Learn Python Programming from Scratch

Topic: While Loops in Python

1. What are While Loops?

While loops continue executing as long as a condition remains True. Think of them as a way to repeat code until something changes. Unlike for loops, which iterate over a known sequence, while loops are perfect when you don't know in advance how many times you need to repeat something - they keep going until a condition becomes False.

2. Basic While Loop Syntax

The basic syntax of a while loop involves a condition that gets checked before each iteration.

In [1]:
```
# Basic while loop syntax
count = 1

while count <= 5:
    print(f"Count: {count}")
    count += 1  # Important: update the condition variable

print("Loop finished!")
```

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
Loop finished!
```

3. Input Validation with While Loops

While loops are excellent for validating user input until correct data is provided.

In [2]:
```
# Input validation example
age = -1  # Initialize with invalid value

while age < 0 or age > 150:
    age = int(input("Enter your age (0-150): "))
    if age < 0 or age > 150:
        print("Invalid age! Please try again.")

print(f"Your age is: {age}")
```

```
Your age is: 57
```

4. Menu-Driven Programs

While loops are perfect for creating menu systems that run until the user chooses to exit.

In [3]:
```python
# Simple menu system
choice = ""

while choice != "3":
    print("\n--- Main Menu ---")
    print("1. Say Hello")
    print("2. Calculate Square")
    print("3. Exit")

    choice = input("Enter your choice: ")

    if choice == "1":
        name = input("Enter your name: ")
        print(f"Hello, {name}!")
    elif choice == "2":
        num = int(input("Enter a number: "))
        print(f"Square of {num} is {num ** 2}")
    elif choice == "3":
        print("Goodbye!")
    else:
        print("Invalid choice!")
```

```
--- Main Menu ---
1. Say Hello
2. Calculate Square
3. Exit
Invalid choice!

--- Main Menu ---
1. Say Hello
2. Calculate Square
3. Exit
Square of 5 is 25

--- Main Menu ---
1. Say Hello
2. Calculate Square
3. Exit
Goodbye!
```

5. Using break and continue

Control the flow of while loops with `break` (exit immediately) and `continue` (skip to next iteration).

In [4]:
```python
# Using break to exit early
count = 1
while True:  # Infinite loop
    print(f"Count: {count}")
    count += 1

    if count > 5:
        print("Breaking out of loop")
        break

# Using continue to skip iterations
number = 0
while number < 10:
    number += 1
    if number % 2 == 0:  # Skip even numbers
        continue
    print(f"Odd number: {number}")
```

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
Breaking out of loop
Odd number: 1
Odd number: 3
Odd number: 5
Odd number: 7
Odd number: 9
```

6. Avoiding Infinite Loops

Always ensure your while loop condition can eventually become False to avoid infinite loops.

```
# Good practice: Always update condition variable
counter = 0
while counter < 5:
    print(f"Counter: {counter}")
    counter += 1  # This makes the condition eventually False

# Safety check for complex conditions
attempts = 0
max_attempts = 10

while some_condition and attempts < max_attempts:
    # Do something
    attempts += 1  # Safety counter
```

Exercises

1. Write a program that counts down from 10 to 1 using a while loop.
2. Create a number guessing game where the user keeps guessing until correct.
3. Calculate the factorial of a number using a while loop.
4. Create a program that keeps asking for numbers until user enters 0, then shows the sum.
5. Build a simple ATM withdrawal system with balance checking.

Practical Examples

Let's explore some practical examples of working with while loops in Python. These examples demonstrate real-world applications of condition-based repetition.

ATM Banking System

Here's a practical example of using while loops to create a simple ATM banking system with multiple operations.

In [5]:

```python
# ATM Banking System using while loops

# Initialize account details
account_balance = 1000.0
pin = "1234"
max_attempts = 3

print("Welcome to Python Bank ATM")
print("=" * 30)

# PIN verification with limited attempts
attempts = 0
pin_verified = False

while attempts < max_attempts and not pin_verified:
    entered_pin = input("Enter your 4-digit PIN: ")

    if entered_pin == pin:
        pin_verified = True
        print("PIN verified successfully!")
    else:
        attempts += 1
        remaining = max_attempts - attempts
        if remaining > 0:
            print(f"Incorrect PIN. {remaining} attempts remaining.")
        else:
            print("Account locked due to too many failed attempts.")

# Main banking operations (only if PIN is verified)
if pin_verified:
    while True:
        print(f"\nCurrent Balance: ${account_balance:.2f}")
        print("\n--- ATM Menu ---")
        print("1. Check Balance")
        print("2. Withdraw Money")
        print("3. Deposit Money")
        print("4. Exit")

        choice = input("Select an option (1-4): ")

        if choice == "1":
            print(f"Your current balance is: ${account_balance:.2f}")

        elif choice == "2":
            try:
                amount = float(input("Enter withdrawal amount: $"))
                if amount <= 0:
                    print("Amount must be positive!")
                elif amount > account_balance:
                    print("Insufficient funds!")
                else:
                    account_balance -= amount
                    print(f"Withdrawal successful! New balance: ${account_balance
            except ValueError:
                print("Please enter a valid amount!")
        elif choice == "3":
            try:
                amount = float(input("Enter deposit amount: $"))
                if amount <= 0:
                    print("Amount must be positive!")
                else:
                    account_balance += amount
                    print(f"Deposit successful! New balance: ${account_balance:.2f
            except ValueError:
                print("Please enter a valid amount!")

        elif choice == "4":
            print("Thank you for banking with us! Have a great day!")
            break

        else:
            print("Invalid option! Please select 1-4.")

print("\nSession ended.")
```

```
Welcome to Python Bank ATM
==============================
PIN verified successfully!

Current Balance: $1000.00

--- ATM Menu ---
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Your current balance is: $1000.00

Current Balance: $1000.00

--- ATM Menu ---
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Withdrawal successful! New balance: $800.00

Current Balance: $800.00

--- ATM Menu ---
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Deposit successful! New balance: $1100.00

Current Balance: $1100.00

--- ATM Menu ---
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Your current balance is: $1100.00

Current Balance: $1100.00

--- ATM Menu ---
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Thank you for banking with us! Have a great day!

Session ended.
```

Number Guessing Game with Statistics

This example demonstrates a while loop-based guessing game that tracks statistics and provides hints to the player.

In [6]:
```python
# Advanced number guessing game with statistics
```

```python
import random

print("Number Guessing Game")
print("=" * 25)

# Game settings
min_number = 1
max_number = 100
max_guesses = 7

# Game statistics
games_played = 0
total_guesses = 0
games_won = 0

play_again = "yes"

while play_again.lower() in ["yes", "y"]:
    # Start new game
    games_played += 1
    secret_number = random.randint(min_number, max_number)
    guesses_made = 0
    game_won = False

    print(f"\nGame {games_played}")
    print(f"I'm thinking of a number between {min_number} and {max_number}")
    print(f"You have {max_guesses} guesses to find it!")

    while guesses_made < max_guesses and not game_won:
        try:
            guess = int(input(f"\nGuess #{guesses_made + 1}: "))
            guesses_made += 1

            if guess == secret_number:
                game_won = True
                games_won += 1
                print(f"Congratulations! You found it in {guesses_made} guesses!"

            elif guess < secret_number:
                remaining = max_guesses - guesses_made
                if remaining > 0:
                    print(f"Too low! {remaining} guesses remaining.")

            else:  # guess > secret_number
                remaining = max_guesses - guesses_made
                if remaining > 0:
                    print(f"Too high! {remaining} guesses remaining.")

            # Provide additional hints based on how close the guess is
            if not game_won and guesses_made < max_guesses:
                difference = abs(guess - secret_number)
                if difference <= 5:
                    print("Very close!")
                elif difference <= 10:
                    print("Getting warmer!")
                elif difference <= 20:
                    print("Getting colder!")

        except ValueError:
            print("Please enter a valid number!")
```

```python
                guesses_made -= 1  # Don't count invalid input as a guess

    # End of game summary
    if not game_won:
        print(f"\nGame over! The number was {secret_number}")

    total_guesses += guesses_made

    # Show game statistics
    print(f"\nGame Statistics:")
    print(f"Games played: {games_played}")
    print(f"Games won: {games_won}")
    print(f"Win rate: {(games_won/games_played)*100:.1f}%")
    if games_played > 0:
        print(f"Average guesses per game: {total_guesses/games_played:.1f}")

    # Ask to play again
    play_again = input("\nWould you like to play again? (yes/no): ")

print("\nThanks for playing! Final Statistics:")
print(f"Total games: {games_played}")
print(f"Total wins: {games_won}")
if games_played > 0:
    print(f"Overall win rate: {(games_won/games_played)*100:.1f}%")
```

```
Number Guessing Game
=========================

Game 1
I'm thinking of a number between 1 and 100
You have 7 guesses to find it!
Too high! 6 guesses remaining.
Getting colder!
Too low! 5 guesses remaining.
Too low! 4 guesses remaining.
Too low! 3 guesses remaining.
Too low! 2 guesses remaining.
Too low! 1 guesses remaining.

Game over! The number was 72

Game Statistics:
Games played: 1
Games won: 0
Win rate: 0.0%
Average guesses per game: 7.0

Game 2
I'm thinking of a number between 1 and 100
You have 7 guesses to find it!
Too high! 6 guesses remaining.
Too high! 5 guesses remaining.
Getting colder!
Too low! 4 guesses remaining.
Very close!
Too high! 3 guesses remaining.
Getting colder!
Too high! 2 guesses remaining.
Too high! 1 guesses remaining.
Getting colder!

Game over! The number was 14

Game Statistics:
Games played: 2
Games won: 0
Win rate: 0.0%
Average guesses per game: 7.0

Thanks for playing! Final Statistics:
Total games: 2
Total wins: 0
Overall win rate: 0.0%
```

Key While Loop Rules to Remember

Let's review the important rules and best practices for working with while loops:

- Always update the condition variable inside the loop to avoid infinite loops
- Use while loops when you don't know the exact number of iterations needed
- Initialize condition variables before the while loop starts
- Use break to exit a loop early when a specific condition is met
- Use continue to skip the current iteration and move to the next one

- Add safety counters to prevent infinite loops in complex conditions
- Test your while loops with different inputs to ensure they terminate properly
- Use meaningful condition expressions that clearly show the loop's purpose
- Be careful with floating-point comparisons in while loop conditions
- Consider using try-except blocks for input validation within while loops
- Use proper indentation (4 spaces) for the loop body
- Document complex while loop conditions with comments for clarity

In [7]:
```python
# Examples of good while loop practices

# Example 1: Safe input validation with multiple conditions
print("User Registration System")
print("=" * 28)

# Validate username
username = ""
while len(username) < 3 or len(username) > 20 or " " in username:
    username = input("Enter username (3-20 chars, no spaces): ")
    if len(username) < 3:
        print("Username too short!")
    elif len(username) > 20:
        print("Username too long!")
    elif " " in username:
        print("Username cannot contain spaces!")

print(f"Username '{username}' is valid!")

# Example 2: Processing data until specific condition
print(f"\nNumber Processing System")
print("=" * 27)

numbers = []
sum_total = 0
positive_count = 0

print("Enter numbers (0 to finish):")

while True:
    try:
        number = float(input("Enter number: "))

        if number == 0:
            print("Finishing input...")
            break

        numbers.append(number)
        sum_total += number

        if number > 0:
            positive_count += 1

    except ValueError:
        print("Please enter a valid number!")
        continue

# Display results
if numbers:
    average = sum_total / len(numbers)
    print(f"\nResults:")
    print(f"Numbers entered: {len(numbers)}")
    print(f"Sum: {sum_total:.2f}")
    print(f"Average: {average:.2f}")
    print(f"Positive numbers: {positive_count}")
    print(f"Negative numbers: {len(numbers) - positive_count}")
else:
    print("No numbers were entered.")
```

```
User Registration System
============================
Username 'John' is valid!

Number Processing System
===========================
Enter numbers (0 to finish):
Finishing input...

Results:
Numbers entered: 3
Sum: 69.00
Average: 23.00
Positive numbers: 3
Negative numbers: 0
```

Course Information

Learn Python Programming from Scratch

Author: Prakash Ukhalkar

Topic: Python Control Flow - While Loops

Built with ❤️ for the Python learning community