IE416 - ROBOT PROGRAMMING
LAB - 1

Group Member :

1. Vaishvik Patel (202201433)
2. Darshan Jogadiya (202201415)

Github Link

Colab Link

NumPy

Visualization

IE416 - ROBOT PROGRAMMING

Group Member :

1. Vaishvik Patel (202201433)
2. Darshan Jogadiya (202201415)

Github Link : https://github.com/Vaishvik79/IE416_Robot_Programming/tree/main/LAB1

Colab Link : https://colab.research.google.com/drive/1qqqYtmI9uGXUaQ7LIRnPlj-qWLDsR9ze#scrollTo=J962SyWiTNYF

NumPy : https://colab.research.google.com/drive/108BZ9Katr9yDEufecatIDNWMCI_Wr-qW

Visualization : https://colab.research.google.com/drive/1aA7ppVlUUBuZGTOs9MVpePy9AtZhtGqL

Q1. Write a function that gives number of days of given year.

```
a = int(input("Enter a year : "))
if(a % 100 == 0):
    if(a % 4 == 0):
        print("Number of days in %d :"%a,"366")
elif(a % 4 == 0):
    if(a % 100 != 0):
        print("Number of days in %d :"%a,"366")
else:
    print("Number of days in %d :"%a,"365")
```

```
Enter a year : 2024
Number of days in 2024 : 366
```

Q2. Count the frequency of each character in a string and store it in a dictionary. An example is given below.

```
s = input("Enter a string :")
freq = {}
for c in s:
  freq[c] = freq.get(c,0) + 1

print(freq)
```

```
Enter a string :DAIICT
{'D': 1, 'A': 1, 'I': 2, 'C': 1, 'T': 1}
```

Q3. Write a program to remove duplicates from a list but keep the first.

```
n = int(input("Enter number of elements : "))
st = set()
for i in range(n):
    a = int(input())
    st.add(a)

print(st)
```

```
Enter number of elements : 5
1
1
2
4
3
{1, 2, 3, 4}
```

Q4. Write a program to sort a stack using only another stack (no other data structures like arrays or linked lists).

```
n = int(input("Enter number of elements : "))
stack = []
for i in range(n):
  x = input("Enter the element :")
  stack.append(x)

sorted = []
while stack:
  x = stack.pop()
  while sorted and sorted[-1] > x:
    stack.append(sorted.pop())
  sorted.append(x)

print(sorted)
```

```
Enter number of elements : 5
Enter the element :2
Enter the element :4
Enter the element :1
Enter the element :5
Enter the element :6
['1', '2', '4', '5', '6']
```

Q5. Make a module “pascal.py” with function “pascalTriangle(numOfRows)” and import into “main.py”.

```
def createPascalTriangle(n):
    # iterate up to n
    for i in range(n):
    # adjust space
      print(' '*(n-i), end='')

    # compute each value in the row
      coef = 1
      for j in range(0, i + 1):
        print(coef, end=' ')
        coef = coef * (i - j) // (j + 1)
      print()

if __name__ == "__main__":
  n = int(input("Enter the number of rows :"))
  createPascalTriangle(n)
```

```
Enter the number of rows :5
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
```

Q6. Create a 6x6 matrix with random values and:

```
Replace all values greater than 0.5 with 1, and all others with 0.
Extract a 3x3 submatrix starting from index (2, 2) and calculate its mean.
```

```
import numpy as np

matrix = np.random.random((6, 6))
matrix[matrix > 0.5] = 1
matrix[matrix <= 0.5] = 0

sub_mat = matrix[2:5, 2:5]
mean_value = np.mean(sub_mat)

print("6x6 Matrix:\n", matrix)
print("Extracted sub-matrix:\n", sub_mat)
print("Mean of sub-matrix:", mean_value)
```

```
Matrix:
 [[0. 1. 0. 1. 0. 0.]
 [1. 1. 0. 0. 0. 0.]
 [1. 0. 1. 0. 1. 1.]
 [0. 1. 1. 1. 0. 1.]
 [0. 0. 0. 1. 0. 1.]
 [1. 0. 1. 0. 0. 0.]]
Extracted sub-matrix:
 [[1. 0. 1.]
 [1. 1. 0.]
 [0. 1. 0.]]
Mean of sub-matrix: 0.5555555555555556
```

Q7. Array Reshaping:

```
Create a 1D array with 16 elements. Reshape it into a 4x4 matrix.

Flatten a 3x3x3 array into a 1D array.

Reshape a matrix into a new shape without changing its data.
```

```
import numpy as np

def transform_array():
    # Create a 1D array with 16 elements
    one_d_array = np.arange(16)
    print("1D Array with 16 Elements:")
    print(one_d_array)

    # Reshape to a 4x4 matrix
    matrix_4x4 = one_d_array.reshape(4, 4)
    print("\nReshaped to 4x4 Matrix:")
    print(matrix_4x4)

    # Create a 3x3x3 array
    three_d_array = np.arange(27).reshape(3, 3, 3)
    print("\nReshaped to 3x3x3 Array:")
    print(three_d_array)

if __name__ == "__main__":
    transform_array()
```

```
1D Array with 16 Elements:
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]

Reshaped to 4x4 Matrix:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

Reshaped to 3x3x3 Array:
[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]]

 [[ 9 10 11]
  [12 13 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]
  [24 25 26]]]
```

Q8. Write a recursive function Fibonacci_sum(n) to calaculate the sum of first n numbers in Fibonacci series 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89.

```
def fibonacci_sum(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci_sum(n - 1) + fibonacci_sum(n - 2) + (n if n > 1 else 0)

if __name__ == "__main__":
    num = int(input("Enter the number of Fibonacci numbers to sum: "))
    print("Sum of first", num, "Fibonacci numbers:", fibonacci_sum(num))
```

```
Enter the number of Fibonacci numbers to sum: 10
Sum of first 10 Fibonacci numbers: 364
```

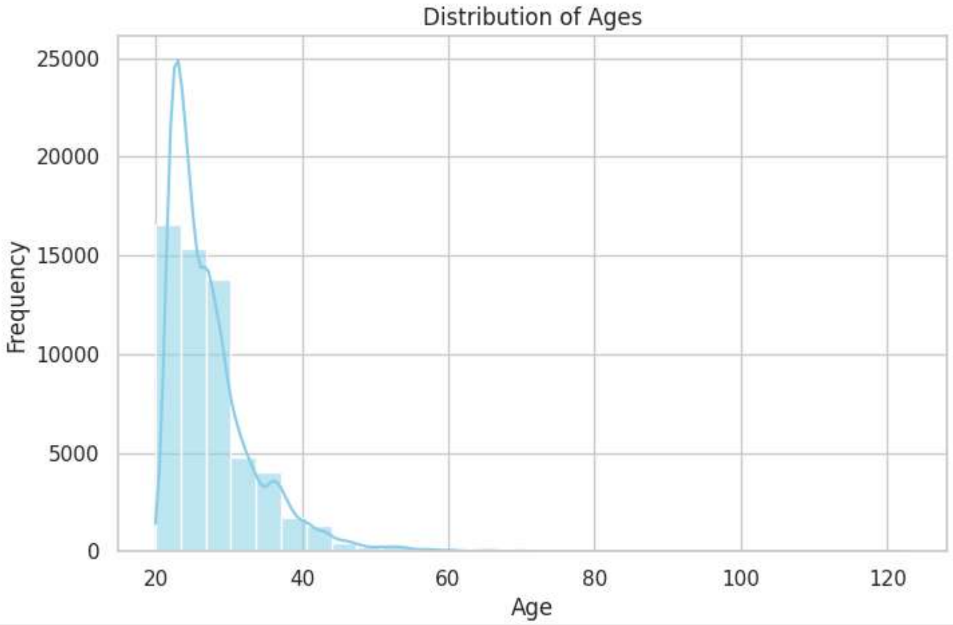
Q9) Define a function get_value_from_dict that takes a dictionary and a key as parameters.If the key is not present in the dictionary, the function should raise a KeyError with a custom error message. Write a main function that calls get_value_from_dict with a dictionary and userprovided key. Handle KeyError and display a user-friendly message if the key is not found

```
def get_dict_value(data_dict, search_key):
    try:
        result = data_dict[search_key]
        return result
    except KeyError:
        raise KeyError("Key not found in dictionary")

if __name__ == "__main__":
    sample_dict = {"l": 10, "m": 20, "n": 30}
    user_key = input("Enter a key: ")
    try:
        retrieved_value = get_dict_value(sample_dict, user_key)
        print(f"The value for key '{user_key}' is {retrieved_value}")
    except KeyError as e:
        print(e)
```

```
Enter a key: v
'Key not found in dictionary'
```

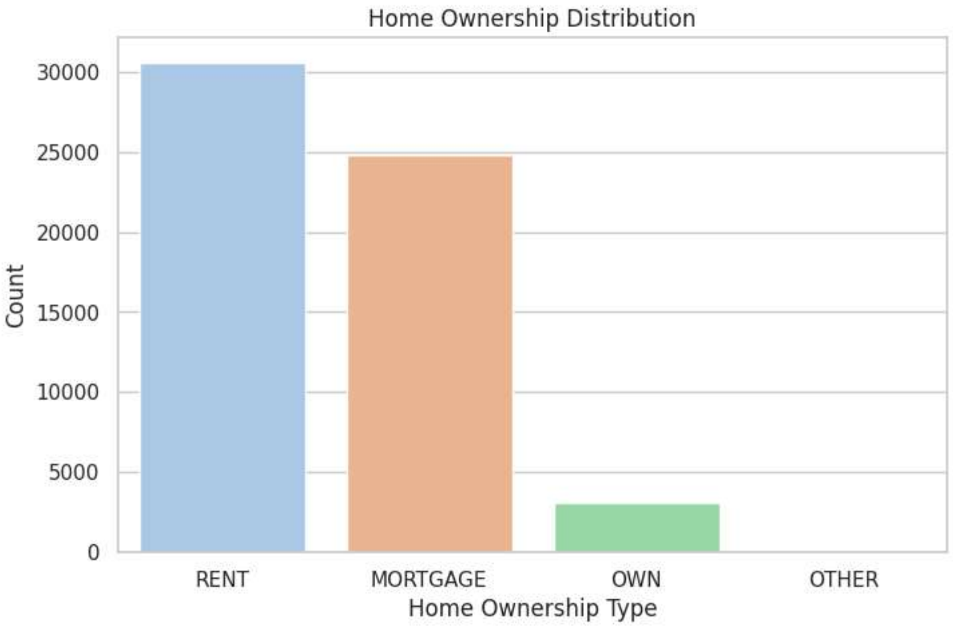
Q10. Using the following dataset, visualize the data with the maximum number of visualization tools available in Python. Create a variety of plots and charts, including but not limited to bar charts, piecharts, line graphs, scatter plots, histograms, and heatmaps. Use libraries such as matplotlib, seaborn, and plotly to explore different ways of presenting the data. Provide clear titles, labels, and legends to enhance the readability of your visualizations.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Raw GitHub URL for the CSV file
data_url = "https://raw.githubusercontent.com/Vaishvik79/IE416_Robot_Programming/refs/heads/main/LAB1/Loan_train.csv"

# Load the dataset
loan_df = pd.read_csv(data_url)

sns.set_theme(style="whitegrid")

plt.figure(figsize=(8, 5))
sns.histplot(loan_df['person_age'], kde=True, bins=30, color='skyblue')
plt.title('Distribution of Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



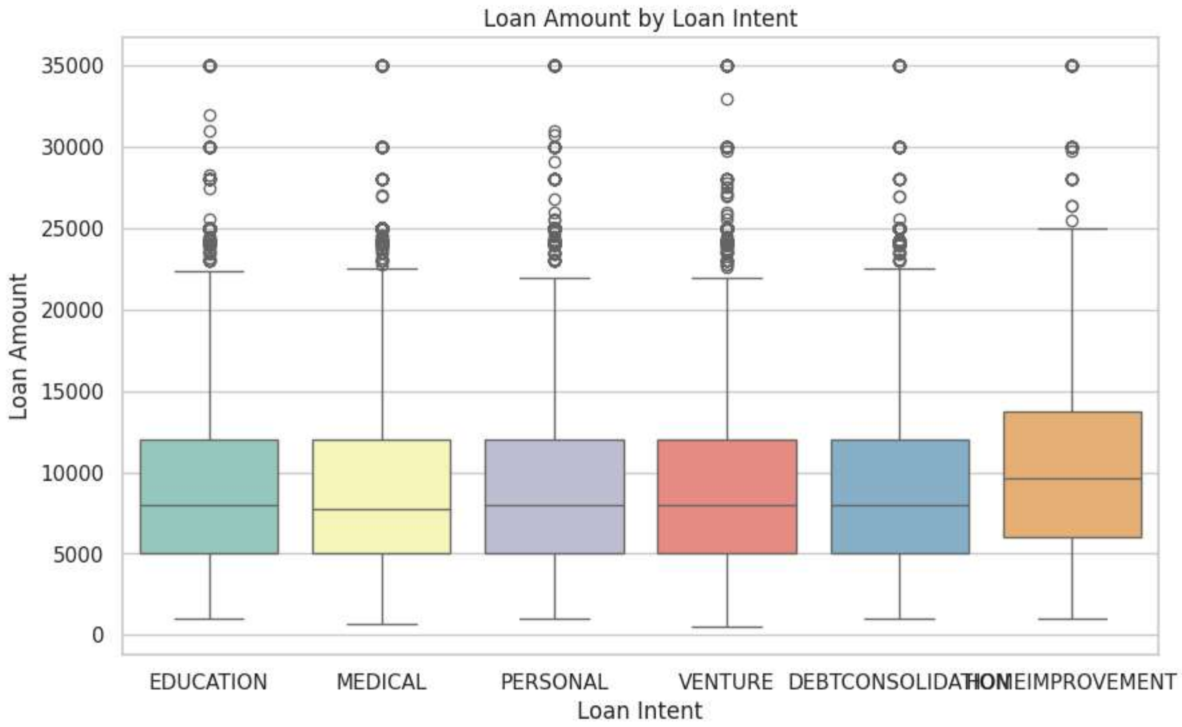
```
plt.figure(figsize=(8, 5))
sns.countplot(x='person_home_ownership', data=loan_df, palette='pastel', order=loan_df['person_home_ownership'].value_counts().index)
plt.title('Home Ownership Distribution')
plt.xlabel('Home Ownership Type')
plt.ylabel('Count')
plt.show()
```

```
<ipython-input-30-8b725718e76f>:2: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
```

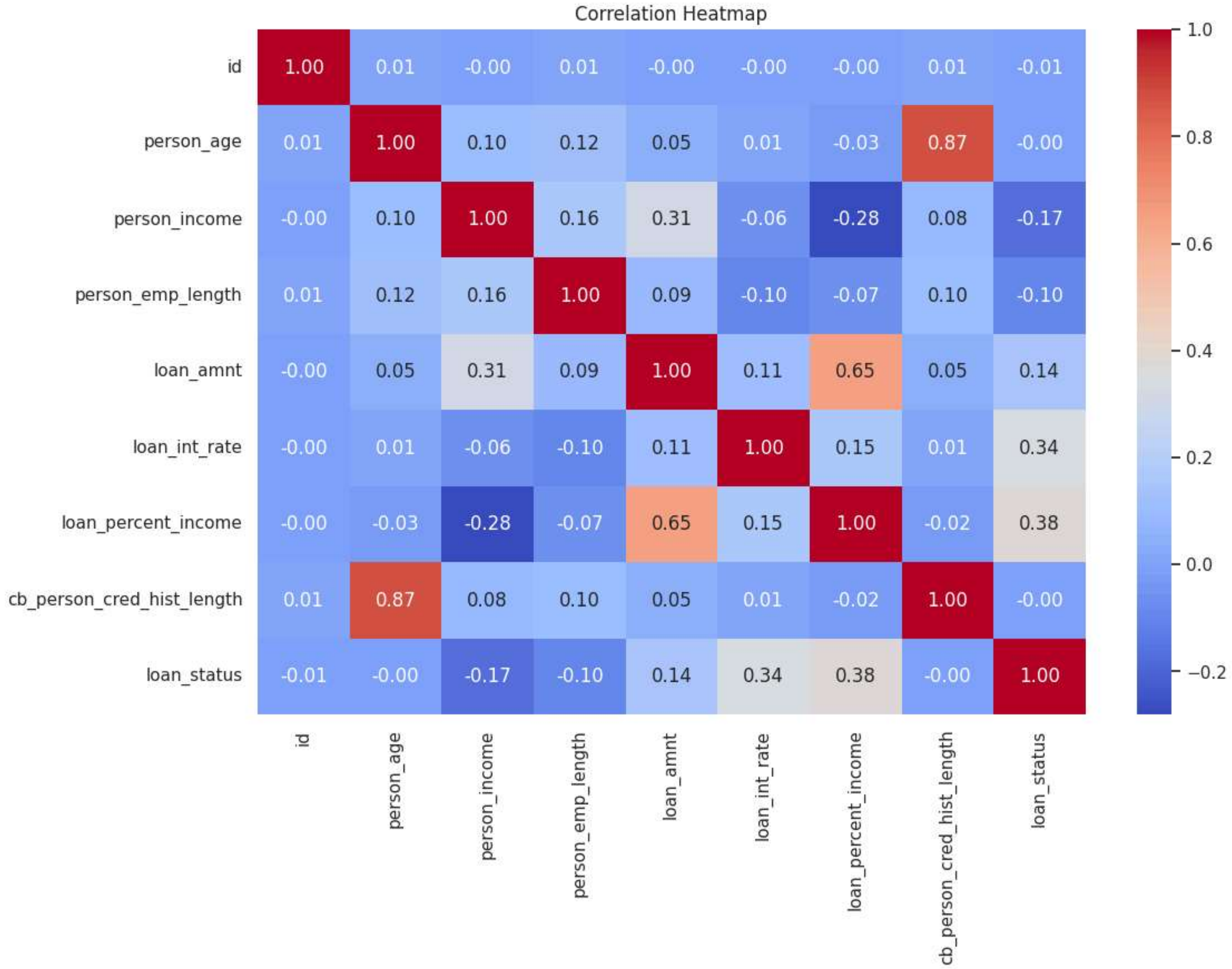


```
plt.figure(figsize=(10, 6))
sns.boxplot(x='loan_intent', y='loan_amnt', data=loan_df, palette='Set3')
plt.title('Loan Amount by Loan Intent')
plt.xlabel('Loan Intent')
plt.ylabel('Loan Amount')
plt.show()
```

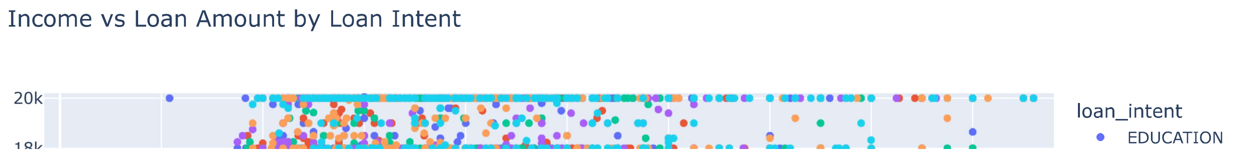
```
<ipython-input-31-e54695e7c5f3>:2: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
```



```
plt.figure(figsize=(12, 8))
numeric_cols = loan_df.select_dtypes(include=['float64', 'int64'])
corr_matrix = numeric_cols.corr()
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm', cbar=True)
plt.title('Correlation Heatmap')
plt.show()
```



```
income_vs_loan_plot = px.scatter(loan_df, x='person_income', y='loan_amnt',
                                 color='loan_intent',
                                 title='Income vs Loan Amount by Loan Intent',
                                 labels={'person_income': 'Income', 'loan_amnt': 'Loan Amount'})
income_vs_loan_plot.show()
```



```
loan_grade_counts = loan_df['loan_grade'].value_counts()
pie_chart_loan_grades = px.pie(values=loan_grade_counts, names=loan_grade_counts.index,
                               title='Loan Grades Distribution',
                               color_discrete_sequence=px.colors.sequential.RdBu)
pie_chart_loan_grades.show()
```

Loan Grades Distribution