In [ ]:
```
Universal Input List
You can use this list for most questions that require a list or numeric collecti

numbers = [5, -2, 12, 7, 0, 33, -10, 8]
words = ["apple", "banana", "kiwi", "fig", "strawberry", "grape"]
```

Functions

In [1]:
```
# 1. Function to return the square of a number
def square_number(n):
    return n * n
print(square_number(4))
```

```
16
```

In [2]:
```
# 2. Function to return the larger of two numbers
def larger_number(a, b):
    return a if a > b else b
print(larger_number(10, 20))
```

```
20
```

In [3]:
```
def print_welcome():
    for _ in range(5):
        print("Welcome to AIML")

print_welcome()
```

```
Welcome to AIML
Welcome to AIML
Welcome to AIML
Welcome to AIML
Welcome to AIML
```

In [4]:
```
def check_even_odd(n):
    if n % 2 == 0:
        return "Even"
    else:
        return "Odd"

print(check_even_odd(7))
print(check_even_odd(8))
```

```
Odd
Even
```

In [5]:
```
def sum_of_list(lst):
    return sum(lst)

numbers = [5, -2, 12, 7, 0, 33, -10, 8]
print(sum_of_list(numbers))
```

```
53
```

Loops

In [6]:
```python
# 1. For loop to print numbers from 1 to 10
print("Numbers from 1 to 10:")
for i in range(1, 11):
    print(i)
```

```
Numbers from 1 to 10:
1
2
3
4
5
6
7
8
9
10
```

In [7]:
```python
# 2. While loop to print all even numbers between 1 and 20
print("\nEven numbers between 1 and 20:")
i = 1
while i <= 20:
    if i % 2 == 0:
        print(i)
    i += 1
```

```
Even numbers between 1 and 20:
2
4
6
8
10
12
14
16
18
20
```

In [8]:
```python
# 3. Loop to print each character of a given string
input_string = "Artificial Intelligence"
print("\nCharacters in the string:")
for char in input_string:
    print(char)
```

```
Characters in the string:
A
r
t
i
f
i
c
i
a
l

I
n
t
e
l
l
i
g
e
n
c
e
```

In [9]:
```python
# 4. Loop that prints the multiplication table of 5
print("\nMultiplication table of 5:")
for i in range(1, 11):
    print(f"5 x {i} = {5 * i}")
```

```
Multiplication table of 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

In [10]:
```python
# 5. Loop that prints only the positive numbers from the list
print("\nPositive numbers from the list:")
for num in numbers:
    if num > 0:
        print(num)
```

```
Positive numbers from the list:
5
12
7
33
8
```

Try and Except

In [11]:
```python
# 1. Function that divides two numbers with divide-by-zero handling
def divide_numbers(a, b):
```

```
    try:
        result = a / b
        return result
    except ZeroDivisionError:
        return "Error: Cannot divide by zero."
print(divide_numbers(10, 2))
```

```
5.0
```

In [ ]:
```
# 2. Ask user for a number and handle non-numeric input

try:
    user_input = input("Enter a number: ")
    number = float(user_input)
    print(f"You entered: {number}")
except ValueError:
    print("Error: That is not a valid number.")
```

In [12]:
```
#Open a file and print custom error if not found

filename = "example.txt"

try:
    with open(filename, 'r') as file:
        content = file.read()
        print(content)
except FileNotFoundError:
    print(f"Error: The file '{filename}' was not found.")
```

```
Error: The file 'example.txt' was not found.
```

In [13]:
```
#4. Add two numbers and raise an error if inputs are not numbers

def add_numbers(a, b):
    if not isinstance(a, (int, float)) or not isinstance(b, (int, float)):
        raise TypeError("Both inputs must be numbers.")
    return a + b

# Example usage
try:
    result = add_numbers(5, "three")
    print(f"Sum: {result}")
except TypeError as e:
    print(f"Error: {e}")
```

```
Error: Both inputs must be numbers.
```

In [ ]:
```
# 5. Read an integer from the user safely using try and except

try:
    user_input = input("Enter an integer: ")
    number = int(user_input)
    print(f"You entered the integer: {number}")
except ValueError:
    print("Error: That is not a valid integer.")
```

Lambda, Map, Filter

In [1]:
```
add_10 = lambda x: x + 10

# Example usage
print(add_10(5))  # Output: 15
```

```
15
```

In [2]:
```
numbers = [5, -2, 12, 7, 0, 33, -10, 8]

squared_numbers = list(map(lambda x: x**2, numbers))
print(squared_numbers)  # Output: [25, 4, 144, 49, 0, 1089, 100, 64]
```

```
[25, 4, 144, 49, 0, 1089, 100, 64]
```

In [3]:
```
numbers = [5, -2, 12, 7, 0, 33, -10, 8]

even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)  # Output: [-2, 12, 0, -10, 8]
```

```
[-2, 12, 0, -10, 8]
```

In [4]:
```
words = ["apple", "banana", "kiwi", "fig", "strawberry", "grape"]

uppercase_words = list(map(lambda x: x.upper(), words))
print(uppercase_words)  # Output: ['APPLE', 'BANANA', 'KIWI', 'FIG', 'STRAWBERRY
```

```
['APPLE', 'BANANA', 'KIWI', 'FIG', 'STRAWBERRY', 'GRAPE']
```

In [5]:
```
numbers = [5, -2, 12, 7, 0, 33, -10, 8]

greater_than_50 = list(filter(lambda x: x > 50, numbers))
print(greater_than_50)  # Output: []
```

```
[]
```

Variables

In [6]:
```
count = 0  # global variable

def increment():
    global count
    count += 1
    print("Inside function, count =", count)

increment()
print("Outside function, count =", count)
```

```
Inside function, count = 1
Outside function, count = 1
```

In [7]:
```
value = 100  # global variable

def demo():
    value = 50  # local variable
    print("Inside function, value =", value)

demo()
print("Outside function, value =", value)
```

```
Inside function, value = 50
Outside function, value = 100
```

In [8]:
```python
total = sum(numbers)  # global variable

def modify_total():
    global total
    total += 10
    print("Modified total inside function =", total)

modify_total()
print("Modified total outside function =", total)
```

```
Modified total inside function = 63
Modified total outside function = 63
```

In [9]:
```python
global_word = "orange"

def set_local_word():
    local_word = "pineapple"
    print("Local word inside function:", local_word)

def print_global_word():
    print("Global word inside function:", global_word)

set_local_word()
print_global_word()
```

```
Local word inside function: pineapple
Global word inside function: orange
```

In [10]:
```python
def create_variable():
    internal_value = "This is local"
    print("Inside function:", internal_value)

create_variable()

# This will cause an error if uncommented, because internal_value is not in glob
# print("Outside function:", internal_value)  # Uncommenting this will raise Nam
```

```
Inside function: This is local
```

In [ ]:
```python

```