|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column 1 | Column2 | Column 3 | Column 4 | Column 5 |
| 10 | 50 | 30 | 40 | 35 |
| 20 | 30 | 10 | 20 | 25 |
| 30 | 20 | 35 | 10 | 10 |
| 25 | 15 | 15 | 15 | 30 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| 12 | 56 | 90 | 45 | 89 |
| 23 | 67 | 12 | 56 | 90 |
| 34 | 78 | 23 | 67 | 12 |
| 45 | 89 | 34 | 78 | 23 |

**Name :** Md.Mahfuzur Rahman

**Edge no:** BPP01-9

Project Idea: The **Water Quality Index (WQI)** is a numerical tool used to assess the overall health of a water body based on various water quality parameters. It simplifies complex water data into a single score, typically ranging from 0 to 100, where higher values indicate better water quality. Common parameters included in WQI calculations are pH, dissolved oxygen, turbidity, biochemical oxygen demand (BOD), and contaminants like fecal coliforms. Different WQI methods may focus on specific uses, such as drinking water, recreational water, or aquatic ecosystems. The WQI helps identify pollution levels, monitor water quality trends, and guide decision-making in water management. A higher WQI score represents safe, clean water, while a lower score indicates pollution or degradation. The index aids in public health decisions, environmental monitoring, and policy development. However, WQI can oversimplify complex water conditions and may miss specific localized issues. Overall, it serves as an important tool for understanding and managing water resources.

Project:

def sub\_index\_ph(ph):

""" Calculate sub-index for pH. """

if 6.5 <= ph <= 8.5:

return 100 # Excellent

elif 6.0 <= ph < 6.5 or 8.5 < ph <= 9.0:

return 80 # Good

else:

return 50 # Poor

def sub\_index\_do(do):

""" Calculate sub-index for Dissolved Oxygen (DO). """

if do >= 8:

return 100 # Excellent

elif 5 <= do < 8:

return 75 # Good

else:

return 50 # Poor

def sub\_index\_turbidity(turbidity):

""" Calculate sub-index for Turbidity. """

if turbidity <= 5:

return 100 # Excellent

elif 5 < turbidity <= 10:

return 75 # Good

else:

return 50 # Poor

def sub\_index\_tds(tds):

""" Calculate sub-index for Total Dissolved Solids (TDS). """

if tds <= 300:

return 100 # Excellent

elif 300 < tds <= 600:

return 75 # Good

else:

return 50 # Poor

def calculate\_wqi(ph, do, turbidity, tds):

""" Calculate the overall Water Quality Index (WQI). """

score\_ph = sub\_index\_ph(ph)

score\_do = sub\_index\_do(do)

score\_turbidity = sub\_index\_turbidity(turbidity)

score\_tds = sub\_index\_tds(tds)

# Weighting the scores for different parameters (optional)

wqi = (0.25 \* score\_ph) + (0.25 \* score\_do) + (0.25 \* score\_turbidity) + (0.25 \* score\_tds)

return wqi

# Function to process list input

def input\_list(prompt):

"""Take a comma-separated string and return a list of floats."""

user\_input = input(prompt) # Get input as string

user\_list = user\_input.split(',') # Split the string into a list by commas

user\_list = [float(item) for item in user\_list] # Convert each item to float

return user\_list

# Example of how you can input lists manually:

try:

# Input lists for each parameter

ph\_values = input\_list("Enter pH values (comma-separated): ")

do\_values = input\_list("Enter Dissolved Oxygen (mg/L) values (comma-separated): ")

turbidity\_values = input\_list("Enter Turbidity (NTU) values (comma-separated): ")

tds\_values = input\_list("Enter Total Dissolved Solids (mg/L) values (comma-separated): ")

# Calculate WQI for each set of values

wqi\_values = []

for i in range(len(ph\_values)):

wqi = calculate\_wqi(ph\_values[i], do\_values[i], turbidity\_values[i], tds\_values[i])

wqi\_values.append(wqi)

# Print the calculated WQI for each input set

for i, wqi in enumerate(wqi\_values):

print(f"Calculated Water Quality Index (WQI) for sample {i + 1}: {wqi}")

# WQI Interpretation for each sample

for i, wqi in enumerate(wqi\_values):

if wqi >= 80:

print(f"Sample {i + 1}: Water quality is Excellent.")

elif 60 <= wqi < 80:

print(f"Sample {i + 1}: Water quality is Good.")

elif 40 <= wqi < 60:

print(f"Sample {i + 1}: Water quality is Fair.")

else:

print(f"Sample {i + 1}: Water quality is Poor.")

except ValueError:

print("Please enter valid numeric values for all inputs.")

Input/output :

