INDEX

1. **Write a python program to check the validity of a password given by the user. The password should satisfy the following criteria.**
    1) **Contain at least one letter between a and z.**
    2) **Contains at least one number between 0 and 9.**
    3) **Contains at least one letter between A and Z.**
    4) **Contains at least one special character from $, #, @**
    5) **Minimum length of password: 6**

Algorithm

Step 1: Start.

Step 2: Take the password as input from the user.

Step 3: Check if the password has at least one lowercase letter.

Step 4: Check if the password has at least one uppercase letter.

Step 5: Check if the password has at least one digit.

Step 6: Check if the password has at least one special character.

Step 7: Check if the password length is at least 6.

Step 8: If all conditions are met, print "Password is valid."

Step 9: Otherwise, print "Password is invalid."

Step 10: Stop.

Source Code

```
def  is_valid_password(password):
    has_lower = False
    has_upper = False
    has_digit = False
    has_special = False
    special_chars = "$#@"

    if len(password) < 6:
        return False

    for char in password:
        if char.islower():
```

```
            has_lower = True
        elif char.isupper():
            has_upper = True
        elif char.isdigit():
            has_digit = True
        elif char in special_chars:
            has_special = True
    return has_lower and has_upper and has_digit and has_special

password = input("Enter a password: ")
if is_valid_password(password):
    print("Password is valid.")
else:
    print("Password is invalid. Make sure it satisfies all the conditions.")
```

Output

```
Enter a password:  Python@123
Password is valid.
```

2. **Write a python program to demonstrate different number data types in Python. And Write a program to perform different Arithmetic Operations on numbers in Python.**

Algorithm

Step 1: Start.

Step 2: Declare variables **int_num, float_num, complex_num**, and **bool_val**.

Step 3: Print the value and type of each variable.

Step 4: Convert **int_num** to float, **float_num** to int, and **bool_val** to int, then print results.

Step 5: Declare variables **a** and **b**, and print their values.

Step 6: Perform and print the results of arithmetic operations on a and b.

Step 7: Stop.

Source Code

```
int_num = 42
print(f"Integer: {int_num}, Type: {type(int_num)}")

float_num = 42.42
print(f"Floating-point: {float_num}, Type: {type(float_num)}")

complex_num = 3 + 4j
print(f"Complex number: {complex_num}, Type: {type(complex_num)}")

bool_val = True
print(f"Boolean: {bool_val}, Type: {type(bool_val)}")

print(f"Convert Integer to Float: {float(int_num)}")
print(f"Convert Float to Integer: {int(float_num)}")
print(f"Convert Boolean to Integer: {int(bool_val)}")

a = 10
b = 3
```

```
print(f"Numbers: a = {a}, b = {b}")
print(f"Addition: a + b = {a + b}")
print(f"Subtraction: a - b = {a - b}")
print(f"Multiplication: a * b = {a * b}")
print(f"Division: a / b = {a / b}")
print(f"Floor Division: a // b = {a // b}")
print(f"Modulo: a % b = {a % b}")
print(f"Exponentiation: a ** b = {a ** b}")
```

Output

```
Integer: 42, Type: <class 'int'>
Floating-point: 42.42, Type: <class 'float'>
Complex number: (3+4j), Type: <class 'complex'>
Boolean: True, Type: <class 'bool'>

Convert Integer to Float: 42.0
Convert Float to Integer: 42
Convert Boolean to Integer: 1

Numbers: a = 10, b = 3
Addition: a + b = 13
Subtraction: a - b = 7
Multiplication: a * b = 30
Division: a / b = 3.3333333333333335
Floor Division: a // b = 3
Modulo: a % b = 1
Exponentiation: a ** b = 1000
```

3. Write a Python program to construct the following pattern, using a nested for loop.

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
```

Algorithm

Step 1: Start.

Step 2: Loop through numbers from 0 to 6 (inclusive) using **i**.

Step 3: Inside the loop, for each value of i, loop through numbers from 0 to **i-1** to print an asterisk **(*).**

Step 4: After printing the asterisks for each **i,** move to the next line.

Step 5: Repeat until the outer loop finishes.

Step 6: End.

Source Code

```
for i in range(7):
    for j in range(i):
        print("*", end=" ")
    print("")
```

Output

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
```

4. **Write a python class named Circle constructed by a radius and two methods which will compute the perimeter of a given circle**

Algorithm

Step 1: Start.
Step 2: Import the **math** module.
Step 3: Define a class **Circle**:

- **Step 3.1**: Define the **__init__** method to initialize the **radius** attribute when a new object is created.
- **Step 3.2**: Define the **perimeter** method to calculate the perimeter using **math.pi** and the object's **radius**.
- **Step 3.3**: Define the **area** method to calculate the area using **math.pi** and the object's **radius**.

Step 4: Ask the user to input the radius of the circle.
Step 5: Create an instance (object) of the **Circle** class, passing the input radius as a parameter.
Step 6: Call the **perimeter** method on the circle object and print the result, rounded to two decimal places.
Step 7: Call the **area** method on the circle object and print the result, rounded to two decimal places.

Source Code

```
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def perimeter(self):
        return 2 * math.pi * self.radius

    def area(self):
        return math.pi * (self.radius ** 2)
```

```
radius = float(input("Enter the radius of the circle: "))

circle = Circle(radius)
print(f"Perimeter of the circle: {circle.perimeter():.2f}")
print(f"Area of the circle: {circle.area():.2f}")
```

Output

```
Enter the radius of the circle:  5
Perimeter of the circle: 31.42
Area of the circle: 78.54
```

5. **Given the file "auto.csv" of automobile data with the fields inbox, company, body-style, wheel-base, length, engine-type, number-of-cylinders, horsepower, average-mileage, and price,. Write Python codes using Pandas to**

    1) **Clean and Update Files**
    2) **Print total cars of all companies.**
    3) **Find the average mileage of all companies**
    4) **Find the highest priced car of all companies**

Algorithm

Step 1: Start.

Step 2: Load the CSV file using Pandas.

Step 3: Clean the data by removing the duplicate rows, filling the missing numeric values with 0, filling missing values in text columns with unknown and by filling the values in the milage column with its mean

Step 4: Count the total number of cars for each company using **value_counts()**.

Step 5: Calculate the average mileage of cars for each company using **groupby()** and **mean()**.

Step 6: Identify the highest-priced car for each company using **groupby()** and **idxmax()**.

Step 7: Print the results.

Step 8: Stop.

Source Code

```
import pandas as pd

data = pd.read_csv("auto.csv")

data = data.drop_duplicates()

data['price'] = data['price'].fillna(0)
data['company'] = data['company'].fillna("Unknown")
data['average-mileage'] = data['average-mileage'].fillna(data['average-mileage'].mean())

print("Cleaned Data:\n", data.head())
```

```
total_cars = data['company'].value_counts()
print("\nTotal Cars of Each Company:\n", total_cars)

average_mileage = data.groupby('company')['average-mileage'].mean()
print("\nAverage Mileage of Each Company:\n", average_mileage)

highest_priced_cars = data.loc[data.groupby('company')['price'].idxmax()]
print("\nHighest Priced Car of Each Company:\n", highest_priced_cars[['company', 'price']])
```

Output

```
Cleaned Data:
   company  price  average-mileage
0  Toyota  25000             25.5
1   Honda  22000             27.3
2    Ford  20000             22.8
3  Toyota  30000             23.6
4   Honda  18000             26.9

Total Cars of Each Company:
 company
Toyota       3
Honda        3
Ford         2
Chevrolet    1
Nissan       1
Name: count, dtype: int64

Average Mileage of Each Company:
 company
Chevrolet    22.400000
Ford         23.450000
Honda        27.433333
Nissan       26.200000
Toyota       23.933333
Name: average-mileage, dtype: float64

Highest Priced Car of Each Company:
      company  price
6  Chevrolet  27000
5       Ford  22000
1      Honda  22000
7     Nissan  21000
8     Toyota  35000
```

6. **Write a python program to add two matrices and find the transpose of the resultant matrix**

Algorithm

Step 1: Start.

Step 2: Define two matrices **matrix1** and **matrix2**.

Step 3: Initialize an empty **resultant_matrix** with the same size as **matrix1** and **matrix2**.

Step 4: Add the corresponding elements of **matrix1** and **matrix2** to **resultant_matrix**.

- For each element at position **(i, j)**, set **resultant_matrix[i][j] = matrix1[i][j] + matrix2[i][j].**

Step 5: Initialize an empty **transpose_matrix** with swapped dimensions **(cols × rows)**.

Step 6: Transpose **resultant_matrix** into **transpose_matrix**.

- For each element at position **(i, j)**, set **transpose_matrix[j][i] = resultant_matrix[i][j].**

Step 7: Print **matrix1, matrix2, resultant_matrix,** and **transpose_matrix**.

Step 8: Stop.

Source Code

```
matrix1 = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]

matrix2 = [[5, 14, 33],[10, 6, 1],[15, 10, 12]]

rows, cols = len(matrix1), len(matrix1[0])
resultant_matrix = [[0 for _ in range(cols)] for _ in range(rows)]

for i in range(rows):
    for j in range(cols):
        resultant_matrix[i][j] = matrix1[i][j] + matrix2[i][j]

for i in range(rows):
    for j in range(cols):
```

```
        transpose_matrix[j][i] = resultant_matrix[i][j]

print("\nResultant Matrix (Matrix1 + Matrix2):")
for row in resultant_matrix:
    print(row)

print("\nTranspose of the Resultant Matrix:")
for row in transpose_matrix:
    print(row)
```

Output

```
Resultant Matrix (Matrix1 + Matrix2):
[6, 16, 36]
[14, 11, 7]
[22, 18, 21]

Transpose of the Resultant Matrix:
[6, 14, 22]
[16, 11, 18]
[36, 7, 21]
```

7. **Write a python program to create a function called list_of_frequency that takes a string and prints the letter in non-increasing order of the frequency of their occurrences. Use dictionaries.**

Algorithm

Step 1: Start

Step 2: Define a function named **list_of_frequency** that takes a string s as input.

Step 3: Create an empty dictionary **frequency** to store letter frequencies.

Step 4: Iterate through each character **char** in the string **s**:

- If **char** is an alphabetic character, convert it to lowercase.
- Increment its count in the **frequency** dictionary (initialize with 0 if not present).

Step 5: Convert the **frequency** dictionary to a list of tuples **frequency_items**.

Step 6: Perform bubble sort on **frequency_items** to sort it in non-increasing order of frequency:

- For each index **i** from **0** to **len(frequency_items) - 1**:
    - For each index **j** from **0** to **len(frequency_items) - i - 2**:
        - Compare the frequencies of adjacent elements.
        - Swap them if the earlier frequency is smaller.

Step 7: Print each letter and its frequency from the sorted list.

Step 8: Read a string input **text** from the user.

Step 9: Call the function **list_of_frequency** with **text** as the argument.

Step 10: Stop

Source Code

```
def list_of_frequency(s):
    # Create a dictionary to count the frequency of each letter
    frequency = {}
    for char in s:
        if char.isalpha():  # Consider only alphabetic characters
            char = char.lower()  # Normalize to lowercase
            frequency[char] = frequency.get(char, 0) + 1
```

```
    # Convert the dictionary to a list of tuples for sorting
    frequency_items = list(frequency.items())

    # Sort the list manually using a nested loop (bubble sort for simplicity)
    for i in range(len(frequency_items)):
        for j in range(len(frequency_items) - i - 1):
            if frequency_items[j][1] < frequency_items[j + 1][1]:  # Compare frequencies
                # Swap the elements
                frequency_items[j], frequency_items[j + 1] = frequency_items[j + 1],
frequency_items[j]

    # Print the letters and their frequencies
    for letter, freq in frequency_items:
        print(f"{letter}: {freq}")

# Example usage
text = input("Enter a string: ")
list_of_frequency(text)
```

Output

Enter a string: Hello World!

l : 3
o : 2
h : 1
e : 1
w : 1
r : 1
d : 1

8. **Write a python program to read the list of numbers and sort the list in a non-descending order without using any built-in functions. Separate function should be written to sort the list wherein the name of its list is passed as parameter.**

Algorithm

Step 1: Start

Step 2: Define a function named **sort_list** that takes a list of numbers as input.

Step 3: Perform bubble sort inside the function:

- Iterate through the list.
- Compare adjacent elements.
- Swap elements if the current element is greater than the next.

Step 4: Read input from the user and convert it into a list of integers.

Step 5: Call the **sort_list** function with the input list as the argument.

Step 6: Print the sorted list returned by the function.

Step 7: Stop

Source Code

```
def sort_list(numbers):
    for i in range(len(numbers)):
        for j in range(len(numbers) - i - 1):
            if numbers[j] > numbers[j + 1]:
                numbers[j], numbers[j + 1] = numbers[j + 1], numbers[j]
    return numbers


numbers = list(map(int, input("Enter numbers separated by spaces: ").split()))
sorted_numbers = sort_list(numbers)
print("Sorted List in Non-Descending Order:", sorted_numbers)
```

Output

```
Enter numbers separated by spaces:  10 20 0 25 10
Sorted List in Non-Descending Order: [0, 10, 10, 20, 25]
```

9. **Given the Sales information of a supermarket as a CSV file with the following fields month_number, Soap, facewash, toothpaste, shampoo, moisturiser, total_units, total_profit. Write Python code to visualize the data as follows.**
    1) **Toothpaste sales data of each month and show it using scatter plot**
    2) **Soap and face wash product sales data and show it using the bar chart**
    3) **Calculate total sale data for the last year for each product and show it using a Pie chart.**

Algorithm

Step 1: Start.

Step 2: Import **pandas** and **matplotlib.pyplot**.

Step 3: Load the sales data from **sales_data.csv**.

Step 4: Plot toothpaste sales using a scatter plot.

Step 5: Plot soap and facewash sales using a bar chart.

Step 6: Plot total sales as a pie chart.

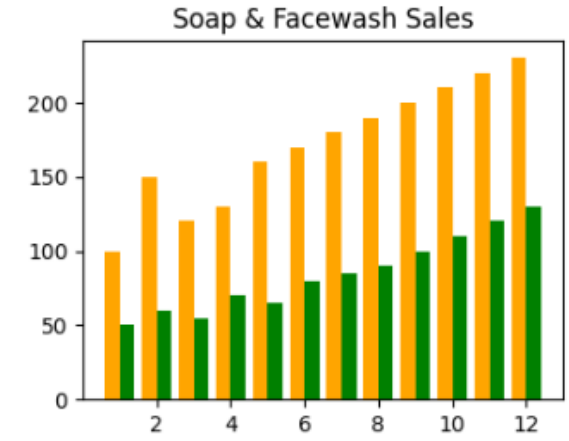
Step 7: Stop.

Source Code

```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("sales_data.csv")

plt.figure(figsize=(4, 3))
plt.scatter(data['month_number'], data['toothpaste'], color='blue')
plt.title('Toothpaste Sales')
plt.show()

plt.figure(figsize=(4, 3))
plt.bar(data['month_number'] - 0.2, data['Soap'], width=0.4, color='orange')
plt.bar(data['month_number'] + 0.2, data['facewash'], width=0.4, color='green')
```

```
plt.title('Soap & Facewash Sales')
plt.show()

plt.figure(figsize=(4, 3))
plt.pie(data.sum()[1:], labels=data.columns[1:], autopct='%1.1f%%')
plt.title('Total Sales')
plt.show()
```

Output



10. Write a Python script named copyfile.py that:

1) **Prompts the user to input two text file names.**
2) **Copies the content of the first file to the second file.**

Algorithm

Step 1: Start.

Step 2: Get the source and destination file names from the user.

Step 3: Open the source file in read mode and the destination file in write mode.

Step 4: Read the content from the source file and write it to the destination file.

Step 5: Print a message confirming the content has been copied.

Step 6: End.

Source Code

```
def copy_file():
    source_file = input("Enter the name of the source file: ")
    destination_file = input("Enter the name of the destination file: ")

    with open(source_file,"r") as src, open(destination_file,"w") as dst:
        content = src.read()
        dest.write(content)
    print(f"Content copied from {source_file} to {destination_file}")

copy_file()
```

Output

```
Enter the name of the source file : source.txt
Enter the name of the destination file : destination.txt
Content copied from source.txt to destination.txt
```

11. Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate dataset for building the decision tree and apply this knowledge to classify a new sample.

Algorithm

Step 1: Start.

Step 2: Import the necessary libraries

Step 3: Load the Iris dataset using **load_iris()**.

Step 4: Split the dataset into features (**X**) and target labels (**y**).

Step 5: Split the data into training and testing sets using **train_test_split()**.

Step 6: Create a DecisionTreeClassifier using **criterion='entropy'**.

Step 7: Train the model using **clf.fit(X_train, y_train)**.

Step 8: Make predictions on the test data using **clf.predict(X_test)**.

Step 9: Calculate the accuracy of the model using **accuracy_score()**.

Step 10: Print the accuracy of the classifier.

Step 11: Create a new sample (feature values).

Step 12: Use the trained classifier to predict the class of the new sample.

Step 13: Print the predicted class for the new sample.

Step 14: Stop.

Source Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris

data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

clf = DecisionTreeClassifier(criterion='entropy', random_state=42)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy of the Decision Tree Classifier: {accuracy * 100:.2f}%')

new_sample = [[5.1, 3.5, 1.4, 0.2]]
predicted_class = clf.predict(new_sample)

print(f'The predicted class for the new sample is: {data.target_names[predicted_class][0]}')
```

Output

```
Accuracy of the Decision Tree Classifier: 97.78%
The predicted class for the new sample is: setosa
```

12. Write a program to implement the naive Bayesian classifier for a sample training dataset stored as a .CSV file. Compute the accuracy of the classifier, considering a few test data sets.

Algorithm

Step 1: Start.

Step 2: Import necessary libraries.

Step 3: Load the dataset from the CSV file.

Step 4: Split the dataset into features (**X**) and target (**y**).

Step 5: Split the data into training and testing sets.

Step 6: Create a Naive Bayes classifier.

Step 7: Train the classifier with the training data.

Step 8: Predict the target values using the test data.

Step 9: Calculate the accuracy of the model.

Step 10: Print the accuracy.

Step 11: Stop.

Source Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

data = pd.read_csv('your_dataset.csv')

X = data.iloc[:, :-1]  # All columns except the last one
y = data.iloc[:, -1]   # The last column as the target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy of the Naive Bayes classifier: {accuracy * 100:.2f}%')
```

Output

```
Accuracy of the Naive Bayes classifier: 85.25%
```