

1.a) What are the different loop control statements available in Python? Explain with suitable examples.

Ans.1.a)

Python Loop Control Statements

Loop control statements change execution from their normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements.

The different Loop Control Statements available in Python -

1. Continue Statement

- The continue statement in Python returns the control to the beginning of the loop.
- It returns the control to the beginning of the loop.

```
# Prints all letters except 'e' and 's'
for letter in 'geeksforgeeks':
    if letter == 'e' or letter == 's':
        continue
    print('Current Letter :', letter)
```

2. Break Statement

- The break statement in Python brings control out of the loop.

```
for letter in 'geeksforgeeks':
    # break the loop as soon it sees 'e'
    # or 's'
    if letter == 'e' or letter == 's':
        break
    print('Current Letter :', letter)
```

3. Pass Statement

- We use pass statement in Python to write empty loops.
- Pass is also used for empty control statements, functions, and classes.

```
# An empty loop
for letter in 'geeksforgeeks':
    pass
print('Last Letter :', letter)
```

1.b) Differentiate between Method Overloading and Method Overriding.

Ans.1.b)

Sr. No.	Method overloading	Method overriding
1.	Compile-time polymorphism.	Run-time polymorphism.
2.	Helps to increase the readability of the program.	Used to grant the specific implementation of the method which is already provided by its parent class or superclass.
3.	It occurs within the class.	It is performed in two classes with inheritance relationships.
4.	May or may not require inheritance.	Always needs inheritance.
5.	Methods must have the same name and different signatures.	Methods must have the same name and same signature.
6.	The return type can or cannot be the same, but we just have to change the parameter.	The return type must be the same or co-variant.
7.	Static binding is being used	Dynamic binding is being used
8.	Poor performance due to compile time polymorphism.	It gives better performance. The reason behind this is that the binding of overridden methods is being done at runtime.
9.	Private and final methods can be overloaded.	Private and final methods can't be overridden.
10.	The argument list should be different while doing method overloading.	The argument list should be the same in method overriding.

2.a) Write Python code to determine whether the given string is a Palindrome or not.

Ans.2.a)

```
# function which return reverse of a string
def isPalindrome(s):
    return s == s[::-1]
# Driver code
s = input("Enter a String: ")
ans = isPalindrome(s)
if ans:
    print("Yes")
else:
    print("No")
```

2.b) Explain the difference between function and module.

Ans.2.b)

Feature	Function	Module
Definition	Defined using the <code>def</code> keyword.	Code stored in a separate file with a <code>.py</code>
Purpose	Performs a specific task or set of tasks.	Organizes related functions, variables, and classes.
Reusability	Can be called multiple times in the code.	Can be imported and reused in different parts of the program.
Scope	Typically smaller in scope, focused on a specific task.	Can have a broader scope, encompassing multiple related functionalities.
Structure	Consists of a block of code with a specific name.	Consists of a file containing one or more functions, variables, and classes.
Invocation	Called using the function name and arguments.	Imported using the <code>import</code> statement.

3.a) Write short note on (any 3)

i. Inheritance ii. Encapsulation

iii. Polymorphism iv. Abstraction

Ans.3.a)

4. Inheritance

Inheritance is the most important aspect of object-oriented programming, which simulates the real-world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object. By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class. It provides the re-usability of the code

ii. Encapsulation

Encapsulation is also an essential aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident

iii. Polymorphism

Polymorphism contains two words “poly” and “morphs”. Poly means many, and morph means shape. By polymorphism, we understand that one task can be performed in different ways. For example – you have a class animal, and all

animals speak. But they speak differently. Here, the “speak” behavior is polymorphic in a sense and depends on the animal. So, the abstract “animal” concept does not actually “speak”, but specific animals (like dogs and cats)

have a concrete implementation of the action “speak”.

iv. Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonyms because data abstraction is achieved through encapsulation. Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

3.b) Write a Python program to create a person class. Include attributes like name, country and date of birth. Implement a method to determine the person's age.

Ans.3.b)

```
from datetime import datetime

class Person:
    def __init__(self):
        self.name = input("Enter name: ")
        self.country = input("Enter country: ")
        date_of_birth_input = input("Enter date of birth (YYYY-MM-DD): ")
        self.date_of_birth = datetime.strptime(date_of_birth_input, "%Y-%m-%d").date()

    def calculate_age(self):
        age = datetime.now().year - self.date_of_birth.year - ((datetime.now().month,
            datetime.now().day) < (self.date_of_birth.month, self.date_of_birth.day))

        return age

# Create a Person instance with user input
person1 = Person()

# Calculate and print the person's age
print(f"{person1.name} is {person1.calculate_age()} years old.")
```

4.a) What is the lambda function? Write the characteristics of a lambda function. Explain the same with an example.

Ans.4.a)

Lambda Function

1. A lambda function is a small anonymous function.
2. A lambda function can take any number of arguments but can only have one expression.
lambda arguments : expression
3. The power of lambda is better shown when you use them as an anonymous function inside another function.
4. Use lambda functions when an anonymous function is required for a short period of time.

The characteristics of a lambda function in Python:

- **Anonymous:** Lambda functions are anonymous, meaning they don't have a name like regular functions defined with def.
- **Expression:** Lambda functions consist of a single expression, and the result of that expression is implicitly returned.
- **Syntax:** The syntax for a lambda function is lambda arguments: expression.
- **Limited Functionality:** Lambda functions are best suited for simple tasks, as they are restricted to a single expression.

Now, let's look at an example:

```
# Regular function
```

```
def add(x, y):  
    return x + y
```

```
result_regular = add(3, 5)  
print(result_regular) # Output: 8
```

```
# Lambda function
```

```
add_lambda = lambda x, y: x + y  
result_lambda = add_lambda(3, 5)  
print(result_lambda) # Output: 8
```

4.b) What do you mean by a constructor? List and describe various constructors used for converting to different data types.

Ans.4.b)

In Python, a constructor is a special method used for initializing an object. It is called automatically when an object is created. The most common constructor in Python is the `__init__` method, which initializes the attributes of the object. Constructors play a crucial role in setting up the initial state of an object.

Regarding converting to different data types, Python provides several constructors (also known as conversion functions) that allow you to convert between different data types.

Various constructors for type conversion:

1. `int(x, base=10)`:
 - Converts `x` to an integer. The `base` parameter specifies the base if `x` is a string.
2. `float(x)`:
 - Converts `x` to a floating-point number.
3. `str(x)`:
 - Converts `x` to a string.
4. `list(iterable)`:
 - Converts an iterable (e.g., a tuple or string) to a list.
5. `tuple(iterable)`:
 - Converts an iterable to a tuple.
6. `set(iterable)`:
 - Converts an iterable to a set.
7. `dict(kwargs)` or `dict(mapping, kwargs)`:
 - Creates a new dictionary. If you pass keyword arguments or a mapping, it initializes the dictionary with those values.

4.c) Write a Python program to create a class representing a Circle. Include methods to calculate its area and perimeter.

Ans.4.c)

```
# Import the math module to access mathematical functions like pi
import math

# Define a class called Circle to represent a circle
class Circle:
    # Initialize the Circle object with a given radius
    def __init__(self, radius):
        self.radius = radius

    # Calculate and return the area of the circle using the formula:  $\pi * r^2$ 
    def calculate_circle_area(self):
        return math.pi * self.radius**2

    # Calculate and return the perimeter (circumference) of the circle using the formula:  $2\pi * r$ 
    def calculate_circle_perimeter(self):
        return 2 * math.pi * self.radius

# Example usage
# Prompt the user to input the radius of the circle and convert it to a floating-point number
radius = float(input("Input the radius of the circle: "))

# Create a Circle object with the provided radius
circle = Circle(radius)

# Calculate the area of the circle using the calculate_circle_area method
area = circle.calculate_circle_area()

# Calculate the perimeter of the circle using the calculate_circle_perimeter method
perimeter = circle.calculate_circle_perimeter()

# Display the calculated area and perimeter of the circle
print("Area of the circle:", area)
print("Perimeter of the circle:", perimeter)
```

5.a) Explain the following functions of NumPy library with the help of examples.

i. max() ii. min() iii. sum() iv. mean() v. std()

Ans.5.a)

i. max

The max() function finds the maximum element from an array.

max element from the whole 1-D array

```
>>> arrayA.max()
```

```
8
```

max element from the whole 2-D array

```
>>> arrayB.max()
```

```
6
```

if axis=1, it gives column wise maximum

```
>>> arrayB.max(axis=1) array([6, 4])
```

if axis=0, it gives row wise maximum

```
>>> arrayB.max(axis=0)
```

```
array([4, 6])
```

ii. min

The min() function finds the minimum element from an array.

```
>>> arrayA.min()
```

```
-3
```

```
>>> arrayB.min()
```

```
2
```

```
>>> arrayB.min(axis=0)
```

```
array([3, 2])
```

iii. sum

The sum() function finds the sum of all elements of an array.

```
>>> arrayA.sum()
```

```
25
```

```
>>> arrayB.sum()
```

```
15
```

#axis is used to specify the dimension

#axis is used to specify the dimension

#means the sum of elements on the first row

#means the sum of elements on the first row

```
array([9, 6])
```

iv. mean

The mean() function finds the average of elements of the array.

```
>>> arrayA.mean()
```

```
3.125
```

```
>>> arrayB.mean()
```

```
3.75
```

```
>>> arrayB.mean(axis=0)
```

```
array([3.5, 4. ])
```

```
>>> arrayB.mean(axis=1)
```

```
array([4.5, 3. ])
```

v. std

The std() function is used to find standard deviation of an array of elements.

```
>>> arrayA.std()
```

```
3.550968177835448
```

```
>>> arrayB.std()
```

```
1.479019945774904
```

```
>>> arrayB.std(axis=0)
```

```
array([0.5, 2. ])
```

```
>>> arrayB.std(axis=1)
```

```
array([1.5, 1. ])
```

5.b) Create the following NumPy arrays:

- i. 1-D array called zeros having 10 elements and all the elements are set to zero.
- ii. 1-D array called vowels having the elements 'a', 'e', 'i', 'o' and 'u'.
- iii. 2-D array called ones having 2 rows and 5 columns and all the elements are set to 1 and dtype as int.

Ans.5.b)

```
import numpy as np

# i. 1-D array called zeros having 10 elements and all elements set to zero.
zeros = np.zeros(10)

# ii. 1-D array called vowels having the elements 'a', 'e', 'i', 'o', and 'u'.
vowels = np.array(['a', 'e', 'i', 'o', 'u'])

# iii. 2-D array called ones having 2 rows and 5 columns, all elements set to 1, and dtype as int.
ones = np.ones((2, 5), dtype=int)

# Printing the arrays
print("i. 1-D array 'zeros':", zeros)
print("ii. 1-D array 'vowels':", vowels)
print("iii. 2-D array 'ones':\n", ones)
```

Explanation:

- `np.zeros(10)` creates a 1-dimensional array with 10 elements, all set to zero.
- `np.array(['a', 'e', 'i', 'o', 'u'])` creates a 1-dimensional array with the specified vowels.
- `np.ones((2, 5), dtype=int)` creates a 2-dimensional array with 2 rows and 5 columns, all elements set to 1, and the data type set to integer.

6.a) Write difference between Pandas series and NumPy arrays

Ans.6.a)

Feature	Pandas Series	NumPy Arrays
Data Types	Supports various data types, including custom ones.	Homogeneous data types; all elements must be of the same type.
Indexing	Can have custom row indices, similar to a dictionary.	Typically integer-based indexing, similar to lists in Python.
Size Mutability	Size can be changed dynamically.	Size is fixed upon creation.
Missing Values	Can handle missing data with NaN (Not a Number).	Does not have built-in support for missing data; uses NaN for floating-point arrays.
Operations	Supports both element-wise operations and vectorized operations.	Primarily supports vectorized operations.
Functionality	Designed for data manipulation and analysis; includes features like label-based indexing.	Designed for numerical and mathematical operations; lacks some data analysis features.
Usage	Suited for working with labeled data and tabular structures.	Suited for numerical and mathematical computations.
Dependencies	Built on top of NumPy, extends its functionality.	Fundamental to scientific computing in Python; provides the foundation for Pandas.

6.b) Explain any 5 attributes of data frames in pandas.

Ans.6.b)

Attribute Name	Purpose	Example
DataFrame.index	to display row labels	<pre>>>> ForestAreaDF.index Index(['GeoArea', 'VeryDense', 'ModeratelyDense', 'OpenForest'], dtype ='object')</pre>
DataFrame.columns	to display column labels	<pre>>>> ForestAreaDF.columns Index(['Assam', 'Kerala', 'Delhi'], dtype='object')</pre>
DataFrame.dtypes	to display data type of each column in the DataFrame	<pre>>>> ForestAreaDF.dtypes Assam int64 Kerala int64 Delhi float64 dtype: object</pre>
DataFrame.shape	to display a tuple representing the dimensionality of the DataFrame	<pre>>>> ForestAreaDF.shape (4, 3)</pre> It means ForestAreaDF has 4 rows and 3 columns.
DataFrame.size	to display a tuple representing the dimensionality of the DataFrame	<pre>>>> ForestAreaDF.size 12</pre> This means the ForestAreaDF has 12 values in it.

6.c) Create the following Data Frame Sales containing year wise sales figures for five: sales persons in INR. Use the years as column labels, and sales person names as row labels.

	2014	2015	2016	2017
Ankit	100.5	1200	2000	4000
Nazneen	150.6	1800	5000	5500
Sumit	200.9	2200	4000	6500
Akshada	3000	5000	7000	8000
Shruti	5000	6000	8000	10000

Ans.6.c)

```
import pandas as pd

# Data
data = {
    'Ankit': [100.5, 1200, 2000, 4000],
    'Nazneen': [150.6, 1800, 5000, 5500],
    'Sumit': [200.9, 2200, 4000, 6500],
    'Akshada': [3000, 5000, 7000, 8000],
    'Shruti': [5000, 6000, 8000, 10000]
}

# Years
years = ['2014', '2015', '2016', '2017']

# Creating the DataFrame
Sales = pd.DataFrame(data, index=years)

# Transposing the DataFrame to have sales persons as row labels
Sales = Sales.T

# Displaying the DataFrame
print(Sales)
```

Now, the DataFrame `Sales` has sales persons as row labels and years as column labels. The output should look like this:

	2014	2015	2016	2017
Ankit	100.5	1200	2000	4000
Nazneen	150.6	1800	5000	5500
Sumit	200.9	2200	4000	6500
Akshada	3000.0	5000	7000	8000
Shruti	5000.0	6000	8000	10000

7.a) Write the installation process of flask setup and discuss the advantages of Flask Framework.

Ans.7.a)

The installation process of flask setup:

1. Prerequisites:

- Make sure you have Python installed on your system. You can download Python from the official website: <https://www.python.org/downloads/>

2. Create a Virtual Environment (Optional but recommended):

- Open a terminal or command prompt.
- Navigate to your project directory.
- Run the following commands to create and activate a virtual environment:

```
# On Windows
python -m venv venv
venv\Scripts\activate

# On macOS/Linux
python3 -m venv venv
source venv/bin/activate
```

3. Install Flask:

- Once the virtual environment is activated, run the following command to install Flask:

```
pip install flask
```

This command installs Flask and its dependencies.

Advantages of Flask Framework:

1. Lightweight and Simple:

- Flask is designed to be simple and easy to use. It doesn't come with a lot of built-in features, allowing developers the flexibility to choose their components. This simplicity makes it easy to learn and quick to set up.

2. Modularity:

- Flask follows a modular design, and it's built around the concept of extensions. Developers can choose and add only the components they need for their specific project, making it a modular and customizable framework.

3. Extensibility:

- Flask is highly extensible. It provides a wide range of extensions that can be easily integrated into the application. These extensions include support for database integration, form validation, authentication, and more.

4. Werkzeug and Jinja2:

- Flask uses the Werkzeug WSGI toolkit for handling requests and responses, providing a solid foundation for web development. It also uses the Jinja2 templating engine, making it easy to render dynamic content in HTML templates.

5. RESTful Support:

- Flask is well-suited for building RESTful APIs. It provides tools and features that make it easy to design and implement RESTful services, making it a popular choice for developing web services and APIs.

6. Active Community:

- Flask has a vibrant and active community. This means there are plenty of resources, tutorials, and extensions available. The community actively contributes to the framework's development and provides support through forums and other channels.

7. Microservices Architecture:

- Flask is well-suited for building microservices due to its lightweight nature and modularity. It allows developers to create small, independent components that can work together, making it an excellent choice for microservices architecture.

8. Flexibility:

- Flask doesn't enforce a specific way of doing things. It gives developers the freedom to choose the tools and libraries they prefer. This flexibility allows developers to create applications tailored to their specific needs.

9. Excellent Documentation:

- Flask has extensive and well-documented official documentation. This makes it easy for developers to get started, find solutions to common problems, and explore advanced features.

These advantages contribute to Flask's popularity and make it a great choice for developers, especially those who value simplicity, flexibility, and modularity in their web development projects.

7.b) Explain the following commands.

i. `app.run(host, port, debug, options)`

ii. `@app.route('/home')`

Ans.7.b)

i. `app.run(host, port, debug, options)`

In the context of web development using a framework like Flask (a popular web framework for Python), `app.run()` is a command used to run the Flask application.

Parameters:

- **host:** Specifies the host where the application will be accessible. For example, setting it to `'0.0.0.0'` makes the app accessible from any external IP address.
- **port:** Specifies the port number on which the application will be running. Common choices include 5000.
- **debug:** If set to `True`, it enables debug mode, providing more detailed error messages and automatic reloading of the server when code changes are detected. In a production environment, it's usually set to `False`.
- **options:** Additional options for running the application.

ii. `@app.route('/home')`

In Flask, the `@app.route()` decorator is used to define routes for different parts of a web application. The `route()` decorator binds a function to a URL.

Parameter:

- **`/home`:** Specifies the URL route for which the decorated function will be executed.

8.a) Explain the following Flask HTTP methods:

i. GET ii. HEAD iii. POST iv. PUT v. DELETE

Ans.8.a)

Flask, like other web frameworks, supports various HTTP methods for data communication. Here are the different HTTP methods used in Flask:

1. GET:

This is the most common method used to retrieve data from a server. It sends data in an unencrypted form to the server. In Flask, the GET method is used by default if no other method is specified.

2. HEAD:

This method is similar to GET, but it is used without the response body. It is often used to retrieve metadata about a resource without actually fetching the resource itself.

4. POST:

The POST method is used to send data, typically in the form of form data, to the server. It is commonly used for submitting forms on websites. The data transmitted using the POST method is not shown in the URL on the browser.

1. PUT

This method is used to replace all the current representation of the target resource with the uploaded content. It is often used for updating existing resources on the server.

2. DELETE:

The DELETE method is used to delete the specified resource on the server. It removes all the current representation of the target resource specified in the URL.

These HTTP methods allow Flask to handle different types of requests and perform various operations on the server.

8.b) Design a code to print the table of 10 by Embedding Python statements in HTML.

Ans.8.b)

```
<!DOCTYPE html>
<html>
<head>
  <title>Table of 10</title>
</head>
<body>

  <h2>Table of 10</h2>

  <table border="1">
    <tr>
      <th>Number</th>
      <th>Result</th>
    </tr>

    <!-- Python code embedded in HTML -->
    {% for i in range(1, 11) %}
    <tr>
      <td>{{ i }}</td>
      <td>{{ i * 10 }}</td>
    </tr>
    {% endfor %}
    <!-- End of Python code -->

  </table>

</body>
</html>
```

9.a) Explain any one regression algorithm.

- i. Simple Linear Regression ii. Multiple Linear Regression
- iii. Support Vector Regression iv. Random Forest Regression

Ans.9.a)

Regression algorithms are a type of supervised machine learning algorithm used for predicting a continuous outcome variable (also known as the dependent variable) based on one or more predictor variables (independent variables).

i. Simple Linear Regression:

Simple Linear Regression is the most basic form of regression. It involves establishing a linear relationship between a single independent variable (feature) and a dependent variable. The goal is to find the best-fit line that minimizes the sum of the squared differences between the actual and predicted values. The equation for a simple linear regression model is typically represented as:

$$y = mx + b$$

where:

- y is the dependent variable,
- x is the independent variable,
- m is the slope of the line,
- b is the y -intercept.

The algorithm aims to find the optimal values for m and b to minimize the prediction errors.

ii. Multiple Linear Regression:

Multiple Linear Regression extends the concept of simple linear regression to include multiple independent variables. The model's equation becomes:

$$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n$$

where:

y is the dependent variable,

x_1, x_2, \dots, x_n are the independent variables,

b_0 is the y -intercept,

b_1, b_2, \dots, b_n are the coefficients for the respective independent variables.

The goal is to find the optimal values for the coefficients to minimize the prediction errors.

iii. Support Vector Regression (SVR):

Support Vector Regression is a regression technique based on Support Vector Machines (SVMs). It aims to find a hyperplane that best represents the relationship between the independent and dependent variables. SVR introduces a margin of tolerance to control the smoothness of the function. It is particularly useful when the relationship between variables is non-linear, and it uses a kernel trick to transform the input space.

iv. Random Forest Regression:

Random Forest Regression is an ensemble learning method that combines multiple decision trees to make predictions. Each tree is trained on a random subset of the data and produces its prediction. The final prediction is an average (for regression) or a voting (for classification) of the individual tree predictions. Random Forest Regression is robust, handles non-linear relationships well, and is less prone to overfitting.

These regression algorithms serve different purposes and are applicable in various scenarios based on the nature of the data and the problem at hand.

9.b) Differentiate between Supervised and Unsupervised learning.

Ans.9.b)

Parameters	Supervised machine learning	Unsupervised machine learning
Input Data	Algorithms are trained using labelled data.	Algorithms are used against data that is not labelled
Computational Complexity	Simpler method	Computationally complex
Accuracy	Highly accurate	Less accurate
No. of classes	No. of classes is known	No. of classes is not known
Data Analysis	Uses offline analysis	Uses real-time analysis of data
Algorithms used	Linear and Logistics regression, Random Forest, Support Vector Machine, Neural Network, etc.	K-Means clustering, Hierarchical clustering, Apriori algorithm, etc.
Output	Desired output is given.	Desired output is not given.
Training data	Use training data to infer model.	No training data is used.
Complex model	It is not possible to learn larger and more complex models than with supervised learning.	It is possible to learn larger and more complex models with unsupervised learning.
Model	We can test our model.	We cannot test our model.
Called as	Supervised learning is also called classification.	Unsupervised learning is also called clustering.
Example	Example: Optical character recognition.	Example: Find a face in an image.

10.a) Explain Naive Bayes Algorithm along with the example.

Ans.10.a)

The Naïve Bayes algorithm is a probabilistic machine learning algorithm based on Bayes' theorem. It is widely used for classification tasks, especially in natural language processing and text classification. The algorithm is considered "naïve" because it makes a strong independence assumption among the features, which simplifies the calculations and makes it computationally efficient.

Here's a step-by-step explanation of the Naïve Bayes algorithm:

3. Bayes' Theorem: Bayes' theorem is a fundamental probability formula that describes the probability of an event based on prior knowledge of conditions that might be related to the event. In the context of classification:
- $$P(\text{Class}|\text{Features})=P(\text{Features})P(\text{Features}|\text{Class})\cdot P(\text{Class})$$
 - $P(\text{Class}|\text{Features})$ is the probability of the class given the features.
 - $P(\text{Features}|\text{Class})$ is the probability of the features given the class.
 - $P(\text{Class})$ is the prior probability of the class.
 - $P(\text{Features})$ is the probability of the features.
4. Naïve Assumption: The Naïve Bayes algorithm assumes that the features are conditionally independent given the class. This means that the presence or absence of a particular feature does not affect the presence or absence of any other feature. While this assumption is often not strictly true in real-world data, it simplifies the computation significantly.
5. Training:
 - Given a labeled dataset with features and corresponding class labels, the algorithm calculates the prior probabilities $P(\text{Class})$ for each class and the conditional probabilities $P(\text{Features}|\text{Class})$ for each feature given the class.
 - These probabilities are used to build the model.
6. Prediction:
 - When a new set of features is presented, the algorithm uses Bayes' theorem to calculate the probability of each class given the features.
 - The class with the highest probability is predicted as the output.

Here's a simple example using text classification:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split

# Sample training data
documents = [
    ("Buy now, great deal!", "Spam"),
    ("Hi, how are you?", "Ham"),
    ("Discount offer", "Spam"),
    ("Meeting tomorrow?", "Ham")
]

# Separate the documents into features (X) and labels (y)
X, y = zip(*documents)

# Vectorize the text data (convert text into numerical features)
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.25, random_state=42)

# Train a Naïve Bayes classifier
classifier = MultinomialNB()
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the performance (e.g., using accuracy)
```

10.b) Differentiate between regression and classification algorithms.

Ans.10.b)

	Classification	Regression
1.	In this problem statement, the target variables are discrete.	In this problem statement, the target variables are continuous.
2.	Problems like Spam Email Classification , Disease prediction like problems are solved using Classification Algorithms.	Problems like House Price Prediction , Rainfall Prediction like problems are solved using regression Algorithms.
3.	In this algorithm, we try to find the best possible decision boundary which can separate the two classes with the maximum possible separation.	In this algorithm, we try to find the best-fit line which can represent the overall trend in the data.
4.	Evaluation metrics like Precision, Recall, and F1-Score are used here to evaluate the performance of the classification algorithms.	Evaluation metrics like Mean Squared Error , R2-Score , and MAPE are used here to evaluate the performance of the regression algorithms.
5.	Here we face the problems like binary Classification or Multi-Class Classification problems.	Here we face the problems like Linear Regression models as well as non-linear models.
6.	Input Data are Independent variables and categorical dependent variable.	Input Data are Independent variables and continuous dependent variable.
7.	Output is Categorical labels.	Output is Continuous numerical values.
8.	Objective is to Predict categorical/class labels.	Objective is to Predicting continuous numerical values.
9.	Example use cases are Spam detection, image recognition, sentiment analysis	Example use cases are Stock price prediction, house price prediction, demand forecasting.