**NAME:** MD. AZIZUL ISLAM

**EDGE ID:** *16*

**INFORMARION:**

**Azizul Islam**  
*Department of Physics, Mawlana Bhashani Science and Technology University*

I am a student in the Department of Physics at Mawlana Bhashani Science and Technology University. I am currently learning Python programming as part of a basic course to enhance my skills in scientific computing and problem-solving.

**PROJECT IDEA:**

This project is a **Radiation Health Effect Warning System** developed using Tkinter, which helps users assess the health risks associated with radiation exposure. The application allows users to input radiation values in units like Roentgen, Curie, Gray, or Sievert, and convert them to Sieverts for accurate assessment. Based on the input, the system provides health warnings, such as mild or severe radiation sickness, and recommends protective actions like minimizing exposure time, using shielding, and seeking medical attention. Additionally, the app compares the calculated dose with annual radiation limits for both occupational and public safety, ensuring that users can monitor their exposure levels and take appropriate measures to protect their health.

|  |
| --- |
| **PROJECT:**  import tkinter as tk from tkinter import messagebox  # Constants for annual radiation dose limits (in Sievert) ANNUAL\_DOSE\_LIMIT\_OCCUPATIONAL = **50** / **1000** # 50 mSv for occupational exposure (converted to Sv) ANNUAL\_DOSE\_LIMIT\_PUBLIC = **1** / **1000** # 1 mSv for the public (converted to Sv)  def convert\_to\_sievert(value**,** unit**,** radiation\_type):  *"""  Converts different radiation units (Roentgen, Curie, Gray) to Sievert, adjusted for radiation type.  """* if unit == "R":  if radiation\_type == "Gamma":  return value \* **0.01** # 1 Roentgen ≈ 0.01 Sievert for Gamma radiation  elif radiation\_type == "Alpha":  return value \* **0.02** # 1 Roentgen ≈ 0.02 Sievert for Alpha radiation  elif radiation\_type == "Beta":  return value \* **0.015** # 1 Roentgen ≈ 0.015 Sievert for Beta radiation  elif unit == "Ci":  if radiation\_type == "Gamma":  return value \* **37** # 1 Curie ≈ 37 Sievert for Gamma radiation  elif radiation\_type == "Alpha":  return value \* **74** # Alpha radiation is more harmful  elif radiation\_type == "Beta":  return value \* **45** # Beta radiation is intermediate  elif unit == "Gy":  return value # 1 Gray = 1 Sievert (assuming it’s gamma radiation, for simplicity)  elif unit == "Sv":  return value # No conversion needed if already in Sievert  else:  raise ValueError("Unknown unit. Please use R (Roentgen), Ci (Curie), Gy (Gray), or Sv (Sievert).")   def get\_radiation\_health\_effects(dose):  *"""  Returns health effects based on the radiation dose in Sievert (Sv).  """* if dose < **0.1**:  return ("No immediate health effects. Long-term exposure may slightly increase the risk of cancer."**,** "Limit exposure and monitor over time. Regular check-ups advised.")  elif **0.1** <= dose < **1**:  return ("Increased risk of cancer with long-term exposure. Acute symptoms unlikely."**,** "Minimize exposure, wear protective clothing, and stay indoors during high radiation events.")  elif **1** <= dose < **2**:  return ("Mild radiation sickness possible, including nausea and fatigue. Higher cancer risk."**,** "Seek medical attention for any symptoms. Use shielding and reduce exposure duration.")  elif **2** <= dose < **6**:  return ("Moderate to severe radiation sickness. Potential damage to internal organs, bone marrow damage."**,** "Immediate medical treatment required. Stay indoors, avoid contaminated areas, and use protective gear.")  elif **6** <= dose < **10**:  return ("Severe radiation sickness. Death is likely without medical intervention."**,** "Emergency medical attention required. Full protection and decontamination necessary.")  else:  return ("Extremely high radiation dose. Death is almost certain within days or weeks."**,** "Immediate evacuation and emergency medical intervention required. Avoid exposure at all costs.")   def radiation\_protection\_solutions():  *"""  Returns general protection solutions for radiation exposure.  """* solutions = {  "Time": "Minimize the time spent in radiation-exposed areas."**,** "Distance": "Increase distance from the radiation source to reduce exposure."**,** "Shielding": "Use protective barriers (lead, concrete, or water) to block radiation."**,** "Monitoring": "Regularly monitor radiation levels in areas where exposure is possible."**,** "Medical Check-ups": "Schedule frequent medical check-ups if exposed to ionizing radiation over time."  }  return solutions  def calculate\_effective\_dose(dose**,** radiation\_type):  *"""  Calculates the effective dose for human health based on radiation type.  """* # Adjust the effective dose based on radiation type (Gamma, Alpha, Beta)  if radiation\_type == "Gamma":  effective\_dose = dose \* **1.0** # Effective dose for gamma radiation  elif radiation\_type == "Alpha":  effective\_dose = dose \* **20** # Alpha radiation is much more harmful internally  elif radiation\_type == "Beta":  effective\_dose = dose \* **10** # Beta radiation is intermediate in its effects  else:  effective\_dose = dose # Default to no adjustment   return effective\_dose  # Tkinter GUI setup class RadiationWarningApp:  def \_\_init\_\_(self**,** aziz):  self.root = aziz  self.root.title("Radiation Health Effect Warning System")  self.root.geometry("600x650")   self.title\_label = tk.Label(aziz**,** text="Radiation Health Effect Warning System"**,** font=("Arial"**, 20,** "bold"))  self.title\_label.pack(pady=**5**)   # Input Fields  self.value\_label = tk.Label(aziz**,** text="Enter the radiation exposure value:")  self.value\_label.pack()   self.value\_entry = tk.Entry(aziz)  self.value\_entry.pack(pady=**5**)   self.radiation\_type\_label = tk.Label(aziz**,** text="Select Radiation Type (Gamma, Alpha, Beta):")  self.radiation\_type\_label.pack()   self.radiation\_type = tk.StringVar()  self.radiation\_type.set("Gamma") # Default to Gamma   self.radiation\_type\_menu = tk.OptionMenu(aziz**,** self.radiation\_type**,** "Gamma"**,** "Alpha"**,** "Beta")  self.radiation\_type\_menu.pack(pady=**5**)   # Frame for Radio Buttons (Unit Selection)  self.unit\_frame = tk.Frame(aziz)  self.unit\_frame.pack(pady=**5**)   self.selected\_unit = tk.StringVar()  self.selected\_unit.set("R") # Default to Roentgen   # Unit Selection Buttons (in one line)  self.roentgen\_button = tk.Radiobutton(self.unit\_frame**,** text="Roentgen (R)"**,** variable=self.selected\_unit**,** value="R")  self.roentgen\_button.pack(side=tk.LEFT**,** padx=**5**)   self.curie\_button = tk.Radiobutton(self.unit\_frame**,** text="Curie (Ci)"**,** variable=self.selected\_unit**,** value="Ci")  self.curie\_button.pack(side=tk.LEFT**,** padx=**5**)   self.gray\_button = tk.Radiobutton(self.unit\_frame**,** text="Gray (Gy)"**,** variable=self.selected\_unit**,** value="Gy")  self.gray\_button.pack(side=tk.LEFT**,** padx=**5**)   self.sievert\_button = tk.Radiobutton(self.unit\_frame**,** text="Sievert (Sv)"**,** variable=self.selected\_unit**,** value="Sv")  self.sievert\_button.pack(side=tk.LEFT**,** padx=**5**)   # Submit Button (Green Color)  self.submit\_button = tk.Button(aziz**,** text="Submit"**,** command=self.display\_results**,** bg="green"**,** fg="white"**,** font=("Arial"**, 12,** "bold"))  self.submit\_button.pack(pady=**5**)   # Output Labels  self.output\_label = tk.Label(aziz**,** text="Results will be displayed here."**,** justify="left")  self.output\_label.pack(pady=**5**)   # Health Warning Section  self.health\_warning\_frame = tk.Frame(aziz**,** bd=**2,** relief="solid"**,** padx=**5,** pady=**5**)  self.health\_warning\_frame.pack(pady=**5,** fill="both")   self.health\_warning\_label = tk.Label(self.health\_warning\_frame**,** text=" Health Warning "**,** font=("Arial"**, 15,** "bold")**,** fg="red"**,** anchor="center")  self.health\_warning\_label.pack(fill="both")   self.health\_warning\_output = tk.Label(self.health\_warning\_frame**,** text=""**,** justify="left"**,** anchor="center")  self.health\_warning\_output.pack()   # Recommended Action Section  self.recommended\_action\_frame = tk.Frame(aziz**,** bd=**2,** relief="solid"**,** padx=**5,** pady=**5**)  self.recommended\_action\_frame.pack(pady=**5,** fill="both")   self.recommended\_action\_label = tk.Label(self.recommended\_action\_frame**,** text=" Recommended Action "**,** font=("Arial"**, 15,** "bold")**,** anchor="center")  self.recommended\_action\_label.pack(fill="both")   self.recommended\_action\_output = tk.Label(self.recommended\_action\_frame**,** text=""**,** justify="left"**,** anchor="center")  self.recommended\_action\_output.pack()   # General Protection Section  self.protection\_solutions\_frame = tk.Frame(aziz**,** bd=**2,** relief="solid"**,** padx=**5,** pady=**5**)  self.protection\_solutions\_frame.pack(pady=**5,** fill="both")   self.protection\_solutions\_label = tk.Label(self.protection\_solutions\_frame**,** text=" General Protection Solutions "**,** font=("Arial"**, 15,** "bold")**,** fg="green"**,** anchor="center")  self.protection\_solutions\_label.pack(fill="both")   self.protection\_solutions\_output = tk.Label(self.protection\_solutions\_frame**,** text=""**,** anchor="center")  self.protection\_solutions\_output.pack()   # Annual Dose Limit Information  self.annual\_dose\_frame = tk.Frame(aziz**,** bd=**2,** relief="solid"**,** padx=**5,** pady=**5**)  self.annual\_dose\_frame.pack(pady=**5,** fill="both")   self.annual\_dose\_label = tk.Label(self.annual\_dose\_frame**,** text=" Annual Dose Limit "**,** font=("Arial"**, 12,** "bold")**,** anchor="center")  self.annual\_dose\_label.pack(fill="both")   self.annual\_dose\_output = tk.Label(self.annual\_dose\_frame**,** text=""**,** justify="left"**,** anchor="center")  self.annual\_dose\_output.pack()   # sign Labels  self.output\_label = tk.Label(aziz**,** text="@Aziz"**,** justify="left"**,** fg="blue")  self.output\_label.pack(padx=**15,**anchor="ne"**,**side="top")   def display\_results(self):  try:  value = float(self.value\_entry.get())  unit = self.selected\_unit.get()  radiation\_type = self.radiation\_type.get()   # Convert the input value to Sievert based on selected radiation type and unit  dose\_in\_sievert = convert\_to\_sievert(value**,** unit**,** radiation\_type)   # Get health effects and recommendations based on the dose  health\_warning**,** health\_solution = get\_radiation\_health\_effects(dose\_in\_sievert)   # Display health warning, action, and solutions  self.health\_warning\_output.config(text=health\_warning)  self.recommended\_action\_output.config(text=health\_solution)  self.protection\_solutions\_output.config(  text="\n".join([f"{key}: {value}" for key**,** value in radiation\_protection\_solutions().items()]))   # Compare with annual dose limit  if dose\_in\_sievert > ANNUAL\_DOSE\_LIMIT\_OCCUPATIONAL:  self.annual\_dose\_output.config(  text=f"Warning: Your dose exceeds the occupational annual dose limit ({ANNUAL\_DOSE\_LIMIT\_OCCUPATIONAL \* **1000**} mSv).")  elif dose\_in\_sievert > ANNUAL\_DOSE\_LIMIT\_PUBLIC:  self.annual\_dose\_output.config(  text=f"Warning: Your dose exceeds the public annual dose limit ({ANNUAL\_DOSE\_LIMIT\_PUBLIC \* **1000**} mSv).")  else:  self.annual\_dose\_output.config(text="Your dose is within safe limits.")   except ValueError:  messagebox.showerror("Invalid Input"**,** "Please enter a valid number for the radiation dose.")  # Create the Tkinter window root = tk.Tk() app = RadiationWarningApp(root) root.mainloop() |

**INPUT/OUTPUT:**

|  |  |
| --- | --- |
| INPUT | OUTPUT |