1a. Write a Python function that checks if a given string is a palindrome.

```
python
Copy code
def is_palindrome(s):
    return s == s[::-1]

# Test the function
print(is_palindrome("racecar"))  # True
print(is_palindrome("hello"))    # False
```

1b. Write a Python function to find all odd numbers up to a given number `n` using a loop.

```
python
Copy code
def find_odd_numbers(n):
    odd_numbers = []
    for i in range(1, n + 1):
        if i % 2 != 0:
            odd_numbers.append(i)
    return odd_numbers

# Test the function
print(find_odd_numbers(10))  # [1, 3, 5, 7, 9]
```

2. Create a number guessing game.

```
python
Copy code
import random

def guessing_game():
    number_to_guess = random.randint(1, 100)
    guess = None
    while guess != number_to_guess:
        guess = int(input("Guess the number between 1 and 100: "))
        if guess < number_to_guess:
            print("Too low!")
        elif guess > number_to_guess:
            print("Too high!")
        else:
            print("Correct! You guessed the number.")

guessing_game()  # Uncomment to play the game
```

3. Create a login system with three attempts.

```
python
Copy code
def login_system():
    correct_username = "user"
    correct_password = "pass"
```

```
    attempts = 0

    while attempts < 3:
        username = input("Enter username: ")
        password = input("Enter password: ")

        if username == correct_username and password == correct_password:
            print("Login successful!")
            return
        else:
            print("Incorrect credentials, try again.")
            attempts += 1

    print("Account locked due to too many failed attempts.")

# login_system()  # Uncomment to test the system
```

4. Create a library management system with a `Book` class and a subclass `Ebook`.

```
python
Copy code
class Book:
    def __init__(self, title, author, ISBN):
        self.title = title
        self.author = author
        self.ISBN = ISBN

    def display(self):
        return f"Title: {self.title}, Author: {self.author}, ISBN:
{self.ISBN}"

class Ebook(Book):
    def __init__(self, title, author, ISBN, file_format):
        super().__init__(title, author, ISBN)
        self.file_format = file_format

    def display(self):
        return super().display() + f", File Format: {self.file_format}"

# Test the classes
ebook = Ebook("Python Programming", "John Doe", "1234567890", "PDF")
print(ebook.display())  # "Title: Python Programming, Author: John Doe, ISBN:
1234567890, File Format: PDF"
```

5. Create a `Tournament` class.

```
python
Copy code
class Tournament:
    def __init__(self):
        self.matches = []

    def add_match(self, team1, team2):
        self.matches.append((team1, team2))
```

```
    def record_result(self, team1, team2, winner):
        for match in self.matches:
            if match == (team1, team2):
                self.matches.remove(match)
                self.matches.append((team1, team2, winner))
                break

    def display_bracket(self):
        for match in self.matches:
            print(match)

# Test the class
tournament = Tournament()
tournament.add_match("Team A", "Team B")
tournament.add_match("Team C", "Team D")
tournament.record_result("Team A", "Team B", "Team A")
tournament.display_bracket()  # Should display matches including the result
of "Team A" vs "Team B"
```

6. Implement a `Gradebook` class.

```
python
Copy code
class Gradebook:
    def __init__(self):
        self.students = {}

    def add_grade(self, student, subject, grade):
        if student not in self.students:
            self.students[student] = {}
        self.students[student][subject] = grade

    def calculate_average(self, student):
        grades = self.students[student].values()
        return sum(grades) / len(grades) if grades else 0

    def highest_scoring_student(self):
        highest_average = 0
        top_student = None
        for student in self.students:
            average = self.calculate_average(student)
            if average > highest_average:
                highest_average = average
                top_student = student
        return top_student

# Test the class
gradebook = Gradebook()
gradebook.add_grade("Alice", "Math", 90)
gradebook.add_grade("Alice", "Science", 95)
gradebook.add_grade("Bob", "Math", 85)
gradebook.add_grade("Bob", "Science", 80)

print(gradebook.calculate_average("Alice"))  # 92.5
print(gradebook.calculate_average("Bob"))    # 82.5
print(gradebook.highest_scoring_student())   # "Alice"
```

1. Write a Python function that calculates the factorial of a given non-negative integer using recursion.

```
python
Copy code
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Test the function
print(factorial(5))  # 120
print(factorial(0))  # 1
```

2. Develop a Python program that reads a text file and counts the frequency of each word. Display the top N most frequent words along with their counts, where N is provided by the user.

```
python
Copy code
from collections import Counter

def top_n_frequent_words(file_path, N):
    with open(file_path, 'r') as file:
        text = file.read()

    words = text.split()
    word_counts = Counter(words)
    most_common_words = word_counts.most_common(N)

    return most_common_words

# Example usage
# top_words = top_n_frequent_words('example.txt', 5)
# for word, count in top_words:
#     print(f"{word}: {count}")
```

3. Develop a Python program that simulates a simple ATM machine.

```
python
Copy code
class ATM:
    def __init__(self, balance=0):
        self.balance = balance

    def check_balance(self):
        print(f"Current balance: ${self.balance}")

    def deposit(self, amount):
        self.balance += amount
        print(f"${amount} deposited. New balance: ${self.balance}")
```

```
    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance!")
        else:
            self.balance -= amount
            print(f"${amount} withdrawn. New balance: ${self.balance}")

    def run(self):
        while True:
            print("\n1. Check Balance")
            print("2. Deposit Money")
            print("3. Withdraw Money")
            print("4. Exit")
            choice = input("Choose an option: ")

            if choice == "1":
                self.check_balance()
            elif choice == "2":
                amount = float(input("Enter amount to deposit: "))
                self.deposit(amount)
            elif choice == "3":
                amount = float(input("Enter amount to withdraw: "))
                self.withdraw(amount)
            elif choice == "4":
                print("Exiting...")
                break
            else:
                print("Invalid choice. Please try again.")

# Example usage
# atm = ATM(1000)
# atm.run()
```

4. Create a Python class called `Rectangle` and a subclass `Square`.

```
python
Copy code
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)

class Square(Rectangle):
    def __init__(self, side_length):
        super().__init__(side_length, side_length)

# Test the classes
rectangle = Rectangle(4, 6)
print(f"Rectangle area: {rectangle.area()}")          # 24
print(f"Rectangle perimeter: {rectangle.perimeter()}") # 20
```

```
square = Square(4)
print(f"Square area: {square.area()}")                # 16
print(f"Square perimeter: {square.perimeter()}")       # 16
```

5. Design a Python class for a basic shopping cart.

```
python
Copy code
class ShoppingCart:
    def __init__(self):
        self.cart = {}

    def add_item(self, item, price):
        if item in self.cart:
            self.cart[item] += price
        else:
            self.cart[item] = price
        print(f"Added {item} for ${price}")

    def remove_item(self, item):
        if item in self.cart:
            del self.cart[item]
            print(f"Removed {item}")
        else:
            print(f"{item} not in cart")

    def view_cart(self):
        for item, price in self.cart.items():
            print(f"{item}: ${price}")

    def total_cost(self):
        return sum(self.cart.values())

    def clear_cart(self):
        self.cart.clear()
        print("Cart cleared")

# Test the class
cart = ShoppingCart()
cart.add_item("Apple", 1.5)
cart.add_item("Banana", 2)
cart.view_cart()
print(f"Total cost: ${cart.total_cost()}")  # 3.5
cart.remove_item("Apple")
cart.view_cart()
cart.clear_cart()
```

6. Write a Python class to represent a bank account.

```
python
Copy code
class BankAccount:
    def __init__(self, account_number, account_holder, balance=0):
        self.account_number = account_number
```

```
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"${amount} deposited. New balance: ${self.balance}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance!")
        else:
            self.balance -= amount
            print(f"${amount} withdrawn. New balance: ${self.balance}")

    def display_account_details(self):
        print(f"Account Number: {self.account_number}")
        print(f"Account Holder: {self.account_holder}")
        print(f"Balance: ${self.balance}")

# Test the class
account = BankAccount("12345678", "John Doe", 500)
account.display_account_details()
account.deposit(200)
account.withdraw(100)
account.display_account_details()
```