```
[ ]: %load_ext ipython_unittest
     # imports
     import pandas as pd
     import unittest
```

Identifying stimulus onsets

After testing the devices' latencies, we concluded that latencies are constant and devices are synchronized. Now, we did the actual experiment with a participant.

Situation: Participant sat on a chair inside a virtual room and performed a visual task looking at some images. Simultaneously, we collected brain and eye-tracking data. We saved the following streams:

Device	Stream Content
EEG	64 channels brain signal
Unity	*ImageInfo* *imageNumber* (0-3), *imageName* (face, body, object or 'fixation', 'grayCanvas' when not displaying an image)
Unity	*VisualDisplayStatus* (0 for images, 1 for fixation, 2 for canvas, and -1 for start of experiment)

Lab Streaming Layer (LSL) was used to store and synchronize the streams timestamps (*time_stamps*)

1. Load Data

```
[ ]: # load csv as dataframe
     df = pd.read_csv("data-onsets/all_streams_ab6397.csv", low_memory=False)
     df
```

```
[ ]: # inspect the unique values in imageName
     df['imageName_ImageInfo'].head(1000).unique()
     # the images names contain other relevant metadata such as:
     # img.1600x1000.date.2022-07-07_19-36-43.hitname.face_4208.rotation.54.distance.3.
       ↪frame.45567
```

Task 1:

Identify the shifts in the imageInfo stream and save them to a new columns called *"shift"*

Hint: We can compare the rows of the *imageName* column with each other identifying every first time it changes from startMessage to grayCanvas or to something else.

Test your results against the unit tests in the next cell.

```
[ ]: # identify the shifts in the imageInfo stream
     df['shift'] = df['imageName_ImageInfo'].shift(1) != df['imageName_ImageInfo']
```

```
[ ]: %%unittest
     # Test your result against the expected outcome
     # when correct, it returns 'Success'
     assert 'shift' in df.head()
     assert len(df[df['shift']]) == 5045
```

Task 2:

Extract the names of objets, bodies, and faces from the image name and store in a separate column *"ob_names"*

Hint: we can use a lambda function on the DataFrame to create new series of values. Here some sources: - Applying a function along an axis of the DataFrame. - Lambdas documentation - Python lambda Tutorial

Test your results against the unit tests in the next cell.

[ ]:
```
# save the names of the object, body or face shown in the image
df['ob_names'] =  df.apply(lambda x: x["imageName_ImageInfo"].split(".")[5]
                if len(x["imageName_ImageInfo"].split(".")) > 7 else '', axis=1)
```

[ ]:
```
%%unittest
# Test your result against the expected outcome
assert 'ob_names' in df.head()
assert len(df['ob_names'].unique()) == 103
```

Task3:

Create a new column called “latency” containing the exact time when an image was first displayed.

These will correspond to the times indicating the stimulus onsets for the even related potential (ERP), which are usually called *“latency”* in the EEG trigger files.

Hint: again we can use a lambda function on the DataFrame to create new series of values.

Test your results against the unit tests in the next cell.

[ ]:
```
# save the starting time (aka 'latency') when image is displayed
df['latency'] = df.apply(lambda x: x['normalized_tstamps_ImageInfo']
                if len(x["imageName_ImageInfo"].split(".")) > 7
                and x['shift'] else '', axis=1)
```

[ ]:
```
%%unittest
# Test your result against the expected outcome
assert 'latency' in df.head()
assert len(df['latency'].unique()) == 1681 # If you get 5046 unique values, you are␣
↪not just looking at times images are displayed, but all shifts
```

Task 4:

Save the type of image that was displayed (face, object, body) in a column called *“type”*

Hint: we can again use a lambda function on the DataFrame to create new series of values depending on the string type in a column and values in a different column.

Test your results against the unit tests in the next cell.

[ ]:
```
# save the type of image displaying (face, object, body)
df['type'] = df.apply(lambda x: 'face' if x['shift'] and 'face'
                        in x['imageName_ImageInfo'].lower()
                        else ('body' if x['shift'] and 'npc'
                        in x['imageName_ImageInfo'].lower()
                        else ('object' if x['shift'] and 'rotation'
                        in x['imageName_ImageInfo'].lower()
                        and 'face|npc' not in x['imageName_ImageInfo'].lower()
                        else '')), axis=1)
```

[ ]:
```
%%unittest
# Test your result against the expected outcome
assert 'type' in df.head()
assert len(df['type'].unique()) == 4
assert len(df[df['type'] == 'body']) == 560
assert len(df[df['type'] == 'face']) == 560
```

```
assert len(df[df['type'] == 'object']) == 560
```

Task 5:

Define the triggers for rotation, distance, and block

Finally, we can define the stimulus onsets for rotation, distance, and block in the same way we have defined it for faces. Afterward, we just need to save the stimulus onsets into a single *"triggers"* dataframe

[ ]:
```
# define the triggers for rotation, distance, and block
df['rotation'] = df.apply(lambda x: x["imageName_ImageInfo"].split(".")[7]
                 if len(x["imageName_ImageInfo"].split(".")) > 7
                 and x['shift'] else '', axis=1)
# distances
df['distance'] = df.apply(lambda x: x["imageName_ImageInfo"].split(".")[9]
                 if len(x["imageName_ImageInfo"].split(".")) > 7
                 and x['shift'] else '', axis=1)
# blocks
df['block'] = df.apply(lambda x: str(x["blockNumber_ImageInfo"])
                 if len(x["imageName_ImageInfo"].split(".")) > 7
                 and x['shift'] else '', axis=1)

# select the trigger columns and non-empty rows
df_selected = df[['latency','type','rotation','distance','block']]
df_triggers = df_selected[df_selected['latency'] != '']
```

[ ]:
```
df_triggers
```

[ ]:
```
```