**1.** **WAP in Go language using blanks imports.**

package main

import (

"fmt"

\_ "image/png"

)

func main() {

fmt.Println("This program demonstrates blank imports in Go.")

fmt.Println("The 'image/png' package is imported to enable PNG support.")

}

**2.** **Write python program to read "car.csv" file.**

import pandas as pd

try:

# Replace 'car.csv' with the path to your file if necessary

data = pd.read\_csv('car.csv')

print("Contents of the CSV file:")

print(data)

except FileNotFoundError:

print("The file 'car.csv' was not found.")

except Exception as e:

print(f"An error occurred: {e}")

**3. WAP in Go language to demonstrate init() function**

package main

import "fmt"

func init() {

fmt.Println("This is the init() function. It runs before the main() function.")

}

Fun4c main() {

fmt.Println("This is the main() function. The program starts here after init().")

}

**4.** **write a python program for aprior algorithm using ARM and print**

**the rule-**

import pandas as pd

import os

dataset=pd.read\_csv("groceries2.csv")

print(dataset.head(10))

transactions = []

# Add all the items from each row in one list( Neglect the 1st columns where all the items are in number (0-9))

for i in range(0, 100):

transactions.append([str(dataset.values[i,u]) for u in range(10, 32)])

from apyori import apriori

rules = apriori(transactions, min\_support=0.0040, min\_confidence=0.2, min\_lift=3, min\_length = 2)

results = list(rules)

# See the items that were bought together with their support

results\_list = []

for i in range(0, len(results)):

results\_list.append('RULE:\t' + str(results[i][0]) + '\nSUPPORT:\t' + str(results[i][1]))

print(dataset.head())

**5.** **Write a program in go language to create structure student. Write a method show() whose receiver is a pointer of struct student.**

**package main**

**import "fmt"**

**type Student struct {**

**Name string**

**Age int**

**Grade string**

**}**

**func (s \*Student) show() {**

**fmt.Printf("Student Details:\n")**

**fmt.Printf("Name: %s\n", s.Name)**

**fmt.Printf("Age: %d\n", s.Age)**

**fmt.Printf("Grade: %s\n", s.Grade)**

**}**

**func main() {**

**student := Student{**

**Name: "Alice",**

**Age: 20,**

**Grade: "A",**

**}**

**student.show()**

**}**

**6.** **Write a program in go language to demonstrate working type switch interface**

**package main**

**import "fmt"**

**func typeSwitchDemo(i interface{}) {**

**switch v := i.(type) {**

**case int:**

**fmt.Printf("The type is int, value: %d\n", v)**

**case string:**

**fmt.Printf("The type is string, value: %s\n", v)**

**case bool:**

**fmt.Printf("The type is bool, value: %t\n", v)**

**case float64:**

**fmt.Printf("The type is float64, value: %f\n", v)**

**default:**

**fmt.Printf("Unknown type, value: %v\n", v)**

**}**

**}**

**func main() {**

**typeSwitchDemo(42) // int**

**typeSwitchDemo("Hello") // string**

**typeSwitchDemo(true) // bool**

**typeSwitchDemo(3.14) // float64**

**typeSwitchDemo([]int{1, 2}) // Unknown type**

**}**

**7.** **Write a program in go language to demonstrate interface concept.**

**package main**

**import "fmt"**

**type Shape interface {**

**Area() float64**

**Perimeter() float64**

**}**

**type Rectangle struct {**

**Length, Width float64**

**}**

**func (r Rectangle) Area() float64 {**

**return r.Length \* r.Width**

**}**

**func (r Rectangle) Perimeter() float64 {**

**return 2 \* (r.Length + r.Width)**

**}**

**type Circle struct {**

**Radius float64**

**}**

**func (c Circle) Area() float64 {**

**return 3.14159 \* c.Radius \* c.Radius**

**}**

**func (c Circle) Perimeter() float64 {**

**return 2 \* 3.14159 \* c.Radius**

**}**

**func printShapeDetails(s Shape) {**

**fmt.Printf("Area: %.2f\n", s.Area())**

**fmt.Printf("Perimeter: %.2f\n", s.Perimeter())**

**}**

**func main() {**

**rect := Rectangle{Length: 10, Width: 5}**

**circle := Circle{Radius: 7}**

**fmt.Println("Rectangle:")**

**printShapeDetails(rect)**

**fmt.Println("\nCircle:")**

**printShapeDetails(circle)**

**}**

**8. Consider following observations/data. And apply simple linear regression and find out estimated coefficients b1 and b1 Also analyse the performance of the model (Use sklearn package)**

**import matplotlib.pyplot as plt**

**from scipy import stats**

**import numpy as np**

**x = np.array([1,2,3,4,5,6,7,8])**

**y = np.array([7,14,15,18,19,21,26,23])**

**slope, intercept, r, p, std\_err = stats.linregress(x, y)**

**def myfunc(x):**

**return slope \* x + intercept**

**mymodel = list(map(myfunc, x))**

**plt.scatter(x, y)**

**plt.plot(x, mymodel)**

**plt.show()**

**9.** **Write a program in go language to create an interface and display it’s values with the help of type assertion.**

**package main**

**import "fmt"**

**type Any interface{}**

**func main() {**

**values := []Any{42, "Hello, World!", 3.14, true}**

**for \_, value := range values {**

**switch v := value.(type) {**

**case int:**

**fmt.Printf("Type: int, Value: %d\n", v)**

**case string:**

**fmt.Printf("Type: string, Value: %s\n", v)**

**case float64:**

**fmt.Printf("Type: float64, Value: %.2f\n", v)**

**case bool:**

**fmt.Printf("Type: bool, Value: %t\n", v)**

**default:**

**fmt.Printf("Unknown Type: %T, Value: %v\n", v, v)**

**}**

**}**

**}**

**10. ) Write a program in go language to demonstrate working embedded interfaces.**

**package main**

**import "fmt"**

**type Reader interface {**

**Read() string**

**}**

**type Writer interface {**

**Write(data string)**

**}**

**type ReadWriter interface {**

**Reader**

**Writer**

**}**

**type File struct {**

**Content string**

**}**

**func (f \*File) Read() string {**

**return f.Content**

**}**

**func (f \*File) Write(data string) {**

**f.Content = data**

**fmt.Println("Data written to file.")**

**}**

**func main() {**

**file := &File{}**

**file.Write("Hello, Go!")**

**fmt.Println("Content of the file:", file.Read())**

**var rw ReadWriter = file**

**rw.Write("New Content")**

**fmt.Println("Updated content of the file:", rw.Read())**

**}**

**11.** **WAP in GO program printout the number from 0 to 5 waiting between 0 & 250 ms after each one using delay function.**

**package main**

**import ("fmt"**

**"math/rand"**

**"time"**

**)**

**func main() {**

**rand.Seed(time.Now().UnixNano())**

**for i := 0; i <= 5; i++ {**

**fmt.Println(i)**

**delay := time.Duration(rand.Intn(250)) \* time.Millisecond**

**time.Sleep(delay)**

**}**

**}**

**12.** **) Write python program to read "studentPerformance.csv" file.**

**solve the following.**

**-display shape of dataset**

**-display top rows of dataset**

**import numpy as np**

**import pandas as pd**

**import os**

**dataset=pd.read\_csv("StudentsPerformance.csv")**

**#To display the shape of dataset.**

**dataset**

**#To display the top rows of the dataset with their columns.**

**dataset.head()**

**#To display the number of rows randomly.**

**dataset.sample(5)**

**#To display the number of columns and names of the columns.**

**rows = len(dataset.axes[0])**

**cols = len(dataset.axes[1])**

**# Print the number of rows and columns**

**print("Number of Rows: " + str(rows))**

**print("Number of Columns: " + str(cols))**

**#name of columns**

**for col in dataset.columns:**

**print(col)**

**13.** **Write a go program to demonstrate channel buffering.**

**package main**

**import "fmt"**

**func main() {**

**ch := make(chan int, 3)**

**ch <- 1**

**ch <- 2**

**ch <- 3**

**fmt.Println("Buffered channel demonstration:")**

**fmt.Println(<-ch)**

**fmt.Println(<-ch)**

**fmt.Println(<-ch)**

**}**

**14.** **WAP in Go to create buffered channel, store few values in it and find channel capacity and length**

**package main**

**import "fmt"**

**func main() {**

**ch := make(chan int, 5)**

**ch <- 10**

**ch <- 20**

**ch <- 30**

**fmt.Println("Channel Capacity:", cap(ch))**

**fmt.Println("Channel Length:", len(ch))**

**fmt.Println("Retrieving values from the channel:")**

**for i := 0; i < len(ch); i++ {**

**fmt.Println(<-ch)**

**}**

**fmt.Println("Channel Length after retrieving all values:", len(ch))**

**}**

**15. ) Write a go program in go implement multiple goroutine function and schedule is determine by scheduler..**

**package main**

**import (**

**"fmt"**

**"math/rand"**

**"sync"**

**"time"**

**)**

**func worker(id int, wg \*sync.WaitGroup) {**

**defer wg.Done() // Mark this goroutine as done when it finishes**

**delay := rand.Intn(1000)**

**time.Sleep(time.Millisecond \* time.Duration(delay))**

**fmt.Printf("Worker %d completed after %d ms\n", id, delay)**

**}**

**func main() {**

**rand.Seed(time.Now().UnixNano())**

**var wg sync.WaitGroup**

**numWorkers := 5**

**for i := 1; i <= numWorkers; i++ {**

**wg.Add(1)**

**go worker(i, &wg)**

**}**

**wg.Wait()**

**fmt.Println("All workers completed")**

**}**

**16.** **WAP in go language how to create anonymous function**

**package main**

**import "fmt"**

**func main() {**

**add := func(a, b int) int {**

**return a + b**

**}**

**result := add(10, 20)**

**fmt.Printf("The sum is: %d\n", result)**

**func(message string) {**

**fmt.Println("Message:", message)**

**}("Hello, Go!")**

**func() {**

**fmt.Println("This is an anonymous function with no parameters.")**

**}()**

**}**

**17.** **WAP in go language have concreate process it has operation that are either or receiving using channel.**

**package main**

**import (**

**"fmt"**

**"time"**

**)**

**func producer(ch chan int) {**

**for i := 1; i <= 5; i++ {**

**fmt.Printf("Producer: Sending %d\n", i)**

**ch <- i // Send data to the channel**

**time.Sleep(500 \* time.Millisecond)**

**}**

**close(ch)**

**}**

**func consumer(ch chan int) {**

**for value := range ch {**

**fmt.Printf("Consumer: Received %d\n", value)**

**time.Sleep(300 \* time.Millisecond)**

**}**

**fmt.Println("Consumer: Channel closed, no more data.")**

**}**

**func main() {**

**ch := make(chan int)**

**go producer(ch)**

**go consumer(ch)**

**time.Sleep(4 \* time.Second)**

**fmt.Println("Main function: Done.")**

**}**

**18.** **WAP in go language to demonstrate import function**

**package main**

**import (**

**"fmt"**

**"example.com/mathutils"**

**)**

**func main() {**

**sum := mathutils.Add(5, 10)**

**fmt.Printf("Sum: %d\n", sum)**

**product := mathutils.Multiply(5, 10)**

**fmt.Printf("Product: %d\n", product)**

**}**

**19.Write a program for K-Means algorithm.**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.cluster import KMeans**

**from sklearn.preprocessing import StandardScaler, normalize**

**from sklearn.decomposition import PCA**

**from sklearn.metrics import silhouette\_score**

**raw\_df = pd.read\_csv("GENERAL.csv")**

**raw\_df = raw\_df.drop('CUST\_ID', axis = 1)**

**raw\_df.fillna(method ='ffill', inplace = True)**

**raw\_df.head(2)**

**20.** **WAP in Go language to generate random number using int31().**

**package main**

**import (**

**"fmt"**

**"math/rand"**

**"time"**

**)**

**func main() {**

**rand.Seed(time.Now().UnixNano())**

**randomNumber := rand.Int31()**

**fmt.Printf("Random Number (Int31): %d\n", randomNumber)**

**}**

**21.** **WAP in go language to demonstrate panic statement**

**package main**

**import "fmt"**

**func main() {**

**defer fmt.Println("Deferred: This will execute even if panic occurs.")**

**fmt.Println("Starting the program...")**

**panic("Something went wrong!")**

**fmt.Println("This will not be printed.")**

**}**

**22.** **WAP in go language to demonstrate errorf().**

**package main**

**import (**

**"fmt"**

**)**

**func divide(a, b int) (int, error) {**

**if b == 0 {**

**return 0, fmt.Errorf("division by zero: cannot divide %d by %d", a, b)**

**}**

**return a / b, nil**

**}**

**func main() {**

**result, err := divide(10, 2)**

**if err != nil {**

**fmt.Println("Error:", err)**

**} else {**

**fmt.Printf("Result: %d\n", result)**

**}**

**result, err = divide(10, 0)**

**if err != nil {**

**fmt.Println("Error:", err)**

**} else {**

**fmt.Printf("Result: %d\n", result)**

**}**

**}**