While Loop

Example 01: Finite loop

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

```
5
4
3
2
1
```

```
n = 5
while n > 10:
  print(n)
  n = n - 1
print('Exit')
```

```
Exit
```

Example 02: infinite loop

```
n = 5
while n > 0:
  print(n)
  n = n + 1
print('Exit')
```

Example 03: Why do we use break

```
a  = input('enter a number: ')
a = int(a)
```

```
enter a number: 34
```

```
type(a)
```

```
int
```

```
while True:
  a = input('enter a number: ')
  a = int(a)
  if a == 2:
    print('okay')
```

```
enter a number: 2
okay
enter a number: 1
enter a number: 4
enter a number: 5
enter a number: 6
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-10-a0447eeb7c88> in <cell line: 1>()
      1 while True:
----> 2   a = input('enter a number: ')
      3   a = int(a)
      4   if a == 2:
      5     print('okay')

1 frames
/usr/local/lib/python3.10/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
    893             except KeyboardInterrupt:
    894                 # re-raise KeyboardInterrupt, to truncate traceback
--> 895                 raise KeyboardInterrupt("Interrupted by user") from None
    896             except Exception as e:
    897                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

```
while True:
  a = input('enter a number: ')
  a = int(a)
  if a == 2:
    print('okay')
  elif a == 4:
    break
```

```
enter a number: 2
okay
enter a number: 3
enter a number: 6
enter a number: 5
enter a number: 4
```

Start coding or generate with AI.

Practice: 01

Background:

In the diverse world of cellular biology, different cell types play unique roles in the body's structure and function. Identifying these cell types is crucial for understanding biological processes, diagnosing diseases, and developing treatments. Neurons, for example, are the building blocks of the nervous system, transmitting information to different parts of the body. An interactive program that can identify cell types based on user input could serve as an educational tool for students and enthusiasts alike.

Task:

Develop a Python program that continuously prompts the user to enter the name of a cell type. The program should specifically recognize the cell type "Neuron" and provide a brief description of its biological significance. If the user enters any cell type other than "Neuron", the program should notify the user that the cell type is not recognized and then terminate. This task aims to demonstrate the use of a continuous loop combined with conditional statements to create a basic interactive application that can engage users in cell biology.

Start coding or generate with AI.

Example: 05 (continue)

```
a = 4
while True:
  if a == 5:
    print('okay')
    continue
  else:
    break
```

Practice: 02

Problem Statement: Sample Identifier Processing System

Background:

In laboratory research, especially within biology and related fields, managing and categorizing biological samples is a fundamental task. Samples often come with unique identifiers that carry specific meanings, such as indicating the sample's verification status or signaling the end of a sample batch.

Task:

Create a Python program that simulates the processing of biological sample identifiers. The program should continuously prompt the user to enter sample IDs. If an ID starts with '#', the program recognizes this as a preliminary sample and skips it. If an ID starts with '$', this signifies the end of the sample batch, and the program terminates. Otherwise, the program should acknowledge the sample ID for processing.

```
sample = '#12345678'
```

```
sample[0]
```

```
'#'
```

For Loop

Example: 06 (For Loop)

```
lst = [1,2,3,4,5]
for i in lst:
  print(i)
```

```
1
2
3
4
5
```

Practice_03

Problem Statement: Student Marks Evaluation System
Background:
In educational institutions, evaluating student performance is crucial for both teachers and students to understand academic progress. Typically, this evaluation involves assessing students' marks against a predefined cutoff to determine whether they have passed or failed a particular subject or exam.

Task:
Develop a Python program that processes a list of student marks and classifies each mark as either 'Passed' or 'Failed' based on a specified cutoff value. The program should iterate through a collection of marks, compare each mark against the cutoff, and print the result for each student. For this task, assume the cutoff mark for passing is 40.

Program Requirements:
Input Data: Start with a list of student marks. For the purpose of this problem, use the provided list: [56, 34, 90, 85, 79, 65, 67, 92, 82, 88].
Cutoff Value: Define a variable for the cutoff marks. In this case, set the cutoff to 40.

```
Start coding or generate with AI.
```

Practice 04
Problem Statement: Classifying Gene Expressions
Background:
Gene expression analysis is a critical component of genetic research and diagnostics. Classifying gene expression levels helps in understanding the roles of genes in normal cellular functions and diseases.

Task:
Develop a Python program to classify a list of gene expression levels. The program should iterate through each expression level and classify it based on predefined thresholds: expressions below a low threshold are classified as "Low", expressions above a high threshold are classified as "High", and all others are classified as "Moderate".

```
Let,
gene_expressions = [0.2, 1.5, 0.75, 2.3, 3.0, 0.4]

And Thresholds for classification:
low_threshold = 0.5
high_threshold = 2.0
```

```
range(10)
```

```
range(0, 10)
```

```
for i in range(1,10,3):
  print(i)
```

```
1
4
7
```

Practice 05
Task:
Develop a Python program to calculate the weighted average of a series of marks obtained by a student in different courses, taking into account the credit value of each course. For this task, assume that all courses have the same credit value of 3. The program should multiply each mark by the credit value, sum these weighted marks, and then divide the total by the sum of all credits to find the weighted average.

Program Requirements:
Input Data: A list of marks obtained by the student in various courses: [56, 34, 90, 85].
Credit Value: A fixed credit value of 3 for each course.

```
marks = [56, 34, 90, 85]
sum = 0
for mark in marks:
  sum = sum + mark
print(sum)
```

```
265
```

Example: 07

	Course 1	Course 2	Course 3
Quiz	8	7	9
Mid	20	23	18
Final	40	55	57
Total	?	?	?

```
courses = [[8,20,40],[7,23,55],[9,18,57]]
for course in courses:
  sum = 0
  for i in course:
    sum = sum + i
  print(sum)
```

```
68
85
84
```

```
Start coding or generate with AI.
```

Practice_06

Problem Statement: Calculating Weighted Average Grade

Background:

In academic settings, courses often have different credit values, and the grades students receive in these courses impact their overall academic performance differently. A course with a higher credit value has a greater impact on the student's Grade Point Average (GPA) or overall grade. To accurately assess a student's performance, it's essential to calculate the weighted average grade, taking into account both the grades received and the credit value of each course.

Task:

Develop a Python program to calculate the weighted average grade for a set of courses. The program should take two lists as input: one containing the grades (marks) received in each course and the other containing the corresponding credit values for these courses. The program should then calculate the total sum of weighted marks and the total sum of credits before computing the final weighted average grade.

```
marks = [56, 34, 90, 85]
credit = [3 , 3, 2, 4]
```

```
len(marks)
```

```
4
```

```
sum_marks = 0
sum_credit = 0
count = 0
for mark in marks:
  temp = mark * credit[count]
  sum_marks = sum_marks + temp
  sum_credit = sum_credit + credit[count]
  count = count + 1
print(sum_marks)
print(sum_credit)
result = sum_marks / sum_credit
print(result)
```

```
790
12
65.83333333333333
```

```
marks = [56, 34, 90, 85]
credit = [3 , 3, 2, 4]
```

```
sum_marks = 0
sum_credit = 0
for i in range(len(marks)):
  temp = marks[i] * credit[i]
  sum_marks = sum_marks + temp
  sum_credit = sum_credit + credit[i]
print(sum_marks)
print(sum_credit)
result = sum_marks / sum_credit
print(result)
```

```
790
12
65.83333333333333
```

Practice_07

Problem Statement: Filtering Erroneous Temperature Readings

Background:

In environmental monitoring and climate research, accurate temperature data collection is crucial. Networks of temperature sensors deployed across various locations frequently gather temperature readings to analyze climate patterns, changes, and anomalies. However, sensor malfunctions or environmental interferences can result in erroneous readings that significantly deviate from expected values, either being too high or too low. Identifying and filtering out these erroneous readings is essential for ensuring the accuracy and reliability of temperature data analysis.

Task:

Develop a Python program that processes a series of temperature readings collected from a network of sensors. The program should identify and ignore any readings that are marked as erroneous due to being outside an acceptable temperature range. The acceptable range is determined by predefined minimum and maximum temperature thresholds. The goal is to filter out these erroneous readings and only analyze valid temperature data.

```
# List of temperature readings from different sensors
temperature_readings = [22, -5, 48, 21, 45, 110, 23, -20, 25, 18]

# Define acceptable temperature range
min_temp = 10
max_temp = 40
```

Start coding or generate with AI.