# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belagavi-590014, Karnataka, India



Report

on

# “Biomedical waste, Hydrogen energy,

# Hotspot"

# Submitted in partial fulfillment of the 5th Semester requirements for the award of the degree

Bachelor of Engineering

In

Computer Science and Engineering

SUBMITTED BY

### NAME:

### USN:



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**IMPACT COLLEGE OF ENGINEERING AND APPLIED SCIENCES**

Sah**a**karnagar, Bengaluru – 560092

2022-2023

**IMPACT COLLEGE OF ENGINEERING AND APPLIED SCIENCES**

Sahakarnagar, Bengaluru – 560092



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that the work entitled **“Python Programming”** carried out by **Shaik Yaseen (1IC21CS042)** bonafied student at Impact College of Engineering and Applied Sciences has submitted in partial fulfillment of requirements in Internship work of 4th Semester **Bachelor of Engineering in Computer Science and Engineering/AI&ML/DS** as prescribed by **Visvesvaraya Technological University**, Belagavi during the academic year 2022-2023.

|  |  |  |
| --- | --- | --- |
|  |  | **Signature of HOD** |
|  |  | **Dr. DHANANJAYA V** |
|  |  |  |

**ACKNOWLEDGEMENT**

The successful completion of Internship in Python programming Language was made possible with the help and guidance received by us from the various quarters. We would like to avail this opportunity to express our sincere gratitude to all of them.

We thank the management and our Principal **Mr Khaleel** for providing us with the necessary infrastructure and good environment to nourish our knowledge.

We would like to sincerely thank **Dr. Dhananjaya V Professor & Head of Department Computer Science and Engineering** for his support and guidance.

We are grateful to our guide **Mr. Vijendra** for their keen interest and encouragement in our course, their guidance and cooperation helped us in nurturing the Internship.

We are also in debt to our parents and friends for their continued moral and material support throughout the course of the Internship

Our heartfelt thanks to all the people who have contributed to accomplish our course.

**NAME:**

# 

# CONTENTS

**Chapter no Description Page no.**

**01 Introduction 1**

**1.1 Biomedical waste**

**1.2 What are biomedical waste**

**02 Effects and Treatment of BMW 2**

**2.1 What are effect of BMW**

**2.2 Technologies for the treatment of BMW**

**03 Hydrogen Energy 3**

**3.1 Introduction to hydrogen energy**

**3.2 Resources to produce Hydrogen energy**

**04 Benefits and application of Hydrogen 4**

**Energy**

**4.1 Benefits of Hydrogen energy 4.2 Application of hydrogen energy in various**

**fields**

**05 Hotspots 6**

**5.1 Introduction to hotspots 5.2 Why biodiversity Hotspots are important 5.3 Criteria to qualify as Hotspots**

**5.4 Hotspots in India**

**Chapter 1**

**Introduction**

**1.1 Biomedical Waste**

Python is a high level general purpose programming language known known for its simplicity readability. It was created by Guido van Rossum and first released in 1991. Python’s design philosophy emphasizes code readability and clean, easy-to-understand syntax, which has contributed to its popularity among developers.

Python versatility makes it suitable for various applications, including web development, data analysis, scientific computing, artificial intelligence, and more. Its extensive standard library and a vibrant community of developers have contributed to its widespread adoption in both academia and industry.

**1.2 Printing the console using print()**

In Python, you can use the print() function to display the text or values in the console. Here’s the example

print(“Hello world)

Output: Hello world

**1.3 Arithmetic operators**

Arithmetic operators in Python are used to perform mathematical operations on numbers. The main arithmetic operators are

1.Addition (+)

2.Subtraction (-)

3.Multiplication (\*)

4.Division (/)

5.Modulas (%)

**1.4 Getting user input()**

In Python, you can use the input() function to get user input from the console. Here’s the example

User\_input = input(“Please enter something:”)

**Chapter 2**

**String splicing and Type casting**

**2.1 String splicing**

String splicing also known as the string slicing, allows you to extract a portion of a string in Python. You can specify a starting and ending index to slice a substring from a larger string.

Here’s the example

Text = “hello, world!”

Substring = text[:5]

print(Substring)

Output: hell

Here we use square brackets [] to access the string character.

**Syntax:**

Variable[staring index : end index:]

**2.2 String methods:**

1. .upper(), .lower() :

String.upper() : Converts all the characters of the string to upper case.

String.lower() : Converts all the characters of the string to lower case.

2. .replace():

String.replace(old, new) : replace a substring in a string with new string.

3. .count():

String.count(character or string) : count the occurrence of character or a string in the main string.

4. .title():

Capitalize the first character of each word in a string.

**2.3 Type casting**

Type casting is a fundamental concept in programming and refers to the process of converting a value from one data type to another. This often necessary in programming when you need to perform operations or assignments involving different data types

Type casting is of two types

1. Implicit Type casting:

Implicit type casting , also known as type coercion, is an automatic conversion performed by the programming language itself. It happens when u mix different data types in an operation, and the language tries to make them compactable.

Example: Adding an integer and a float, where the integer is implicitly cast to a float for the addition to occur.

1. Explicit Type casting:

Explicit type casting is a manual conversion where you specifically instruct the programming language to change the data type of the value.

Example: converting a integer to string using casting operators like str()

Different functions of type casting are:

int() : Converts the data type of a variable to Integer.

str(): converts the data type of a variable to String.

float(): Converts the data type of a variable to Float.

Bool(): Converts the data type of a variable to Boolean.

**Chapter 3**

**Comparison operators**

1. **Greater Than (>):** This operator checks if the value on the left is greater than the value on the right.
2. **Less Than (<):** This operator checks if the value on the left is less than the value on the right.
3. **Less Than or Equal To (<=):** This operator checks if the value on the left is less than or equal to the value on the right.
4. **Greater Than or Equal To (>=):** This operator checks if the value on the left is greater than or equal to the value on the right.
5. **Equality Operator (==):** This operator checks if x is equal to y.
6. **Inequality Operator (!=):** This operator checks if x is not equal to y.
7. **Logical AND (and):** This operator returns True if both x and y are True.
8. **Logical OR (or):** This operator returns True if at least one of x or y is True.
9. **Logical NOT (not):** This operator returns the opposite of the Boolean value of x.
10. **Bitwise AND (&):** This operator performs a bitwise AND operation between the binary representations of x and y.
11. **Bitwise OR (|):** This operator performs a bitwise OR operation between the binary representations of x and y.
12. **Bitwise XOR (^):** This operator performs a bitwise XOR (exclusive OR) operation between the binary representations of x and y.

**Chapter 4**

**Conditional Statements**

What are conditional statements?

Conditional statements, also known as control statements, are used to make decision in code. They allow a program to execute different actions based on whether a specified condition is true or false.

**4.1 Types of Conditional statements**

There are typically three common type of statements/

**if statement :** it execute a lock of code if condition is true.

**Syntax:**

if condition:

statements

**else statement:** it is used with if statement to execute when the condition is false

**Syntax:**

if condition:

statement 1

else :

statement 2

**elif statement:** Short for “else if,” it allow you to check multiple condition one by one

**Syntax:**

if condition:

Statement 1

elif condition:

Statement2

else:

Statement3

**4.2 Performing simple calculator by basic arithmetic operation**s

a = int(input("Enter First number: "))

b = input("Enter the operator")

c = int(input("Enter Second number: "))

if c == "+":

d = a+ b

print(d)

elif c == "-":

d = a- b

print(d)

elif c == "\*":

d = a\* b

print(d)

elif c == "/":

d = a/ b

print(d)

elif c == "%":

d = a% b

print(d)

else:

print(" Entered wrong choice: ")

These are the fundamental conditional statements in Python. They allow you to make decisions and control the flow of program based on various conditions.

**Chapter 5**

**Loops and Loop control statement**

**Loops**

In python loops are used to repeatedly execute a block of code.

In some cases we need to repeat the statement more then twice in this kind of cases loops are used.

**5.1 Types of loops**

Loops are of two types

1. For loop

A for loop is used to iterate over a sequence (such as list, tuple, string, or range) or any iterable object. It execute a block of code for each item in sequence.

Syntax:

For variable in sequence:

#code to execute

Example:

for i in range(0,3):

print(“hello world”)

Output: hello world

hello world

hello world

1. While loop

A while loop repeatedly executes a block of code as long as a specified condition is true. Its used when you don’t know beforehand how many times the loop should run.

Syntax:

while condition:

#statement to execute while the condition is true.

Exanple:

Count = 0

while count < 5:

print(Count)

count += 1

Output: 0

1

2

3

4

**5.2 Control Statements:**

Loop control statements in Python are used to modify the behavior of loops (such as for and while loops) during their execution. They allow you to control when to exit a loop prematurely, skip the current iteration, or force the loop to continue to the next iteration.

1. break Statement:

The break statement is used to exit a loop prematurely, even if the loop condition is still True. It is typically used to terminate the loop when a certain condition is met.

Example:

i = 0

while(1):

if I == 0:

break;

1. continue Statement:

The continue statement is used to skip the rest of the current iteration of a loop and move to the next iteration. It is typically used to skip certain iterations based on a condition.

Example:

i = 0

while(1):

if I == 0:

break;

else:

continue;

1. pass Statement:

The pass statement is a placeholder statement that does nothing. It is often used when syntactically required but no action is needed. It can be used in loops or conditional statements.

Example:

key = “Python”

If key == “Python”:

Pass

**Chapter 9**

**File Handling**

**Introduction:**

File handling is a cornerstone of computer programming, and in the world of Python, it's a skill that every developer must master. Files are vessels that store data, and the ability to read, write, and manipulate this data is essential for creating versatile and practical software applications. In this essay, we will embark on a journey through the realm of file handling in Python, exploring the principles, methods, and best practices that make it such a valuable skill.

**9.1 Understanding Files and File Modes:**

In Python, a file is a named location on disk that stores data. Files can be of various types, including text files, binary files, and more. To interact with a file, we use the 'open()' function, specifying both the file name and the mode in which we intend to operate on the file. Common file modes include:

1. 'r' (Read Mode):

Opens the file for reading.

Raises an error if the file does not exist.

2. 'w' (Write Mode):

Opens the file for writing.

If the file exists, it truncates its content (deletes everything).

If the file does not exist, it creates a new one.

3. 'a' (Append Mode):

Opens the file for writing, but appends new data to the end.

If the file does not exist, it creates a new file.

'b' (Binary Mode):

Used in conjunction with other modes for working with binary files (e.g., 'rb' for reading binary).

**9.2 Opening and Closing Files:**

Before you can read from or write to a file, you must open it using the 'open()' function. Once you are done with the file, it's crucial to close it using the 'close()' method. Failing to close files can lead to resource leaks and potential data corruption.

file = open("example.txt", "r")

# Perform file operations here

file.close()

Reading Data from Files:

Python offers various methods to read data from files:

'read()': Reads the entire content of the file as a string.

'readline()': Reads a single line from the file.

'readlines()': Reads all lines of the file into a list of strings.

file = open("example.txt", "r")

content = file.read()

file.close()

**Syntax**

open(name of the file,mode)

"x" to open empty text file

"w" to write in text file

"r" to read from text file

"a" to append in text file

"r+" to read and write

"w+" to write and read

close()

**9.3 Different types of Files**

**1.Text File Handling:**

Reading Text Files ('r' Mode): Involves opening and reading plain text files line by line or as a whole.

Writing Text Files ('w' or 'a' Mode): Involves creating, writing, or appending text data to text files.

with open("example.txt", "r") as file:

content = file.read()

**2.Binary File Handling:**

Reading Binary Files ('rb' Mode): Involves reading binary data from files, such as images or executable files.

Writing Binary Files ('wb' or 'ab' Mode): Involves creating, writing, or appending binary data to files.

with open("image.jpg", "rb") as binary\_file:

image\_data = binary\_file.read()

Example code:

f=open("file.text","r")

count=0

for i in f:

print(i)

count=count+1

if count==3:

break

file handling is a fundamental skill in programming and data management. It's essential to understand the principles, techniques, and best practices associated with file operations to work effectively with data in various applications.

**Chapter 10**

**Plotting Graphs**

**Introduction:**

Data visualization is a crucial component of data analysis and presentation. It enables us to communicate insights and patterns effectively, making complex data more accessible and understandable. Python, a versatile and widely used programming language, offers several libraries for creating visualizations, including bar graphs and pie charts. In this essay, we will explore the process of installing and using libraries for bar graphs and pie charts in Python.

**10.1 Installing Required Libraries:**

To create bar graphs and pie charts in Python, we commonly use two popular libraries: Matplotlib and Seaborn. We'll walk through the installation process for both of these libraries.

**Matplotlib:**

Matplotlib is a widely used library for creating static, animated, and interactive visualizations in Python. To install Matplotlib, you can use the Python package manager, pip, by running the following command:

pip install matplotlib

**10.2 Plotting a Graph**

import matplotlib.pyplot as plt

days = [1,2,3,4,5,6,7]

temp = [35,30,38,33,40,42,32]

plt.plot(days,temp)

Output:

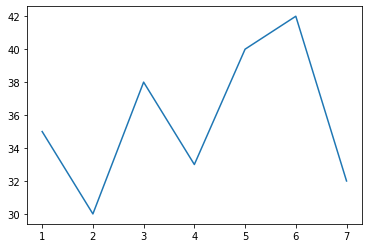


Figure 10.1 plotting a graph

**10.3 Plotting a Pie Chart:**

To create a pie chart, we can use Matplotlib's plt.pie() function

Example:

import matplotlib.pyplot as plt

temp=[35,30,38,33,40,42,32]

plt.pie(temp)

plt.show()

Output:

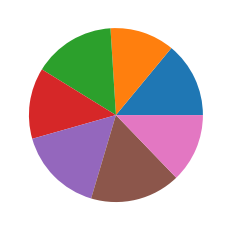


Figure 10.2 Plotting pie chart

**10.4 Plotting a Bar Graph:**

To create a bar graph, we can use Matplotlib's plt.bar() function.

Example:

import matplotlib.pyplot as plt

days=[1,2,3,4,5,6,7]

temp=[35,30,38,33,40,42,32]

plt.title("weather monitering")

plt.xlabel("days")

plt.ylabel("temp")

plt.bar(days,temp)

plt.show()

Output:

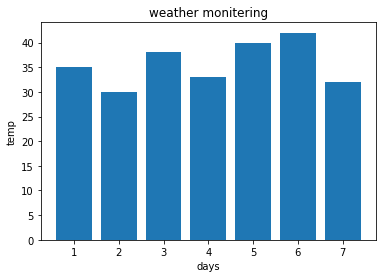


Figure 10.3 Plotting a bar graph

**Chapter 11**

**Graphical User Interface(GUI)**

Creating a complete GUI application using PyQt5 involves designing the user interface (UI) and adding functionality to the UI components.

**11.1 Steps involved in application making**

Make sure you have PyQt5 installed. You can install it using pip:

pip install PyQt5

Step 1: Import Required Modules

Step 2: Create the Application and Main Window

Step 3: Create UI Components (Button and Label)

Step 4: Create a Layout for UI Components

Step 5: Create a Function to Handle Button Clicks

Step 6: Set the Layout for the Main Window

Step 7: Run the Application.

**11.2 Opening the QT Designer**

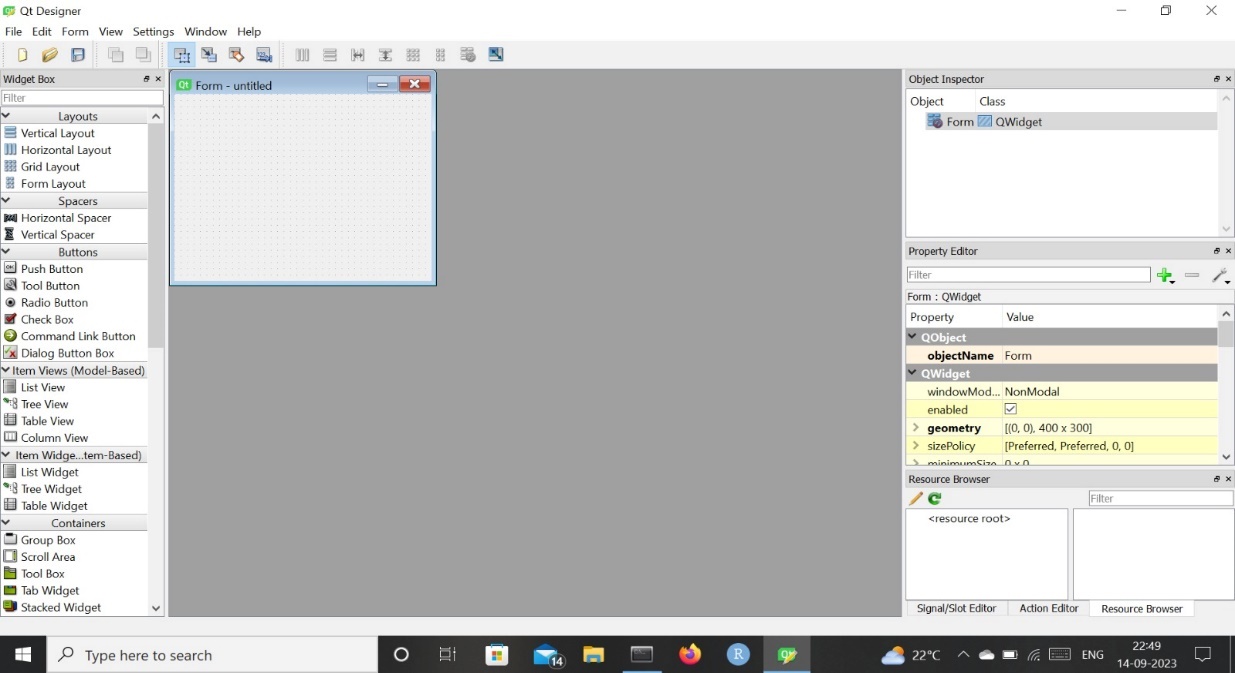


Figure 11.1 Opening QT designer

**11.3 A simple GUI application:**

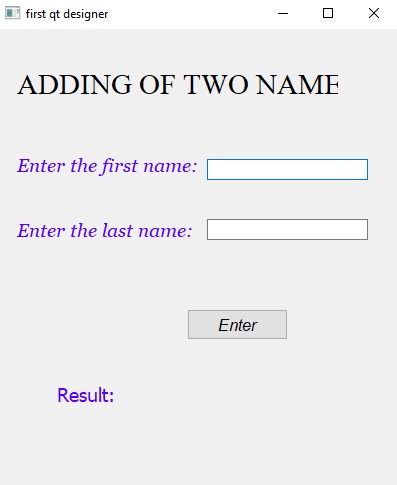
****

Figure 11.2 A simple GUI application