Loop

Finite loop

```
n = 5
while n > 0:
  print(n)
  n = n - 1
print("Exit")
```

```
5
4
3
2
1
Exit
```

```
n = 0
while n > 0:
  print(n)
print("Exit")
```

```
Exit
```

Example_02 Infinite loop

```
n = 10
while n > 0:
  print(n)
print("Exit")
```

Breaking out of a loop
Example: 05 (aftershowing the flowchart)
Problem Statement: Interactive Greeting Response System
Background:
Creating an interactive program that responds to user inputs is a foundational skill in programming. Such programs can range from simple command-line interfaces to complex chatbots. Understanding how to handle user inputs and provide responses based on specific conditions is crucial in developing user-friendly software.

Task:
Develop a Python program that continuously prompts the user for input and responds based on the input received. The program should specifically recognize and respond to the greeting "Hi". If the user inputs "Hi", the program should respond with "okay" and then wait for the next input. If any other input is received, the program should terminate. This task aims to demonstrate the use of a continuous loop (while) combined with conditional statements (if-else) to create a basic interactive interface.

```
while True:
  x = input(">")
  if x == "Hi":
    print("okay")
  else:
    break
```

```
>Hi
okay
>j
```

```
while True:
  x = input("enter the name of the cell type:")
  if x == "Neuron":
    print("Cell identified as Neuron. Neurons are fundamental units of the brain and nervous system.")
  else:
    print("Cell type not recognized. Exiting.")
    break
```

```
enter the name of the cell type:hi
Cell type not recognized. Exiting.
```

need a simplified version for better understanding

```
while True:
    dna_sequence = input("> ").upper()

    # Exit condition
    if dna_sequence == "EXIT":
        break

    # Check if the sequence contains only valid nucleotides
    is_valid_sequence = True
    for nucleotide in dna_sequence:
        if nucleotide not in "ACGT":
            print("Invalid DNA sequence.")
            is_valid_sequence = False
            break

    if not is_valid_sequence:
      continue
    # Calculate the complementary sequence
    complementary_sequence = ""
    for nucleotide in dna_sequence:
        if nucleotide == 'A':
            complementary_sequence += 'T'
        elif nucleotide == 'T':
            complementary_sequence += 'A'
        elif nucleotide == 'C':
            complementary_sequence += 'G'
        elif nucleotide == 'G':
            complementary_sequence += 'C'

    # Calculate GC content
    gc_count = dna_sequence.count('G') + dna_sequence.count('C')
    gc_content = (gc_count / len(dna_sequence)) * 100

    print("Complementary Sequence:", complementary_sequence)
    print("GC Content: {:.2f}%".format(gc_content))
```

```
> acgt
Complementary Sequence: TGCA
GC Content: 50.00%
> aatt
Complementary Sequence: TTAA
GC Content: 0.00%
> er
Invalid DNA sequence.
> exit
```

Example_06 (combination of conditional statement and loop)

Problem Statement: Loop and Conditional-Based Output Demonstration

Background:

Understanding loops and conditional statements is fundamental in programming. These concepts are essential for tasks that require repetitive actions and decisions based on certain conditions. Mastering these concepts allows for efficient and dynamic code that can adapt to varying inputs and conditions.

Task:

Develop a Python program that demonstrates the use of a loop combined with a conditional statement. The program should iterate over a sequence of numbers (in this case, from 0 to 4) and perform checks at each iteration. If a number in the sequence is greater than 2, the program should output a specific conditional message for that number. The objective is to showcase how loops and conditionals can work together in a program.

```
x = 5
for i in range(5):
  print(i)
  if i > 2:
    print("the conditional output is: ", i)
print("all done")
```

```
0
1
2
3
the conditional output is:  3
4
the conditional output is:  4
all done
```

Example: 10

Problem Statement: Maximal Heart Rate of Animals

Background:

Heart rate, typically measured in beats per minute (BPM), varies significantly across different animal species. This rate can be indicative of an animal's metabolic rate and overall physiology. In biology, comparing such physiological parameters across species can provide insights into their ecological adaptations and evolutionary processes.

Task:

Create a Python program that helps find the animal with the highest heart rate from a given list of animals and their corresponding heart rates. Your program should take a list of tuples as input, with each tuple containing the name of an animal and its average heart rate. The program should then determine the animal with the highest heart rate and display its name along with its heart rate.

Data Input:

Define a list of tuples, each containing an animal's name and its average heart rate. Find Maximum Heart Rate: Write a program to identify the animal with the highest heart rate in the list. Output: Display the name of the animal with the highest heart rate and the rate itself

```
 # Sample Data
 animal_heart_rates = [
     ("Hummingbird", 1200),
     ("Elephant", 30),
     ("Rabbit", 205),
     ("Blue Whale", 9),
     ("Cheetah", 120)
 ]
 # Expected Output:
 # The animal with the highest heart rate is the Hummingbird with 1200 BPM.
```

```
# Sample data: list of tuples with animal names and their heart rates
animal_heart_rates = [
    ("Hummingbird", 1200),
    ("Elephant", 30),
    ("Rabbit", 205),
    ("Blue Whale", 9),
    ("Cheetah", 120)
]

# Initialize variables to store the animal with the highest heart rate
max_heart_rate = 0
animal_with_max_rate = ""

# Iterate through each tuple in the list
for animal, heart_rate in animal_heart_rates:
    # Check if the current animal's heart rate is greater than the max found so far
    if heart_rate > max_heart_rate:
        max_heart_rate = heart_rate
        animal_with_max_rate = animal

# Output the result
print(f"The animal with the highest heart rate is the {animal_with_max_rate} with {max_heart_rate} BPM.")
```