Last Groups code plus interprep:

<https://github.com/EmRobotics/RadRobo>

Activity\_Analysis

1. [**Spectrum\_analysis.py**](https://github.com/EmRobotics/RadRobo/blob/main/Activity_Analysis/Spectrum_analysis.py)
2. [**Activity\_analysis.py**](https://github.com/EmRobotics/RadRobo/blob/main/Activity_Analysis/activity_analysis.py)
3. [**Calibrate\_with\_sources.py**](https://github.com/EmRobotics/RadRobo/blob/main/Activity_Analysis/calibrate_with_sources.py)
4. [**eff.CVS**](https://github.com/EmRobotics/RadRobo/blob/main/Activity_Analysis/eff.CVS)

Summaries:

1: Python script for spectrum analysis of data from a .n42 file, file format storing radiation measurement data.

* Imports:
  + List, Tuple: Importing types for type hints.
  + numpy (np alias): Used for numerical operations.
  + matplotlib.pyplot (plt alias): Used for plotting.
  + xml.etree.ElementTree (ET alias): Used for parsing XML data.
  + pdb: The Python Debugger, used for debugging.
  + os: Provides a way to use operating system-dependent functionality.
  + glob: Helps in searching for files matching a specified pattern.
  + scipy.signal: Importing the find\_peaks and savgol\_filter functions.
* Isotopes Information Loading:
  + Reads information about isotopes from a file (isotopes\_xray.txt) and stores it in a dictionary (isotopes).
* Class Definitions:
  + N42: Represents information about a .n42 measurement file, including real-time, live-time, and spectrum data.
  + Spectrum: Represents spectrum information such as identified peaks, isotope guesses, and methods for analyzing the spectrum.
  + Peak: Represents individual peak information, including energy, background count, total count, estimated count, etc.
* File Path and Isotope Information Path:
  + file\_path: Variable storing the path to the .n42 file.
  + isotopes\_xray\_path: Variable storing the path to the isotopes information file.
* Reading Isotope Information:
  + Reads information about isotopes from isotopes\_xray.txt and stores it in the isotopes dictionary.
* Parsing .n42 File:
  + The N42 class is used to parse the header and data from the .n42 file, storing relevant information.
* Parsing Isotope X-rays Information:
  + The parse\_isotope\_xrays function reads information about isotope x-rays from the isotopes\_xray\_path file.
* Spectrum Analysis:
  + The Spectrum class performs analysis on the spectrum data, including finding peaks, calculating FWHM, estimating background counts, and plotting the spectrum.
* Peak Information Calculation:
  + The Peak class is responsible for calculating FWHM, estimating peak counts, and printing peak information.
* Height Threshold and Isotope Guessing:
* The find\_height function calculates a height threshold based on the derivative of counts.
* The determine\_isotope function attempts to guess the isotope based on detected peaks and their characteristics.
* Plotting Spectrum:
* The plot\_spectrum method in the Spectrum class generates a plot of the spectrum, including detected peaks, background, and clean peak counts.

2:driver for activity analysis based on radiation spectrum data.

Imports:

* + json: Used for JSON data processing.
  + datetime: Provides functionalities for working with dates and times.
  + numpy (np alias): Used for numerical operations.
  + Spectrum\_analysis as sp: Imports a module or script (Spectrum\_analysis.py) for spectrum analysis.
  + csv: Handles CSV file operations.
  + glob: Assists in finding all pathnames matching a specified pattern.
  + os: Provides a way to use operating system-dependent functionality.
  + pandas as pd: Used for data manipulation and analysis.
* Efficiency Lookup Table:
  + Reads efficiency data from a CSV file (efficiency.csv) into a Pandas DataFrame.
  + Creates a lookup table for efficiency based on energy and angle.
* Interpolated Efficiency Function:
  + Defines a function (interpolated\_efficiency) to interpolate efficiency based on angle and energy.
  + Interpolates efficiency values by considering the closest angles and energies from the lookup table.
* Main Script:
  + Reads a .n42 file and instantiates objects for spectrum analysis using the sp.N42 and sp.Spectrum classes.
  + Calls the find\_peaks method to analyze the spectrum and identify peaks.
* Activity Analysis Loop:
  + For each identified peak in the spectrum:
    - Retrieves information such as energy, counts, and name from the peak.
    - Calculates interpolated efficiency using the defined function.
    - Computes solid angle, detected counts, and emitted counts.
    - Estimates current activity of the radiation source based on emitted counts, live time, and branching ratio.
* Output:
  + Prints the detected counts, emitted counts, and estimated activity for each identified peak.

3: calculate the efficiency of a radiation detector based on a folder containing .n42 files.

Imports:

* + json: Used for JSON data processing.
  + datetime: Provides functionalities for working with dates and times.
  + numpy (np alias): Used for numerical operations.
  + Spectrum\_analysis as sp: Imports a module or script (Spectrum\_analysis.py) for spectrum analysis.
  + csv: Handles CSV file operations.
  + glob: Assists in finding all pathnames matching a specified pattern.
  + os: Provides a way to use operating system-dependent functionality.
* File Paths and Data Initialization:
  + isotopes\_xray\_path: Path to the "isotope\_xray.txt" file.
  + source\_info\_path: Path to a JSON file containing source information.
  + path: Path to the folder containing .n42 files.
  + output\_file\_path: Path to the output CSV file for efficiency data.
* Functions:
  + parse\_json: Parses a JSON file and returns the data.
  + parse\_isotope\_xrays: Parses the "isotope\_xray.txt" file and returns relevant data.
  + write\_efficiencies\_to\_csv: Writes efficiency data to a CSV file.
  + calculate\_efficiency\_for\_spectrum: Calculates efficiency for each peak in a spectrum.
  + interpolated\_efficiency: Interpolates efficiency based on angle and energy.
* Efficiency Calculation Loop:
  + Loops through each .n42 file in the specified folder.
  + Determines the detector face angle from the filename.
  + Parses the .n42 file, finds peaks, and calculates efficiency for each peak.
  + Writes the efficiency data to the output CSV file.
* Output:
  + Prints the calculated intrinsic efficiency for each energy peak.
  + Writes the energy, efficiency, and degree data to a CSV file.

4: N/A

ComptonImaging:

1. [**Bif\_to\_array.py**](https://github.com/EmRobotics/RadRobo/blob/main/comptonimaging/bif_to_array.py)
2. [**Rad\_data\_bif\_comptoncone\_map.py**](https://github.com/EmRobotics/RadRobo/blob/main/comptonimaging/rad_data_bif_comptoncone_map.py)

1:performing operations on data obtained from a Compton camera. process the data, extract relevant information, and then create a matrix map for visualization.

main steps:

* Reading Compton Camera Data:
  + Importing necessary libraries.
  + Reading Compton camera data from three different files using the BIFReader class.
* Data Processing:
  + Collapsing polar bins into one single bin per azimuthal direction.
  + Normalizing the data and plotting the resulting spectrum.
  + Extracting x and y values from the plot.
* Creating Matrix Map:
  + Creating a 20x20 matrix map and populating it with values from the spectrum data at specific degrees.
  + Performing this operation for multiple degrees to create a directional matrix map.
* Visualization:
  + Visualizing the matrix map using plt.matshow().
* Matrix Operations:
  + Rotating the matrix map and performing addition operations between matrices.
* Visualization of Updated Matrix:
  + Displaying the updated matrix after rotations and additions.
* Future Work:
  + There are commented-out sections that seem to be part of future work, such as creating a 3D map.

2: check this one:

Counter Imaging: is a matlab code

IsotopeID:

1. [**Isotope\_algorithm.py**](https://github.com/EmRobotics/RadRobo/blob/main/isotopeid/isotope_algorithm.py)
2. [**Isotopes\_xray.txt**](https://github.com/EmRobotics/RadRobo/blob/main/isotopeid/isotopes_xray.txt)
3. [**Peakfinding\_restructured.py**](https://github.com/EmRobotics/RadRobo/blob/main/isotopeid/peakfinding_restructured.py)
4. [**Xmlwrapper.py**](https://github.com/EmRobotics/RadRobo/blob/main/isotopeid/xmlwrapper.py)

1: isotope id in radiation spectra data, including parsing XML files, finding peaks, and determining isotopes based on a provided isotope database.

* Import Libraries:
  + Imports necessary libraries for file handling, XML parsing, plotting, and numerical operations.
* Spectra Class:
  + Defines a class to encapsulate spectrum information, including file name, uncompressed counts, and derived parameters.
* Functions for Zeros Expansion and Height Threshold:
  + expand\_zeros: Expands zeros in compressed spectra.
  + find\_height: Determines a height threshold based on counts.
* Function for Determining Isotope Name:
  + Determines isotopes given the name and adjusts the name according to the isotope-number naming convention.
* Function for Isotope Peaks:
  + isotope\_peaks: Given an isotope name, finds the true isotope peaks and their probabilities.
* Function for Scoring Isotope Identification:
  + score\_isotope: Scores isotope identification based on the guess and true names.
* Function for X-ray List and Isotope Guess:
  + xray\_list: Creates a list of x-ray peaks from the isotope database.
  + determine\_isotope: Guesses the isotope based on the found peaks.
* Function for Plotting Spectra:
  + plot\_spectra: Plots the spectra with identified peaks.
* Main Isotope Identification Function:
  + Initializes a Spectra object, finds peaks, and identifies isotopes.
* Main Part - Read Isotope Database and Process Data:
  + Reads the isotope database, specifies the file path, checks the file type, and loops through measurements in the file.
* File Correction and XML Parsing:
  + Checks the file type, corrects the file format for XML parsing, and reads in the XML file.
* Visualization and Isotope Identification:
  + Loops through measurements in the file, performs isotope identification, and visualizes the results.

* Import Libraries:
  + import os: Provides a way to interact with the operating system.
  + import xmlwrapper: A custom module for reading XML files.
  + import matplotlib.pyplot as plt: Used for creating plots.
  + import numpy as np: Provides support for arrays and mathematical functions.
  + from scipy.signal import find\_peaks, savgol\_filter: Functions for finding peaks in a signal and applying a Savitzky–Golay filter.
* Define the Spectra Class:
  + Represents a spectrum with methods for initializing and processing the spectrum.
  + Reads a file, counts, and a compression flag.
  + Processes the counts, extracts relevant parameters, and calculates a height threshold.
* Function expand\_zeros:
  + Expands compressed spectra by replacing zero counts with appropriate repetitions.
  + Used in the Spectra class for decompressing zero counts.
* Function find\_height:
  + Determines a height threshold for the given spectrum counts.
  + Uses standard deviation and a Savitzky–Golay filter to filter the counts.
  + Returns a height array based on certain conditions.
* Function determine\_name:
  + Modifies isotope names for consistency and splits them into a list.
  + Used in the isotope\_peaks function.
* Function isotope\_peaks:
  + Takes an isotope name and an isotope dictionary.
  + Maps isotope names to peaks and probabilities using the dictionary.
  + Returns lists of peaks and probabilities.
* Function score\_isotope:
  + Compares a guessed isotope name with a true isotope name.
  + Calculates a score based on the number of matching isotopes.
* Function xray\_list:
  + Creates a list of X-ray peaks from the isotope dictionary.
* Function determine\_isotope:
  + Determines the isotope based on the peaks found.
  + Compares peaks with isotope database parameters and assigns a score.
  + Returns the identified peaks and isotope name.
* Function plot\_spectra:
  + Plots the given spectrum along with identified peaks.
  + Optionally, plots the true peaks and threshold.
* Function isotopeID:
  + Takes a file path, counts, isotope dictionary, and compression flag.
  + Initializes a Spectra object and finds peaks.
  + Identifies isotopes based on the peaks and plots the spectrum.
* Read Isotope Database:
  + Reads isotope information from a text file specified by isotopes\_xray\_path.
* Check File and Read Measurements:
  + Checks if the specified file is of type ".n42" and exists.
  + Reads measurements from the file using the custom xmlwrapper module.
* Loop Through Measurements:
  + Iterates through the measurements in the file.
  + Calls the isotopeID function for each measurement.
* Error Handling:
  + Provides an error message if the file type is incorrect.
* Import necessary libraries: os, xmlwrapper, matplotlib.pyplot, numpy, and functions from scipy.signal.
* Define a class Spectra for describing spectra.
* Define functions for expanding zeros in compressed spectra, finding a height threshold, determining isotope names, finding true isotope peaks, scoring isotope identification, defining X-rays from file, determining isotopes based on identified peaks, and plotting spectra.
* Create a class instance Spectra and plot spectra for each input file in a specified directory.
* Print the final score based on isotope identification.

Note: The code uses some external modules, such as xmlwrapper, which is not a standard Python module. It also relies on specific XML file structures. Make sure you have the required dependencies installed before running the code.

If you have any specific questions or if there's something specific you'd like assistance with, please let me know!

* Attributes Class:
  + Attributes class is used to abstract the dictionary-style access (dict['key']) using the '.' operator. It is initialized with a dictionary and provides a more convenient way to access attributes.
* XmlParse Class:
  + XmlParse class is designed to abstract etree.findall, etree.attrib, and etree.text operations in one place using the '.' operator. It is initialized with the root of an XML element.
  + It also handles namespaces (xmlns) and provides a more accessible way to navigate through XML elements and their attributes.
* get\_tag Function:
  + get\_tag function extracts the tag name of an XML element without the namespace.
* xmlread Function:
  + xmlread function reads an XML file, modifies it slightly to handle namespaces, and returns an instance of the XmlParse class.
  + If the file ends with '.n42', it reads the file content, adds a namespace (xmlns:H3D), and creates an ElementTree. It then returns an instance of the XmlParse class with the root of the ElementTree.

Example usage of the code:

python

Copy code

# Example usage of xmlread function

file\_path = "/path/to/your/file.n42"

parsed\_xml = xmlread(file\_path)

# Accessing attributes and elements

attrib\_value = parsed\_xml.XMLname.Attributes.attrib1

element\_text = parsed\_xml.XMLname.Element.text

# Handling multiple elements with the same name

different\_element\_1\_attrib = parsed\_xml.XMLname.DifferentElement[1].Attributes.attrib2

different\_element\_2\_text = parsed\_xml.XMLname.DifferentElement[2].Text

This code provides a convenient way to access attributes and elements in an XML document using Python. If you have any specific questions or if there's something specific you'd like assistance with, please let me know!