Assignment

Python Programming Questions and Answers

1. What is the difference between static and dynamic variables in Python?
   Answer: In Python, static variables are defined within a function and retain their value between function calls, typically using mutable data types. Dynamic variables, on the other hand, are created at runtime, allowing for flexible memory allocation and scope changes. For example, a static variable can be used to count function calls, while a dynamic variable can be created based on user input, changing its value and existence during program execution.

```
class MyClass:
    static_var = 0  # Static variable

    def __init__(self, dynamic_var):
        self.dynamic_var = dynamic_var  # Dynamic variable
```

2. **Mutable vs Immutable Data Types**:

   ○ i. **Mutable**: Data types that can be changed after creation. Example: Lists (list), Dictionaries
   ii. (dict).
   ○ iii. **Immutable**: Data types that cannot be changed after creation. Example: Tuples (tuple), Strings
   iv. (str).

3. **Dictionary Methods**:

   ○ i. pop(): Removes and returns an element from the dictionary.
   ○ ii. popitem(): Removes and returns the last inserted key-value pair.
   ○ iii. clear(): Removes all elements from the dictionary.

4. **Frozen Set**:

   ○ i. A frozenset is an immutable version of a set. Example:

   ii. fs = frozenset([1, 2, 3])

5. **Immutable Data Types**:

   ○ i. Data types that cannot be modified after creation. Example: Tuples, Strings.

6. 5. **__init_ in Python**:

- i. It is a constructor method used to initialize objects. Example:

  ii. class Example:
  iii. def _init
  (self, value):
  1. s
  e
  lf
  .
  v
  a
  l
  u
  e
  =
  v
  a
  l
  u
  e

7. 6. **Docstring and __repr_**:

- i. **Docstring**: A string literal used to document a module, class, or function.
- ii. **_repr_**: A method used to provide a string representation of an object.

8. **Unit Tests in Python**:

- i. Unit tests are a way to test individual units of source code to determine if they are fit for use. Python's unittestmodule is commonly used for this purpose.

9. **Break, Continue, and Pass in Python**:

- i. **Break**: Exits the current loop.
- ii. **Continue**: Skips the rest of the code inside the loop for the current iteration and moves to the next iteration.
- iii. **Pass**: A null statement used as a placeholder.

10. **Use of selfin Python**:

- i. selfrepresents the instance of the class. It is used to access variables and methods associated with the current object.

11. **Tuples**:
    a. Tuples are immutable sequences in Python. Example:

    b. my_tuple = (1, 2, 3)

12. What are lists and tuples? What is the key difference between the two?

Lists and tuples are both data structures in Python used to store collections of items. The key difference is that lists are mutable (can be changed), while tuples are immutable (cannot be changed after creation).

13. What is an Interpreted language & dynamically typed language? Write 5 differences between them.

    An interpreted language is executed line-by-line by an interpreter, while a dynamically typed language determines variable types at runtime.

    Execution: Interpreted languages run code directly; compiled languages convert code to machine code.
    Type checking: Dynamic typing checks types at runtime; static typing checks at compile time.
    Performance: Interpreted languages are generally slower than compiled.
    Error detection: Interpreted languages catch errors during execution; compiled languages catch errors during compilation.
    Flexibility: Dynamic typing allows more flexibility in coding.

14. What are Dict and List comprehensions?

    Dict and List comprehensions provide a concise way to create dictionaries and lists in Python.
    List comprehension: [expression for item in iterable]
    Dict comprehension: {key: value for item in iterable}
    Example:

```
squares = [x**2 for x in range(10)]
square_dict = {x: x**2 for x in range(10)}
```

15. What are decorators in Python? Explain it with an example. Write down its use cases.
    Decorators are functions that modify the behavior of another function. They are used for logging, access control, and caching.
    Example:

```
def my_decorator(func):
  def wrapper():
    print("Something is happening before the function is called.")
    func()
    print("Something is happening after the function is called.")
  return wrapper

@my_decorator
def say_hello():
  print("Hello!")

say_hello()
```

16. How is memory managed in Python?
    Memory management in Python involves automatic garbage collection, which frees up memory by removing unused objects. Python uses reference counting and a cyclic garbage collector to manage memory efficiently.

17. What is lambda in Python? Why is it used?

    A lambda function is a small anonymous function defined with the lambda keyword. It can take any number of arguments but has a single expression.

Example:
```
add = lambda x, y: x + y
```
It's used for short, throwaway functions, often in higher-order functions like map() and filter().

18. **Explain split() and join() functions in Python.**
    split(): Splits a string into a list based on a delimiter.
    ```
    text = "Hello World"
    words = text.split() # ['Hello', 'World']
    ```
    join(): Joins elements of a list into a string with a specified delimiter.
    ```
    words = ['Hello', 'World']
    text = " ".join(words) # 'Hello World'
    ```

19. **What are iterators, iterable & generators in Python?**
    Iterable: An object that can return an iterator (e.g., lists, tuples).
    Iterator: An object that implements the iterator protocol, with __iter__() and __next__() methods.
    Generator: A special type of iterator created using functions with the yield statement.

20. **What is the difference between xrange and range in Python?**
    In Python 2, range() returns a list, while xrange() returns an iterator, which is more memory efficient. In Python 3, range() behaves like xrange() from Python 2, returning an iterator.

21. **Pillars of OOP**
    The four pillars of Object-Oriented Programming (OOP) are:
    Encapsulation: Bundling data and methods that operate on the data.
    Abstraction: Hiding complex implementation details and showing only essential features.
    Inheritance: Creating new classes from existing ones.
    Polymorphism: Allowing different classes to be treated as instances of the same class through a common interface.

22. **How will you check if a class is a child of another class?**
    You can use the issubclass() function to check if a class is a child of another class.
    Example:
    ```
    class Parent: pass
    class Child(Parent): pass
    print(issubclass(Child, Parent)) # True
    ```

23. **How does inheritance work in Python? Explain all types of inheritance with an example.**
    Inheritance allows a class to inherit attributes and methods from another class.
    Types of inheritance:
    Single Inheritance: One class inherits from one parent class.
    ```
    class Parent: pass
    class Child(Parent): pass
    ```
    Multiple Inheritance: One class inherits from multiple parent classes.
    ```
    class Parent1: pass
    class Parent2: pass
    class Child(Parent1, Parent2): pass
    ```
    Multilevel Inheritance: A class inherits from another derived class.
    ```
    class Grandparent: pass
    class Parent(Grandparent): pass
    class Child(Parent): pass
    ```

Hierarchical Inheritance: Multiple classes inherit from the same parent class.
```
class Parent: pass
class Child1(Parent): pass
class Child2(Parent): pass
```

24. **What is encapsulation? Explain it with an example.**
    Encapsulation is the bundling of data and methods that operate on that data within a single unit, or class. It restricts direct access to some of an object's components.
    Example:
```
class Encapsulated:
  def __init__(self):
    self.__private_var = 42 # Private variable
  def get_private_var(self):
    return self.__private_var # Accessing private variable
obj = Encapsulated()
print(obj.get_private_var()) # 42
```

25. **What is polymorphism? Explain it with an example.**
    Polymorphism allows methods to do different things based on the object it is acting upon, even if they share the same name.
    Example:
```
class Dog:
  def speak(self):
    return "Woof!"
class Cat:
  def speak(self):
    return "Meow!"
def animal_sound(animal):
  print(animal.speak())
animal_sound(Dog()) # Woof!
animal_sound(Cat()) # Meow!
```

Question 1.(2)

Which of the following identifier names are invalid and why?

- **a) Serial_no.**: Invalid because it ends with a period.
- **b) lst_Rooms**: Valid.
- **c) total_marks**: Valid.
- **d) Total_Marks**: Valid.
- **e) total-Marks**: Invalid because it contains a hyphen.
- **f) _Marks**: Valid.
- **g) True**: Invalid because it’s a reserved word in Python.
- **h) Percentage**: Valid.

Question 1.(3)

Perform the following operations on the list:

```
name = ["Mohan", "das", "karam", "chandra", "gandhi", "Bapu"]
```

- **a) Find out the output of the following and explain:**

  ```
  print(name[2:6])
  ```

  Output: ['karam', 'chandra', 'gandhi', 'Bapu'] **Explanation**: This slices the list from index 2 to 5 (6 is excluded).

  b) find the output of the following ,and explain how?

  ```
  name = [“freedomFighter”,"Bapuji","MOhan" "dash", "karam",
  "chandra","gandhi"]
  length1=len((name[-len(name)+1:-1:2]))
  length2=len((name[-len(name)+1:-1]))
  print(length1+length2)
  ```

- **c) Add an element ‘freedom_fighter’ at the 0th index:**

  ```
  name.insert(0, 'freedom_fighter')
  ```

- **d) Add two more elements in the name list:**

  ```
  name.extend(['Bapuji',
               'H.H'])
  ```

Perform the following operations on the list:

```
name = ["freedom_fighter", "Bapuji",
"H.H"] length = len(name +
["karamchand"]) name.insert(0,
"Gandhi") name.extend(["Nehru"])
print(len(name[1:len(name)-1]))
```

- **c) Add two more elements at the end of the name list:**

  ```
  name.extend(['NetaJi', 'Bose'])
  ```

Question 1.(4)

Find the output of the following:

```
animal = ['Human', 'cat', 'mat', 'cat', 'rat', 'Human', 'Lion']
```

- **a) Count the number of times ‘Human’ appears in the list:**

  ```
  print(animal.count('Human'))
  ```
  Output: 2

- **b) Find the index of the element 'rat':**

  ```
  print(animal.index('rat'))
  ```

  Output: 4

- **c) Calculate the length of the list:**

  ```
  print(len(animal))
  ```

  Output: 7

Question 1.5

Tuple Operations:

```
tuple1 = (10, 20, "Apple", 3.4, 'a', ["master", "ji"], ("sita", "geeta", 22), [{"roll_no": 1, "name": "Navneet"}])
```

- **a) Print the length of the tuple:**

  ```
  print(len(tuple1))
  ```

  Output: 8

- **b) Print the name from the dictionary inside the tuple:**

  ```
  print(tuple1[-1][-1]["name"])
  ```

  Output: Navneet

- **c) Fetch the value of roll_no from this tuple:**

  ```
  roll_no = tuple1[-1][-1]["roll_no"]
  print(roll_no)
  ```

  Output: 1

- **d) Print the second element of the sixth item in the tuple:**

  ```
  print(tuple1[-3][1])
  ```

  Output: ji

- **e) Fetch the element "22" from this tuple:**

  ```
  element_22 = tuple1[-2][-1]
  ```

```
p
r
i
n
t
(
e
l
e
m
e
n
t
_
2
2
)
```

O
u
t
p
u
t
:
2
2

Question 1.6

Write a program to display the appropriate message as per the color of signal (RED-Stop/Yellow-Stay/Green-Go)
at the road crossing:

```
signal = input("Enter the signal color (RED/Yellow/Green): ").strip().lower()
if signal == "red":
	print("Stop")
elif signal == "yellow":
	print("Stay")
elif signal == "green":
	print("Go")
else:
	print("Invalid signal color")
```

Question 1.7

*Write a program to create a simple calculator performing only four basic operations (+, -, , /):

```
def calculator(a, b, operation):
	if operation == '+':
```

```
              r
              e
              t
              u
              r
              n
              a
              +
              b
            elif
          opera
           tion
          == '-':
              r
              e
              t
              u
              r
              n
              a
              -
              b
            elif
          opera
           tion
          == '*':
              r
              e
              t
              u
              r
              n
              a
              *
              b
            elif
          opera
           tion
          == '/':
           retur
           n a /
             b
           else:
                return "Invalid
                  operation"

a = float(input("Enter first number: "))
b = float(input("Enter second number: "))
operation = input("Enter operation (+, -, *, /): ")
print("Result:", calculator(a, b, operation))
```

Question 1.8

Write a program to find the larger of the three pre-specified numbers using ternary operators:

```
a, b, c = 10, 20, 15
largest = a if (a > b and a > c) else (b if b > c else c)
print("The largest number is:", largest)
```

Question 1.9

Write a program to find the factors of a whole number using a while loop:

```
number = int(input("Enter a number: "))
i = 1
factors = []
while i <= number:
    if number % i == 0:
        factors.append(i)
    i += 1
print("Factors of", number, "are:", factors)
```

Question 1.10

Write a program to find the sum of all the positive numbers entered by the user. As soon as the user enters a negative number, stop taking in any further input from the user and display the sum:

```
total_
sum =
0
while
True:
    num = int(input("Enter a number: "))
    if num < 0:
        break
    total_sum += num
print("The sum of all positive numbers is:", total_sum)
```

Question 1.11

Write a program to find prime numbers between 2 to 100 using nested for loops:

```
for num in range(2, 101):
    is_prime = True
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            is_prime =
                False
            break
    if is_prime:
        print(num, end=" ")
```

Question 1.12

Write the programs for the following:

- **Accept the marks of the student in five major subjects and display the same:**

  ```
  marks = []
  for i in range(5):
      mark = float(input(f"Enter marks for subject {i+1}: "))
      marks.append(mark)
  print("Marks:", marks)
  ```

- **Calculate the sum of the marks of all subjects. Divide the total marks by number of subjects (i.e. 5), calculate percentage = total marks/5 and display the percentage:**

  ```
  total_marks = sum(marks)
  percentage = total_marks / 5
  print("Total Marks:", total_marks)
  print("Percentage:", percentage)
  ```

- **Find the grade of the student as per the following criteria. Hint: Use Match & case for this:**

  ```
  grade = ""
   if percentage >= 85:
      grade = "A"
  elif percentage >= 75:
      grade = "B"
  elif percentage >= 50:
      grade = "C"
  elif percentage >= 30:
      grade = "D"
  else:
      grade = "Reappear"
  print("Grade:", grade)
  ```

Question 1.13

Write a program for VIBGYOR Spectrum based on their Wavelength using Python:

```
spectrum = {
    "Violet": (400,
    440), "Indigo":
    (440, 460),
    "Blue": (460,
    500), "Green":
    (500, 570),
    "Yellow": (570,
    590),
    "Orange":
```

```
        (590, 620),
        "Red": (620,
        720)
}

for color, (start, end) in spectrum.items():
        print(f"{color}: {start}nm - {end}nm")
```

Question 1.14

Calculate the gravitational attractions between Earth, Moon, and Sun in our solar system:

```
G = 6.67430e-11  # Gravitational constant in m^3 kg^-1 s^-
2 mass_earth = 5.972e24  # Mass of Earth in kilograms
mass_moon = 7.348e22  # Mass of Moon in kilograms
distance_earth_moon = 3.844e8  # Average distance between Earth and Moon in meters

mass_sun = 1.989e30  # Mass of Sun in kilograms
distance_earth_sun = 1.496e11  # Average distance between Earth and Sun in meters

# Calculate gravitational force between Earth and Moon
force_earth_moon = G * (mass_earth * mass_moon) / (distance_earth_moon ** 2)
print(f"Gravitational force between Earth and Moon: {force_earth_moon} N")

# Calculate gravitational force between Earth and Sun
force_earth_sun = G * (mass_earth * mass_sun) / (distance_earth_sun ** 2)
print(f"Gravitational force between Earth and Sun: {force_earth_sun} N")

# Compare the forces
if force_earth_sun > force_earth_moon:
         print("The gravitational force between Earth and Sun is stronger.")
else:
        print("The gravitational force between Earth and Moon is stronger.")


# Explain which celestial body (Earth or Moon) is more attracted to the other based on the comparison:
```

- The gravitational force between Earth and Sun is stronger than the force between Earth and Moon. Therefore, Earth is more attracted to the Sun compared to the Moon.

Question 2

Design and implement a Python program for managing student information using object-oriented principles:

```
class Student:
        def _init_(self, name, age, roll_number):
                self._name =
                name self._age
                = age
                self._roll_number = roll_number
        # Getter methods
        def get_name(self):
                return self._name

        def get_age(self):
                return self._age

        def get_roll_number(self):
                return self._roll_number

        # Setter methods
        def set_name(self, name):
                self._name = name

        def set_age(self, age):
                self._age = age

        def set_roll_number(self, roll_number):
                self._roll_number = roll_number

        # Method to display student information
        def display_info(self):
                print(f"Name: {self._name}, Age: {self._age}, Roll Number: {self._roll_number}")

        # Method to update student details
        def update_info(self, name=None, age=None, roll_number=None):
                if name:
                        self.set_name(name)
                if age:
                        self.set_age(age)
                if roll_number:
                        self.set_roll_number(roll_number)

# Creating instances and testing
student1 = Student("John Doe", 20, "A123")
student1.display_info()
student1.update_info(name="Jane  Doe",
age=21) student1.display_info()
```

Question 3

Develop a Python program for managing library resources efficiently:

```
class LibraryBook:
        def _init_(self, book_name, author, availability=True):
                self._book_name =
                book_name self._author =
                author
                self._availability = availability
```

```
# Method to borrow a
book def
borrow_book(self):
        if self._availability:
                self._availability = False
                print(f"The book '{self._book_name}' has been borrowed.")
            else:
                print(f"The book '{self._book_name}' is currently unavailable.")

    # Method to return a book def
    return_book(self):
            self._availability = True
            print(f"The book '{self._book_name}' has been returned.")

    # Method to display book details def
    display_book_info(self):
            status = "Available" if self._availability else "Unavailable"
            print(f"Book: {self._book_name}, Author: {self._author}, Status: {status}")

# Creating instances and testing
book1 = LibraryBook("1984", "George Orwell")
book1.display_book_info()
book1.borrow_book()
book1.display_book_info()
book1.return_book()
book1.display_book_info()
```

Question 4

Create a simple banking system using object-oriented concepts in Python:

```
class BankAccount:
    def _init_(self, account_number, balance=0): self.
            account_number = account_number self._balance
            = balance

    def deposit(self, amount):
            self._balance += amount
            print(f"Deposited {amount}. New balance is {self._balance}.")

    def withdraw(self, amount):
            if amount <= self._balance:
                self._balance -= amount
                print(f"Withdrew {amount}. New balance is {self._balance}.")
            else:
                print("Insufficient funds.")

    def get_balance(self):
            return self._balance

class SavingsAccount(BankAccount):
    pass

class CheckingAccount(BankAccount):
    pass

# Creating instances and testing savings =
SavingsAccount("S123") checking =
CheckingAccount("C123", 500)
savings.deposit(100) checking.withdraw(200)
print(f"Savings Balance: {savings.get_balance()}")
print(f"Checking Balance: {checking.get_balance()}")
```

Question 5

Write a Python program that models different animals and their sounds:

```
class Animal:
    def make_sound(self):
        pass

class Dog(Animal):
    def make_sound(self):
        print("Woof!")

class Cat(Animal):
     def make_sound(self):
         print("Meow!")

# Creating instances and testing dog =
Dog()
cat = Cat()
dog.make_sound()
cat.make_sound()
```

Question 6

Write a code for Restaurant Management System Using OOPS:

```
class MenuItem:
  def _init_(self, name, description, price, category):
    self._name = name
    self._description = description
    self._price = price
    self._category = category

  # Getter methods
  def get_name(self):
    return self._name

  def get_description(self):
    return self._description

  def get_price(self):
    return self._price

  def get_category(self):
    return self._category

  # Setter methods
  def set_name(self, name):
    self._name = name

  def set_description(self, description):
    self._description = description

  def set_price(self, price):
    self._price = price

  def set_category(self, category):
    self._category = category

  # Method to display menu item
  information def
  display_info(self):
        print(f"Name: {self._name}, Description: {self._description}, Price: {self._price}, Category: {self.
        category}")

# Inheriting from MenuItem to create FoodItem and
BeverageItem classes class FoodItem(MenuItem):
    pass

class BeverageItem(MenuItem):
    pass

# Creating instances and testing
food1 = FoodItem("Burger", "A delicious beef burger", 5.99, "Main Course")
beverage1 = BeverageItem("Coke", "Refreshing cola drink", 1.99, "Drink")
food1.display_info()
beverage1.display_info()
```

Question 7

Write a code for Hotel Management System using OOPS:

```
class Room:
    def _init_(self, room_number, room_type, rate, availability=True):
        self._room_number = room_number
        self._room_type = room_type
        self._rate = rate
        self._availability = availability

    # Method to
    book a room def
    book_room(self):
        if self._availability:
            self._availability = False
            print(f"Room {self._room_number} has been booked.")
        else:
            print(f"Room {self._room_number} is already booked.")

    # Method to check in
    a guest def
    check_in(self):
        if not self._availability:
            print(f"Guest checked into room {self._room_number}.")
        else:
            print(f"Room {self._room_number} is not booked yet.")

    # Method to check
    out a guest def
    check_out(self):
        if not self._availability:
            self._availability = True
            print(f"Guest checked out of room {self._room_number}.")
        else:
            print(f"Room {self._room_number} is already available.")

    # Method to display room
    information def
    display_info(self):
        status = "Available" if self._availability else "Booked"
        print(f"Room Number: {self._room_number}, Type: {self._room_type}, Rate: {self._rate}, Status:
        {status}")

# Creating instances and
testing room1 =
Room(101, "Deluxe",
150)
room1.display_info()
room1.book_room()
room1.display_info()
room1.check_in()

room1.check_out()
room1.display_info()
```

Question 8

Write a code for Fitness Club Management System using OOPS:

```
class Member:
    def _init_(self, name, age, membership_type, membership_status=True):
        self._name = name
        self._age = age
        self._membership_type = membership_type
        self._membership_status = membership_status

    # Method to register a
    new member def
    register_member(self):
        print(f"Member {self._name} has been registered.")

    # Method to renew
    membership def
    renew_membership(
    self):
        self._membership_status = True
        print(f"Membership for {self._name} has been renewed.")

    # Method to cancel
    membership def
    cancel_membership(s
    elf):
        self._membership_status = False
        print(f"Membership for {self._name} has been cancelled.")

    # Method to display member
    information def
    display_info(self):
        status = "Active" if self._membership_status else "Inactive"
        print(f"Name: {self._name}, Age: {self._age}, Membership Type: {self._membership_type}, Status:
        {status}")

# Creating instances and
testing member1 =
Member("Alice", 30, "Gold")
member1.display_info()
member1.register_member(
)
member1.renew_membersh
ip()
member1.cancel_membersh
ip() member1.display_info()
```

Question 9

Write a code for College Enrollment System Using OOPS:

```
class Student:
    def _init_(self, name, id_number):
        self._name = name
        self._id_number = id_number
        self._classes = []

    # Method to add a
    new student def
    add_student(self):
        print(f"Student {self._name} has been added.")

    # Method to update student information
    def update_student(self, name=None, id_number=None):
        if name:
            self._name = name
        if id_number:
            self._id_number = id_number
        print(f"Student information updated to Name: {self._name}, ID: {self._id_number}")

    # Method to remove
    a student def
    remove_student(self)
    :
        print(f"Student {self._name} has been removed.")

    # Method to enroll in a class
    def enroll_in_class(self, class_name):
        if class_name not in self._classes:
            self._classes.append(class_name)
            print(f"Student {self._name} enrolled in {class_name}")
        else:
            print(f"Student {self._name} is already enrolled in {class_name}")

    # Method to display student
    information def
    display_info(self):
        print(f"Name: {self._name}, ID: {self._id_number}, Classes: {', '.join(self._classes)}")

# Creating instances and
testing student1 =
Student("Bob", "S123")
student1.display_info()
student1.add_student()
student1.enroll_in_class("Ma
th")
student1.update_student(na
me="Robert")
student1.display_info()
student1.remove_student()
```

Question 10

Write a code for Airline Reservation System using OOPS:

```
class Flight:
    def _init_(self, flight_number, departure, arrival, seats):
        self._flight_number = flight_number
        self._departure = departure
        self._arrival = arrival
        self._seats = seats
        self._booked_seats = 0
    # Method to
    book a seat def
    book_seat(self):
        if self._booked_seats < self._seats:
            self._booked_seats += 1
            print(f"Seat booked on flight {self._flight_number}.")
        else:
            print(f"No seats available on flight {self._flight_number}.")


    #
    Meth
    od to
    cance
    l a
    reserv
    ation
    def
    cance
    l_rese
    rvatio
    n(self)
    :
        if self._booked_seats > 0:
            self._booked_seats -= 1
            print(f"Reservation cancelled on flight {self._flight_number}.")
        else:
            print(f"No reservations to cancel on flight {self._flight_number}.")

    # Method to get
    the remaining
    available seats
    def
    get_available_se
    ats(self):
        return self._seats - self._booked_seats

    # Method
    to display
    flight
    informatio
    n def
    display_inf
    o(self):
        print(f"Flight Number: {self._flight_number}, Departure: {self._departure}, Arrival: {self._arrival},
        Available Seats: {self.get_available_seats()}")

# Creating instances and testing
flight1 = Flight("AI101", "New York", "London", 200)
flight1.display_info()
flight1.book_seat()
flight1.display_info()
flight1.cancel_reservation()
flight1.display_info()
```

Question 13

Write a Python program using str.format() method for comma separation:

```
#
comma_sep
aration.py
```

```
number =
1000000
formatted_number = "{:,}".format(number)
print(formatted_number)
```

Question 14

Implement a Python package calculator.py containing functions for addition, subtraction, multiplication, and division:

```
#
calcul
ator.
py
def
add(a
, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b != 0:
        re
    turn a
    / b
    else:
        return "Division by zero is not allowed"
```

Question 15

Implement another module string_utilities.py with functions for string manipulation such as reversing a string:

```
    #
string_utilitie
  s.py def
reverse_strin
 g(s): return
   s[::-1]

def capitalize_string(s):
    return s.capitalize()
```

Question 16

Write and compile module named file_operations.py containing functions for writing, appending data to file and calculating statistics:

```
# file_operations.py
def write_file(file_path, data):
    with open(file_path, 'w') as file:
        file.wri
        te(data
          )

def append_file(file_path, data):
    with open(file_path, 'a') as file:
        file.wri
        te(data
           )

def calculate_statistics(numbers):
    mean = sum(numbers) / len(numbers)
    variance = sum((x - mean) ** 2 for x in numbers) / len(numbers)
    return mean, variance
```

Question 17

Develop a Python script to create an existing text file "employees.txt" and write the details of employees including their name, age, salary into the file:

```
#
create_employ
ees_file.py
employees = [
    {"name": "John Doe", "age": 30, "salary": 50000},
    {"name": "Jane Smith", "age": 25, "salary": 60000}
]

with open("employees.txt", 'w') as file:
    for employee in employees:
        file.write(f"Name: {employee['name']}, Age: {employee['age']}, Salary: {employee['salary']}\n")
```

Question 18

Write a Python program that opens an existing named "inventory.txt" in read mode and displays the contents of the file line by line:

```
# read_inventory_file.py
with open("inventory.txt", 'r') as file:
    for line in file:
        print(line.strip())
```

Question 19

Create a Python script that reads a text file named "expenses.txt" and counts the total amount spent on different expenses displayed in the text in human approachable order:

```
#
calculate_e
xpenses.py
total_expe
nses = 0

with open("expenses.txt", 'r') as file:
    for line in file:
        expense = float(line.strip().split()[-1])
        total_expenses +=
expense print(f"Total Expenses:
{total_expenses}")
```

Question 20

What do you mean by Measures of Central Tendency? Give measures of dispersion:

Measures of Central Tendency are statistical measures that describe the center or typical value of a dataset. The main measures are:

- **Mean**: The average of all data points.
- **Median**: The middle value when data points are ordered.

- **Mode**: The most frequently occurring value.

Measures of Dispersion describe the spread or variability of a dataset. The main measures are:

- **Range**: The difference between the highest and
- lowest values. **Variance**: The average of the squared
- differences from the mean. **Standard Deviation**: The
  square root of the variance.

Question 21
What do you mean by skewness? Explain its types and how to show skewness if it can be calculated: Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable. It
indicates whether the data points are skewed to the left (negative skew) or to the right (positive skew).

- **Positive Skew (Right Skew)**: The tail on the right side of the distribution is longer or fatter than the left
  side.
- **Negative Skew (Left Skew)**: The tail on the left side of the distribution is longer or fatter than the right side.

Graphs can be used to visualize skewness, showing how data points are distributed around the mean.

Question 22

Define PROBABILITY MASS FUNCTION (PMF) AND PROBABILITY DENSITY FUNCTION (PDF) and what is their connection between them? Explain what is meant by random experiment? Why continuous events like speed or light intensity cannot be explained in terms of PMFs?

A **Probability Mass Function (PMF)** gives the probability that a discrete random variable is exactly equal to some value. For a discrete random variable (X), the PMF is defined as:

$$ P(X = x) $$

A **Probability Density Function (PDF)** describes the likelihood of a continuous random variable to take on a particular value. The PDF is used to specify the probability of the random variable falling within a particular range of values.

For example, the PDF of a normal distribution is given by:

$$ f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} $$

where:

- (μ) is the mean.
- (σ) is the standard deviation.

Random Experiment: An experiment or process for which the outcome cannot be predicted with certainty. Continuous events like speed or light intensity cannot be explained in terms of PMFs because they take on an
infinite number of possible values within a given range, making it necessary to use PDFs instead.

Question 23: What is correlation? Explain its types in detail. What are the methods of determining correlation?

Correlation is a statistical measure that describes the extent to which two variables change together. If two variables tend to increase or decrease in tandem, they are said to have a positive correlation. If one variable tends to increase when the other decreases, they have a negative correlation.

Types of Correlation:

1. **Positive Correlation**: Both variables move in the same direction.

2. **Negative Correlation**: Variables move in opposite directions.
3. **No Correlation**: No predictable relationship between the variables.

Methods of Determining Correlation:

4. **Pearson's Correlation Coefficient**: Measures linear correlation between two variables.
5. **Spearman's Rank Correlation**: Measures the strength and direction of association between two ranked variables.
6. **Kendall's Tau**: Measures the association between two variables based on the ranks.

Question 24: Calculate the coefficient of correlation between the marks obtained by 10 students in Accountancy and Statistics.

Let's calculate the Pearson correlation coefficient using Python:

```
import numpy as np

# Marks obtained by students
accountancy = [45, 35, 70, 65, 40, 95, 50, 75, 80, 50]
statistics = [35, 30, 70, 60, 40, 90, 45, 60, 85, 60]

# Calculate the Pearson correlation coefficient
correlation_matrix = np.corrcoef(accountancy, statistics)
correlation_coefficient = correlation_matrix[0, 1]

print(f"The Pearson correlation coefficient is: {correlation_coefficient}")
```

Question 25

Discuss the 4 differences between correlation and regression:

1. **Purpose**:
   - **Correlation**: Measures the strength and direction of the relationship between two
   - variables. **Regression**: Predicts the value of a dependent variable based on the value of one or more independent variables.
2. **Symmetry**:
   - **Correlation**: Symmetric; the correlation between X and Y is the same as between Y and X.
   - **Regression**: Asymmetric; the regression of Y on X is different from the regression of X on Y.
3. **Units**:
   - **Correlation**: Dimensionless; it has no units.
   - **Regression**: Has units; the units of the dependent variable per unit of the independent variable.
4. **Interpretation**:
   - **Correlation**: Indicates the degree of linear relationship.
   - **Regression**: Provides an equation to predict the dependent variable.

Question 26

Find the most likely price at Delhi corresponding to the price of Rs. 70 at Agra from the following data:

Given:

- Coefficient of correlation ((r)) = 0.8
- Coefficient of variation ((CV_x)) = (\frac{10}{80} = \frac{1}{8})
- Coefficient of variation ((CV_y)) = (\frac{18}{144} = \frac{1}{8})

Using the formula for regression: [y = \overline{y} + r \left(\frac{\sigma_y}{\sigma_x} \right) (x -
\ove
rline
{x})]

Given (x = 70), (\overline{x} = 80), (\overline{y} = 144), (\sigma_x = 10), (\sigma_y = 18): [y = 144 + 0.8 \left(\frac{18}{10} \right) (70 - 80)] [y = 144 + 0.8 \times 1.8 \times (-10)] [y = 144 - 14.4] [y = 129.6]

The most likely price at Delhi is Rs.
129.6.

**Ques
tion
27**

In a partially destroyed laboratory record of an analysis of correlation data, the following results only are legible: Variance of x = 9, Regression equations are: (i) 8x-10y = -66; (ii) 40x - 18y = 214. What are (a) the mean values of x and y, (b) the coefficient of correlation between x and y, © the a of y:

(a) **Mean values of x and y**: From the regression equations: [8\overline{x} - 10\overline{y} = -66] [40\overline{x} - 18\overline{y} =
214]

Solving these equations simultaneously: [\overline{x} = 6, \overline{y} =
9]

(b) **Coefficient of correlation ®**: Given variance of x ((\sigma_x^2)) = 9, so (\sigma_x = 3).

From the regression coefficients: [b_{yx} = \frac{\sigma_y}{\sigma_x} \times r] [b_{xy} = \frac{\sigma_x}{\sigma_y} \times r]

Using the regression equations: [b_{yx} = \frac{10}{8} = 1.25] [b_{xy} = \frac{18}{40} = 0.45] [r = \sqrt{b_{yx} \times b_{xy}} = \sqrt{1.25 \times 0.45} = 0.75]
© **The a of y**: [a = \overline{y} - b_{yx} \times \overline{x}] [a = 9 - 1.25 \times 6 = 9 - 7.5 = 1.5]

Question 28

What is Normal Distribution? What are the four Assumptions of Normal Distribution? Explain in detail:
Normal Distribution is a continuous probability distribution characterized by a symmetric, bell-shaped curve.
It is defined by its mean (μ) and standard deviation (σ).

Four Assumptions of Normal Distribution:

1. **Symmetry**: The distribution is symmetric around the mean.
2. **Unimodal**: It has a single peak.
3. **Asymptotic**: The tails approach the horizontal axis but never touch it.
4. **Mean, Median, Mode**: All are equal and located at the center of the distribution.

Question 29

Write all the characteristics or Properties of the Normal Distribution Curve:

1. **Symmetry**: The curve is symmetric about the mean.
2. **Bell-shaped**: The highest point is at the mean.
3. **Mean, Median, Mode**: All are equal.
4. **Asymptotic**: The tails approach but never touch the horizontal axis.

5. **68-95-99.7 Rule**: Approximately 68% of data falls within 1 standard deviation, 95% within 2, and 99.7%
w
i
t
h
i
n
3
.

Question 30

Which of the following options are correct about Normal Distribution Curve:

(a) Within a range 0.6745 of a on both sides the middle 50% of the observations occur i.e., mean =0.67450 covers 50% area 25% on each side.

(b) Mean +1S.D. (i.e., $\mu + \sigma$) covers 68.268% area, 34.134 % area lies on either side of the mean.

© Mean +2S.D. (i.e., $\mu + 2\sigma$) covers 95.45% area, 47.725% area lies on either side of the mean.

(d) Mean +3 S.D. (i.e., $\mu + 3\sigma$) covers 99.73% area, 49.856% area lies on the either side of the mean. (e) Only 0.27% area is outside the range $\mu + 3\sigma$.

Question 31

The mean of a distribution is 60 with a standard deviation of 10. Assuming that the distribution is normal, what percentage of items be (i) between 60 and 72, (ii) between 50 and 60, (iii) beyond 72 and (iv) between 70 and 80:

(i) **Between 60 and 72**: [z = \frac{72 - 60}{10} = 1.2] Using z-tables, the area between 0 and 1.2 is approximately 38.21%.

(ii) **Between 50 and 60**: [z = \frac{60 - 50}{10} = 1] Using z-tables, the area between 0 and 1 is approximately 34.13%.

(iii) **Beyond 72**: [z = \frac{72 - 60}{10} = 1.2] Using z-tables, the area beyond 1.2 is approximately 11.51%.

(iv) **Between 70 and 80**: [z = \frac{70 - 60}{10} = 1] [z = \frac{80 - 60}{10} = 2] Using z-tables, the area between 1 and 2 is approximately 13.59%.

Question 32

15000 students sat for an examination. The mean marks was 49 and the distribution of marks had a standard deviation of 6. Assuming that the marks were normally distributed what proportion of students scored (a) more than 55 marks, (b) more than 70 marks:

(a) **More than 55 marks**: [z = \frac{55 - 49}{6} = 1] Using z-tables, the area beyond 1 is approximately 15.87%.

(b) **More than 70 marks**: [z = \frac{70 - 49}{6} = 3.5] Using z-tables, the area beyond 3.5 is approximately 0.02%.

Question 33

If the height of 500 students are normally distributed with mean 65 inch and standard deviation 5 inch. How many students have height: (a) greater than 70 inch, (b) between 60 and 70 inch:

(a) **Greater than 70 inch**: [z = \frac{70 - 65}{5} = 1] Using z-tables, the area beyond 1 is approximately 15.87%. [0.1587 \times 500 = 79.35 \approx 79]

(b) **Between 60 and 70 inch**: [z = \frac{60 - 65}{5} = -1] [z = \frac{70 - 65}{5} = 1] Using z-tables, the area between -1 and 1 is approximately 68.27%. [0.6827 \times 500 = 341.35 \approx 341]

Question 34

What is the statistical hypothesis? Explain the errors in hypothesis testing. Explain the Sample. What are Large Samples & Small Samples:

Statistical Hypothesis: A statistical hypothesis is an assumption or claim about a population parameter. It can be tested using statistical methods to determine whether it is likely to be true.

Errors in Hypothesis Testing:

1. **Type I Error (α)**: Rejecting the null hypothesis when it is true.
2. **Type II Error (β)**: Failing to reject the null hypothesis when it is false.

Sample: A sample is a subset of a population used to make inferences about the entire population.

Large Samples: Typically, a sample size greater than 30 is considered large. Large samples provide more reliable and accurate estimates of population parameters.

Small Samples: A sample size less than 30 is considered small. Small samples may not accurately represent the population and can lead to less reliable estimates.

Question 35

A random sample of size 25 from a population gives the sample standard deviation to be 9.0. Test the hypothesis that the population standard deviation is 10.5. Hint: Use chi-square distribution:

Given:

- Sample size ((n)) = 25
- Sample standard deviation ((s)) = 9.0
- Population standard deviation ((\sigma)) = 10.5

The test statistic for the chi-square test is: [\chi^2 = \frac{(n - 1) s^2}{\sigma^2}]

[\chi^2 = \frac{(25 - 1) \times 9.0^2}{10.5^2}] [\chi^2 = \frac{24 \times 81}{110.25}] [\chi^2 = \frac{1944}{110.25}] [\chi^2 \approx 17.63]

Using the chi-square distribution table, compare the calculated chi-square value with the critical value at the desired significance level (e.g., α = 0.05) for 24 degrees of freedom to determine whether to reject or fail to reject the null hypothesis.

Question 37

100 students of a PW IO1 obtained the following grades in Data Science paper: Grade [A, B, C, D, E] Total Frequency: [15, 17, 30, 22, 16, 100]. Using the chi-square test, examine the hypothesis that the distribution of grades is uniform:

Given:

- Observed frequencies: [15, 17, 30, 22, 16]
- Expected frequencies (assuming uniform distribution): [20, 20, 20, 20, 20]

The chi-square test statistic is: [\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}]
[\chi^2 = \frac{(15 - 20)^2}{20} + \frac{(17 - 20)^2}{20} + \frac{(30 - 20)^2}{20} + \frac{(22 - 20)^2}{20} + \frac{(16 - 20)^2}{20}] [\chi^2 = \frac{25}{20} + \frac{9}{20} + \frac{100}{20} + \frac{4}{20} + \frac{16}{20}] [\chi^2 = 1.25 + 0.45 + 5 + 0.2 + 0.8] [\chi^2 = 7.7]

Using the chi-square distribution table, compare the calculated chi-square value with the critical value at the desired significance level (e.g., α = 0.05) for 4 degrees of freedom to determine whether to reject or fail to reject the null hypothesis.

Question 38: ANOVA Test[2]

To study the performance of three detergents and three different water temperatures, the following whiteness readings were obtained with specially designed equipment[3].

Water temp:

- Cold
- Wat
- er
  War
  m
  Wat
  er
  Hot
  Wat
  er

Detergents:

- A: 57, 49, 54
- B: 55, 52, 46
- C: 67, 68, 58

Answer:

```
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

#
D
a
t
a
d
a
t
a
=
{
    'Detergent': ['A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C'],
    'WaterTemp': ['Cold', 'Warm', 'Hot', 'Cold', 'Warm', 'Hot', 'Cold', 'Warm', 'Hot'],
    'Whiteness': [57, 49, 54, 55, 52, 46, 67, 68, 58]
}

df = pd.DataFrame(data)

# ANOVA
model = ols('Whiteness ~ C(Detergent) + C(WaterTemp) + C(Detergent):C(WaterTemp)', data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)
```

Question 39: Basic Flask Route[4]

Answer:

```
from flask import
Flask app = Flask(
name_)
@app.route('/')
def hello_world():
    return 'Hello, World!'
if _name
== '_main_':
app.run(debug=True)
```

Question 40: Handling Form Submissions in Flask

Answer:from flask import Flask, request, render_template app = Flask(_name_)

```
@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST': data =
        request.form['data'] return f'Data
        received: {data}'
    return render_template('form.html')

if _name
== '_main_':
app.run(debug=True)
```

Question 41: Flask Route with URL Parameter

Answer:

```
from flask import Flask app
= Flask(_name_)
@app.route('/<name>')
def hello_name(name):



    return f'Hello, {name}!'
if _name_== '_main_':
app.run(debug=True)
```

Question 42: User Authentication in Flask

Answer:

```
from flask import Flask, request, redirect, url_for, session from
flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(_name_) app.config['SECRET_KEY'] = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db' db =
SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), unique=True, nullable=False)
    password = db.Column(db.String(150), nullable=False)

@app.route('/login', methods=['GET', 'POST'])
def login():
        if request.method == 'POST':
                username = request.form['username']
                password = request.form['password']
                user = User.query.filter_by(username=username).first()
                if user and check_password_hash(user.password,  password):
                        session['user_id']  = user.id
                        return redirect(url_for('dashboard'))
        return render_template('login.html')

@app.route('/dashboard')
def dashboard():
        if 'user_id' in session:
                return 'Welcome to your dashboard!'
        return redirect(url_for('login'))
if _name
== '_main_':
app.run(debug=True)
```

Question 43: Connecting Flask to SQLite with SQLAlchemy

Answer:

```
from flask import Flask
from flask_sqlalchemy  import SQLAlchemy

app = Flask(_name_)
app.config['SQLALCHEMY_DATABASE_URI']  = 'sqlite:///database.db' db =
SQLAlchemy(app)

class User(db.Model):
        id = db.Column(db.Integer,  primary_key=True)
        name = db.Column(db.String(80),  nullable=False)

@app.route('/')
def index():
        return 'Database connected!'
if _name_== '_main_':
app.run(debug=True)
```

Question 44: RESTful API Endpoint in Flask

Answer:

```
from flask import Flask, jsonify app =
Flask(_name_)
@app.route('/api/data',  methods=['GET'])
def get_data():
        data = {'key': 'value'}
        return jsonify(data)
if _name_== '_main_':
app.run(debug=True)
```

Question 45: Using Flask-WTF for Form Validation

Answer:

```
from flask import Flask, render_template,  request from
flask_wtf import FlaskForm
from wtforms import StringField, SubmitField from
wtforms.validators  import DataRequired

app = Flask(_name_)
```

```
app.config['SECRET_KEY'] = 'your_secret_key'

class MyForm(FlaskForm):
        name = StringField('Name', validators=[DataRequired()])
        submit = SubmitField('Submit')

@app.route('/form', methods=['GET', 'POST'])
def form():
        form = MyForm()
        if form.validate_on_submit():
                return f'Hello, {form.name.data}!'
        return render_template('form.html', form=form)


if _name
== '_main_':
app.run(debug=True)
```

Question 46: Implementing File Uploads in Flask

Answer:

```
from flask import Flask, request, redirect, url_for from
werkzeug.utils import secure_filename
import os

app = Flask(_name_)
app.config['UPLOAD_FOLDER'] = 'uploads/'
@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
        if request.method == 'POST':
                file = request.files['file']
                if file:
                        filename = secure_filename(file.filename)
                        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename)) return
                        'File uploaded successfully!'
        return '''
        <!doctype html>
        <title>Upload a File</title>
        <h1>Upload a File</h1>
        <form method=post enctype=multipart/form-data>
            <input type=file name=file>
            <input type=submit value=Upload>
        </form>
        '''


if _name
== '_main_':
app.run(debug=True)
```

Question 47: Creating a Flask Blueprint[5]

Answer:

```
from flask import Flask, Blueprint app =
Flask(_name_)
bp = Blueprint('bp', _name_)

@bp.route('/hello')
def hello():
        return 'Hello from the blueprint!'
```

```
  app.register_blueprint(bp,  url_prefix='/bp')
  if _name
== '_main_':
app.run(debug=True)
```

Question 48: Deploying Flask with Gunicorn and Nginx[5]

Answer:

1. **Install Gunicorn:**

```
pip install gunicorn
```

2. **Run Flask App with Gunicorn:**

```
gunicorn -w 4 -b 0.0.0.0:8000 app:app
```

3. **Nginx Configuration:**

```
server {
    listen 80;
    server_name your_domain.com;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

50. Question: What is the difference between Series & DataFrames?

Series:
- A one-dimensional labeled array capable of holding any data type.
- Similar to a column in a table.

DataFrame:
- A two-dimensional labeled data structure with columns of potentially different types.
- Essentially a table with rows and columns.
- Question: Create a database named `Travel_Planner` in MySQL, and create a table named `bookings` with attributes (user_id INT, flight_id INT, hotel_id INT, activity_id INT, booking_date DATE). Fill with some dummy values. Read the content of this table using pandas as a DataFrame. Show the output.

Here's the SQL code to create the database and table, and insert dummy values:

```
CREATE DATABASE IF NOT EXISTS Travel_Planner;
USE Travel_Planner;

CREATE TABLE IF NOT EXISTS bookings (
  user_id INT,
  flight_id INT,
  hotel_id INT,
  activity_id INT,
  booking_date DATE
);

INSERT INTO bookings (user_id, flight_id, hotel_id, activity_id, booking_date)
VALUES
(1, 12345, 54321, 67890, '2021-01-01'),
(2, 12346, 54322, 67891, '2021-01-02');
```

To read this table using pandas as a DataFrame:

```
import pandas as pd
import sqlalchemy

# Replace 'user' and 'password' with your actual MySQL username and password
engine = sqlalchemy.create_engine('mysql+pymysql://user:password@localhost/Travel_Planner')
df = pd.read_sql_table('bookings', engine)
print(df)
```

- Explain the concept of cross-validation

Cross-validation is a technique used to evaluate the performance of a machine learning model. It involves splitting the dataset into multiple subsets, training the model on some subsets, and validating it on the remaining subsets. This process is repeated several times to ensure the model's performance is consistent and not dependent on a particular train-test split.

- What is the difference between a classification and a regression problem?
- **Classification**: Predicts categorical outcomes (e.g., spam or not spam).
- **Regression**: Predicts continuous outcomes (e.g., house prices).
- Explain the concept of ensemble learning

Ensemble learning combines multiple models to improve overall performance. Techniques include:

- **Bagging**: Reduces variance by training multiple models on different subsets of data.
- **Boosting**: Reduces bias by sequentially training models, each correcting errors of the previous one.
- What is gradient descent and how does it work?

Gradient descent is an optimization algorithm used to minimize the cost function in machine learning models. It iteratively adjusts model parameters in the direction of the steepest descent of the cost function.

- Describe the difference between batch gradient descent and stochastic gradient descent
- **Batch Gradient Descent**: Uses the entire dataset to compute gradients.
- **Stochastic Gradient Descent (SGD)**: Uses one data point at a time to compute gradients, making it faster but noisier.
- What is the curse of dimensionality in machine learning?

The **curse of dimensionality** refers to the challenges that arise when working with high-dimensional data. As the number of features increases, the volume of the feature space grows exponentially, making it difficult for models to generalize well.

- Explain the difference between L1 and L2 regularization
- **L1 Regularization (Lasso)**: Adds the absolute value of coefficients as a penalty term to the loss function, promoting sparsity.
- **L2 Regularization (Ridge)**: Adds the squared value of coefficients as a penalty term, promoting small but non-zero coefficients.
- What is a confusion matrix and how is it used?

A **confusion matrix** is a table used to evaluate the performance of a classification model. It shows the counts of true positive, true negative, false positive, and false negative predictions, helping to calculate metrics like precision, recall, and accuracy.

- Define AUC-ROC curve

The **AUC-ROC curve** is a graphical representation of a classifier's performance. The **ROC curve** plots the true positive rate against the false positive rate at various threshold settings. The **AUC** (Area Under the Curve) measures the overall performance, with a value closer to 1 indicating a better model.

- Explain the k-nearest neighbors algorithm

The **k-nearest neighbors (KNN)** algorithm classifies a data point based on the majority class of its k-nearest neighbors in the feature space. It is a simple, instance-based learning method.

- Explain the basic concept of a Support Vector Machine (SVM)

A **Support Vector Machine (SVM)** is a supervised learning algorithm that finds the optimal hyperplane to separate different classes in the feature space. It maximizes the margin between the closest points of different classes.

- How does the kernel trick work in SVM?

The **kernel trick** allows SVMs to perform non-linear classification by transforming the input features into a higher-dimensional space using a kernel function, making it easier to find a linear separation.

- What are the different types of kernels used in SVM and when would you use each?
- **Linear Kernel**: Used when data is linearly separable.
- **Polynomial Kernel**: Used for polynomial relationships between features.
- **Radial Basis Function (RBF) Kernel**: Used for non-linear relationships, most common.
- **Sigmoid Kernel**: Used for neural network-like behavior.
- What is the hyperplane in SVM and how is it determined?

The **hyperplane** is the decision boundary that separates different classes in the feature space. It is determined by maximizing the margin between the closest points (support vectors) of different classes.

- What are the pros and cons of using a Support Vector Machine (SVM)?

Pros:

- Effective in high-dimensional spaces.
- Works well with clear margin of separation.

Cons:

- Not suitable for large datasets.
- Sensitive to the choice of kernel and hyperparameters.
- Explain the difference between a hard margin and a soft margin SVM
- **Hard Margin SVM**: Assumes data is linearly separable with no misclassification.
- **Soft Margin SVM**: Allows some misclassification to handle noisy data and achieve better generalization.
- Describe the process of constructing a decision tree

1. **Select the best feature** to split the data based on a criterion (e.g., information gain).
2. **Split the data** into subsets based on the selected feature.
3. **Repeat the process** for each subset until a stopping condition is met (e.g., maximum depth).

- Describe the working principle of a decision tree

A **decision tree** splits the data into subsets based on the value of input features. Each internal node represents a feature, each branch represents a decision rule, and each leaf node represents an outcome.

- What is information gain and how is it used in decision trees?

Information gain measures the reduction in entropy or impurity after a dataset is split on a feature. It is used to select the best feature for splitting the data at each node.

- Explain Gini impurity and its role in decision trees

Gini impurity measures the likelihood of incorrect classification of a randomly chosen element. It is used as a criterion to select the best feature for splitting the data in decision trees.

- What are the advantages and disadvantages of decision trees?

Advantages:

- Easy to interpret and visualize.
- Handles both numerical and categorical data.

Disadvantages:

- Prone to overfitting.
- Sensitive to noisy data.
- How do random forests improve upon decision trees?

Random forests improve upon decision trees by combining multiple trees to reduce overfitting and improve generalization. Each tree is trained on a random subset of the data and features.

- How does a random forest algorithm work?

4. **Bootstrap sampling**: Create multiple subsets of the data by sampling with replacement.
5. **Train decision trees**: Train a decision tree on each subset.
6. **Aggregate predictions**: Combine the predictions of all trees (e.g., majority voting for classification).

- What is bootstrapping in the context of random forests?

Bootstrapping is a sampling technique where multiple subsets of the data are created by sampling with replacement. Each subset is used to train a different decision tree in the random forest.

- Explain the concept of feature importance in random forests

Feature importance measures the contribution of each feature to the model's predictions. In random forests, it is calculated based on the average decrease in impurity or accuracy when the feature is used for splitting.

- What are the key hyperparameters of a random forest and how do they affect the model?
- **Number of trees**: More trees generally improve performance but increase computational cost.
- **Maximum depth**: Controls the depth of each tree, affecting model complexity and overfitting.
- **Minimum samples split**: Minimum number of samples required to split a node, affecting tree growth.
- Describe the logistic regression model and its assumptions

Logistic regression is a linear model used for binary classification. It assumes a linear relationship between the input features and the log-odds of the outcome.

- How does logistic regression handle binary classification problems?

Logistic regression uses the sigmoid function to map the linear combination of input features to a probability between 0 and 1, which is then used for binary classification.

- What is the sigmoid function and how is it used in logistic regression?

The **sigmoid function** maps any real-valued number to a value between 0 and 1. In logistic regression, it is used to convert the linear combination of input features into a probability.

- Explain the concept of the cost function in logistic regression

The **cost function** in logistic regression measures the difference between the predicted probabilities and the actual outcomes. It is minimized during training to improve model performance.

- How can logistic regression be extended to handle multiclass classification?

Logistic regression can be extended to multiclass classification using techniques like **one-vs-rest (OvR)** or **softmax regression**.

- What is the difference between L1 and L2 regularization in logistic regression?
- **L1 Regularization (Lasso)**: Adds the absolute value of coefficients as a penalty term, promoting sparsity.
- **L2 Regularization (Ridge)**: Adds the squared value of coefficients as a penalty term, promoting small but non-zero coefficients.
- What is XGBoost and how does it differ from other boosting algorithms?

XGBoost is an optimized implementation of gradient boosting. It differs from other boosting algorithms by providing regularization, handling missing values, and offering parallel processing.

- Explain the concept of boosting in the context of ensemble learning

Boosting is an ensemble technique that combines weak learners sequentially, with each learner correcting the errors of the previous ones. It aims to create a strong learner from weak learners.

- How does XGBoost handle missing values?

XGBoost handles missing values by learning the best direction to take when encountering a missing value during training, allowing it to make informed decisions during prediction.

1. Key Hyperparameters in XGBoost and Their Impact on Model Performance
2. **Learning Rate (eta)**: Controls the step size during the update of weights. Lower values make the model more robust but require more boosting rounds.
    - **Typical values**: 0.01 to 0.3
    - **Impact**: Smaller values prevent overfitting but increase training time[11].
3. **Max Depth**: The maximum depth of a tree. Increasing this value makes the model more complex.
    - **Typical values**: 3 to 10
    - **Impact**: Higher values can lead to overfitting[11].
4. **Min Child Weight**: Minimum sum of instance weight (hessian) needed in a child.
    - **Typical values**: 1 to 10
    - **Impact**: Higher values prevent overfitting[11].
5. **Subsample**: The fraction of samples used for fitting the individual base learners.
    - **Typical values**: 0.5 to 1.0
    - **Impact**: Lower values prevent overfitting[11].
6. **Colsample_bytree**: The fraction of features used for fitting the individual base learners.
    - **Typical values**: 0.5 to 1.0
    - **Impact**: Lower values prevent overfitting[11].
7. **Gamma**: Minimum loss reduction required to make a further partition on a leaf node.
    - **Typical values**: 0 to 10
    - **Impact**: Higher values make the algorithm more conservative[11].
8. Process of Gradient Boosting in XGBoost
9. **Initialization**: Start with an initial model, usually a simple one like the mean of the target values.
10. **Iterative Improvement**: Add new trees sequentially. Each new tree is trained to correct the errors of the combined ensemble of all previous trees.
11. **Gradient Calculation**: Compute the gradient of the loss function with respect to the predictions of the current ensemble.
12. **Tree Construction**: Fit a new tree to the negative gradient (residuals).
13. **Update Model**: Add the new tree to the ensemble with a weight that minimizes the overall loss.
14. **Repeat**: Continue adding trees until a stopping criterion is met (e.g., a fixed number of trees or no further improvement)[13].
15. Advantages and Disadvantages of Using XGBoost

Advantages:

- **High Performance**: Achieves state-of-the-art results on many machine learning tasks.
- **Efficient Handling of Missing Values**: Automatically learns the best way to handle missing data.
- **Built-in Regularization**: Helps prevent overfitting.
- **Scalability**: Can handle large datasets efficiently.
- **Feature Importance**: Provides insights into the importance of different features[67].

Disadvantages:

- **Complexity**: Requires careful tuning of hyperparameters.
- **Computationally Intensive**: Training can be slow, especially with large datasets.
- **Prone to Overfitting**: If not properly regularized, it can overfit the training data[67].

16. Python Code for Reading a MySQL Table into a DataFrame

Here's an example of how to create a database and table in MySQL, insert dummy values, and read the table into a pandas DataFrame:

```
CREATE DATABASE IF NOT EXISTS Travel_Planner;
USE Travel_Planner;

CREATE TABLE IF NOT EXISTS bookings (
   user_id INT,
   flight_id INT,
   hotel_id INT,
   activity_id INT,
   booking_date DATE
);

INSERT INTO bookings (user_id, flight_id, hotel_id, activity_id, booking_date)
VALUES
(1, 12345, 54321, 67890, '2021-01-01'),
(2, 12346, 54322, 67891, '2021-01-02');
import pandas as pd
import sqlalchemy

# Replace 'user' and 'password' with your actual MySQL username and password
engine = sqlalchemy.create_engine('mysql+pymysql://user:password@localhost/Travel_Planner')
df = pd.read_sql_table('bookings', engine)
print(df)
```