My Notice for my sae 1.5

- Create a program that will automate the execution of your

Data processing in Python:

First of all,

Certainly! Let's go through the code step by step:

1. **File Reading:**

  ```python
  try:
      with open("Dumpfile.txt", encoding="utf8") as fh:
          res = fh.read()
  except:
      print("The file does not exist %s", os.path.abspath('fichieratraiter.txt'))
  ```

  - It attempts to open the file "Dumpfile.txt" with utf-8 encoding.

  - If successful, it reads the content of the file into the variable `res`.

  - If the file does not exist, it prints an error message.

2. **Data Processing:**

  ```python
  ress = res.split('\n')
  ```

  - It splits the content of the file (`res`) into a list of lines, using the newline character '\n' as the delimiter.

3. **CSV File Initialization:**

  ```python
  ```

fic = open("coco.csv", "w")

- It opens a new CSV file named "coco.csv" in write mode (`"w"`).

4. **Header Writing:**

```python

evenement = "DATE ; SOURCE ; PORT ; DESTINATION ; FLAG ; SEQ ; ACK ; WIN ; OPTIONS ; LENGTH"

fic.write(evenement + "\n")

```

- It defines the header (`evenement`) for the CSV file with column names.

- Writes the header to the CSV file.

5. **Event Processing Loop:**

```python

for event in ress:

```

- Iterates through each line in `ress`.

- for the data that came back I used the csv files

  This Python script uses the pandas library to read a CSV file named "coco.csv" and then prints the most common value in each column of the DataFrame.

  Here's a breakdown of the code:

  1. `import pandas as pd`: Imports the pandas library and assigns the alias 'pd' for easier reference.

  2. `df = pd.read_csv("coco.csv", delimiter=";")`: Reads the CSV file "coco.csv" into a DataFrame named `df`. The delimiter is set to semicolon (;).

  3. The `for` loop iterates over each column in the DataFrame:

     ```python
     for column in df.columns:
     ```

4. Inside the loop, it calculates the most common value for each column using the `mode()` function and selects the first value using `iloc[0]`:

  ```python
  most_common_value = df[column].mode().iloc[0]
  ```

5. Finally, it prints the column name and its most common value:

  ```python
  print(f"Most common value in {column}: {most_common_value}")
  ```

The results you provided indicate the most common values in each column of the DataFrame:

- **DATE**: Most common value is "11:42:04.766656"
- **SOURCE**: Most common value is "BP-Linux8"
- **PORT**: Most common value is "https"
- **DESTINATION**: Most common value is "184.107.43.74.http:"
- **FLAG**: Most common value is "."
- **SEQ**: Most common value is "1419:2837"
- **ACK**: Most common value is "1322.0"
- **WIN**: Most common value is "512.0"
- **OPTIONS**: Most common value is "nop,nop,TS val 1512074367 ecr 102927777"
- **LENGTH**: Most common value is "1448"

These values represent the most frequently occurring entries in each respective column of your dataset.

- FOR MY PAGE HTML

```
python
•  file = open("C:/Users/amand/Music/Mon projet SAE15/EXCEL/Dumpfile.txt",
"r")
```

The code opens the text file (`Dumpfile.txt`) in read mode (`"r"`).

- **Initialize lists and counters:**

```
python
•  source_ips = []
destination_ips = []
lengths = []
flags = []
sequences = []
timestamps = []

flags_P_counter = 0
flags_S_counter = 0
flags_total_counter = 0
```

```
images_counter = 0
requests_counter = 0
responses_counter = 0
sequential_counter = 0
global_counter = 0
winner_counter = 0
```

Lists (`source_ips`, `destination_ips`, `lengths`, `flags`, `sequences`, `timestamps`) are initialized to store extracted data. Counters are also initialized to track different events in the file.

- **File reading loop:**

```
python
•  for line in file:
    #...
```

The code reads each line of the file, extracts, and processes relevant information (source IP address, destination IP, flags, sequence, etc.).

- **Calculate statistics:**

```
python
•  P = flags_P_counter / flags_total_counter
S = flags_S_counter / flags_total_counter
A = flags_total_counter / global_counter

total_req_rep = responses_counter + requests_counter
req = requests_counter / total_req_rep
rep = responses_counter / total_req_rep
```

Statistics are calculated, such as the proportion of [P], [S], [.] flags relative to the total, as well as the proportion of ICMP requests and responses.

- **Create graphs with Matplotlib:**

```
python
•  # Pie chart for flags
plt.pie(data, explode=explode_data, labels=labels_data, autopct='%1.1f%%',
startangle=90, shadow=True)
plt.axis('equal')
plt.savefig("C:/Users/amand/Music/Mon projet SAE15/EXCEL/graph1.png")

# Pie chart for requests and responses
plt.pie(data2, explode=explode_data, labels=labels_data2,
autopct='%1.1f%%', startangle=90, shadow=True)
plt.savefig("C:/Users/amand/Music/Mon projet SAE15/EXCEL/graph2.png")
```

Pie charts are generated using the Matplotlib library to visually represent the proportions of flags and requests/responses.

- **Generate HTML content:**

```
python
•  html_content = '''
```

```
<html>
   <!-- ... HTML Content ... -->
</html>
```

The code generates HTML content with tags incorporating calculated statistics and created graphs.

- **Write data to CSV files:**

```
python
•  with open('C:/Users/amand/Music/Mon projet SAE15/EXCEL/data.csv', 'w', 
newline='') as csv_file:
    # ... write extracted data to a CSV file ...

with open('C:/Users/amand/Music/Mon projet SAE15/EXCEL/Stats.csv', 'w', 
newline='') as file2:
    # ... write statistics to a CSV file ...
```

Extracted data and statistics are written to two separate CSV files.

- **Create a web page:**

```
python
•  with open("C:/Users/amand/Music/Mon projet SAE15/EXCEL/data.html", "w") 
as html_file:
    html_file.write(html_content)
    print("Web page created successfully")
```

The code creates an HTML file containing the previously generated content.

- **Close the initial file:**

```
python
      file.close()
```

The initial file is closed after processing all the lines.