VaultofCodes

Python Programming Assignment: "Secret Code Generator"

Objective:

Get creative with Python by building a fun **secret code generator** that can convert regular text into a coded message and decode it back. This will help you practice using **functions**, **loops**, **string manipulation**, and **dictionaries**.

Assignment: Create a Secret Code Generator

Your task is to write a Python program that can:

- **Encode** a message by shifting the letters in the alphabet (similar to a Caesar cipher).
- **Decode** a message back to its original form.
- Allow the user to choose between **encoding** or **decoding**.

Requirements:

1. **Basic Functionality**:
    - **Encode a message**: Ask the user for a message and a shift number. Each letter in the message should be shifted by that number (e.g., A -> C with a shift of 2).
    - **Decode a message**: Take a coded message and a shift number, and shift the letters back to their original form.
    - The user should be able to input both **uppercase** and **lowercase** letters, and they should be encoded correctly.
2. **User Menu**:
    - Create a simple menu that asks the user if they want to:
        - Encode a message.
        - Decode a message.
        - Exit the program.
3. **Data Handling**:
    - Use string manipulation to **shift** each letter.
    - Ignore or skip characters that aren't letters (e.g., spaces, punctuation marks).

4. **Functions**:
    - Write functions to:

        - **Encode** a message.
        - **Decode** a message.
        - **Handle user input** and menu choices.
5. **Edge Cases**:
    - Handle scenarios like wrapping around the alphabet (e.g., shifting Z forward by 2 should result in B).
    - Handle invalid inputs, like trying to decode without entering a valid shift.

Instructions:

- Focus on **clean code** and **modular functions**.
- Submit your Python code in pdf file with comments & output screenshots explaining your logic.(Copy your code, paste in ms word, save as pdf, upload on app)

OR

Programming Assignment: "Virtual Pet Simulator"

Objective:

Create a **Virtual Pet Simulator** where users can interact with a virtual pet by feeding it, playing with it, and monitoring its happiness and hunger levels. This assignment will help you practice using **functions**, **loops**, **conditionals**, and **data structures** like dictionaries.

Assignment: Build a Virtual Pet Simulator

Your task is to build a simple command-line pet simulator that allows users to:

- **Feed** their pet.
- **Play** with their pet.
- Check the pet's **happiness** and **hunger** levels.
- **Quit** the game when they're done.

Requirements:

1. **Pet Status**:
    - The pet should have attributes like:
        - **Happiness level** (starts at 50).
        - **Hunger level** (starts at 50).
    - Both levels should range from **0 to 100**.
    - When the user feeds the pet, its hunger decreases, and when they play with it, its happiness increases.
    - If hunger gets too high (e.g., >80), the pet becomes sad, and happiness decreases.

2. **Actions**:
    - The user should be able to:
        - **Feed** the pet, which decreases hunger but slightly decreases happiness.
        - **Play** with the pet, which increases happiness but slightly increases hunger.
        - **Check Status**: Display current hunger and happiness levels.
3. **User Menu**:
    - Create a menu where the user can:
        - Feed the pet.
        - Play with the pet.
        - Check the pet's status.
        - Quit the game.
4. **Automatic Changes**:
    - As time passes (for example, every few user actions), the pet's hunger should gradually increase, and happiness should decrease, simulating the need for constant care.
5. **Game Over Conditions**:
    - If hunger reaches **100** or happiness reaches **0**, the game should display a message indicating that the pet has become too hungry or sad and end the simulation.

Bonus (Optional):

- Allow the user to **name** the pet at the start.
- Add more actions like giving the pet a **toy** or **medicine**.
- Add **multiple pets** that the user can manage.
- Implement a **random event system** where occasionally unexpected things happen (e.g., the pet finds a snack, or it gets sick).

Instructions:

- Write **clean code** and use **functions** for each major action (feed, play, check status).
- Ensure your program handles **user input** and provides clear feedback.
- Submit your Python code in pdf file with comments & output screenshots explaining your logic.(Copy your code, paste in ms word, save as pdf, upload on app)