While Loops in Python

A **loop** is a programming structure that repeats a set of instructions as long as a specified condition is True. In Python, the `while` **loop** allows you to repeatedly execute a block of code as long as the condition is True.

1. The Basic Structure of a `while` Loop

The `while` loop repeatedly executes a block of code as long as the condition is `True`.

Syntax:

```
while condition:
    # Code to execute as
```

Example:

Let's print numbers from 1 to 5 using a `while` loop.

```
i = 1
while i <= 5:
    print(i)
    i += 1  # Incrementin
```

- The loop starts with `i = 1` and checks if `i <= 5`.
- As long as this condition is `True`, it prints the value of `i` and increases it by 1 (`i += 1`).
- The loop ends when `i` becomes 6, as the condition `i <= 5` becomes `False`.

Output:

```
1
2
3
4
5
```

2. Common Example: Counting Sheep

Let's relate this to a common example: Imagine you're counting sheep to fall asleep.

```
sheep_count = 1
while sheep_count <= 10:
    print(f"Sheep {sheep_
    sheep_count += 1
```

This prints `"Sheep 1"`, `"Sheep 2"`, and so on, until `"Sheep 10"`. After that, the loop stops.

3. Avoiding Infinite Loops

A `while` **loop** can run indefinitely if the condition is always `True`. To prevent this, ensure that the condition eventually becomes `False`.

Example of an Infinite Loop:

```
i = 1
while i <= 5:
    print(i)
    # Forgot to update i,
```

In this case, the loop will keep printing `1` forever because `i` is never incremented, so the condition `i <= 5` will always be `True`.

To avoid this, make sure to **update the variable** that controls the condition within the loop.

4. Using `break` to Exit a `while` Loop

You can use the `break` statement to exit a loop when a certain condition is met.

Example:

Let's stop counting sheep after 5 sheep, even though the condition allows counting up to 10:

```
sheep_count = 1
while sheep_count <= 10:
    print(f"Sheep {sheep_
    if sheep_count == 5:
        print("That's enc
        break
    sheep_count += 1
```

- The loop stops after `"Sheep 5"` because of the `break` statement, even though the condition was `sheep_count <= 10`.

5. Using `continue` to Skip an Iteration

The `continue` statement is used to skip the current iteration and move on to the next one.

Example:

Let's say you want to skip counting sheep that are number 4:

```
sheep_count = 1
while sheep_count <= 5:
    if sheep_count == 4:
        sheep_count += 1
        continue
    print(f"Sheep {sheep_
    sheep_count += 1
```

Here, when `sheep_count` is 4, the `continue` statement skips printing `"Sheep 4"`, and the loop continues with `sheep_count = 5`.

Output:

6. Using `while` Loops for User Input

You can use a `while` loop to repeatedly ask the user for input until they provide valid data.

Example:

Let's ask the user for a PIN until they enter the correct one:

```
pin = ""
correct_pin = "1234"
while pin != correct_pin:
    pin = input("Enter yc
    if pin != correct_pin
        print("Incorrect
print("PIN accepted. You
```

- The loop keeps running until the user enters the correct PIN.
- If the user enters an incorrect PIN, they are prompted to try again.

7. Real-life Example: KSRTC Bus Seats Availability

Let’s say you want to simulate a KSRTC bus seat booking system. The bus has 5 available seats. Each time a seat is booked, the available seats decrease.

```
available_seats = 5

while available_seats > 0
    print(f"{available_se
    booking = input("Do y

    if booking == "yes":
        available_seats -
        print("Seat booke
    else:
        print("No booking

print("All seats are book
```

Here, the loop keeps running until all seats are booked. It checks the available seats and asks the user if they want to book one. The loop stops when there are no more seats available.

8. Nested `while` Loops

You can also nest `while` loops inside each other. This can be useful in more complex scenarios, such as checking multiple conditions or dealing with multi-level data.

Example:

Let's simulate a snack machine that allows users to buy snacks as long as both the machine has snacks and the user has money:

```python
snacks_available = 3
money = 10

while snacks_available >
    print(f"Snacks availa
    buy = input("Do you w

    if buy == "yes" and m
        snacks_available
        money -= 5
        print("Snack purc
    else:
        print("No purchas

print("Either snacks are
```

Homework

1. **Basic Counting with `while` Loop**:
    - Write a program that counts from 1 to 10 using a `while` loop.
2. **Odd Numbers Printer**:
    - Create a program that prints all odd numbers between 1 and 20 using a `while` loop.
3. **Ticket Booking Simulation**:
    - Write a program that simulates a bus ticket booking system. The bus has 8 seats. Each time a seat is booked, the available seats decrease. When there are no seats left, the loop stops and displays a message saying "All seats are booked."