differences[i + 1][j - 1] - differences[i][j - 1]

result = forward\_difference(x\_values, y\_values, x\_target)

print(f"f({x\_target}) is approximately {result}")

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A** Roll No. :- 3 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 15 Class :- SYBCS

**Q.1. Attempt any two of the following. [10]**

1. Using for loop on Python, find range from 1 to 11 integers.

ANS:

for i in range(1, 12):

print(i)

2. Use Python code to find,

(a) sin75

(b) pi/2

(c) e

(d) cos56

ANS:

import math

sin\_75 = math.sin(math.radians(75))

print(f"(a) sin(75 degrees) = {sin\_75}")

pi\_over\_2 = math.pi / 2

print(f"(b) π/2 = {pi\_over\_2}")

euler\_number = math.e

print(f"(c) e (Euler's number) = {euler\_number}")

cos\_56 = math.cos(math.radians(56))

print(f"(d) cos(56 degrees) = {cos\_56}")

3. Write Python program to find diameter, area, circumference of the circle with radius

is 5.

ANS:

import math

radius = 5

diameter = 2 \* radius

area = math.pi \* (radius \*\* 2)

circumference = 2 \* math.pi \* radius

print(f"Radius: {radius}")

print(f"Diameter: {diameter}")

print(f"Area: {area}")

print(f"Circumference: {circumference}")

**Q.2. Attempt any two of the following. [10]**

1. Using Python code construct any three matrices A,B and C to show that

(A+B)+C=A+(B+C).

ANS:

import numpy as np

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

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

C = np.array([[9, 10], [11, 12]])

result1 = (A + B) + C

result2 = A + (B + C)

if np.array\_equal(result1, result2):

print("(A + B) + C is equal to A + (B + C):")

print(result1)

else:

print("(A + B) + C is not equal to A + (B + C)")

2. Using python find the eigenvalues and corresponding eigenvectors of the matrix

ANS:

import numpy as np

matrix = np.array([[3, -2],

[6, -4]])

eigenvalues, eigenvectors = np.linalg.eig(matrix)

print("Eigenvalues:")

for eigenvalue in eigenvalues:

print(eigenvalue)

print("\nEigenvectors:")

for i, eigenvector in enumerate(eigenvectors.T):

print(f"Eigenvector {i + 1}:")

print(eigenvector)

3. Generate all prime numbers between 1000 to 2000 using Python program.

ANS:

def is\_prime(n):

if n <= 1:

return False

if n <= 3:

return True

if n % 2 == 0 or n % 3 == 0:

return False

i = 5

while i \* i <= n:

if n % i == 0 or n % (i + 2) == 0:

return False

i += 6

return True

start = 1000

end = 2000

prime\_numbers = [x for x in range(start, end + 1) if is\_prime(x)]

print("Prime numbers between 1000 and 2000:")

print(prime\_numbers)

**Q.3. a. Attempt any one of the following. [7]**

1. Write Python program to estimate the value of the integral x*dx* using Simpson’s

(1/3 )*rd* rule (n=6).

ANS:

import math

def f(x):

return math.exp(x)

n = 6

h = 6 / n

integral\_sum = f(0) + f(6) # Sum of the first and last points

for i in range(1, n):

x = i \* h

if i % 2 == 0:

integral\_sum += 2 \* f(x)

else:

integral\_sum += 4 \* f(x)

integral\_estimate = (h / 3) \* integral\_sum

print("Estimated Integral:", integral\_estimate)

2. Write Python program to estimate a root of an equation *f*(*x*) = 3*x* − *cos*(*x*) − 1

using Newton–Raphson method correct up to four decimal places.

ANS:

import math

def f(x):

return 3 \* x - math.cos(x) - 1

def f\_prime(x):

return 3 + math.sin(x)

x0 = 1.0

tolerance = 1e-4

max\_iterations = 100

iteration = 0

while iteration < max\_iterations:

x1 = x0 - f(x0) / f\_prime(x0)

if abs(x1 - x0) < tolerance:

root = round(x1, 4) # Round to four decimal places

print(f"Approximate Root: {root}")

break

x0 = x1

iteration += 1

if iteration == max\_iterations:

print("Newton-Raphson did not converge within the maximum number of iterations.")

**b. Attempt any one of the following. [8]**

1. Write Python program to obtained the approximate real root of *x*3 −4*x*−9 = 0 by

using Regula-falsi method.

ANS:

def f(x):

return x\*\*3 - 4\*x - 9

x0 = 2.0

x1 = 3.0

tolerance = 1e-6

max\_iterations = 100

iteration = 0

while iteration < max\_iterations:

x2 = (x0 \* f(x1) - x1 \* f(x0)) / (f(x1) - f(x0))

if abs(x2 - x1) < tolerance:

print(f"Approximate Root: {x2}")

break

if f(x0) \* f(x2) < 0:

x1 = x2

else:

x0 = x2

iteration += 1

if iteration == max\_iterations:

print("Regula-Falsi did not converge within the maximum number of iterations.")

2. Write Python program to evaluate interpolate value f(2.2) of the given data

ANS:

f(2)=0.593,f(2.5)=0.816,f(3)=1.078.

x\_values = [2, 2.5, 3]

y\_values = [0.593, 0.816, 1.078]

x\_target = 2.2

def linear\_interpolation(x, x0, x1, y0, y1):

return y0 + (x - x0) \* (y1 - y0) / (x1 - x0)

for i in range(len(x\_values) - 1):

if x\_values[i] <= x\_target <= x\_values[i + 1]:

x0, x1 = x\_values[i], x\_values[i + 1]

y0, y1 = y\_values[i], y\_values[i + 1]

break

interpolated\_value = linear\_interpolation(x\_target, x0, x1, y0, y1)

print(f"Interpolated value f({x\_target}) = {interpolated\_value}")

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A**  Roll No. :- 3 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 16 Class :- SYBCS

**Q.1. Attempt any two of the following.**

1. Write Python program to find absolute value of a given real number(n). The estimated integral value is: 2.079365

n = float(input("Enter a real number: "))

absolute\_value = abs(n)

print(f"The absolute value of {n} is {absolute\_value}")

>> Enter a real number: -5.67

The absolute value of -5.67 is 5.67

2. Using Python program

List1 = [5, 10, 15, 20, 25, 30] and List2 = [7, 14, 21, 28, 35, 42]

Evaluate

(a) List1 + List2

(b) 7\*List1

(c) 11\*List2

ANS:

List1 = [5, 10, 15, 20, 25, 30]

List2 = [7, 14, 21, 28, 35, 42]

result\_a = List1 + List2

result\_b = [7 \* x for x in List1]

result\_c = [11 \* x for x in List2]

print("(a) List1 + List2:", result\_a)

print("(b) 7\*List1:", result\_b)

print("(c) 11\*List2:", result\_c)

output:

(a) List1 + List2: [5, 10, 15, 20, 25, 30, 7, 14, 21, 28, 35, 42]

(b) 7\*List1: [35, 70, 105, 140, 175, 210]

(c) 11\*List2: [77, 154, 231, 308, 385, 462]

3. Write Python program to find the area and circumference of a circle(r=5).

ANS:

import math

radius = 5

area = math.pi \* (radius \*\* 2)

circumference = 2 \* math.pi \* radius

print(f"Radius of the circle: {radius}")

print(f"Area of the circle: {area:.2f}")

print(f"Circumference of the circle: {circumference:.2f}")

output:

Radius of the circle: 5

Area of the circle: 78.54

Circumference of the circle: 31.42

**Q.2. Attempt any two of the following.**

1. Using Python code, find percentage of marks 70,80, 55, 78, 65 in five subject out of 100 each.

ANS:

marks = [70, 80, 55, 78, 65]

max\_marks = 100

total\_marks = sum(marks)

percentage = (total\_marks / (len(marks) \* max\_marks)) \* 100

print(f"The percentage of marks is: {percentage:.2f}%")

output:

The percentage of marks is: 71.60%

2. Using sympy module of python, find the following terms of vector x = [1, -5, 0] and

y = [2, 3, -1].

a. 5x

b. x+y

c. x-3y

ANS:

import sympy

x = sympy.Matrix([1, -5, 0])

y = sympy.Matrix([2, 3, -1])

result\_a = 5 \* x

result\_b = x + y

result\_c = x - 3 \* y

print("(a) 5x:", result\_a)

print("(b) x + y:", result\_b)

print("(c) x - 3y:", result\_c)

output:

(a) 5x: Matrix([[5], [-25], [0]])

(b) x + y: Matrix([[3], [-2], [-1]])

(c) x - 3y: Matrix([[-5], [-14], [3]])

3. Write python code to find the determinant and inverse of matrices

A= and B=

import sympy

A = sympy.Matrix([[1, 0, 5],[2, 1, 6],[3, 4, 0]])

B = sympy.Matrix([[2, 5],[-1, 4]])

determinant\_A = A.det()

determinant\_B = B.det()

inverse\_A = A.inv()

inverse\_B = B.inv()

print("Matrix A:")

print(A)

print(f"Determinant of A: {determinant\_A}")

print(f"Inverse of A:")

print(inverse\_A)

print("\nMatrix B:")

print(B)

print(f"Determinant of B: {determinant\_B}")

print(f"Inverse of B:")

print(inverse\_B)

output:

Matrix A:

Matrix([[1, 0, 5],[2, 1, 6],[3, 4, 0]])

Determinant of A: -23

Inverse of A:

Matrix([[8/23, -20/23, 5/23],[4/23, 3/23, -1/23],[-1/23, 8/23, -2/23]])

Matrix B: Matrix([[2, 5],[-1, 4]])

Determinant of B: 13

Inverse of B:

Matrix([[4/13, -5/13],

[1/13, 2/13]])

**Q.3. a. Attempt any one of the following.**

**1. Write Python program to estimate the value of the integral using Simpson’s rule (n=6).**

**ANS:import math**

def f(x):

return math.sin(x)

n = 6

a = 0

b = math.pi

h = (b - a) / n

odd\_sum = 0

even\_sum = 0

for i in range(1, n, 2):

x = a + i \* h

odd\_sum += f(x)

for i in range(2, n - 1, 2):

x = a + i \* h

even\_sum += f(x)

result = (h / 3) \* (f(a) + 4 \* odd\_sum + 2 \* even\_sum + f(b))

print(f"The estimated integral value is: {result:.6f}")

output:

The estimated integral value is: 2.000000

**2. Write Python program to estimate a root of an equation *f* (*x*) = + 5*x* + 1 using Newton–Raphson method in the interval [-1,0].**

ANS:

def f(x):

return x\*\*5 + 5\*x + 1

def f\_prime(x):

return 5\*x\*\*4 + 5

def newton\_raphson(initial\_guess, tolerance, max\_iterations):

x = initial\_guess

iteration = 0

while iteration < max\_iterations:

f\_x = f(x)

f\_prime\_x = f\_prime(x)

if abs(f\_prime\_x) < tolerance:

print("Derivative is close to zero. Newton-Raphson method failed.")

return None

x\_new = x - f\_x / f\_prime\_x

if abs(x\_new - x) < tolerance:

return x\_new

x = x\_new

iteration += 1

print("Newton-Raphson method did not converge within the specified iterations.")

return None

initial\_guess = -1.0

tolerance = 1e-6

max\_iterations = 100

root = newton\_raphson(initial\_guess, tolerance, max\_iterations)

if root is not None:

print(f"Estimated root: {root:.6f}")

output:

Estimated root: -0.735048

**b. Attempt any one of the following. [8]**

**1. Write Python program to obtained the approximate real root of −2*x*−1 = 0 by using Regula-falsi method in the interval [2,3].**

**ANS:**

def f(x):

return x\*\*2 - 2\*x - 1

def regula\_falsi(a, b, tolerance, max\_iterations):

if f(a) \* f(b) >= 0:

print("Regula Falsi method cannot be applied to the given interval.")

return None

iteration = 0

while iteration < max\_iterations:

c = (a \* f(b) - b \* f(a)) / (f(b) - f(a)

if abs(f(c)) < tolerance:

return c

if f(c) \* f(a) < 0:

b = c

else:

a = c

iteration += 1

print("Regula Falsi method did not converge within the specified iterations.")

return None

a = 2

b = 3

tolerance = 1e-6

max\_iterations = 100

root = regula\_falsi(a, b, tolerance, max\_iterations)

if root is not None:

print(f"Estimated root: {root:.6f}")

output:

Estimated root: 2.791287

**2. Write Python program to estimate the value of the integral *dx* using Trapezoidal rule (n=8)**.

ANS:

def f(x):

return 1 / (1 + x)

def trapezoidal\_rule(a, b, n):

h = (b - a) / n

integral = (f(a) + f(b)) / 2

for i in range(1, n):

x\_i = a + i \* h

integral += f(x\_i)

integral \*= h

return integral

a = 2

b = 10

n = 8

result = trapezoidal\_rule(a, b, n)

print(f"The estimated integral value is: {result:.6f}")

output:

The estimated integral value is: 2.079365

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A**  Roll No. :- 3 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 17 Class :- SYBCS

**Q.1. Attempt any two of the following. [10]**

1. Write the Python code to print ‘Python is bad’ and ‘Python is wonderful’ , where wonderful is global variable and bad is local variable.

ANS:

def print\_messages():

bad = "bad"

print(f'Python is {bad}')

print(f'Python is {wonderful}')

print\_messages()

2. Write Python code to evaluate eigen value and eigen vector of the following matrix.

A=

import numpy as np

A = np.array([[1, 1, 1],

[0, 1, 1],

[0, 0, 1]])

eigenvalues, eigenvectors = np.linalg.eig(A)

print("Eigenvalues:")

for eigenvalue in eigenvalues:

print(eigenvalue)

for eigenvector in eigenvectors.T:

print(eigenvector)

3. Write Python code, find a, b and c such that *a2+b2=c2*.(where 1 ≤ *a, b, c* ≤ 50)

ANS:

max\_value = 50

for a in range(1, max\_value + 1):

for b in range(1, max\_value + 1):

c = (a \*\* 2 + b \*\* 2) \*\* 0.5 # Calculate c using the equation

if c.is\_integer() and c <= max\_value

print(f"a = {a}, b = {b}, c = {int(c)}")

**Q.2. Attempt any two of the following. [10]**

1. Using Python code construct any two matrices *A* and *B* to show that

(AB)-1 =B-1 A-1

ANS:

import numpy as np

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

B = np.array([[5, 1],[6, 2]])

AB = np.dot(A, B)

AB\_inverse = np.linalg.inv(AB)

B\_inverse = np.linalg.inv(B)

A\_inverse = np.linalg.inv(A)

if np.allclose(AB\_inverse, np.dot(B\_inverse, A\_inverse)):

print("(AB)^(-1) is equal to B^(-1)A^(-1)")

else:

print("(AB)^(-1) is not equal to B^(-1)A^(-1)")

2. Use linsolve code in python to solve the following system of linear equations.

*x* − 2*y* + 3*z* = 7

2*x* + *y* + *z* = 4

−3*x* + 2*y* − 2*z* = −10

ANS:

from sympy import symbols, Eq, linsolve

x, y, z = symbols('x y z')

eq1 = Eq(x - 2\*y + 3\*z, 7)

eq2 = Eq(2\*x + y + z, 4)

eq3 = Eq(-3\*x + 2\*y - 2\*z, -10)

solution = linsolve([eq1, eq2, eq3], x, y, z)

x\_solution, y\_solution, z\_solution = solution.args[0]

print("Solution:")

print(f"x = {x\_solution}")

print(f'y = {y\_solution}')

print(f'z = {z\_solution}')

3. Write python code to find trace and transpose of the matrix

A=

ANS:

import numpy as np

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

trace\_A = np.trace(A)

transpose\_A = np.transpose(A)

print("Matrix A:")

print(A)

print("Trace of A:", trace\_A)

print("Transpose of A:")

print(transpose\_A)

**Q.3. a. Attempt any one of the following. [7]**

1. Write Python program to find f(3) of the functional value f(1)=2, f(2)=10, f(4)=68 by using Lagrange method.

ANS:

def lagrange\_interpolation(x\_values, y\_values, x):

if len(x\_values) != len(y\_values):

raise ValueError("x\_values and y\_values must have the same number of elements")

n = len(x\_values)

result = 0

for i in range(n):

term = y\_values[i]

for j in range(n):

if j != i:

term \*= (x - x\_values[j]) / (x\_values[i] - x\_values[j])

result += term

return result

x\_values = [1, 2, 4]

y\_values = [2, 10, 68]

x = 3

result = lagrange\_interpolation(x\_values, y\_values, x)

print(f"f({x}) = {result}")

2. Write Python program to estimate a root of an equation x5 *+5x* + 6 = 0 using

Newton–Raphson method correct up to four decimal places.

ANS:

def f(x):

return x\*\*5 + 5\*x + 6

def df(x):

return 5\*x\*\*4 + 5

def newton\_raphson(initial\_guess, tolerance=1e-4, max\_iterations=100):

x = initial\_guess

iteration = 0

while iteration < max\_iterations:

iteration += 1

x\_new = x - f(x) / df(x)

if abs(x\_new - x) < tolerance:

return round(x\_new, 4) # Corrected to four decimal places

x = x\_new

raise Exception("Newton-Raphson method did not converge")

initial\_guess = 1.0

root = newton\_raphson(initial\_guess)

print("Estimated root:", root)

**b. Attempt any one of the following. [8]**

1. Write Python program to obtained the approximate real root of x2−2*x*−1 = 0 by

using Regula-falsi method in the interval [1,3].

ANS:

def f(x):

return x\*\*2 - 2\*x - 1

def regula\_falsi(a, b, tolerance=1e-4, max\_iterations=100):

if f(a) \* f(b) >= 0:

raise ValueError("The Regula-Falsi method may not converge with the given initial interval.")

iteration = 0

while iteration < max\_iterations:

iteration += 1

c = (a \* f(b) - b \* f(a)) / (f(b) - f(a)

if abs(f(c)) < tolerance:

return round(c, 4) # Corrected to four decimal places

if f(a) \* f(c) < 0:

b = c

else:

a = c

raise Exception("Regula-Falsi method did not converge")

a = 1

b = 3

root = regula\_falsi(a, b)

print("Approximate root:", root)

2. Write Python program to estimate the value of the integral 2*dx* using Trapezoidal rule (n=10).

ANS:

def trapezoidal\_rule(func, a, b, n):

h = (b - a) / n

result = (func(a) + func(b)) / 2

for i in range(1, n):

result += func(a + i \* h)

result \*= h

return result

def func(x):

return x\*\*2

a = 0

b = 1

n = 10

integral\_value = trapezoidal\_rule(func, a, b, n)

print("Estimated integral value:", integral\_value)

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A** Roll No. :- 3 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 18 Class :- SYBCS

**Q.1. Attempt any two of the following. [10]**

1. Use Python code to find minimum value from the given numbers

16,3,5,48,2,4,5,6,78,12,5,6,24.

ANS:

numbers = [16, 3, 5, 48, 2, 4, 5, 6, 78, 12, 5, 6, 24]

min\_value = min(numbers)

print("Minimum value:", min\_value)

2. Use Python code to find hypotenuse of triangle whose sides are 12 and 5.

import math

ANS:

side1 = 12

side2 = 5

hypotenuse = math.sqrt(side1\*\*2 + side2\*\*2)

print("Hypotenuse of the triangle:", hypotenuse)

3. Use Python code to remove all digits after decimal of the given Number 125312.3142.

ANS:

number = 125312.3142

number\_without\_decimal = int(number)

print("Number without decimal:", number\_without\_decimal)

**Q.2. Attempt any two of the following. [10]**

1. Using Python code, find the below matrices, where

A= and B=

(a) A+B

(b) AT

(c) A-1

ANS:

import numpy as np

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

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

result\_addition = A + B

result\_transpose = np.transpose(A)

result\_inverse = np.linalg.inv(A)

print("(a) A + B:")

print(result\_addition)

print("\n(b) A^T (Transpose of A):")

print(result\_transpose)

print("\n(c) A^(-1) (Inverse of A):")

print(result\_inverse)

2. Use while code on Python to find sum of first twenty natural number.

ANS:

n = 1 # Start with the first natural number

sum\_of\_numbers = 0

count =

while count < 20:

sum\_of\_numbers += n

n += 1

count += 1

print("Sum of the first twenty natural numbers:", sum\_of\_numbers)

3. Write Python program to diagonalize the matrix

and find matrix P and D.

ANS:

import numpy as np

A = np.array([[3, -2],[6, -4]])

eigenvalues, eigenvectors = np.linalg.eig(A)

D = np.diag(eigenvalues)

P = eigenvectors

print("Matrix A:")

print(A)

print("\nMatrix D (diagonal matrix of eigenvalues):")

print(D)

print("\nMatrix P (matrix of eigenvectors as columns):")

print(P)

**Q.3. a. Attempt any one of the following. [7]**

1. Write Python program to estimate the value of the integral *dx* using Simpson’s (1/3 )rdrule (n=8).

ANS:

def f(x):

return 1 / x

def simpsons\_one\_third\_rule(func, a, b, n):

h = (b - a) / n

x\_values = [a + i \* h for i in range(n + 1)]

result = func(a) + func(b)

for i in range(1, n, 2):

result += 4 \* func(x\_values[i])

for i in range(2, n - 1, 2):

result += 2 \* func(x\_values[i])

result \*= h / 3

return result

a = 1

b = 3

n = 8

integral\_value = simpsons\_one\_third\_rule(f, a, b, n)

print("Estimated integral value:", integral\_value)

2. Write Python program to evaluate interpolate value *f*(2*.*9) of the given data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 1 | 2 | 3 | 4 |
| Y=f(x) | 11 | 9 | 27 | 64 |

ANS:

def lagrange\_interpolation(x\_values, y\_values, x):

n = len(x\_values)

result = 0.0

for i in range(n):

term = y\_values[i]

for j in range(n):

if i != j:

term \*= (x - x\_values[j]) / (x\_values[i] - x\_values[j])

result += term

return result

x\_values = [1, 2, 3, 4]

y\_values = [11, 9, 27, 64]

x\_to\_interpolate = 2.9

interpolated\_value = lagrange\_interpolation(x\_values, y\_values, x\_to\_interpolate)

print(f"f({x\_to\_interpolate}) ≈ {interpolated\_value}")

**b. Attempt any one of the following. [8]**

1. Write Python program to obtained the approximate real root of x3− 5*x* − 9 = 0 in

[2,3] using Regula-falsi method.

ANS:

def func(x):

return x\*\*3 - 5\*x - 9

def regula\_falsi(a, b, tolerance, max\_iterations):

if func(a) \* func(b) >= 0:

print("Regula Falsi method may not converge as the initial interval [a, b] does not satisfy the conditions.")

return None

for i in range(max\_iterations):

c = (a \* func(b) - b \* func(a)) / (func(b) - func(a))

if abs(func(c)) < tolerance:

return c

if func(c) \* func(a) < 0:

b = c

else:

a = c

print("Regula Falsi method did not converge within the specified number of iterations.")

return None

a = 2

b = 3

tolerance = 1e-6

max\_iterations = 100

root = regula\_falsi(a, b, tolerance, max\_iterations)

if root is not None:

print(f"Approximate root: {root}")

2. Write Python program to estimate the value of the integral *dx* using Trapezoidal rule (n=5).

ANS:

import math

def f(x):

return math.cos(x)

def trapezoidal\_rule(a, b, n):

h = (b - a) / n

integral = (f(a) + f(b)) / 2.0

for i in range(1, n):

x = a + i \* h

integral += f(x)

integral \*= h

return integral

a = 0

b = 1

n = 5

result = trapezoidal\_rule(a, b, n)

print(f"Estimated integral value: {result:.6f}")

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A** Roll No. :- 3 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 19 Class :- SYBCS

**Q.1. Attempt any two of the following. [10]**

1. Write python code to display multiplication tables of numbers 2 to 10.

ANS:

start = 2

end = 10

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

print(f"Multiplication table for {num}:")

for i in range(1, 11):

product = num \* i

print(f"{num} x {i} = {product}")

print()

2. Write Python code to check if a number is Zero, Odd or Even.

ANS:

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

if number == 0:

print("The number is zero.")

elif number % 2 == 0:

print("The number is even.")

else:

print("The number is odd.")

3. Write Python code to list name and birth date of 5 students in your class.

ANS:

students = [

{"name": "Student 1", "birthdate": "January 5, 2000"},

{"name": "Student 2", "birthdate": "March 12, 2001"},

{"name": "Student 3", "birthdate": "May 20, 1999"},

{"name": "Student 4", "birthdate": "July 8, 2002"},

{"name": "Student 5", "birthdate": "November 15, 1998"}

]

for student in students:

print(f"Name: {student['name']}, Birthdate: {student['birthdate']}")

**Q.2. Attempt any two of the following. [10]**

1. Write python code to find transpose and inverse of matrix

A=

import numpy as np

A = np.array([[1, 2, 2],[2, 1, 2],[2, 2, 1]])

A\_transpose = np.transpose(A)

A\_inverse = np.linalg.inv(A)

print("Matrix A:")

print(A)

print("\nTranspose of A:")

print(A\_transpose)

print("\nInverse of A:")

print(A\_inverse)

2. Declare the matrix

A=

find a row echelon form and the rank of matrix *A*.

ANS:

R1/5

1 2/5 1 4/5

10 3 4 6

2 0 -1 11

R2-10R1

1 2/5 1 4/5

0 -17/5 -6 10/5

2 0 -1 11

R3-2R1

1 2/5 1 4/5

0 -17/5 -6 10/5

0 -4/5 -3 9/5

R2/(-17/5)

1 2/5 1 4/5

0 1 30/17 -10/17

0 -4/5 -3 9/5

(4/5)R2+R3

1 2/5 1 4/5

0 1 30/17 -10/17

0 0 9/17 -26/17

Rank of matrix:3

3. Declare the matrix

A=

find the matrices *L* and *U* such that *A* = *LU*.

ANS:

R2-(1/2)R1

1 -1/2 1 7/2

0 8.5 1 -3.5

4 2 0 -1

R3-(1/2)R1

1 -1/2 1 7/2

0 8.5 1 -3.5

0 3 -2 -9

R3-R2(1/8.5)

1 -1/2 1 7/2

0 1 1/8.5 -7/17

0 0 -17/17 -44/17

L MATRIX: 1 0 0 0

2 1 0 0

2 -3.5 1 0

2 -1/2 1 7/2

0 1 1/8.5 -7/17

0 0 -17/17 -44/17

**Q.3. a. Attempt any one of the following. [7]**

1. Write Python program to estimate the value of the integral 2)*dx* by using Simpson’s (3/8 )thrule (n=6).

ANS:

def f(x):

return 1 / (1 + x) \*\* 2

def simpsons\_3\_8\_rule(a, b, n):

h = (b - a) / n

sum\_result = f(a) + f(b)

for i in range(1, n):

x = a + i \* h

if i % 3 == 0:

sum\_result += 2 \* f(x)

else:

sum\_result += 3 \* f(x)

integral = (3 \* h / 8) \* sum\_result

return integral

a = 0

b = 1

n = 6

result = simpsons\_3\_8\_rule(a, b, n)

print(f"Estimated integral value: {result:.6f}")

2. Write Python program to obtained the approximate real root of x3 − 8*x* − 4 = 0

using Regula-falsi method.

ANS:

def func(x):

return x\*\*3 - 8\*x - 4

def regula\_falsi(a, b, tolerance, max\_iterations):

if func(a) \* func(b) >= 0:

print("Regula Falsi method may not converge as the initial interval [a, b] does not satisfy the conditions.")

return None

for i in range(max\_iterations):

c = (a \* func(b) - b \* func(a)) / (func(b) - func(a))

if abs(func(c)) < tolerance:

return c

if func(c) \* func(a) < 0:

b = c

else:

a = c

print("Regula Falsi method did not converge within the specified number of iterations.")

return None

a = 2

b = 3

tolerance = 1e-6

max\_iterations = 100

root = regula\_falsi(a, b, tolerance, max\_iterations)

if root is not None:

print(f"Approximate root: {root:.6f}")

**b. Attempt any one of the following. [8]**

1. Write Python program to estimate the value of the integral 2 *dx* using Trapezoidal

rule (n=5).

ANS:

def f(x):

return x\*\*2

def trapezoidal\_rule(a, b, n):

h = (b - a) / n

integral = 0.5 \* (f(a) + f(b))

for i in range(1, n):

x = a + i \* h

integral += f(x)

integral \*= h

return integral

a = 0

b = 1

n = 5

result = trapezoidal\_rule(a, b, n)

print(f"Estimated integral value: {result:.6f}")

2. Write python program to find *sin*(42)0 using Newton backward interpolation formula

for the data:

*sin*300 = 0*.*5*, sin*350 = 0*.*5736*, sin*400 = 0*.*6428*, sin*450 = 0*.*7071.

ANS:

def newton\_backward\_interpolation(x\_values, y\_values, target\_x):

n = len(x\_values)

h = x\_values[1] - x\_values[0]

difference\_table = [[0] \* n for \_ in range(n)]

for i in range(n):

difference\_table[i][0] = y\_values[i]

for j in range(1, n):

for i in range(n - j):

difference\_table[i][j] = difference\_table[i + 1][j - 1] - difference\_table[i][j - 1]

result = difference\_table[0][0]

coefficient = 1

for i in range(1, n):

coefficient \*= (target\_x - x\_values[n - i]) / (i \* h)

result += coefficient \* difference\_table[n - i][i]

return result

x\_values = [30, 35, 40, 45]

y\_values = [0.5, 0.5736, 0.6428, 0.7071]

target\_x = 42

sin\_42 = newton\_backward\_interpolation(x\_values, y\_values, target\_x)

print(f"sin(42°) is approximately: {sin\_42:.4f}")

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Baravkar Pushkar Hiraman **Batch No. :-** **A** Roll No. :- 3 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 20 Class :- SYBCS

**Q.1. Attempt any two of the following. [10]**

1. Write Python code to print first n natural numbers and their square roots of input integer n.

ANS:

import math

n = int(input("Enter a positive integer n: "))

if n <= 0:

print("Please enter a positive integer.")

else:

print("Natural Number | Square Root")

print("-" \* 30)

for i in range(1, n + 1):

square\_root = math.sqrt(i)

print(f"{i} | {square\_root:.4f}")

2. Use Python code to find sum of square of first twenty five natural numbers.

ANS:

n = 25

sum\_of\_squares = (n \* (n + 1) \* (2 \* n + 1)) / 6

print(f"The sum of the squares of the first {n} natural numbers is: {sum\_of\_squares}")

3. Write Python code to find all positive divisors of given number n.

ANS:

n = int(input("Enter a positive integer: "))

if n <= 0:

print("Please enter a positive integer.")

else:

print(f"Positive divisors of {n} are:")

for i in range(1, n + 1):

if n % i == 0:

print(i)

**Q.2. Attempt any two of the following. [10]**

1. Write python code to display tuple ‘I am Indian ’ and the second letter in this tuple

my\_tuple = ('I am Indian',)

second\_letter = my\_tuple[0][1]

print("Tuple:", my\_tuple[0])

print("Second Letter:", second\_letter)

2. Write python code to display the matrix whose all entries are 10 and order is (4,6).

ANS:

rows = 4

columns = 6

matrix = [[10 for \_ in range(columns)] for \_ in range(rows)]

for row in matrix:

print(row)

3. Write Python program to diagonalize the matrix

and find matrix P and D.

ANS:

import numpy as np

A = np.array([[3, -2],[6, -4]])

eigenvalues, eigenvectors = np.linalg.eig(A)

eigenvalue1, eigenvalue2 = eigenvalues

eigenvector1, eigenvector2 = eigenvectors.T

P = np.column\_stack((eigenvector1, eigenvector2))

D = np.diag(eigenvalues)

print("Matrix A:")

print(A)

print("\nEigenvalues:")

print(D)

print("\nMatrix P:")

print(P)

**Q.3. a. Attempt any one of the following. [7]**

1. Write Python program to estimate the value of the integral  *dx* using Simpson’s

(3/8 )*th* rule (n=6).

ANS:

import math

def f(x):

return math.cos(x)

a = 1

b = 3

n = 6

h = (b - a) / n

result = f(a) + f(b)

for i in range(2, n, 2):

result += 2 \* f(a + i \* h)

for i in range(1, n, 2):

result += 3 \* f(a + i \* h)

result = (3 \* h / 8) \* result

print(f"The estimated value of the integral ∫[1, 3] cos(x) dx using Simpson's 3/8 rule (n=6) is: {result:.6f}")

2. Write Python program to evaluate interpolate value *f* (5) of the given data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 1 | 2 | 3 | 6 |
| Y=f(X) | 2 | 6 | 12 | 42 |

ANS:

def lagrange\_interpolation(x, x\_values, y\_values):

result = 0

n = len(x\_values)

for i in range(n):

term = y\_values[i]

for j in range(n):

if i != j:

term \*= (x - x\_values[j]) / (x\_values[i] - x\_values[j])

result += term

return result

x\_values = [1, 2, 3, 6]

y\_values = [2, 6, 12, 42]

x\_interpolate = 5

result = lagrange\_interpolation(x\_interpolate, x\_values, y\_values)

print(f"f(5) is approximately {result:.2f}")

**b. Attempt any one of the following. [8]**

1. Write Python program to obtained the approximate real root of x3 − 5*x* − 9 = 0 in [2,3] using Regula-falsi method.

ANS:

def f(x):

return x\*\*3 - 5\*x - 9

def regula\_falsi(a, b, tol, max\_iter):

for i in range(max\_iter):

c = (a \* f(b) - b \* f(a)) / (f(b) - f(a)

if f(c) == 0 or abs(f(c)) < tol:

return c

elif f(c) \* f(a) < 0:

b = c

else:

a = c

return None

a = 2

b = 3

tolerance = 1e-6

max\_iterations = 100

root = regula\_falsi(a, b, tolerance, max\_iterations)

if root is not None:

print(f"Approximate root: {root:.6f}")

else:

print("Regula Falsi method did not converge within the specified maximum iterations.")

2. Write Python program to estimate the value of the integral 3-5x+2*dx* using

Trapezoidal rule (n=5).

ANS:

return x\*\*3 - 5\*x + 2

def trapezoidal\_rule(a, b, n):

h = (b - a) / n

integral = (f(a) + f(b)) / 2 # Initial and final points

for i in range(1, n):

integral += f(a + i \* h)

integral \*= h

return integral

a = 1

b = 5

n = 5

result = trapezoidal\_rule(a, b, n)

print(f"Estimated integral value: {result:.4f}")

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Jadhav Nikhilk Savleram **Batch No. :-** **B**  Roll No. :- 24 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 21 Class :- SYBCS

**Q.1. Attempt any two of the following.**

1. Write Python code to display multiplication tables of numbers 20 to 30

for i in range(20, 31):

print(f"Multiplication Table for {i}:")

for j in range(1, 11):

product = i \* j

print(f"{i} x {j} = {product}")

print()

2. Write Python code to list name and birth date of 5 students in your class.

student\_data = [

{"name": "Nikhil", "birthdate": "2003-04-15"},

{"name": "Aditya", "birthdate": "2004-09-21"},

{"name": "Pushkar", "birthdate": "2003-12-05"},

{"name": "Vedant", "birthdate": "2004-02-28"},

{"name": "Jaydev", "birthdate": "2003-07-10"}

]

for student in student\_data:

print(f"Name: {student['name']}, Birth Date: {student['birthdate']}")

>> Name: Nikhil, Birth Date: 2003-04-15

Name: Aditya, Birth Date: 2005-01-03

Name: Pushkar, Birth Date: 2004-12-05

Name: Vedant, Birth Date: 2004-02-28

Name: Jaydev, Birth Date: 2003-07-10

3. Write Python function f(a, b) = (4a+b) /3(a–6b) , find the value of f(12, 25).

def fun(a,b):

print((4\*a+b)/(3\*(a-6\*b)))

fun(12,25)

>> -0.17632850241545894

**Q.2. Attempt any two of the following.**

1. Using Python construct the following matrices.

1. Matrix of order 5×6 with all entries 1.

import numpy as np

matrix=np.ones((5,6))

print(matrix)

>>[[1. 1. 1. 1. 1. 1.]

[1. 1. 1. 1. 1. 1.]

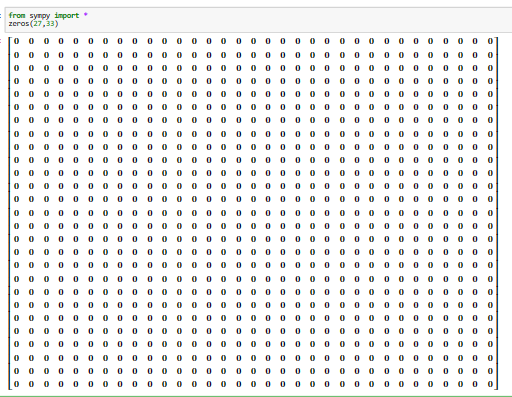
[1. 1. 1. 1. 1. 1.]

[1. 1. 1. 1. 1. 1.]

[1. 1. 1. 1. 1. 1.]]

2. Zero matrix of order 27 × 33.

🡪



3. Identity matrix of order 5.

import numpy as np

matrix3 = np.eye(5)

print(matrix3)

>>[[1. 0. 0. 0. 0.]

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

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

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

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

2. Write python code to perform the R2 + 2R1 row operation on given matrix

R=[[1, 1, 1,],[ 2, 2, 2, ], [3, 3, 3,]]

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

R[1]=R[1]+2\*R[0]

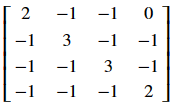
print(R)

>>[[1 1 1]

[4 4 4]

[3 3 3]]

3. Write python code to find all the eigen values and the eigen vectors of the matrix



import numpy as np

# Define your matrix

matrix = np.array([[4, -2], [1, 1]])

# Find eigenvalues and eigenvectors

eigenvalues, eigenvectors = np.linalg.eig(matrix)

# Print the eigenvalues

print("Eigenvalues:")

print(eigenvalues)

# Print the eigenvectors

print("\nEigenvectors:")

print(eigenvectors)

**Q.3. a. Attempt any one of the following.**

1. Write Python program to find the approximate root of the equation x 5 +3x+1 = 0, by using Newton Raphson method.

def f(x):

return x\*\*5 + 3\*x + 1

def f\_prime(x):

return 5\*x\*\*4 + 3

x0 = 0.5 # Initial guess

tolerance = 1e-6 # Tolerance for stopping criterion

max\_iterations = 100 # Maximum number of iterations

for i in range(max\_iterations):

x1 = x0 - f(x0) / f\_prime(x0)

if abs(x1 - x0) < tolerance:

break

x0 = x1

print("Approximate root:", x1)

>>Approximate root: -0.3479134856809244

2. Write a Python program to evaluate interpolate value f(3) of the given data.

x 1 2 3 4

Y=f(x) 11 22 33 66

x\_data = [1, 2, 3, 4]

y\_data = [11, 22, 33, 66]

x\_interpolate = 3

for i in range(len(x\_data) - 1):

if x\_data[i] <= x\_interpolate <= x\_data[i + 1]:

y\_interpolate = y\_data[i] + (x\_interpolate - x\_data[i]) \* (y\_data[i + 1] - y\_data[i]) / (x\_data[i + 1] - x\_data[i])

break

print("Interpolated value at x = 3:", y\_interpolate)

>>Interpolated value at x = 3: 44.0

**b. Attempt any one of the following.**

1. Write Python program to obtained the approximate real root of xsin(x)+cos(x) = 0 by using Regula-falsi method.

import math

def f(x):

return x \* math.sin(x) + math.cos(x)

a = 0.0 # Lower bound of the interval

b = 1.0 # Upper bound of the interval

tolerance = 1e-6 # Tolerance for stopping criterion

max\_iterations = 100 # Maximum number of iterations

for i in range(max\_iterations):

fa = f(a)

fb = f(b)

x = (a \* fb - b \* fa) / (fb - fa)

fx = f(x)

if fx == 0.0 or abs(b - a) < tolerance:

break

if fx \* fa < 0:

b = x

else:

a = x

print("Approximate real root:", x)

>>Approximate real root: 0.7390851362148053

2. Write Python program to find sin(37)0 using Newton backward interpolation formula for the data: sin300 = 0.5, sin350 = 0.5736, sin400 = 0.6428, sin450 = 0.7071

import math

x\_data = [300, 350, 400, 450]

y\_data = [0.5, 0.5736, 0.6428, 0.7071]

x\_interpolate = 370

n = len(x\_data)

h = x\_data[1] - x\_data[0]

diff\_table = [[0 for \_ in range(n)] for \_ in range(n)]

for i in range(n):

diff\_table[i][0] = y\_data[i]

for j in range(1, n):

for i in range(n - j):

diff\_table[i][j] = diff\_table[i + 1][j - 1] - diff\_table[i][j - 1]

result = diff\_table[0][0]

u = (x\_interpolate - x\_data[0]) / h

for i in range(1, n):

result += (u / math.factorial(i)) \* diff\_table[0][i]

print("Approximate sin(37):", result)

>>Approximate sin(37): 0.5922494312500004

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Jadhav Nikhilk Savleram **Batch No. :-** **B**  Roll No. :- 24 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 22 Class :- SYBCS

**Q.1. Attempt any two of the following.**

1. Write Python code to sort a tuples in ascending order (49, 17, 23, 54, 36, 72).

**ANS:**

t = (49, 17, 23, 54, 36, 72)

sorted\_t = tuple(sorted(t))

print(sorted\_t)

**OUTPUT:**

(17, 23, 36, 49, 54, 72)

2. Find the values of the following expressions if x and y are true and z is false.

(a) (x or y) and z.

(b) (x and y) or not z.

(c) (x and not y) or (x and z).

**ANS:**

x = True

y = True

z = False

expression\_a = (x or y) and z

expression\_b = (x and y) or (not z)

expression\_c = (x and (not y)) or (x and z)

print("Expression (a):", expression\_a)

print("Expression (b):", expression\_b)

print("Expression (c):", expression\_c)

**OUTPUT:**

Expression (a): False

Expression (b): True

Expression (c): False

3. Write Python code to find the tuple ‘MATHEMATICS’ from range 3 to 9.

**ANS:**

text = "MATHEMATICS"

result = tuple(text[3:9])

print(result)

OUTPUT: ('H', 'E', 'M', 'A', 'T', 'I')

**Q.2. Attempt any two of the following**

1. Write Python program that prints whether the given number is positive, negative or zero.

**ANS:**

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

if num > 0:

print("Positive")

elif num < 0:

print("Negative")

else:

print("Zero")

**OUTPUT:**

Enter a number:-20

Negative

2. Write Python program to find the sum of first n natural numbers.

**ANS:**

n = int(input("Enter a positive integer n:"))

sum\_natural = (n \* (n + 1)) // 2

print("Sum of the first", n, "natural numbers is:", sum\_natural)

**OUTPUT:**

Enter a positive integer n:90

Sum of the first 90 natural numbers is: 4095

3. Using Python accept the matrix

A=([[1, -3, 2, -4],

[-3, 9, -1, 5],

[5, -2, 6, -3],

[-4, 12, 2, 7]])

Find the Null space, Column space and rank of the matrix.

**ANS**: import sympy as sp

A = sp.Matrix([[1, -3, 2, -4],

[-3, 9, -1, 5],

[5, -2, 6, -3],

[-4, 12, 2, 7]])

rank\_A = A.rank()

null\_space\_A = A.nullspace()

column\_space\_A = A.columnspace()

print("Rank of the matrix A:", rank\_A)

print("Null space of the matrix A:", null\_space\_A)

print("Column space of the matrix A:", column\_space\_A)

**Q.3. a. Attempt any one of the following.**

1. Write Python program to find the approximate root of f(x) = x 3 − 10x 2 + 5 = 0, using Newton Raphson method. Take x0 = 0.5.

**ANS:**

def f(x):

return x\*\*3 - 10\*x\*\*2 + 5

def f\_prime(x):

return 3\*x\*\*2 - 20\*x

x0 = 0.5

tolerance = 1e-6

max\_iterations = 100

for i in range(max\_iterations):

x1 = x0 - f(x0) / f\_prime(x0)

if abs(x1 - x0) < tolerance:

break

x0 = x1

if i < max\_iterations:

print(f"{x1:.6f}")

else:

print("Method did not converge.")

**OUTPUT:** 0.734604

2. Write Python program to evaluate interpolate value f(2) of the given data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 11 | 12 | 13 | 14 |
| Y=f(x) | 21 | 19 | 27 | 64 |

**ANS:**

X = [11, 12, 13, 14]

Y = [21, 19, 27, 64]

x\_interpolate = 2

def linear\_interpolation(x, x0, x1, y0, y1):

return y0 + (y1 - y0) \* (x - x0) / (x1 - x0)

interpolated\_value = linear\_interpolation(x\_interpolate, X[0], X[1], Y[0], Y[1])

print(f"{interpolated\_value:.2f}")

**OUTPUT:** 39.00

**b. Attempt any one of the following.**

1. Write Python program to obtained the approximate real root of x 3 − x 2 − 2 = 0 in [1,2], using Regula-falsi method.

**ANS:**

def f(x):

return x\*\*3 - x\*\*2 - 2

a = 1.0

b = 2.0

tolerance = 1e-6

max\_iterations = 100

if f(a) \* f(b) >= 0:

print("The Regula Falsi method cannot be applied to the given interval.")

else:

for i in range(max\_iterations):

c = (a \* f(b) - b \* f(a)) / (f(b) - f(a))

if abs(f(c)) < tolerance:

break

if f(a) \* f(c) < 0:

b = c

else:

a = c

if i < max\_iterations:

print(f"Approximate root: {c:.6f}")

else:

print("Regula Falsi method did not converge within the maximum number of iterations.")

**OUTPUT:**

Approximate root: 1.695621

2. Using python accept the matrix

A = np.array([[1, 2, 3],

[2, 5, 3],

[1, 0, 8]])

Find the transpose of the matrix, determinant, inverse of the matrix. Also reduce the matrix to reduced row echelon form and diagonalize it.

**ANS:**

import numpy as np

import sympy as sp

A = np.array([[1, 2, 3],

[2, 5, 3],

[1, 0, 8]])

transpose\_A = np.transpose(A)

determinant\_A = np.linalg.det(A)

try:

inverse\_A = np.linalg.inv(A)

except np.linalg.LinAlgError:

inverse\_A = None

rref\_A, \_ = sp.Matrix(A).T.rref()

diagonal\_A = np.diag(np.linalg.eigvals(A))

print(transpose\_A)

print(determinant\_A)

if inverse\_A is not None:

print(inverse\_A)

else:

print("Matrix A is not invertible.")

print(np.array(rref\_A, dtype=float))

print(diagonal\_A)

**OUTPUT:**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Jadhav Nikhilk Savleram **Batch No. :-** **B**  Roll No. :- 24 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 23 Class :- SYBCS

Q.1. Attempt any two of the following.

1. Write Python program to find the sum of first n natural numbers.

**ANS:** n = int(input("Enter a positive integer: "))

sum = n \* (n + 1) // 2

print("The sum of the first", n, "natural numbers is", sum)

**OUTPUT:** Enter a positive integer: 29

The sum of the first 29 natural numbers is 435

2. Write Python code to prints all integers between 1 to 100 that are divisible by 3 and 7.

**ANS:** for i in range(1, 101):

if i % 3 == 0 and i % 7 == 0:

print(i)

**OUTPUT:** 21

42

63

84

3. Write Python code to prints all integers between 1 to n, which are relatively prime to n.

**ANS:** import math

def are\_relatively\_prime(a, b):

return math.gcd(a, b) == 1

n = int(input("Enter a positive integer n: "))

print(f"Integers relatively prime to {n} between 1 and {n} are:")

for i in range(1, n + 1):

if are\_relatively\_prime(i, n):

print(i)

**OUTPUT:** Enter a positive integer n: 30

Integers relatively prime to 30 between 1 and 30 are:

1

7

11

13

17

19

23

2

**Q.2. Attempt any two of the following.**

1. Write Python code to find determinant, transpose and inverse of matrix

A=([[1, 2, 3], [2, 5, 7], [4, 9, 11]])

**ANS:** import sympy as sp

A = sp.Matrix([[1, 2, 3], [2, 5, 7], [4, 9, 11]])

determinant = A.det()

print(determinant)

transpose = A.T

print(transpose)

if determinant != 0:

inverse = A.inv()

print(inverse)

else:

print("\nThe matrix is singular, and its inverse does not exist.")

**OUTPUT:** -2

Matrix([[1, 2, 4], [2, 5, 9], [3, 7, 11]])

Matrix([[4, -5/2, 1/2], [-3, 1/2, 1/2], [1, 1/2, -1/2]])

2. Write Python program to find the roots of the quadratic equation ax2 + bx + c = 0.

**ANS:** import math

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

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

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

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

if discriminant > 0:

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

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

print(f"Root 1: {root1}")

print(f"Root 2: {root2}")

elif discriminant == 0:

root = -b / (2\*a)

print(f"Root: {root}")

else:

realPart = -b / (2\*a)

imaginaryPart = math.sqrt(abs(discriminant)) / (2\*a)

print(f"Root 1: {realPart} + {imaginaryPart}i")

print(f"Root 2: {realPart} - {imaginaryPart}i")

**OUTPUT:** Enter the coefficient a: 2

Enter the coefficient b: 4

Enter the coefficient c: -3

Root 1: 0.5811388300841898

Root 2: -2.58113883008419

3. Using Python solve the following system of equations using LU – Factorization method

3x − 7y − 2z = −7

−3x + 5y + z = 5

6x − 4y = 2

**ANS:** import numpy as np

import scipy.linalg as la

A = np.array([[3, -7, -2], [-3, 5, 1], [6, -4, 0])

B = np.array([-7, 5, 2])

P, L, U = la.lu(A)

Y = la.solve\_triangular(L, P.dot(B), lower=True)

X = la.solve\_triangular(U, Y)

print(X)

**OUTPUT:** [-5.5 -9.5 24. ]

**3. a. Attempt any one of the following.**

1. Write Python program to estimate the value of the integral R 3 1 1 x dx by using Simpson’s ( 1 3 ) rd rule (n=8).

**ANS:** def f(x):

return 1/x

def simpsons\_one\_third\_rule(a, b, n):

h = (b - a) / n

x = [a + i \* h for i in range(n + 1)]

y = [f(xi) for xi in x]

integral = y[0] + y[n]

for i in range(1, n):

if i % 2 == 0:

integral += 2 \* y[i]

else:

integral += 4 \* y[i]

integral = integral \* h / 3

return integral

a = 1

b = 3

n = 8

result = simpsons\_one\_third\_rule(a, b, n)

print(f"Estimated integral value: {result:.6f}")

**OUTPUT:** Estimated integral value: 1.098725

2. Write Python program to obtained the approximate real root of x 4 − 8x 2 − 4 = 0 using Regula-falsi method.

**ANS:** def f(x):

return x\*\*4 - 8\*x\*\*2 - 4

def regula\_falsi(a, b, tol, max\_iter):

if f(a) \* f(b) >= 0:

print("Regula-Falsi method may not converge on the given interval.")

return None

for i in range(max\_iter):

c = (a \* f(b) - b \* f(a)) / (f(b) - f(a)

if abs(f(c)) < tol:

return c

if f(a) \* f(c) < 0:

b = c

else:

a = c

return None

a = 1.0 # Left endpoint of the interval

b = 3.0 # Right endpoint of the interval

tolerance = 1e-6 # Tolerance for stopping criteria

max\_iterations = 100 # Maximum number of iterations

root = regula\_falsi(a, b, tolerance, max\_iterations)

if root is not None:

print(f"Approximate real root: {root:.6f}")

else:

print("Regula-Falsi method did not converge within the specified iterations.")

**OUTPUT:** Approximate real root: 2.910693

**b. Attempt any one of the following.**

1. Write Python program to estimate the value of the integral R 1 0 x 5dx using Trapezoidal rule (n=10).

**ANS:**

def f(x):

return x\*\*5

def trapezoidal\_rule(a, b, n):

h = (b - a) / n

x = [a + i \* h for i in range(n + 1)]

y = [f(xi) for xi in x]

integral = (y[0] + y[n]) / 2

for i in range(1, n):

integral += y[i]

integral = integral \* h

return integral

a = 0

b = 1

n = 10

result = trapezoidal\_rule(a, b, n)

print(f"Estimated integral value: {result:.6f}")

**OUTPUT:** Estimated integral value: 0.170825

2. Write Python program to find sin(35)0 using Newton backward interpolation formula for the data: sin300 = 0.5, sin350 = 0.5736, sin400 = 0.6428, sin450 = 0.7071.

**ANS:**

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Jadhav Nikhilk Savleram **Batch No. :-** **B**  Roll No. :- 24 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 24 Class :- SYBCS

**Q.1. Attempt any two of the following.**

1. Write Python program to calculate the surface area of sphere A = 4πr2 .

**ANS:** def area\_of\_sphere(r):

Area=4\*3.14\*(r\*\*2)

return Area

area\_of\_sphere(2)

**OUTPUT:** 50.24

2. Use Python code to find the remainder after dividing by ‘n’ to any integers.

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

n = int(input("Enter n: "))

remainder = integer % n

print("Remainder:", remainder)

**OUTPUT:** Enter an integer: 23

Enter n: 4

Remainder: 3

3. Write Python program to prints all integers between 1 to 50 that are divisible by 3 and 7.

**ANS:** for i in range(1, 51):

if i % 3 == 0 and i % 7 == 0:

print(i)

**OUTPUT:** 21

42

**Q.2. Attempt any two of the following.**

1. Write Python program to find perfect square between 1 to 100.

**ANS:** for num in range(1, 101):

sqrt = int(num \*\* 0.5)

if sqrt \* sqrt == num:

print(num)

**OUTPUT:** 1

4

9

16

25

36

49

64

81

100

2. Write Python program to prints whether the given natural number is divisible by 5 and less than 100.

**ANS:** num = int(input("Enter a natural number: "))

if num % 5 == 0 and num < 100 and num > 0:

print(f"{num} is divisible by 5 and less than 100.")

else:

print(f"{num} is not divisible by 5 or not less than 100.")

**OUTPUT:** *Enter a natural number: 20*

*20 is divisible by 5 and less than 100*.

3. Write Python program to diagonalize the matrix

A= ([[2, -3],

[4, -6]])

and find matrix P and D.

**ANS:** import numpy as np

A = np.array([[2, -3],

[4, -6]])

eigenvalues, eigenvectors = np.linalg.eig(A)

P = eigenvectors

D = np.diag(eigenvalues)

print("Matrix P:")

print(P)

print("Matrix D:")

print(D)

**OUTPUT:** Matrix P

[ 0.31622777 0.9486833 ]

[ 0.9486833 -0.31622777]

Matrix D

[[0. 0.]

[0. 4.]]

**Q.3. a. Attempt any one of the following.**

1. Write Python program to estimate the value of the integral R 3 1 cos(x)dx using Simpson’s ( 3 8 ) th rule (n=5).

**ANS:** import math

def f(x):

return math.cos(x)

a = 1

b = 3

n = 5

h = (b - a) / n

integral = f(a) + f(b)

for i in range(1, n):

x = a + i \* h

if i % 3 == 0:

integral += 2 \* f(x)

else:

integral += 3 \* f(x)

integral = (3 \* h / 8) \* integral

print("Estimated integral:", integral)

**OUTPUT:** Estimated integral: 1.846884345043672

2. Write Python program to evaluate f(1.9) by using backward difference formula of the given data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 1 | 2 | 3 | 4 |
| Y | 11 | 10 | 15 | 10 |

**ANS:** X = [1, 2, 3, 4]

Y = [11, 10, 15, 10]

x0 = 1.9

n = len(X)

backward\_diff = [[0 for \_ in range(n)] for \_ in range(n)]

for i in range(n):

backward\_diff[i][0] = Y[i]

for j in range(1, n):

for i in range(n - j):

backward\_diff[i][j] = backward\_diff[i + 1][j - 1] - backward\_diff[i][j - 1]

k = 0

while k < n - 1 and X[k + 1] <= x0:

k += 1

h = X[1] - X[0]

s = (x0 - X[k]) / h

result = backward\_diff[k][0]

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

result += (s - j + 1) \* backward\_diff[k - j][j] / math.factorial(j)

print(f"f({x0}) ≈ {result:.5f}")

**OUTPUT:** f(1.9) ≈ 11.20000

**b. Attempt any one of the following**.

1. Write Python program to obtained the approximate real root of x 3 − 5x − 9 = 0 in [2,4] using Regula-falsi method.

**ANS:** **d**ef f(x):

return x\*\*3 - 5\*x - 9

a = 2

b = 4

tolerance = 1e-6

max\_iterations = 100

iterations = 0

x0 = a

x1 = b

while iterations < max\_iterations:

x2 = x1 - (f(x1) \* (x1 - x0)) / (f(x1) - f(x0))

if abs(x2 - x1) < tolerance:

break

if f(x0) \* f(x2) < 0:

b = x2

else:

a = x2

x0 = x1

x1 = x2

iterations += 1

root\_approximation = x2

print("Approximate Root:", root\_approximation)

**OUTPUT:** Approximate Root: 2.1931789269130723

2. Write Python program to evaluate interpolate value f(17) of the given data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 12 | 22 | 32 | 62 |
| Y | 25 | 65 | 125 | 425 |

**ANS**: def lagrange\_interpolation(x, X, Y):

result = 0

for i in range(len(X)):

term = Y[i]

for j in range(len(X)):

if j != i:

term \*= (x - X[j]) / (X[i] - X[j])

result += term

return result

X = [12, 22, 32, 62]

Y = [25, 65, 125, 425]

x\_interpolate = 17

interpolated\_value = lagrange\_interpolation(x\_interpolate, X, Y)

print(f"f({x\_interpolate}) = {interpolated\_value}")

**OUTPUT:** f(17) = 43.125

**Sahakar Maharshi Bhausaheb Santuji Thorat**

**Remark**

**Demonstrators**

**Signature**

**Date :- / /2023**

**College., Sangamner**

**DEPARTMENT OF COMPUTER SCIENCE**

**MATHEMATICS**

**Name :-** Jadhav Nikhilk Savleram **Batch No. :-** **B**  Roll No. :- 24 Date :- …./…/2023

Title :- Python Practical Expt.No. :- 25 Class :- SYBCS

**Q.1. Attempt any two of the following.**

1. Write Python program to print the integers between 1 and 1000 which are multiples of 7.

**ANS:** for i in range(1, 1001):

if i % 7 == 0:

print(i)

**OUTPUT:**

2. Write Python program to prints whether the given number is divisible by 3 or 5 or 7.

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

if num % 3 == 0:

print("Divisible by 3")

if num % 5 == 0:

print("Divisible by 5")

if num % 7 == 0:

print("Divisible by 7")

**OUTPUT:** Divisible by 3

Divisible by 5

3. Write Python code to find A + B and B ∗ A for the given Matrices.

**ANS:** import numpy as **np**

A = np.array([[4, 2, 4], [4, -1, 1], [2, 4, 2]])

B = np.array([[5, 2, 3], [3, -7, 5], [3, 1, -1])

sum\_result = A + B

product\_result = np.dot(B, A)

print("Sum (A + B):\n", sum\_result)

print("Product (B \* A):\n", product\_result)

**OUTPUT:**

**Q.2. Attempt any two of the following.**

1. Write Python program to find the area and circumference of a circle with radius r.

**ANS:** import math

r = float(input("Enter the radius of the circle: "))

area = math.pi \* (r \*\* 2)

circumference = 2 \* math.pi \* r

print("The area of the circle is:", area)

print("The circumference of the circle is:", circumference)

**OUTPUT**: Enter the radius of the circle: 5

The area of the circle is: 78.53981633974483

The circumference of the circle is: 31.41592653589793

2. Use Python code to solve the following system of equations by gauss elimination method

x + y + 2z = 7

−x − 2y + 3z = 6

3x − 7y + 6z = 1

**ANS**: import numpy as np

coefficients = np.array([[1, 1, 2, 7],

[-1, -2, 3, 6],

[3, -7, 6, 1]], dtype=float)

n = len(coefficients)

for i in range(n):

max\_row = i

for j in range(i + 1, n):

if abs(coefficients[j, i]) > abs(coefficients[max\_row, i]):

max\_row = j

coefficients[[i, max\_row]] = coefficients[[max\_row, i]]

for j in range(i + 1, n):

factor = coefficients[j, i] / coefficients[i, i]

coefficients[j, i:] -= factor \* coefficients[i, i:]

x = np.zeros(n)

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

x[i] = (coefficients[i, -1] - np.dot(coefficients[i, i+1:n-1], x[i+1:])) / coefficients[i, i]

print("Solution:")

print("x =", x[0])

print("y =", x[1])

print("z =", x[2])

**OUTPUT:** Solution:

x = 1.0

y = 2.0

z = 3.0

3. Write Python code to find eigen values, eigen vectors of the matrix and determine whether the matrix is diagonalizable.

A=([[1, -1, 1],

[-1, 1, -1],

[1, -1, 1]])

**ANS:** import numpy as np

A = np.array([[1, -1, 1],

[-1, 1, -1],

[1, -1, 1]])

eigenvalues, eigenvectors = np.linalg.eig(A)

is\_diagonalizable = not np.iscomplexobj(eigenvalues).any()

print("Eigenvalues:")

print(eigenvalues)

print("\nEigenvectors:")

print(eigenvectors)

print("\nIs the matrix diagonalizable?", is\_diagonalizable)

**OUTPUT:** **Eigenvalues:**

[ 0. 2. -0.]

Eigenvectors:

[[-0.57735027 0. -0.57735027]

[ 0.57735027 0. -0.57735027]

[-0.57735027 1. -0.57735027]]

Is the matrix diagonalizable? True

**Q.3. a. Attempt any one of the following.**

1. Write Python program to find the approximate root of the equation f(x) = x 2 − 50 by using Newton Raphson method.

**ANS:** def f(x):

return x\*\*2 - 50

def df(x):

return 2 \* x

x0 = 6.0 # Initial guess

tolerance = 1e-6

max\_iterations = 100

for i in range(max\_iterations):

x1 = x0 - f(x0) / df(x0)

if abs(x1 - x0) < tolerance:

break

x0 = x1

print("Approximate root:", x1)

**OUTPUT:** Approximate root: 7.071067811865475

2. Write Python program to evaluate f(2.4) by forward difference formula of the given data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 0 | 1 | 2 | 3 |
| Y | 11 | 10 | 11 | 21 |

**ANS:** X = [0, 1, 2, 3]

Y = [11, 10, 11, 21]

x = 2.4

n = len(X)

forward\_diff\_table = [[0] \* n for \_ in range(n)]

for i in range(n):

forward\_diff\_table[i][0] = Y[i]

for j in range(1, n):

for i in range(n - j):

forward\_diff\_table[i][j] = forward\_diff\_table[i + 1][j - 1] - forward\_diff\_table[i][j - 1]

h = X[1] - X[0]

s = (x - X[0]) / h

result = forward\_diff\_table[0][0]

term = 1

for i in range(1, n):

term \*= s - i + 1

term /= i

result += term \* forward\_diff\_table[0][i]

print("f(2.4) =", result)

**OUTPUT:** f(2.4) = 11.88

**b. Attempt any one of the following.**

1. Write Python program to estimate the value of the integral R 1 0 sin2 (πx)dx using Simpson’s ( 1 3 ) rd rule (n=6).

**ANS:** import math

def f(x):

return math.sin(math.pi \* x)\*\*2

n = 6

a = 0

b = 1

h = (b - a) / n

result = f(a) + f(b)

for i in range(1, n, 2):

result += 4 \* f(a + i \* h)

for i in range(2, n-1, 2):

result += 2 \* f(a + i \* h)

result \*= h / 3

print("Estimated integral value:", result)

**OUTPUT:** Estimated integral value: 0.5

2. Write Python program to find f(4) using Lagranges interpolation formula from the data: f(1) = 6, f(2) = 9, f(5) = 30, f(7) = 54.

**ANS:** def lagrange\_interpolation(x, data\_points):

n = len(data\_points)

result = 0

for i in range(n):

term = data\_points[i][1]

for j in range(n):

if j != i:

term \*= (x - data\_points[j][0]) / (data\_points[i][0] - data\_points[j][0])

result += term

return result

data\_points = [(1, 6), (2, 9), (5, 30), (7, 54)]

x = 4

result = lagrange\_interpolation(x, data\_points)

print("f(4) =", result)

**OUTPUT:** f(4) = 15.6