ASSIGNMENT – 2

BY: D AARTI

Programming Assignment: "Virtual Pet Simulator"

Objective:

Develop a Virtual Pet Simulator using fundamental programming concepts such as functions, loops, conditionals, and dictionaries. The simulator will allow users to interact with a virtual pet by feeding it, playing with it, and monitoring its key attributes such as happiness and hunger levels. The solution aims to provide an engaging and interactive experience while reinforcing core programming skills through the implementation of logical structures and data manipulation.

Requirements:

This section lists the tools, technologies, and any prerequisites needed to run the project.

Example:

- **Python 3.x installed**
- **Basic knowledge of Python programming**
- **Code editor Python IDLE**
- **Terminal/Command prompt to run the script**

Functionality / Features

Briefly describe what the program does and the features it includes.

Example:

- **Feed the pet to reduce hunger level**
- **Play with the pet to increase happiness level**
- **Display the current status of the pet**
- **Continuous loop to interact until the user exits**

- **Alerts when hunger or happiness levels become critical**

Input:

```
import time
import random
# Define a class for the virtual pet
class VirtualPet:
    def __init__(self, name):
    # Set the pet's name and initial status

        self.name = name
        self.happiness = 50 # Starts at 50
        self.hunger = 50  # Starts at 50
        self.actions_count = 0 # Count how many actions the user has taken

    def feed(self):
    # Feeding decreases hunger but slightly decreases happiness

        self.hunger = max(0, self.hunger - 15)# Avoid going below 0
        self.happiness = max(0, self.happiness - 5)
        print(f"You fed {self.name}. Hunger decreased, but happiness slightly decreased.")

    def play(self):
    # Playing increases happiness but slightly increases hunger

        self.happiness = min(100, self.happiness + 15)  # Avoid going above 100
        self.hunger = min(100, self.hunger + 5)
        print(f"You played with {self.name}. Happiness increased, but hunger slightly
increased.")
```

```
def check_status(self):
# Display current happiness and hunger

    print(f"\n{self.name}'s Status:")
    print(f"Happiness: {self.happiness}")
    print(f"Hunger: {self.hunger}")
        # If hunger is too high, happiness drops more

    if self.hunger > 80:
        print(f"{self.name} is very hungry and is becoming sad!")
        self.happiness = max(0, self.happiness - 10)

def time_passes(self):
# Increase hunger and decrease happiness every few actions

    self.actions_count += 1
    if self.actions_count % 3 == 0:
        self.hunger = min(100, self.hunger + 5)
        self.happiness = max(0, self.happiness - 5)
        print(f"\nTime passes... {self.name} is getting hungrier and less happy.")

def is_game_over(self):
        # Check if hunger or happiness hits a critical point

    if self.hunger >= 100:
        print(f"\nOh no! {self.name} got too hungry. Game over.")
        return True
```

```
        elif self.happiness <= 0:
            print(f"\nOh no! {self.name} became too sad. Game over.")
            return True
        return False


def pet_simulator():
    # Main game function

    print("!!!!!!!!.....Welcome to the Virtual Pet Simulator.....!!!!!!!")
    pet_name = input("What would you like to name your pet? ")
    pet = VirtualPet(pet_name)
    # Main game loop

    while True:
        print("\n--- Menu ---")
        print("1. Feed Pet")
        print("2. Play with Pet")
        print("3. Check Pet's Status")
        print("4. Quit Game")

        choice = input("Choose an action (1-4): ")

        if choice == '1':
            pet.feed()
        elif choice == '2':
            pet.play()
        elif choice == '3':
```

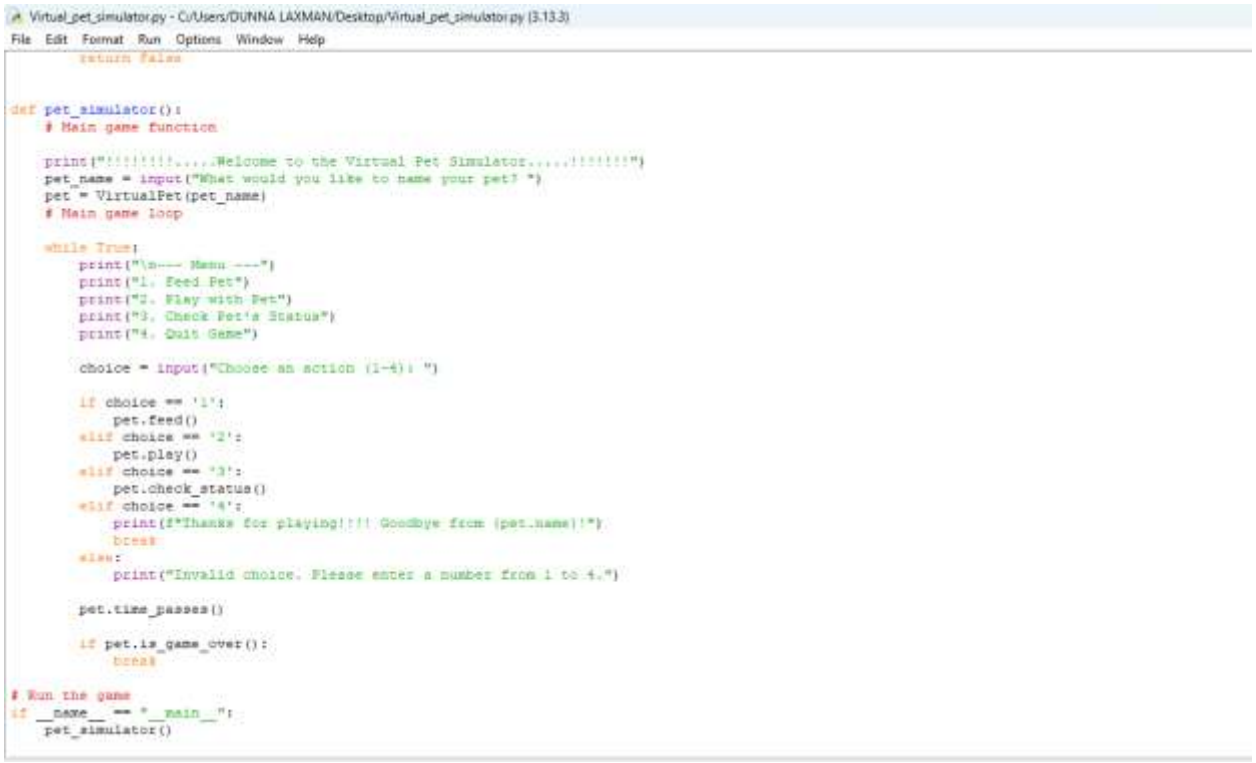
```
            pet.check_status()
        elif choice == '4':
            print(f"Thanks for playing!!!! Goodbye from {pet.name}!")
            break
        else:
            print("Invalid choice. Please enter a number from 1 to 4.")

        pet.time_passes()

        if pet.is_game_over():
            break

# Run the game
if __name__ == "__main__":
    pet_simulator()
```

```
File  Edit  Format  Run  Options  Window  Help
import time
import random
# Define a class for the virtual pet
class VirtualPet:
    def __init__(self, name):
    # Set the pet's name and initial status

        self.name = name
        self.happiness = 50 # Starts at 50
        self.hunger = 50  # Starts at 50
        self.actions_count = 0 # Count how many actions the user has taken

    def feed(self):
    # Feeding decreases hunger but slightly decreases happiness

        self.hunger = max(0, self.hunger - 15)# Avoid going below 0
        self.happiness = max(0, self.happiness - 5)
        print(f"You fed {self.name}. Hunger decreased, but happiness slightly decreased.")

    def play(self):
    # Playing increases happiness but slightly increases hunger

        self.happiness = min(100, self.happiness + 15)  # Avoid going above 100
        self.hunger = min(100, self.hunger + 5)
        print(f"You played with {self.name}. Happiness increased, but hunger slightly increased.")

    def check_status(self):
    # Display current happiness and hunger

        print(f"\n{self.name}'s Status:")
        print(f"Happiness: {self.happiness}")
        print(f"Hunger: {self.hunger}")
                # If hunger is too high, happiness drops more

        if self.hunger > 80:
            print(f"{self.name} is very hungry and is becoming sad!")
            self.happiness = max(0, self.happiness - 10)

    def time_passes(self):
    # Increase hunger and decrease happiness every few actions
```

Ln: 77 C

Virtual_pet_simulator.py - C:\Users\DUNNA LAXMAN\Desktop\Virtual_pet_simulator.py (3.13.3)

```
File  Edit  Format  Run  Options  Window  Help
            self.happiness = max(0, self.happiness - 10)

    def time_passes(self):
    # Increase hunger and decrease happiness every few actions

        self.actions_count += 1
        if self.actions_count % 3 == 0:
            self.hunger = min(100, self.hunger + 5)
            self.happiness = max(0, self.happiness - 5)
            print(f"\nTime passes... {self.name} is getting hungrier and less happy.")

    def is_game_over(self):
                # Check if hunger or happiness hits a critical point

        if self.hunger >= 100:
            print(f"\nOh no! {self.name} got too hungry. Game over.")
            return True
        elif self.happiness <= 0:
            print(f"\nOh no! {self.name} became too sad. Game over.")
            return True
        return False


def pet_simulator():
    # Main game function

    print("!!!!!!!.....Welcome to the Virtual Pet Simulator.....!!!!!!!")
    pet_name = input("What would you like to name your pet? ")
    pet = VirtualPet(pet_name)
    # Main game loop

    while True:
        print("\n--- Menu ---")
        print("1. Feed Pet")
        print("2. Play with Pet")
        print("3. Check Pet's Status")
        print("4. Quit Game")

        choice = input("Choose an action (1-4)): ")

        if choice == '1':
```

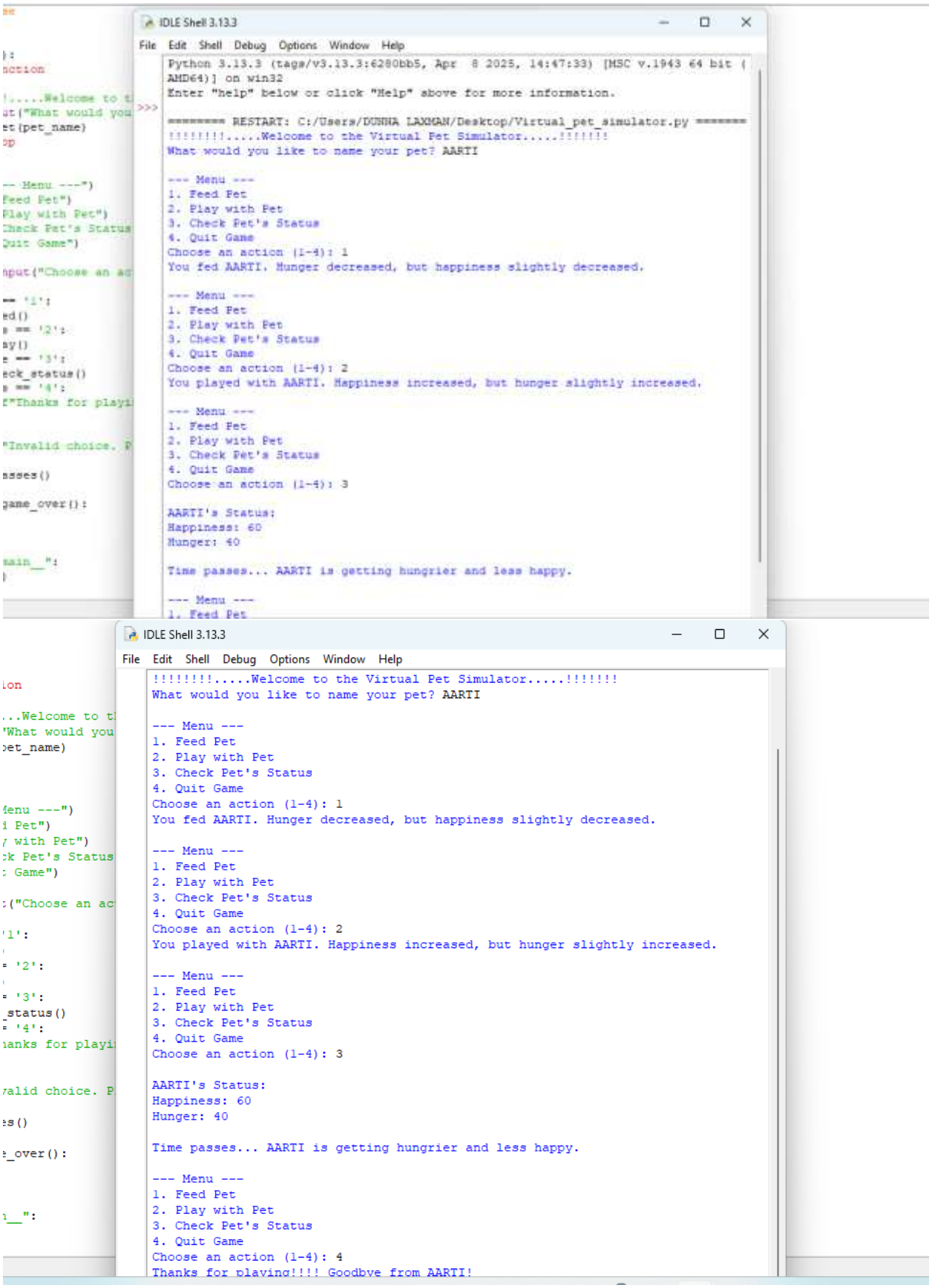


Output / Expected Output

Explain what kind of output the program produces and how the user interacts with it.

Example:

- **The program displays a menu of options: feed, play, check status, or exit.**
- **Based on user input, the pet's hunger and happiness levels are updated.**
- **Status messages and alerts are printed to inform the user of the pet's condition.**


IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
======== RESTART: C:/Users/DUNNA LAXMAN/Desktop/Virtual_pet_simulator.py ========
!!!!!!!!.....Welcome to the Virtual Pet Simulator.....!!!!!!!
What would you like to name your pet? AARTI
--- Menu ---
1. Feed Pet
2. Play with Pet
3. Check Pet's Status
4. Quit Game
Choose an action (1-4): 1
You fed AARTI. Hunger decreased, but happiness slightly decreased.
--- Menu ---
1. Feed Pet
2. Play with Pet
3. Check Pet's Status
4. Quit Game
Choose an action (1-4): 2
You played with AARTI. Happiness increased, but hunger slightly increased.
--- Menu ---
1. Feed Pet
2. Play with Pet
3. Check Pet's Status
4. Quit Game
Choose an action (1-4): 3
AARTI's Status:
Happiness: 60
Hunger: 40
Time passes... AARTI is getting hungrier and less happy.
--- Menu ---
1. Feed Pet
IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
!!!!!!!!.....Welcome to the Virtual Pet Simulator.....!!!!!!!
What would you like to name your pet? AARTI
--- Menu ---
1. Feed Pet
2. Play with Pet
3. Check Pet's Status
4. Quit Game
Choose an action (1-4): 1
You fed AARTI. Hunger decreased, but happiness slightly decreased.
--- Menu ---
1. Feed Pet
2. Play with Pet
3. Check Pet's Status
4. Quit Game
Choose an action (1-4): 2
You played with AARTI. Happiness increased, but hunger slightly increased.
--- Menu ---
1. Feed Pet
2. Play with Pet
3. Check Pet's Status
4. Quit Game
Choose an action (1-4): 3
AARTI's Status:
Happiness: 60
Hunger: 40
Time passes... AARTI is getting hungrier and less happy.
--- Menu ---
1. Feed Pet
2. Play with Pet
3. Check Pet's Status
4. Quit Game
Choose an action (1-4): 4
Thanks for playing!!!! Goodbye from AARTI!

Conclusion:

Summarize what was achieved in the project.

Example:

This Virtual Pet Simulator successfully demonstrates user interaction through input/output handling, use of functions for code organization, and conditionals to control the game flow. It serves as a good foundation for learning more complex programming concepts.