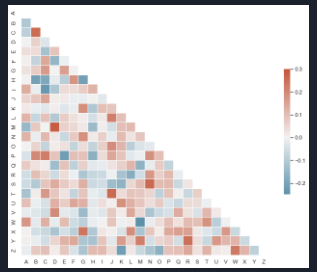
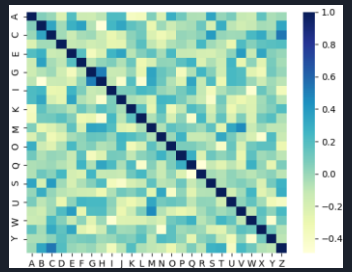
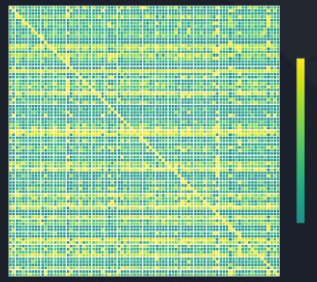






Matrix Visualization in Python

Marek Jesienouski & Omarion Diallo

We need more *Libraries* for plotting

```
import numpy as np
```

- **numpy (np)** → Handles numbers and arrays.

```
import pandas as pd
```

- **pandas (pd)** → Create and manage the table (DataFrame), just like spread sheet!

```
import matplotlib.pyplot as plt
```

- **matplotlib.pyplot (plt)** → Draw stuff.

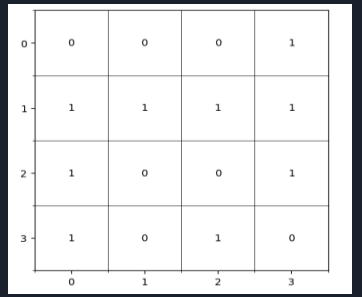
Setting a random DataFrame

This is the code to define a randomly generated matrix

```python
# ----- IMPORT LIBRARIES -----
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


# ----- CREATE SAMPLE DATAFRAME -----
# Matrix is now a pandas DataFrame with each element randomized
matrix_df = pd.DataFrame(
    np.random.randint( 2, size=(4, 4)), # Generate 4x4 array of random 0s or 1s
    columns=['0', '1', '2', '3'],
    index=['3', '2', '1', '0'] # Changed index to align with 0,0 at top-left
)
```

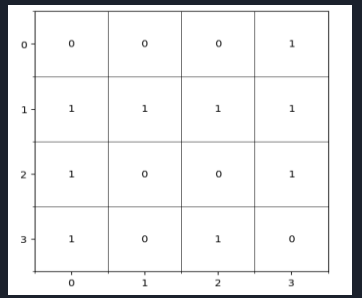
This is the matrix to be plotted:



Setting the size and design

```
plt.figure(figsize=(5, 5))
plt.imshow(np.ones(matrix_df.shape), cmap='gray_r', origin='upper', extent=[-0.5, matrix_df.shape[1]-0.5, matrix_df.shape[0]-0.5, -0.5]) # White background
```

This is the matrix to be plotted:



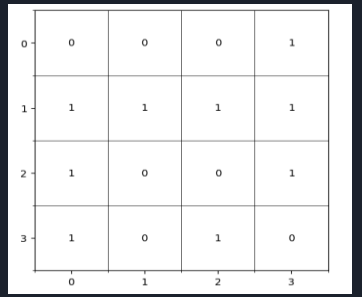
Setting grid lines and display values

```
# Add grid lines
ax = plt.gca()
ax.set_xticks(np.arange(-0.5, matrix_df.shape[1], 1), minor=True)
ax.set_yticks(np.arange(-0.5, matrix_df.shape[0], 1), minor=True)
ax.grid(which='minor', color='black', linestyle='-', linewidth=0.5)

# Display values
for i in range(matrix_df.shape[0]):
    for j in range(matrix_df.shape[1]):
        plt.text(j, i, matrix_df.iloc[i, j], ha='center', va='center', color= 'Black')

plt.xticks(np.arange(matrix_df.shape[1]), matrix_df.columns, rotation=0)
plt.yticks(np.arange(matrix_df.shape[0]), matrix_df.index, rotation=0)
plt.show()
```

This is the matrix to be plotted:



Generating position codes

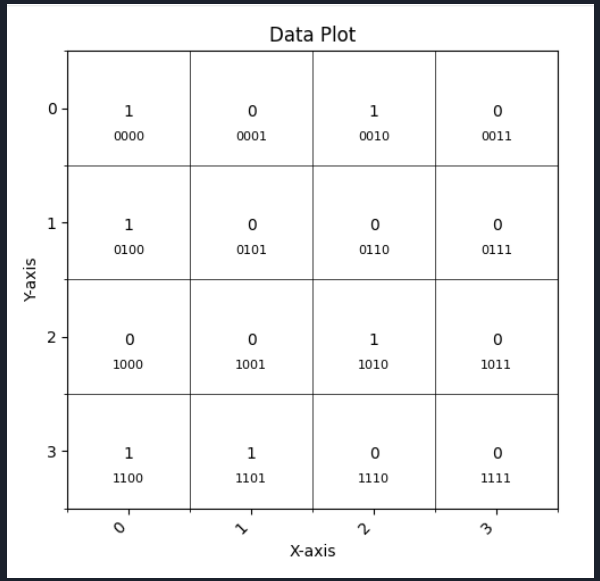
```
# ------ Numbers under placement ------
# Generate position_codes
num_cols, num_rows  = matrix_df.shape
position_codes = []

for r in range(num_rows):
    row_codes = []
    for c in range(num_cols):
        # Get current row and column labels and convert to integer
        row_label_int = int(matrix_df.index[r])
        col_label_int = int(matrix_df.columns[c])

        # Convert to 2-bit binary representation
        row_binary = bin(row_label_int)[2:].zfill(2)
        col_binary = bin(col_label_int)[2:].zfill(2)

        # Concatenate to form the new binary string
        binary_string = row_binary + col_binary
        row_codes.append(binary_string)
    position_codes.append(row_codes)
```

This is the matrix to be plotted:



Adding position codes to the plot

This is the matrix to be plotted:

```
# Add grid lines
ax = plt.gca()
ax.set_xticks(np.arange(-0.5, matrix_df.shape[1], 1), minor=True)
ax.set_yticks(np.arange(-0.5, matrix_df.shape[0], 1), minor=True)
ax.grid(which='minor', color='black', linestyle='-', linewidth=0.5)

# Display values
for i in range(matrix_df.shape[0]):
    for j in range(matrix_df.shape[1]):
        plt.text(j, i, matrix_df.iloc[i, j], ha='center', va='center', color= 'Black')

plt.xticks(np.arange(matrix_df.shape[1]), matrix_df.columns, rotation=0)
plt.yticks(np.arange(matrix_df.shape[0]), matrix_df.index, rotation=0)
plt.show()
```