KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING – DATA SCIENCE

APPLIED PYTHON PROGRAMMING LABORATORY

LAB MANUAL

Subject Code:

Regulation: KGR21

Academic Year: 2022-2023

I B. TECH II SEMESTER

COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

KG REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

Affiliated To JNTUH, Chilkur (V), Moinabad(M) R. R Dist, TS-501504

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-DATA SCIENCE

VISION AND MISSION OF THE INSTITUTION

VISION:

To become an institution which is internationally recognized for its holistic approach to engineering, innovative teaching and learning culture, research and entrepreneurial ecosystem, and sustainable social impact in the community.

MISSION:

• To offer undergraduate and post-graduate programs which are supported through industry relevant curriculum and innovative teaching and learning processes that would help students succeed in their professional careers.
• To provide faculty and students with an ecosystem that fosters innovation, research, entrepreneurship, and international exposure through strategic partnerships with government organizations and collaboration with industries.
• To provide holistic learning environment to students which will contribute to their personal and professional growth and enable them to become leaders in their respective fields.
• To contribute to the development of the region by using our technological expertise to work with nearby communities and support them in their social and economic development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

VISION AND MISSION OF THE DEPARTMENT

VISION:

To be recognized as a department of excellence by stimulating a learning environment in which students and faculty will thrive and grow to achieve their professional, institutional and societal goals.

MISSION:

• To provide high quality technical education to students that will enable life-long learning and build expertise in advanced technologies in Computer Science and Engineering.
• To promote research and development by providing opportunities to solve complex engineering problems in collaboration with industry and government agencies.
• To encourage professional development of students that will inculcate ethical values and leadership skills through entrepreneurship while working with the community to address societal issues.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

PROGRAM EDUCATIONAL OBJECTIVES

PEO 1: Graduates will provide solutions to difficult and challenging issues in their profession by applying computer science and engineering theory and principles.
PEO 2: Graduates have successful careers in computer science and engineering fields or will be able to successfully pursue advanced degrees.
PEO 3: Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism, moral and ethical responsibility.
PEO 4: Graduates will develop the ability to understand and analyze engineering issues in a broader perspective with ethical responsibility towards sustainable development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

PROGRAM OUTCOMES

PO I: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO II: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO III: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO IV: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO V: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO VI: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO VII: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of and need for sustainable development.
PO VIII: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO IX: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO X: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO XI: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO XII: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

PROGRAM SPECIFIC OUTCOMES

PSO1: Analyze, design, develop, test and apply statistical models, tools, Mathematical foundations and management principles in the development of intelligent systems with computational solutions, make them to expert in designing the Software and hardware.
PSO2: Apply suitable techniques and adaptive algorithms to perform data analysis and Visualization Techniques for solving complex problems and effective decision making from inter-disciplinary domains.
PSO3: Exhibit domain knowledge and expertise for enhancing research capability to transform innovative ideas into reality with Societal Issues, Ethics, entrepreneurship and higher studies.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING - DATA SCIENCE

Name of the Laboratory:

GENERAL LABORATORY INSTRUCTIONS:

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis /program / experiment details.
3. Student should enter into the laboratory with:
   a) Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
   b) Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.
   c) Proper Dress code and Identity card.
   d) Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
   e) Execute your task in the laboratory, and record the results / output in the lab observation note book, andget certified by the concerned faculty.
4. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
5. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
6. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
7. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
8. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

HoD, CSE-CSD **Principal**

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING - DATA SCIENCE

Name of the Laboratory:

Things to DO:

1. Be on time. At the start of the lab period, there will be a short introduction to the experiment you will perform that day. It is unfair to your partner and to others in the lab if you are not up to speed when the work begins.
2. Inform the instructor and/or TA if there is a problem. You will have their immediate attention if you have cut yourself (even if you consider it minor), if something broke and needs cleaning up, or if you are onfire.
3. Be aware of all the safety devices. Even though the instructor and TA will take care of emergencies, you should know where to find the first aid kit, the chemical spill kit, the eye wash and the safety shower.
4. Keep clutter to a minimum. There is a coat rack to hang your jackets and there are empty cabinets to store your backpacks. Anything left in the aisles is likely to be stepped on and is a hazard to everyone.
5. Wash your hands before you leave the lab for the day.
6. Be aware of others in the lab. Areas of the room may be crowded at times and you should take care not to disturb the experiments of others in the lab.
7. Bring your lab notebook and an open mind to every lab meeting.

Things NOT TO DO:

1. Do not eat, drink, chew gum, smoke or apply cosmetics in the lab. Just being in lab makes your hands dirtier than you can imagine and you don't want to accidentally eat any reagent (see item 5 on 'things to do' list).
2. Do not put pieces of lab equipment in your mouth. It sounds obvious but you'd be surprised!
3. Do not work with chemicals until you are sure of their safe handling. This includes some awareness of their flammability, reactivity, toxicity, and disposal.
4. Do not use the phone or computer with gloves on your hands.

HoD, CSE-CSD **Principal**

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING - DATA SCIENCE
OPERATING SYSTEMS LAB MANUAL (KG21CS404)TABLE OF

CONTENTS

	b) Read text from a file and return a list of all n letter words beginning with a vowel c) Finding a secret message hidden in a paragraph of text d) Plot a histogram of words according to their length from text read from a file	
	7. Installing OS on Raspberry Pi a) Installation using PiImager b) Installation using image file Downloading an Image• Writing the image to an SD card• using Linux• using Windows• Booting up• Follow the instructions given in the URL https://www.raspberrypi.com/documentation/computers/getting-started.html	
8	Accessing GPIO pins using Python a) Installing GPIO Zero library. First, update your repositories list: sudo apt update Then install the package for Python 3: sudo apt install python3-gpiozero b) Blinking an LED connected to one of the GPIO pin c) Adjusting the brightness of an LED d) Adjust the brightness of an LED (0 to 100, where 100 means maximum brightness) using the in-built PWM wavelength.	
9	Collecting Sensor Data a) DHT Sensor interface ◦ Connect the terminals of DHT GPIO pins of Raspberry Pi. ◦ Import the DHT library using import Adafruit_DHT ◦ Read sensor data and display it on screen.	

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING - DATA SCIENCE

Cycle - 1

1. Downloading and Installing Python and Modules

To download and install Python and modules, follow these steps:

1. Go to the official Python website (https://www.python.org/) and download the latest version of Python for your operating system.
2. Once the download is complete, run the installer and follow the installation instructions.
3. After installing Python, you can open a command prompt or terminal and type "python" to access the Python interpreter.
4. To install modules, you can use pip, the default package installer for Python. To install a module, open a command prompt or terminal and type "pip install <module_name>". Replace "<module_name>" with the name of the module you want to install.
5. If you want to install a specific version of a module, you can use the command "pip install <module_name>==<version>". Replace "<version>" with the version number you want to install.
6. If you want to see a list of all installed modules, you can use the command "pip list".
7. You can also create a virtual environment to isolate your Python environment and module dependencies. To create a virtual environment, open a command prompt or terminal and type "python -m venv <env_name>". Replace "<env_name>" with the name of your virtual environment.
8. To activate the virtual environment, type "source <env_name>/bin/activate" on Linux or macOS, or "<env_name>\Scripts\activate" on Windows.
9. Once the virtual environment is activated, you can install modules using pip as described above.
10. To exit the virtual environment, type "deactivate" in the command prompt or terminal.

Note: It's important to keep your Python and module versions up-to-date to ensure compatibility and security. You can use pip to update modules by typing "pip install --upgrade <module_name>". You can also update Python itself by downloading and installing the latest version from the official Python website.

a) Python 3 on Linux Follow the instructions given in the URL
https://docs.python guide.org/starting/install3/linux/

here are the instructions to install Python 3 on Linux:

1. Open the terminal on your Linux machine. You can do this by pressing Ctrl+Alt+T or by searching for "Terminal" in your applications menu.
2. Check if Python is already installed on your system by running the following command in the terminal:
   python3 –version

3. If Python is already installed, the output of the above command will show the version number. If not, you can install Python 3 by running the following command:

   sudo apt-get update

   sudo apt-get install python3

4. Once the installation is complete, you can verify the installation by running the following command:

   python3 –version

   You can now start using Python 3 by running the **python3** command in the terminal. For example, to run a Python script named **myscript.py**, navigate to the directory containing the script in the terminal and run the following command:

   python3 myscript.py

   That's it! You now have Python 3 installed on your Linux machine and can start using it for your development projects.

b) Python 3 on Windows Follow the instructions given in the URL https://docs.python.org/3/using/windows.html (Please remember that Windows installation of Python is harder!)

Here are the steps you can follow to install Python 3 on Windows:

1. Go to the official Python website (https://www.python.org/downloads/) and click on the "Download Python" button.
2. Scroll down to the "Python Releases for Windows" section and click on the link for the latest version of Python 3.x.
3. On the download page, scroll down to the "Files" section and select the appropriate installer for your system. Choose the 32-bit or 64-bit version depending on your system architecture.
4. Once the installer has finished downloading, double-click on it to run it.
5. On the first screen of the installer, make sure that the "Add Python 3.x to PATH" option is selected, and then click "Install Now".
6. The installer will then begin installing Python 3 on your system. This may take a few minutes.
7. Once the installation is complete, you can launch the Python interpreter by opening a command prompt and typing "python" (without the quotes). You should see the Python version information displayed, indicating that Python 3 is now installed on your system.

That's it! You've now successfully installed Python 3 on your Windows machine. You can now use Python to write and run programs on your system.

c) pip3 on Windows and Linux Install the Python package installer by following the instructions given in the URL https://www.activestate.com/resources/quick-reads/how-to-install-and-use-pip3/

To install pip3 on Windows:

1. Download the latest version of Python from the official website: https://www.python.org/downloads/
2. During the installation process, make sure to check the box "Add Python x.x to PATH" (replace "x.x" with the version of Python you're installing)
3. Open a command prompt by pressing the Windows key + R and typing "cmd" followed by Enter.
4. Type the command **python -m ensurepip --upgrade** to upgrade pip to the latest version.
5. Verify that pip is installed correctly by typing **pip --version** in the command prompt. You should see the version number printed out.

To install pip3 on Linux:

1. Open a terminal window.
2. Run the command **sudo apt-get update** to update your system's package list.
3. Install pip3 by running the command **sudo apt-get install python3-pip**
4. Verify that pip3 is installed correctly by typing **pip3 --version** in the terminal. You should see the version number printed out.

That's it! You now have pip3 installed on your system.

d) Installing numpy and scipy You can install any python3 package using the command pip3 install

you can install the numpy and scipy packages using the pip3 command in Python 3.

To install numpy, you can run the following command:

pip3 install numpy

To install scipy, you can run the following command:

pip3 install scipy

Make sure you have the latest version of pip3 installed on your system before running these commands. You can upgrade pip3 using the following command:

pip3 install --upgrade pip

It's also recommended to create a virtual environment before installing any Python packages. This can help you avoid conflicts with existing packages and provide a clean environment to work in. You can create a virtual environment using the venv module in Python 3:

python3 -m venv myenv

This will create a virtual environment named "myenv" in the current directory. You can activate the environment by running the following command:

source myenv/bin/activate

After activating the environment, you can install the packages using the pip3 command as described above.

e) Installing jupyterlab

Install from pip using the command pip install jupyterlab

To install JupyterLab using pip, you can follow these steps:

1. Open a command prompt or terminal window on your computer.
2. Type **pip install jupyterlab** and press enter.
3. Wait for the installation process to complete.
4. Once the installation is complete, you can launch JupyterLab by typing **jupyter lab** in the command prompt or terminal window.

Note: Make sure you have Python and pip installed on your computer before running the above command.

2.Introduction to Python3

a) Printing your biodata on the screen

```
#python program to print biodata
print ("Biodata")
a=input("Enter your Name:")
b=input("Enter your Father's Name:")
c=input("Enter your Locality:")
d=int(input("Enter your Date of Birth:"))
e=int(input("Enter your Age:"))
f=input("Enter your nation:")
print("Name:",a, end=" ")
print("Father's name:",b, end=" ")
print("Locality:",c, end=" ")
print("Date of Birth:",d, end=" ")
print("Age:",e, end=" ")
print("Nation:",f, end=" ")
```

b) Printing all the primes less than a given number

```
class Solution:
  def solve(self, n):
    sieve = [True] * (n + 1)
    primes = []
    for i in range(2, n + 1):
      if sieve[i]:
        primes.append(i)
        for j in range(i, n + 1, i):
          sieve[j] = False
    return primes
ob = Solution()
print(ob.solve(12))
```

c)Finding all the factors of a number and show whether it is a perfect number, i.e., the sum of all its factors (excluding the number itself) is equal to the number itself

```
n = int(input("Enter any number: "))
sum1 = 0
for i in range(1, n):
    if(n % i == 0):
        sum1 = sum1 + i
if (sum1 == n):
    print("The number is a Perfect number!")
else:
    print("The number is not a Perfect number!")
```

3.Defining and Using Functions

a) Write a function to read data from a file and display it on the screen

```
a=str(input("Enter the name of the file with .txt extension:"))
file2=open(a,'r')
line=file2.readline()
while(line!=""):
    print(line)
    line=file2.readline()
file2.close()
```

b)Define a Boolean function is palindrome

```
def is_palindrome(x):
    x = "".join(x.split()).lower()
    if (len(x) <= 1): return True
    if x[0] != x[-1]: return False
    return is_palindrome(x[1:-1])
```

c) Write a function collatz(x) which does the following: if x is odd, x = 3x + 1; if x is even, then x = x/2. Return the number of steps it takes for x = 1

```
def printCollatz(n):
    # We simply follow steps
    # while we do not reach 1
    while n != 1:
        print(n, end = ' ')
        # If n is odd
        if n & 1:
            n = 3 * n + 1
        # If even
        else:
            n = n // 2

    # Print 1 at the end
    print(n)
# Driver code
printCollatz(6)
```

d)Write a function N(m, s) = exp(-(x-m)2 /(2s2))/sqrt(2π)s that computes the Normal distribution

```
import math
def N(m, s):
    def normal_distribution(x):
        exponent = -1 * ((x - m)**2 / (2 * s**2))
        coefficient = 1 / (math.sqrt(2 * math.pi) * s)
        return coefficient * math.exp(exponent)
    return normal_distribution
```

4)The package numpy

a) Creating a matrix of given order m x n containing random numbers in the range 1 to 99999

```
import numpy as np
m=int(input("enter the number of rows"))
n=int(input("enter the number of cloumns"))
a=np.random.randint(1,99999,size=(m,n))
print(a)
```

b) Write a program that adds, subtracts and multiplies two matrices. Provide an interface such that, based on the prompt, the function (addition, subtraction, multiplication) should be performed
Matrix addition, subtraction, and multiplication program

```
def add_matrices(matrix1, matrix2):
    result = []
    for i in range(len(matrix1)):
        row = []
        for j in range(len(matrix1[0])):
            row.append(matrix1[i][j] + matrix2[i][j])
        result.append(row)
    return result

def subtract_matrices(matrix1, matrix2):
    result = []
    for i in range(len(matrix1)):
        row = []
        for j in range(len(matrix1[0])):
            row.append(matrix1[i][j] - matrix2[i][j])
        result.append(row)
    return result

def multiply_matrices(matrix1, matrix2):
    result = []
    for i in range(len(matrix1)):
        row = []
```

```
        for j in range(len(matrix2[0])):
            total = 0
            for k in range(len(matrix2)):
                total += matrix1[i][k] * matrix2[k][j]
            row.append(total)
        result.append(row)
    return result

# Main program
while True:
    print("Enter 'q' to quit")
    print("Enter 'add' to add two matrices")
    print("Enter 'subtract' to subtract two matrices")
    print("Enter 'multiply' to multiply two matrices")

    user_choice = input("What operation would you like to perform? ")

    if user_choice == 'q':
        break

    # Get input matrices from user
    matrix1 = []
    matrix2 = []

    rows1 = int(input("Enter the number of rows for matrix 1: "))
    cols1 = int(input("Enter the number of columns for matrix 1: "))

    print("Enter matrix 1 row by row:")
    for i in range(rows1):
        row = []
        for j in range(cols1):
            num = int(input())
            row.append(num)
        matrix1.append(row)

    rows2 = int(input("Enter the number of rows for matrix 2: "))
    cols2 = int(input("Enter the number of columns for matrix 2: "))
```

```
print("Enter matrix 2 row by row:")
 for i in range(rows2):
    row = []
    for j in range(cols2):
       num = int(input())
       row.append(num)
    matrix2.append(row)

 # Perform the operation based on user input
 if user_choice == 'add':
    if rows1 == rows2 and cols1 == cols2:
       result = add_matrices(matrix1, matrix2)
       print("The result is:")
       for row in result:
          print(row)
    else:
       print("The matrices must have the same dimensions to add them.")

 elif user_choice == 'subtract':
    if rows1 == rows2 and cols1 == cols2:
       result = subtract_matrices(matrix1, matrix2)
       print("The result is:")
       for row in result:
          print(row)
    else:
       print("The matrices must have the same dimensions to subtract them.")

 elif user_choice == 'multiply':
    if cols1 == rows2:
       result = multiply_matrices(matrix1, matrix2)
       print("The result is:")
       for row in result:
          print(row)
    else:
       print("The number of columns in matrix 1 must match the number of rows in matrix 2 to
multiply them.")

 else:       print("Invalid input. Please try again.")
```

c)Write a program to solve a system of n linear equations in n variables using matrix inverse
import numpy as np

```
def solve_system(A, b):
    """
    Solves a system of linear equations Ax = b using matrix inverse method.

    Args:
    A (numpy.ndarray): An n x n matrix representing the coefficients of the variables in the equations.
    b (numpy.ndarray): An n x 1 matrix representing the constants in the equations.

    Returns:
    numpy.ndarray: An n x 1 matrix representing the values of the variables that solve the system.
    """
    # Compute the inverse of A
    A_inv = np.linalg.inv(A)

    # Solve for x
    x = np.dot(A_inv, b)
    return x

# Example system of equations:
# x + 2y + 3z = 7
# 2x + 5y + 2z = 8
# 3x + 2y + 9z = 12
A = np.array([[1, 2, 3], [2, 5, 2], [3, 2, 9]])
b = np.array([[7], [8], [12]])
# Solve the system of equations
x = solve_system(A, b)
# Print the solution
print("x =\n", x)
```

```
x =
 [[ 1.]
 [-2.]
 [ 2.]]
```

5)The package scipy and pyplot

a) Finding if two sets of data have the same mean value

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
# Generate two sets of data
set1 = np.random.normal(loc=10, scale=2, size=100)
set2 = np.random.normal(loc=10, scale=2, size=100)
# Calculate the means of each set
mean1 = np.mean(set1)
mean2 = np.mean(set2)
# Plot the data
plt.hist(set1, alpha=0.5, label='Set 1')
plt.hist(set2, alpha=0.5, label='Set 2')
plt.legend()
plt.show()
# Perform a t-test
t, p = stats.ttest_ind(set1, set2)
# Print the results
if p < 0.05:
    print("The means of the two sets are significantly different.")
else:
    print("The means of the two sets are not significantly different.")
```

b) Plotting data read from a file

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize

# Load data from file
data = np.loadtxt('data.txt')

# Define function to fit data to
def linear_func(x, a, b):
    return a * x + b

# Perform curve fitting
params, _ = optimize.curve_fit(linear_func, data[:,0], data[:,1])

# Create plot
plt.plot(data[:,0], data[:,1], 'ro', label='Data')
plt.plot(data[:,0], linear_func(data[:,0], *params), 'b-', label='Fit')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```

c)Fitting a function through a set a data points using polyfit function

```
import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import curve_fit

x_data = np.array([1.0, 2.0, 3.0, 4.0, 5.0])

y_data = np.array([1.0, 3.0, 5.0, 7.0, 9.0])

def f(x, a, b):

    return a * x + b
```

```
popt, pcov = curve_fit(f, x_data, y_data)
plt.scatter(x_data, y_data)
plt.plot(x_data, f(x_data, *popt), 'r-', label='fit')
plt.legend()
plt.show()
```

(OR)

```
import numpy as np
from scipy import polyfit
import matplotlib.pyplot as plt

# Generate data points
x = np.linspace(0, 10, num=20)
y = np.sin(x)

# Add noise to data
noise = np.random.normal(0, 0.1, size=len(x))
y += noise
# Fit a polynomial function to the data
p = polyfit(x, y, deg=3)
# Evaluate the fitted function over a range of x values
x_fit = np.linspace(0, 10, num=100)
y_fit = np.polyval(p, x_fit)
```

```
# Plot the original data points and the fitted function
plt.plot(x, y, 'o', label='data')
plt.plot(x_fit, y_fit, label='fit')
plt.legend()
plt.show()
```

d)Plotting a histogram of a given data set

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
# Generate some random data
data = np.random.normal(0, 1, 1000)
# Calculate some descriptive statistics
mean = np.mean(data)
median = np.median(data)
mode = stats.mode(data)[0][0]
std_dev = np.std(data)
# Plot the histogram
plt.hist(data, bins=50, color='blue', alpha=0.5)
plt.axvline(mean, color='red', linestyle='dashed', linewidth=2, label='Mean')
plt.axvline(median, color='green', linestyle='dashed', linewidth=2, label='Median')
plt.axvline(mode, color='purple', linestyle='dashed', linewidth=2, label='Mode')
plt.legend()
plt.xlabel('Value')
```

```
plt.ylabel('Frequency')

plt.title(f'Histogram of Data (mean={mean:.2f}, median={median:.2f}, mode={mode:.2f}, std
dev={std_dev:.2f})')

plt.show()
```

6. The strings package

a) Read text from a file and print the number of lines, words and characters

```
import string

filename = 'example.txt'

with open(filename, 'r') as file:
    text = file.read()

num_lines = len(text.split('\n'))

num_words = len(text.split())

num_chars = len(text)

print(f'Number of lines: {num_lines}')

print(f'Number of words: {num_words}')

print(f'Number of characters: {num_chars}')
```

b)Read text from a file and return a list of all n letter words beginning with a vowel

```
import string

def get_vowel_words(filename, n):
    vowel_set = set(['a', 'e', 'i', 'o', 'u'])
    words = []

    with open(filename, 'r') as file:
        for line in file:
            for word in line.strip().split():
                # check if the word starts with a vowel and has length n
                if word[0].lower() in vowel_set and len(word) == n:
```

```
                words.append(word)

        return words

filename = 'example.txt'
n = 4
vowel_words = get_vowel_words(filename, n)
print(vowel_words)
```

c) Finding a secret message hidden in a paragraph of text

```
import string

# Define the paragraph of text to search for the secret message

text = "This is a paragraph of text with a secret message hidden within it."

# Define the secret message to search for

secret_message = "secret"

# Convert the text to lowercase for case-insensitive search

text = text.lower()

# Remove all punctuation from the text

text = text.translate(str.maketrans("", "", string.punctuation))

# Find the index of the secret message within the text

index = text.find(secret_message)

# Check if the secret message was found

if index != -1:

    print("The secret message was found at index", index)

else:

    print("The secret message was not found in the text.")
```

d) Plot a histogram of words according to their length from text read from a file

```
import matplotlib.pyplot as plt
# Read text from file
with open('filename.txt', 'r') as file:
    text = file.read()
# Split text into words
words = text.split()
# Calculate word lengths
word_lengths = [len(word) for word in words]
# Plot histogram
plt.hist(word_lengths, bins=range(min(word_lengths), max(word_lengths) + 2))
plt.title('Histogram of Word Lengths')
plt.xlabel('Word Length')
plt.ylabel('Frequency')
plt.show()
```

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING - DATA SCIENCE

Cycle -2

7. Installing OS on Raspberry Pi

a) Installation using PiImager

b) Installation using image file

- Downloading an Image
- Writing the image to an SD card
- using Linux
- using Windows
- Booting up
- Follow the instructions given in the URL https://www.raspberrypi.com/documentation/computers/getting-started.html

To install an operating system (OS) on a Raspberry Pi, there are two main methods: using the Pi Imager software or using an image file. Both methods involve downloading an OS image, writing it to an SD card, and booting up the Raspberry Pi.

a) Installation using PiImager:

1. Download and install Pi Imager software from the Raspberry Pi website.
2. Insert the SD card into your computer's card reader.
3. Open the Pi Imager software and select the OS image you want to install.
4. Select the SD card as the target drive.
5. Click "Write" and wait for the process to complete.
6. Insert the SD card into the Raspberry Pi and power it on.

b) Installation using an image file:

1. Download the OS image file from the Raspberry Pi website.
2. Insert the SD card into your computer's card reader.
3. Download and install an image writer software like Etcher, Win32DiskImager, or Rufus.
4. Open the image writer software and select the OS image file you want to install.
5. Select the SD card as the target drive.
6. Click "Write" and wait for the process to complete.
7. Insert the SD card into the Raspberry Pi and power it on.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING - DATA SCIENCE

Note: The process for writing an image to an SD card may vary depending on the operating system you are using. Below are the instructions for using Linux or Windows.

Using Linux:

1. Insert the SD card into your computer's card reader.
2. Open the terminal and enter the command "sudo fdisk -l" to list the available disks.
3. Identify the SD card's device name (e.g. /dev/sdc).
4. Unmount the SD card by entering the command "sudo umount /dev/sdc1" (replace "sdc1" with the correct device name).
5. Write the image to the SD card by entering the command "sudo dd bs=4M if=/path/to/image.img of=/dev/sdc" (replace "/path/to/image.img" and "sdc" with the correct paths and device name).
6. Wait for the process to complete and insert the SD card into the Raspberry Pi.

Using Windows:

1. Insert the SD card into your computer's card reader.
2. Download and install an image writer software like Etcher, Win32DiskImager, or Rufus.
3. Open the image writer software and select the OS image file you want to install.
4. Select the SD card as the target drive.
5. Click "Write" and wait for the process to complete.
6. Insert the SD card into the Raspberry Pi and power it on.

8. Accessing GPIO pins using Python

a) Installing GPIO Zero library. First, update your repositories list: sudo apt update Then install the package for Python 3: sudo apt install python3-gpiozero

To install the GPIO Zero library, you need to update your repositories list by running the command:

sudo apt update

Then, install the package for Python 3 by running the command:

sudo apt install python3-gpiozero

b) Blinking an LED connected to one of the GPIO pin

To blink an LED connected to one of the GPIO pins using GPIO Zero, you can use the following code:

```
from gpiozero import LED

from time import sleep

led = LED(17) # Replace 17 with the GPIO pin number you are using

while True:

    led.on()

    sleep(1)

    led.off()

    sleep(1)
```

This code will blink the LED connected to GPIO pin 17 with a 1 second interval.

a. Adjusting the brightness of an LED

To adjust the brightness of an LED connected to one of the GPIO pins using GPIO Zero, you can use the following code:

```
from gpiozero import PWMLED

from time import sleep
```

```
led = PWMLED(17) # Replace 17 with the GPIO pin number you are using
while True:
    for i in range(0, 101):
        led.value = i/100
        sleep(0.1)
    for i in range(100, -1, -1):
        led.value = i/100
        sleep(0.1)
```

This code will gradually increase and decrease the brightness of the LED connected to GPIO pin 17.

c) Adjust the brightness of an LED (0 to 100, where 100 means maximum brightness) using the in-built PWM wavelength.

To adjust the brightness of an LED (0 to 100, where 100 means maximum brightness) using the in-built PWM wavelength, you can use the following code:

```
from gpiozero import PWMLED
from time import sleep
led = PWMLED(17) # Replace 17 with the GPIO pin number you are using
while True:
    for i in range(0, 101):
        led.value = i/100
        sleep(0.1)
```

This code will gradually increase the brightness of the LED connected to GPIO pin 17 from 0% to 100%.

9. Collecting Sensor Data

a) DHT Sensor interface

◦ Connect the terminals of DHT GPIO pins of Raspberry Pi.

◦ Import the DHT library using import Adafruit_DHT

◦ Read sensor data and display it on screen.

To collect data from a DHT sensor using a Raspberry Pi, you can follow these steps:

1. Connect the DHT sensor to the Raspberry Pi's GPIO pins. The DHT sensor typically has three pins: VCC, GND, and DATA. Connect VCC to a 3.3V pin on the Raspberry Pi, GND to a ground pin, and DATA to a GPIO pin of your choice.
2. Install the Adafruit DHT library using the following command:

```
sudo pip3 install Adafruit_DHT
```

3. Import the library in your Python code using the following line:

```
import Adafruit_DHT
```

4. Read the sensor data using the **read_retry** method of the DHT library. This method takes two arguments: the sensor type (either **DHT11** or **DHT22**) and the GPIO pin number to which the sensor is connected. The method will retry up to 15 times if it fails to read the data the first time.

```
humidity, temperature = Adafruit_DHT.read_retry(sensor_type, gpio_pin)
```

5. Display the data on the screen using the **print** function.

```
print("Temperature: {:.1f}°C".format(temperature))

print("Humidity: {:.1f}%".format(humidity))
```

Here's an example code snippet that collects data from a DHT22 sensor connected to GPIO pin 4:

```
import Adafruit_DHT


sensor_type = Adafruit_DHT.DHT22

gpio_pin = 4


humidity, temperature = Adafruit_DHT.read_retry(sensor_type, gpio_pin)
```

```
if humidity is not None and temperature is not None:
    print("Temperature: {:.1f}°C".format(temperature))
    print("Humidity: {:.1f}%".format(humidity))
else:
    print("Failed to retrieve data from DHT sensor.")
```

Note: Make sure that you have the necessary permissions to access the GPIO pins on your Raspberry Pi.