#1

```
class BankAccount:
    def __init__(self, account_number, account_holder_name, balance=0.0):
        self.account_number = account_number
        self.account_holder_name = account_holder_name
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        return f"Deposited ${amount}. Current balance: ${self.balance}"

    def withdraw(self, amount):
        if amount > self.balance:
            return "Insufficient funds"
        else:
            self.balance -= amount
            return f"Withdrew ${amount}. Current balance: ${self.balance}"

    def display_balance(self):
        return f"Account Number: {self.account_number}\nAccount Holder:
{self.account_holder_name}\nCurrent Balance: ${self.balance}"

# Demonstrate the functionality
account1 = BankAccount("123456789", "John Doe", 1000.0)
print(account1.display_balance())

print(account1.deposit(500.0))
print(account1.withdraw(200.0))
print(account1.display_balance())
```

2

```
class Student:
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

    def update_grade(self, new_grade):
        self.grade = new_grade
        return f"{self.name}'s grade updated to {new_grade}"

    def display_student_info(self):
        return f"Name: {self.name}\nAge: {self.age}\nGrade: {self.grade}"

# Create instances and showcase functionality
student1 = Student("Alice", 18, "A")
print(student1.display_student_info())

print(student1.update_grade("B"))
print(student1.display_student_info())

# Create another instance
student2 = Student("Bob", 17, "C")
print(student2.display_student_info())

print(student2.update_grade("A"))
print(student2.display_student_info())
```

3

```
class Book:
    def __init__(self, title, author, isbn):
        self.title = title
        self.author = author
        self.isbn = isbn

    def update_author(self, new_author):
        self.author = new_author
        return f"Author updated to {new_author}"

    def display_book_info(self):
        return f"Title: {self.title}\nAuthor: {self.author}\nISBN: {self.isbn}"

# Create instances and showcase functionality
book1 = Book("The Great Gatsby", "F. Scott Fitzgerald", "9780141182636")
print(book1.display_book_info())

print(book1.update_author("Francis Scott Fitzgerald"))
print(book1.display_book_info())

# Create another instance
book2 = Book("To Kill a Mockingbird", "Harper Lee", "9780061120084")
print(book2.display_book_info())

print(book2.update_author("Nelle Harper Lee"))
print(book2.display_book_info())
```

4

```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def update_year(self, new_year):
        self.year = new_year
        return f"Year updated to {new_year}"

    def display_car_details(self):
        return f"Make: {self.make}\nModel: {self.model}\nYear: {self.year}"

# Create instances and showcase functionality
car1 = Car("Toyota", "Camry", 2020)
print(car1.display_car_details())

print(car1.update_year(2022))
print(car1.display_car_details())

# Create another instance
car2 = Car("Honda", "Civic", 2018)
print(car2.display_car_details())
```

```
print(car2.update_year(2021))
print(car2.display_car_details())
```

5

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

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

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

# Instantiate objects and compute area/perimeter
rectangle1 = Rectangle(5, 10)
area1 = rectangle1.calculate_area()
perimeter1 = rectangle1.calculate_perimeter()

rectangle2 = Rectangle(8, 6)
area2 = rectangle2.calculate_area()
perimeter2 = rectangle2.calculate_perimeter()

# Print results
print(f"Rectangle 1 - Area: {area1}, Perimeter: {perimeter1}")
print(f"Rectangle 2 - Area: {area2}, Perimeter: {perimeter2}")
```

6

```
class Employee:
    def __init__(self, name, employee_id, basic_pay, hra_percentage, da_percentage):
        self.name = name
        self.employee_id = employee_id
        self.basic_pay = basic_pay
        self.hra_percentage = hra_percentage
        self.da_percentage = da_percentage

    def calculate_salary(self):
        hra = (self.hra_percentage / 100) * self.basic_pay
        da = (self.da_percentage / 100) * self.basic_pay
        salary = self.basic_pay + hra + da
        return salary

    def display_employee_info(self):
        return f"Name: {self.name}\nEmployee ID: {self.employee_id}\nBasic Pay: ${self.basic_pay}
\nSalary: ${self.calculate_salary()}"

# Instantiate objects and display employee information
employee1 = Employee("John Doe", "E123", 50000, 20, 15)
print(employee1.display_employee_info())

employee2 = Employee("Jane Smith", "E456", 60000, 18, 12)
print(employee2.display_employee_info())
```

7

```
import math

class Shape:
    def calculate_area(self):
        pass

    def calculate_perimeter(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        return math.pi * self.radius**2

    def calculate_perimeter(self):
        return 2 * math.pi * self.radius

class Square(Shape):
    def __init__(self, side_length):
        self.side_length = side_length

    def calculate_area(self):
        return self.side_length**2

    def calculate_perimeter(self):
        return 4 * self.side_length

class Triangle(Shape):
    def __init__(self, side1, side2, side3):
        self.side1 = side1
        self.side2 = side2
        self.side3 = side3

    def calculate_area(self):
        s = (self.side1 + self.side2 + self.side3) / 2
        return math.sqrt(s * (s - self.side1) * (s - self.side2) * (s - self.side3))

    def calculate_perimeter(self):
        return self.side1 + self.side2 + self.side3

# Instantiate objects and compute area/perimeter
circle = Circle(5)
square = Square(4)
triangle = Triangle(3, 4, 5)

# Print results
print(f"Circle - Area: {circle.calculate_area()}, Perimeter: {circle.calculate_perimeter()}")
print(f"Square - Area: {square.calculate_area()}, Perimeter: {square.calculate_perimeter()}")
print(f"Triangle - Area: {triangle.calculate_area()}, Perimeter: {triangle.calculate_perimeter()}")
```