PP QB Sessional Answers

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

Ans.1)

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.

```
# 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)
```

2. Differentiate between Method Overloading and Method Overriding.

Ans.2)

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.

3. Write short note on

i. Inheritance ii. Encapsulation

iii. Polymorphism iv. Abstraction

Ans.3)

i. Inheritance

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

ii. Encapsulation

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

iii. Polymorphism

Polymorphism contains two words "poly" and "morphs". Poly means many, andmorph means shape. By polymorphism, we understand that one task canbeperformed in different ways. For example - you have a class animal, andall

animals speak. But they speak differently. Here, the "speak" behavior ispolymorphic in a sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs andcats)

have a concrete implementation of the action "speak".

iv. Abstraction

Data abstraction and encapsulation both are often used as synonyms. Botharenearly 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 namecaptures the core of what a function or a whole program does.

4. Explain the difference between function and module.

Ans.4)

Feature	Function	Module
Definition	Defined using the def keyword.	Code stored in a separate file with a .py
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 import statement.

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

i. max ii. min iii. sum

iv. mean v. std

Ans.5)

i. max

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

```
# max element form the whole 1-D array
>>> arrayA.max()
8
 # max element form 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. ])
```

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

Ans.6)

```
# 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")
```

7. Explain Naïve Bayes Algorithm along with the example.

Ans.7)

Certainly! 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:

1. **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.**

2. **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.**

3. **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.**

4. **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)
```

8. Differentiate between regression and classification algorithms.

Ans.8)

	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.

9. Differentiate between Supervised and Unsupervised learning.

Ans.9)

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. Explain the following Flask HTTP methods

i. GET ii. HEAD iii. POST

iv. PUT v. DELETE

Ans.10)

i. GET

It is the most common method which can be used to send data in the unencryptedform to the server.

ii. HEAD

It is similar to the GET but used without the response body.

iii. POST

It is used to send the form data to the server. The server does not cache thedatatransmitted using the post method.

iv. PUT

It is used to replace all the current representation of the target resource withtheuploaded content.

v. DELETE

It is used to delete all the current representation of the target resource specifiedintheURL

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

Ans.11)

12. Write Difference between Pandas Series and NumPy Arrays.

Ans.12)

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.

13. Explain any 5 attributes of data frames in pandas.

Ans.13)

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

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

Ans.14)

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
```

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

Ans.15)

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.**

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

Ans.16)

```
# 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: π * 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: 2π * 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)
```