Comprehensive Review Packet

First & Last Name:

Date:

Module 1: Data Types in Python

Introduction:

This module provides an **in-depth introduction** to **data types** in Python, which are essential for **programming** as they determine the kinds of operations you can perform on your data and how it is stored. Understanding data types is crucial for effective coding and data manipulation. This module covers basic data types such as **strings**, **integers**, **floats**, and **booleans**, as well as more complex data types like **lists**, **tuples**, **sets**, and **dictionaries**. Through practical examples and exercises, you will learn how to use these data types effectively in your programs.

Key Terms and Concepts:

1. **String**: A sequence of characters used to represent text.
    - Example: `greeting = "Hello, World!"`
2. **Integer (int)**: Whole numbers without a fractional component.
    - Example: `age = 25`
3. **Floating Point Number (float)**: Numbers with a decimal point.
    - Example: `pi = 3.14`
4. **Boolean (bool)**: Represents one of two values: True or False.
    - Example: `is_student = False`
5. **List**: An ordered collection of items that can be of different types.
    - Example: `fruits = ['apple', 'banana', 'cherry']`
6. **Tuple**: An ordered, immutable collection of items.
    - Example: `coordinates = (10.0, 20.0)`
7. **Set**: An unordered collection of unique items.
    - Example: `unique_numbers = {1, 2, 3}`
8. **Dictionary**: A collection of key-value pairs where each key is unique.
    - Example: `student = {'name': 'Alice', 'age': 25}`
9. **Variable**: A name that refers to a value stored in a computer's memory.
    - Example: `score = 100`
10. **Function**: A reusable block of code that performs a specific task.
    - Example:

```
def greet(name):
return "Hello, " + name + "!"
```

Practice Problems:

1. **Write a Python program to define a string variable and print it.**

2. **Write a Python program to perform arithmetic operations on two integer variables.**

3. Write a Python program to calculate the area of a circle given its radius.

4. **Write a Python function that takes a list of numbers and returns the maximum value.**

5. **Write a Python program to create a tuple with the names of your favorite movies and print each name.**

6. **Write a Python program to demonstrate the use of sets by removing duplicates from a list.**

7. **Write a Python function that takes a dictionary and returns a list of its keys.**

8. **Write a Python program to demonstrate the use of boolean values and logical operators.**

9. **Write a Python function to concatenate two strings.**

10. **Write a Python program to convert a string to an integer and perform a calculation.**

Multiple Choice Questions:

1. **Which of the following is a string in Python?**

   a) 42

   b) 3.14

   c) “Hello, World!”

   d) True

2. **What will be the output of the following code?**

```
a = 10
b = 3
result = a / b
print(result)
```

   a) 3

   b) 3.0

   c) 3.33

   d) 3.3333333333333335

3. **Which of the following is a list in Python?**

a) {‘apple’, ‘banana’, ‘cherry’}

b) [‘apple’, ‘banana’, ‘cherry’]

c) (‘apple’, ‘banana’, ‘cherry’)

d) {“apple”: “red”, “banana”: “yellow”, “cherry”: “red”}

4. **What is the result of the following code?**

```
bool1 = True
bool2 = False
result = bool1 and bool2
print(result)
```

a) True

b) False

c) 1

d) 0

5. **Which method can be used to add an element to the end of a list?**

a) add()

b) append()

c) insert()

d) extend()

6. **What will be the output of the following code?**

```
numbers = {1, 2, 2, 3, 4, 4, 5}
print(len(numbers))
```

a) 5

b) 6

c) 7

d) 8

7. **How do you access the value associated with a key in a dictionary?**

a) dictionary.key

b) dictionary[key]

c) dictionary(key)

d) dictionary.get(key)

8. **Which of the following is a tuple in Python?**

   a) [1, 2, 3]

   b) (1, 2, 3)

   c) {1, 2, 3}

   d) {'1': 1, '2': 2, '3': 3}

9. **What is the data type of the following variable?**

```
my_var = 3.14
```

   a) int

   b) float

   c) str

   d) bool

10. **Which of the following statements is true about sets in Python?**

    a) Sets are ordered collections of elements.

    b) Sets can contain duplicate elements.

    c) Sets are immutable.

    d) Sets are unordered collections of unique elements.

Module 2: Lists, Loops, and Dictionaries in Python

Introduction:

This module provides a comprehensive introduction to **lists**, **loops**, and **dictionaries** in Python, essential constructs for **data manipulation** and **control flow** in programming. You will learn how to create and manipulate lists, use different types of loops to iterate over data, and utilize dictionaries

for efficient key-value pair storage. Practical examples and exercises will help you solidify your understanding of these concepts, preparing you for more advanced topics and real-world applications.

Key Terms and Concepts:

1. **List**: An ordered collection of items (elements) that can be of different types (e.g., integers, strings, floats). Lists are mutable, meaning their elements can be changed.
   - Example: `fruits = ['apple', 'banana', 'cherry']`
2. **Loop**: A programming construct that repeats a block of code multiple times based on certain conditions. Common types are for loops and while loops.
   - Example: `for i in range(5): print(i)`
3. **Dictionary**: A collection of key-value pairs where each key is unique. Dictionaries are mutable and can be used to store and retrieve data efficiently.
   - Example: `student = {'name': 'Alice', 'age': 25, 'courses': ['Math', 'Science']}`
4. **Function**: A reusable block of code that performs a specific task.

```
def greet(name):
print("Hello, "name + "!")
```

5. **For Loop**: A loop that iterates over a sequence (e.g., list, string, range) and executes a block of code for each element in the sequence.
   - Example: `for color in colors: print(color)`
6. **While Loop**: A loop that continues to execute a block of code as long as a specified condition is true.
   - Example:

```
i = 1
 while i < 5:
   print(i)
   i += 1
```

7. **Iteration**: The process of executing a set of statements repeatedly. Each repetition is called an iteration.
   - Example: `for number in range(10): print(number)`
8. **Range**: A built-in function that generates a sequence of numbers, commonly used in for loops.
   - Example: `range(5)` generates `[0, 1, 2, 3, 4]`
9. **Key**: An identifier used to access a value in a dictionary. Keys must be unique and immutable.
   - Example: In `student = {'name': 'Alice', 'age': 25}`, `'name'` and `'age'` are keys.

10. **Value**: The data associated with a key in a dictionary. Values can be of any data type.
    - Example: In `student = {'name': 'Alice', 'age': 25}`, `'Alice'` and `25` are values.

Practice Problems:

1. **Write a Python program to create a list of your favorite fruits and print each fruit using a for loop.**

2. **Write a Python function that takes a list of numbers and returns the sum of all the numbers.**

3. **Write a Python program to find the largest number in a list using a for loop.**

4. **Write a Python function that takes a list of strings and returns a new list with all the strings converted to uppercase.**

5. **Write a Python program to create a dictionary that maps the names of your friends to their favorite colors and print each friend's name and their favorite color.**

6. **Write a Python function that takes a dictionary and returns a list of all the keys.**

7. **Write a Python program to count the number of vowels in a string using a for loop.**

8. **Write a Python function that takes a list of integers and returns a new list containing only the even numbers.**

9. **Write a Python program to create a nested list (list of lists) and print each element using nested loops.**

10. **Write a Python program to create a dictionary from two lists, one containing keys and the other containing values, using a loop.**

Multiple Choice Questions:

1. **Which of the following is a list in Python?**

   a) {1, 2, 3}

   b) [1, 2, 3]

   c) (1, 2, 3)

   d) None of the above

2. **What will be the output of the following code?**

```
colors = ['red', 'green', 'blue']
print(colors[1])
```

   a) red

   b) green

   c) blue

   d) None of the above

4. **Which keyword is used to create a loop in Python?**

a) loop

b) for

c) while

d) both b and c

5. **What is the output of the following code?**

```
numbers = [1, 2, 3, 4, 5]
total = 0
for num in numbers:
    total += num
print(total)
```

a) 10

b) 11

c) 12

d) 15

6. **Which of the following is a correct dictionary in Python?**

a) {‘name’: ‘Alice’, ‘age’: 25}

b) [‘name’: ‘Alice’, ‘age’: 25]

c) (‘name’: ‘Alice’, ‘age’: 25)

d) {‘name’, ‘Alice’, ‘age’, 25}

7. **How do you access the value associated with a key in a dictionary?**

a) dictionary.key

b) dictionary[key]

c) dictionary(key)

d) dictionary.value(key)

8. **Which function is used to add an element to the end of a list in Python?**

a) append()

b) add()

c) insert()

d) extend()

9. What will be the output of the following code?

```
i = 1
while i < 4:
   print(i)
   i += 1
```

a) 1 2 3

b) 1 2 3 4

c) 1 2 3 4 5

d) None of the above

10. How do you create an empty dictionary in Python?

a) []

b) ()

c) {}

d) dict()

11. What does the following code print?

```
fruits = ['apple', 'banana', 'cherry']
for fruit in fruits:
   if fruit == 'banana':
       break
   print(fruit)
```

a) apple banana cherry

b) apple banana

c) apple

d) apple cherry

Module 3: Dictionaries, Inputs, and Loops in Python

Introduction:

This module covers the advanced usage of **dictionaries**, **user inputs**, and **loops** in Python. You will learn how to create, access, and manipulate dictionaries, handle user inputs, and use loops to perform complex data operations. By integrating these concepts, you will be able to develop more dynamic and interactive Python programs. The module includes detailed explanations, practical examples, and exercises to enhance your understanding and skills.

Key Terms and Concepts:

1. **Dictionary**: A collection of key-value pairs where each key is unique. Dictionaries are mutable, allowing for dynamic data storage and retrieval.
    - Example: `student = {'name': 'Alice', 'age': 25}`
2. **Key**: An identifier used to access a value in a dictionary.
    - Example: In `student = {'name': 'Alice', 'age': 25}`, `'name'` and `'age'` are keys.
3. **Value**: The data associated with a key in a dictionary.
    - Example: In `student = {'name': 'Alice', 'age': 25}`, `'Alice'` and `25` are values.
4. **For Loop**: A loop that iterates over a sequence (e.g., list, string, range) and executes a block of code for each element in the sequence.
    - Example: `for color in colors: print(color)`
5. **While Loop**: A loop that continues to execute a block of code as long as a specified condition is true.
    - Example:

```
i = 1
while i < 5:
   print(i)
   i += 1
```

6. **Input Function**: A function that reads a line from input, converts it to a string (unless a different type is specified), and returns it.
    - Example: `name = input("Enter your name: ")`
7. **Append Method**: Adds an element to the end of a list.
    - Example: `fruits.append('date')`
8. **Pop Method**: Removes and returns an item at a specified position.
    - Example: `item = list.pop(2)`
9. **List**: An ordered collection of items which can be of different types. Lists are mutable.

- Example: `fruits = ['apple', 'banana', 'cherry']`

10. **Nested Loop**: A loop inside another loop. The inner loop runs completely for each iteration of the outer loop.
    - Example:

```
for i in range(3):
    for j in range(2):
        print(i, j)
```

Practice Problems:

1. **Write a Python program to create a dictionary with the names of your friends and their ages. Print each friend's name and age.**

2. **Write a Python function that takes a dictionary and returns a list of all the keys.**

3. **Write a Python program to count the number of vowels in a string using a for loop.**

4. **Write a Python function that takes a list of integers and returns a new list containing only the even numbers.**

5. **Write a Python program to create a nested list (list of lists) and print each element using nested loops.**

6. **Write a Python function that takes a dictionary and prints each key-value pair on a new line.**

7. **Write a Python program to add a new key-value pair to a dictionary.**

8. **Write a Python function that removes a key-value pair from a dictionary.**

9. **Write a Python program to reverse a list using a for loop.**

10. **Write a Python program to filter a list of strings, keeping only those with more than 3 characters.**

Multiple Choice Questions:

1. **Which of the following is a correct way to define a dictionary in Python?**

   a) {‘name’: ‘Alice’, ‘age’: 25}

   b) [‘name’: ‘Alice’, ‘age’: 25]

   c) (‘name’: ‘Alice’, ‘age’: 25)

   d) {“name”, “Alice”, “age”, 25}

2. **What will be the output of the following code?**

```
student = {'name': 'Alice', 'age': 25}
print(student['age'])
```

   a) name

b) Alice

c) age

d) 25

3. **Which method is used to remove a key-value pair from a dictionary?**

a) remove()

b) delete()

c) del

d) pop()

4. **What is the output of the following code?**

```
numbers = [1, 2, 3, 4, 5]
result = []
for num in numbers:
   if num % 2 == 0:
       result.append(num)
print(result)
```

a) [1, 3, 5]

b) [2, 4]

c) [1, 2, 3, 4, 5]

d) []

5. **How do you access the value associated with a key in a dictionary?**

a) dictionary.key

b) dictionary[key]

c) dictionary(key)

d) dictionary.get(key)

6. **What does the following code print?**

```
fruits = ['apple', 'banana',
'cherry']
for fruit in fruits:
```

```
    if fruit == 'banana':
        continue
    print(fruit)
```

a) apple banana cherry

b) apple cherry

c) apple

d) banana cherry

7. **Which function is used to add an element to the end of a list in Python?**

a) append()

b) add()

c) insert()

d) extend()

8. **What will be the output of the following code?**

```
i = 1
while i < 4:
print(i)
i += 1
```

a) 1 2 3

b) 1 2 3 4

c) 1 2 3 4 5

d) None of the above

9. **Which of the following is a set in Python?**

a) {1, 2, 3}

b) [1, 2, 3]

c) (1, 2, 3)

d) {'1': 1, '2': 2, '3': 3}

10. Which of the following statements is true about lists in Python?

a) Lists are immutable.

b) Lists can contain duplicate elements.

c) Lists are unordered collections of elements.

d) Lists cannot be nested.

Module 4: Pandas and Data Frames

Introduction:

This module focuses on using **Pandas**, a powerful library in Python, to work with **data frames**. Pandas is essential for **data analysis** and **manipulation**, providing data structures and functions needed to clean, analyze, and visualize data. This module covers the creation, manipulation, and analysis of data frames, offering practical examples and exercises to solidify your understanding of these concepts.

Key Terms and Concepts:

1. **Pandas**: A Python library used for data manipulation and analysis.
    - Example: `import pandas as pd`
2. **Data Frame**: A two-dimensional, size-mutable, and heterogeneous tabular data structure with labeled axes (rows and columns).
    - Example:

```
data = {'Name': ['Alice', 'Bob',
'Charlie'], 'Age': [25, 30, 35]}
df = pd.DataFrame(data)
```

3. **Series**: A one-dimensional labeled array capable of holding any data type.
    - Example:

```
os = pd.Series([1, 2, 3, 4, 5,])
```

4. **Index**: The labels or identifiers for the rows in a data frame.
    - Example: `df.index`
5. **Column**: The labels or identifiers for the columns in a data frame.
    - Example: `df.columns`
6. **Reading Data**: Importing data from various file formats like CSV, Excel, SQL, etc.
    - Example:

```
df = pd.read_csv('data.csv')
```

7. **Filtering Data**: Selecting rows and columns that meet certain conditions.
    - Example:

```
filtered_df = df[df['Age'] > 30]
```

8. **Grouping Data**: Aggregating data based on certain criteria.
    - Example:

```
grouped_df = df.groupby('Age').mean()
```

9. **Merging Data**: Combining two data frames based on a common column.
    - Example:

```
merged_df = pd.merge(df1, df2, on='Name')
```

10. **Pivot Table**: A data summarization tool used to sort, group, and aggregate data.
    - Example:

```
pivot_df = df.pivot_table(values='Age', index='Name', aggfunc='mean')
```

Practice Problems:

1. **Write a Python program to create a DataFrame from a dictionary of lists and print it.**

```

```

2. **Write a Python program to calculate the mean age of a DataFrame.**

```

```

3. **Write a Python program to create a pivot table from a DataFrame.**

```

```

4. **Write a Python program to group data by the 'Name' column and calculate the mean age for each group.**

5. **Write a Python program to sort a DataFrame by the 'Age' column in descending order.**

6. **Write a Python program to add a new column 'Score' to a DataFrame.**

7. **Write a Python program to select rows where the age is greater than 30.**

8. **Write a Python program to read a CSV file into a DataFrame and print the first 5 rows.**

9. **Write a Python program to merge two DataFrames on a common column 'Name'.**

10. **Write a Python program to remove rows with missing values in a DataFrame.**

Multiple Choice Questions:

1. **What is a DataFrame in Pandas?**

   a) A one-dimensional array

b) A two-dimensional array

c) A two-dimensional, size-mutable, and heterogeneous tabular data structure

d) A collection of key-value pairs

2. **How do you read a CSV file into a DataFrame?**

   a) `pd.read_excel('data.csv')`

   b) `pd.read_sql('data.csv')`

   c) `pd.read_json('data.csv')`

   d) `pd.read_csv('data.csv')`

3. **How do you select rows where the age is greater than 30?**

   a) `df[df['Age'] > 30]`

   b) `df['Age'] > 30`

   c) `df[df['Age'] < 30]`

   d) `df['Age'] < 30`

4. **What function is used to calculate the mean of a column in a DataFrame?**

   a) `df.mean()`

   b) `df.median()`

   c) `df.mode()`

   d) `df.sum()`

5. **How do you group data by a column and calculate the mean for each group?**

   a) `df.groupby('column_name').sum()`

   b) `df.groupby('column_name').count()`

   c) `df.groupby('column_name').mean()`

   d) `df.groupby('column_name').min()`

6. **How do you merge two DataFrames on a common column 'Name'?**

a) `pd.concat([df1, df2])`

b) `df1.join(df2, on='Name')`

c) `pd.merge(df1, df2, on='Name')`

d) `df1.append(df2)`

7. **How do you create a pivot table in Pandas?**

a) `df.pivot('Age', 'Name')`

b) `df.pivot_table(values='Age', index='Name', aggfunc='mean')`

c) `df.pivot_table(index='Age', columns='Name')`

d) `df.pivot_table(index='Age', values='Name')`

8. **How do you add a new column 'Score' to a DataFrame?**

a) `df.insert('Score', [85, 90, 95])`

b) `df.append('Score', [85, 90, 95])`

c) `df['Score'] = [85, 90, 95]`

d) `df.add_column('Score', [85, 90, 95])`

9. **How do you remove rows with missing values in a DataFrame?**

a) `df.dropna(inplace=True)`

b) `df.remove_na()`

c) `df.dropna()`

d) `df.remove_na(inplace=True)`

10. **How do you sort a DataFrame by the 'Age' column in descending order?**

a) `df.sort_values(by='Age')`

b) `df.sort(by='Age', ascending=False)`

c) `df.sort_values(by='Age', ascending=False)`

d) `df.order_by('Age', ascending=False)`

Module 5: Functions and Modules in Python

Introduction:

This module explores the fundamentals of defining and using **functions** and **modules** in Python. Functions allow you to encapsulate code into **reusable blocks**, making your programs more organized and manageable. Modules enable you to structure your code into separate files and reuse code across different projects. This module includes detailed explanations, practical examples, and exercises to help you master these concepts. You will learn how to define functions, pass arguments, return values, and document functions using docstrings. Additionally, you will learn how to create and use modules to organize your code effectively.

Key Terms and Concepts:

1. **Function**: A reusable block of code that performs a specific task.

Example:

```
def greet(name):
    return "Hello, " + name + "!"
```

2. **Parameter**: A variable in a function definition that receives an argument passed to the function.
    - Example: `def add(a, b):`
3. **Argument**: A value passed to a function when it is called.
    - Example: `add(5, 3)`
4. **Return Statement**: A statement used in a function to send a value back to the caller.
    - Example: `return result`
5. **Docstring**: A string that describes a function's purpose, parameters, and return value.

Example:

```
def add(a, b):
    "
    Adds two numbers and returns the
result.
    Parameters:
    a (int): The first number.
    b (int): The second number.
    Returns:
    int: The sum of the two numbers.
    "
    return a + b
```

6. **Module**: A file containing Python code that can be imported and used in other Python programs.
    - Example: `import math`
7. **Import Statement**: A statement used to include the functionality of a module in a program.
    - Example: `import os`
8. **Standard Library**: A collection of modules included with Python that provides useful functionalities.
    - Example: `import random`
9. **Third-Party Library**: A collection of modules developed by third parties that can be installed and used in Python programs.
    - Example: `import numpy as np`
10. **Namespace**: A container that holds a set of identifiers (variable and function names) and their associated values.
    - Example: Global namespace, local namespace

Practice Problems:

1. **Write a Python function that calculates the factorial of a number.**

2. **Write a Python function that checks if a given string is a palindrome.**

3. **Write a Python program that imports the math module and uses it to calculate the square root of a number.**

4. **Write a Python function that takes a list of numbers and returns the average.**

5. **Write a Python program that imports a custom module and uses a function from it.**

6. **Write a Python function that takes a string and returns the number of vowels in it.**

7. **Write a Python program that imports the random module and uses it to generate a random number between 1 and 10.**

8. **Write a Python function that takes a list of words and returns a dictionary with the words as keys and their lengths as values.**

9. **Write a Python function that takes two numbers and returns their greatest common divisor (GCD).**

Multiple Choice Questions:

1. What is the correct syntax for defining a function in Python?

   a) `def function_name:`

   b) `def function_name():`

   c) `function function_name():`

   d) `def function_name{}`

2. Which statement is used to import a module in Python?

   a) `import module_name`

   b) `include module_name`

   c) `require module_name`

   d) `module module_name`

3. What is a docstring in Python?

   a) A string that describes the purpose of a module.

   b) A string that describes the purpose of a function.

   c) A string that describes the purpose of a variable.

d) A string that describes the purpose of a loop.

4. How do you calculate the square root of a number using the math module?

   a) `math.square(num)`

   b) `math.sqrt(num)`

   c) `math.root(num)`

   d) `math.sqroot(num)`

5. What is a module in Python?

   a) A reusable block of code that performs a specific task.

   b) A file containing Python code that can be imported and used in other Python programs.

   c) A collection of key-value pairs.

   d) A built-in function in Python.

6. What does the import statement do in Python?

   a) It includes the functionality of a module in a program.

   b) It defines a new function.

   c) It creates a new variable.

   d) It prints a message to the console.

7. How do you return a value from a function in Python?

   a) `send value`

   b) `give value`

   c) `return value`

   d) `output value`

8. Which of the following is part of the Python standard library?

   a) `numpy`

   b) `random`

   c) `requests`

   d) `beautifulsoup`

9. What is the purpose of the `random.randint(a, b)` function?

   a) To generate a random integer between `a` and `b` (inclusive).

   b) To generate a random integer between `a` and `b` (exclusive).

   c) To generate a random float between `a` and `b` (inclusive).

   d) To generate a random float between `a` and `b` (exclusive).

10. How do you create a custom module in Python?

    a) By defining a function inside a Python file.

    b) By creating a Python file with functions and importing it in another program.

    c) By installing a third-party library.

    d) By writing code inside the main script.

Module 6: Introduction to Machine Learning

Introduction:

This module explores the fundamentals of **Machine Learning (ML)**, a subset of artificial intelligence that enables computers to learn from data and make decisions. Understanding ML involves knowing different learning types, key concepts, and their applications. This module includes supervised, unsupervised, and reinforcement learning, and delves into practical examples and exercises to help solidify your understanding of these concepts.

Key Terms and Concepts:

1. **Machine Learning (ML)**: A subset of artificial intelligence that trains algorithms to make predictions or decisions based on data.
   - Example: Predicting house prices based on features like location, size, and number of bedrooms.
2. **Supervised Learning**: Uses labeled data to train the model. Examples include classification and regression tasks.
   - Example: Classifying emails as spam or not spam.
3. **Unsupervised Learning**: Uses unlabeled data to identify patterns. Examples include clustering and dimensionality reduction.
   - Example: Grouping customers into different segments based on purchasing behavior.
4. **Reinforcement Learning**: Uses trial and error to learn from actions and rewards. Often used in game playing and robotics.
   - Example: Training a robot to navigate a maze by rewarding it for reaching the end.
5. **Overfitting**: When a model learns the noise in the training data instead of the actual pattern, resulting in poor performance on new data.

    - Example: A model that performs exceptionally well on training data but poorly on test data.
6. **Underfitting**: When a model is too simple to capture the underlying pattern in the data, resulting in poor performance on both training and test data.
    - Example: A linear model trying to fit a non-linear data pattern.
7. **Bias**: The error introduced by approximating a real-world problem, which may be complex, by a simplified model.
    - Example: A model consistently predicting the same incorrect value regardless of the data.
8. **Variance**: The error introduced by the model's sensitivity to small fluctuations in the training set.
    - Example: A model that changes significantly with small changes in the training data.
9. **Feature Engineering**: The process of using domain knowledge to create new features that improve the performance of machine learning models.
    - Example: Creating a new feature by combining existing features, like "total_sales" by multiplying "unit_price" and "quantity_sold".
10. **Cross-Validation**: A technique for assessing how the results of a statistical analysis will generalize to an independent dataset.
    - Example: Splitting the data into k-folds and using each fold as a validation set while training on the remaining k-1 folds.

Practice Problems:

1. **Write a Python program to demonstrate a simple unsupervised learning task using k-means clustering.**

2. **Write a Python program to handle missing data by filling in missing values with the mean of the column.**

3. **Write a Python program to split a dataset into training and testing sets.**

4. **Write a Python program to evaluate a machine learning model using cross-validation.**

5. **Write a Python program to demonstrate a simple supervised learning task using linear regression.**

Multiple Choice Questions:

1. **What is Machine Learning (ML)?**

   a) A technique for building neural networks

   b) A programming language used for data analysis

   c) A subset of artificial intelligence that trains algorithms to make predictions or decisions based on data

   d) A method for storing large amounts of data

2. **Which of the following is NOT a type of Machine Learning?**

   a) Supervised Learning

   b) Unsupervised Learning

   c) Reinforcement Learning

   d) Transfer Learning

3. **What is the primary goal of supervised learning?**

   a) To group data into clusters

   b) To predict outcomes based on labeled input data

   c) To perform dimensionality reduction

   d) To identify patterns in unlabeled data

4. **Which of the following best describes overfitting?**

   a) The model performs well on both training and test data

b) The model performs poorly on training data but well on test data

c) The model learns the noise in the training data, resulting in poor performance on new data

d) The model is too simple to capture the underlying pattern in the data

5. **What is the purpose of cross-validation in machine learning?**

   a) To train the model on all available data

   b) To split the data into training and test sets

   c) To assess how well the model generalizes to an independent dataset

   d) To reduce the dimensionality of the data

6. **Which metric is NOT typically used for evaluating a classification model?**

   a) Accuracy

   b) Precision

   c) Recall

   d) Mean Squared Error

7. **What is feature engineering?**

   a) The process of training a machine learning model

   b) The process of creating new features from raw data to improve model performance

   c) The process of evaluating a model's performance

   d) The process of splitting the dataset into training and test sets

8. **Which algorithm is commonly used for clustering tasks?**

   a) Linear Regression

   b) K-Means

   c) Decision Trees

   d) Random Forest

9. **What is the main difference between supervised and unsupervised learning?**

   a) Supervised learning uses labeled data, while unsupervised learning uses unlabeled data

b) Supervised learning is faster than unsupervised learning

c) Supervised learning is used for clustering, while unsupervised learning is used for classification

d) There is no difference between supervised and unsupervised learning

10. **What is the purpose of the `fit()` function in scikit-learn?**

    a) To make predictions on new data

    b) To transform the data

    c) To train the machine learning model on the training data

    d) To evaluate the model's performance

Module 7: File Handling in Python

Introduction:

This module focuses on **file handling** in Python, covering how to **open**, **read**, **write**, and **close** files. File handling is an essential skill for data processing, allowing you to work with external data sources, store program output, and perform various file operations. You will learn about different file modes, context managers, and various methods to handle files effectively. This module provides practical examples and exercises to help you master file handling techniques in Python.

Key Terms and Concepts:

1. **File Object**: An object that represents a file and provides methods to work with it.
   - Example: `file = open('example.txt', 'r')`
2. **Opening a File**: The process of making a file available for reading or writing.
   - Example: `file = open('example.txt', 'r')`
3. **Reading a File**: The process of extracting data from a file.
   - Example: `content = file.read()`
4. **Writing to a File**: The process of adding data to a file.
   - Example: `file.write('Hello, World!')`
5. **Closing a File**: The process of freeing the resources associated with an open file.
   - Example: `file.close()`
6. **File Modes**: Modes that determine how a file is opened (e.g., read, write, append).
   - Example: `'r'` (read), `'w'` (write), `'a'` (append)
7. **Context Manager**: A construct that ensures resources are properly managed, often used with file operations.

Example:

```
with open('example.txt', 'r') as
file:
    content = file.read()
```

8. **Reading Lines**: The process of reading a file line by line.
    - Example: `lines = file.readlines()`
9. **Appending to a File**: Adding data to the end of an existing file without overwriting its content.

Example:

```
with open('example.txt', 'a') as
file:
    file.write('New line')
```

10. **File Path**: The location of a file in the file system.
    - Example: `file = open('/path/to/example.txt', 'r')`

Practice Problems:

1. **Write a Python program to open a file, read its content, and print it.**

2. **Write a Python program to write a string to a file.**

3. **Write a Python program to read a file line by line and print each line.**

4. **Write a Python program to append a string to an existing file.**

5. **Write a Python program to create a new file and write a list of strings to it.**

6. **Write a Python function that reads a file and returns its content as a list of lines.**

7. **Write a Python program to check if a file exists and print a message if it does not.**

8. **Write a Python program to count the number of words in a file.**

9. **Write a Python program to read a file and print only the first 5 lines.**

10. **Write a Python program to copy the content of one file to another file.**

Multiple Choice Questions:

1. **What is the correct syntax for opening a file in read mode in Python?**

   a) `open('file.txt', 'r')`

   b) `open('file.txt', 'w')`

   c) `open('file.txt', 'a')`

   d) `open('file.txt')`

2. **Which method is used to read the entire content of a file as a string?**

   a) `file.readlines()`

   b) `file.read()`

   c) `file.readline()`

   d) `file.readall()`

3. **How do you properly close a file in Python?**

   a) `file.end()`

   b) `file.close()`

c) `file.exit()`

d) `file.stop()`

4. **What is the purpose of the `'a'` mode in file handling?**

   a) To open a file for reading.

   b) To open a file for writing.

   c) To open a file for appending.

   d) To open a file for updating.

5. **What does the following code do?**

```
with open('file.txt', 'r') as file:
    content = file.read()
```

   a) It reads the content of 'file.txt' and closes the file.

   b) It writes content to 'file.txt' and closes the file.

   c) It appends content to 'file.txt' and closes the file.

   d) It deletes 'file.txt' and closes the file.

6. **How do you write multiple lines to a file in Python?**

   a) `file.write('line1\nline2\n')`

   b) `file.writelines(['line1', 'line2'])`

   c) `file.write_lines(['line1', 'line2'])`

   d) `file.writeall('line1', 'line2')`

7. **What is the purpose of the `os.path.exists()` function?**

   a) To check if a file is readable.

   b) To check if a file is writable.

   c) To check if a file or directory exists.

   d) To check if a file is executable.

8. **How do you read a file line by line in Python?**

   a) `file.read()`

b) `file.readlines()`

c) `file.readline()`

d) `for line in file:`

9. What is a context manager used for in file handling?

a) To open a file for reading.

b) To write content to a file.

c) To ensure that resources are properly managed.

d) To close a file.

10. What does the following code do?

```
with open('source.txt', 'r') as
source_file:
    content = source_file.read()
with open('destination.txt', 'w') as
destination_file:
    destination_file.write(content)
```

a) It reads content from ‘source.txt’ and appends it to ‘destination.txt’.

b) It reads content from ‘source.txt’ and writes it to ‘destination.txt’.

c) It writes content to both ‘source.txt’ and ‘destination.txt’.

d) It deletes ‘source.txt’ and ‘destination.txt’.