Exercise 1: Create a function with a default argument

Write a program to create a function show_employee() using the following conditions.

It should accept the employee's name and salary and display both.

If the salary is missing in the function call then assign default value 9000 to salary

In [32]:
```
def show_employee(employee_s_name, salary=9000):
    return (f"Name : {employee_s_name}, Salary : {salary}")

# Call the function with explicit salary value (12000) for employee "Ben"
result1 = show_employee("Ben", 12000)
print(result1)  # Print the formatted result for "Ben"

# Call the function without explicitly providing a salary value for employee "
result2 = show_employee("Jessa")
print(result2)  # Print the formatted result for "Jessa"
```

```
Name : Ben, Salary : 12000
Name : Jessa, Salary : 9000
```

Exercise 2: Create an inner function to calculate the addition in the following way

Create an outer function that will accept two parameters, a and b

Create an inner function inside an outer function that will calculate the addition of a and b

At last, an outer function will add 5 into addition and return it

In [33]:
```
def outer_function(a, b):
    def inner_function():
        addition = a + b  # Calculate the sum of 'a' and 'b'
        return addition   # Return the sum
    result = inner_function()  # Call the inner function and store its result
    return result + 5  # Return the result of the inner function plus 5

# Test the outer function by calling it with arguments 3 and 7
result = outer_function(3, 7)

# Print the final result
print(result)
```

```
15
```

Exercise 3: Generate a Python list of all the even numbers between 4 to 30

In [34]:
```
def even_num(a, b):
    # List comprehension to generate a list of even numbers in the specified r
    # 'a' is the starting point (inclusive), 'b' is the ending point (exclusive
    return [item for item in range(a, b) if item % 2 == 0]

# Call the function to get the list of even numbers between 4 and 30
# The result will be a list of even numbers from 4 to 29
result = even_num(4, 30)

# Print the list of even numbers
print(result)
```

```
[4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

Exercise 4: Lambda Function to Check if value is in a List

Given a list, the task is to write a Python program to check if the value exists in the list or not. using the lambda function.

In [32]:
```
a = [4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

# def valu_exists_in_list(a, b):
#     for item in a:
#         if b in a:  # This checks if b is in the entire list 'a'
#             return "yes"
#         else:
#             return "not"

# result = valu_exists_in_list(a, 10)
# print(result)

a = [4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

valu_exists_in_list = lambda lst, val: "yes" if any(item == val for item in ls

result = valu_exists_in_list(a, 10)
print(result)
```

```
yes
```

Exercise 5: Sort list of tuples with their sum

Sort the points based on their sum of elements in the tuples

points = [(1, 2), (5, 3), (0, 7), (3, 1)]

In [24]:
```
# Define a list of points, where each point is represented as a tuple (x-coord
points = [(1, 2), (5, 3), (0, 7), (3, 1)]

# Use the sort() function to sort the 'points' list based on a custom sorting k
# The sorting key is defined using a lambda function that takes a point 'a' as
points.sort(key=lambda a: a[0] + a[1])

# After sorting, the 'points' list will be rearranged based on the sum of x-coo

# Print the sorted list of points
print(points)
```

```
[(1, 2), (3, 1), (0, 7), (5, 3)]
```

Exercise 6 :

Write a python function, which will find all such numbers between 1000 and 3000 (both included) such that each digit of the number is an even number. Return the results as a list

In [33]:
```
def find_even_digit_numbers(start, end):
    even_digit_numbers = []  # Initialize an empty list to store even-digit nu

    # Iterate through the range of numbers from 'start' to 'end' (both include
    for num in range(start, end + 1):
        # Check if the number is even (divisible by 2)
        if num % 2 == 0:
            even_digit_numbers.append(num)  # If even, add the number to the l
        else:
            pass  # If not even, do nothing (skip to the next number)

    return even_digit_numbers  # Return the list of even-digit numbers

# Call the function with the specified range
result = find_even_digit_numbers(1000, 1010)

# Print the list of even-digit numbers
print(result)
```

```
[1000, 1002, 1004, 1006, 1008, 1010]
```

Exercise 7 :

Write a python function that accepts a sentence and calculate and return the number of letters and digits. Suppose the following input is supplied to the program: hello world! 123 Then, the output should be:

LETTERS 10

DIGITS 3

In [74]:
```
def calculate_num_letter(input_1):
    digit = 0  # Initialize a counter for digits
    letters = 0  # Initialize a counter for letters

    # Iterate through each character in the input string
    for item in input_1:
        if item.isalpha():  # Check if the character is a letter using isalpha
            letters = letters + 1  # Increment the letter counter

        elif item.isdigit():  # Check if the character is a digit using isdigi
            digit = digit + 1  # Increment the digit counter

        else:
            pass  # If neither a letter or a digit, skip to the next character

    # Print the original user input, digit count, and letter count
    print(f"user input: {input_1}")
    print(f"DIGITS: {digit}")
    print(f"LETTERS: {letters}")

# Call the function with the input string "hello world! 123"
calculate_num_letter("hello world! 123")
```

```
user input: hello world! 123
DIGITS: 3
LETTERS: 10
```

Exercise 8 MAP:

Write a Python program to convert all the characters into uppercase and lowercase and eliminate duplicate letters from a given sequence. Use the map() function

In [34]:
```python
def process_string(input_string):
    # Convert input to uppercase and print
    uppercase_result = input_string.upper()
    print("Uppercase:", uppercase_result)

    # Convert input to lowercase and print
    lowercase_result = input_string.lower()
    print("Lowercase:", lowercase_result)

    # Initialize an empty string to store unique characters
    unique_result = ""

    # Iterate through each character in the input string
    for unique in input_string:
        if unique not in unique_result:
            unique_result += unique  # Use += to concatenate unique characters
        else:
            pass

    # Print the result of unique characters
    print("Unique letters:", unique_result)

# Get input from the user
input_sequence = input("Enter a sequence of characters: ")

# Call the function to process the input sequence
process_string(input_sequence)
```

```
Enter a sequence of characters: muhammed anu rashik
Uppercase: MUHAMMED ANU RASHIK
Lowercase: muhammed anu rashik
Unique letters: muhaed nrsik
```

In [35]:
```
def process_string(input_string):
    # Define a function to process each character
    def process_char(char):
        return char.upper(), char.lower()

    # Apply the process_char function to each character using map()
    uppercase_results, lowercase_results = map(list, zip(*map(process_char, in

    # Join the results to create strings
    uppercase_result = ''.join(uppercase_results)
    lowercase_result = ''.join(lowercase_results)

    # Create a set from the input string to remove duplicates and join the set
    unique_result = ''.join(set(input_string))

    # Print the results
    print("Uppercase:", uppercase_result)
    print("Lowercase:", lowercase_result)
    print("Unique letters:", unique_result)

# Get input from the user
input_sequence = input("Enter a sequence of characters: ")

# Call the function to process the input sequence
process_string(input_sequence)
```

```
Enter a sequence of characters: muhammed anu rashik
Uppercase: MUHAMMED ANU RASHIK
Lowercase: muhammed anu rashik
Unique letters: ndsrikua emh
```

Exercise 9 MAP:

Write a Python program to add two given lists and find the difference between them. Use the map() function

In [6]:
```
# Define two lists
list1 = [10, 20, 30, 40, 50]
list2 = [5, 10, 15, 20, 25]

# Define functions for addition and subtraction
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

# Use map() to add corresponding elements from both lists
sum_result = list(map(add, list1, list2))

# Use map() to subtract corresponding elements from both lists
diff_result = list(map(subtract, list1, list2))

print("Sum:", sum_result)   # Output: Sum: [15, 30, 45, 60, 75]
print("Difference:", diff_result)  # Output: Difference: [5, 10, 15, 20, 25]
```

```
Sum: [15, 30, 45, 60, 75]
Difference: [5, 10, 15, 20, 25]
```

Exercise 10 Filter:

Write a Python program to filter the height and weight of students, which are stored in a dictionary using lambda.

In [63]:
```
# Student data stored in a dictionary
student_data = {
    'Cierra Vega': (6.2, 71),
    'Alden Cantrell': (5.9, 65),
    'Kierra Gentry': (6.0, 68),
    'Pierre Cox': (5.8, 66)
}

# Filter student data using filter(), lambda, and dict()
filtered_students = dict(filter(lambda x: x[1][0] >= 6.0 and x[1][1] >= 70, st

# Print the filtered student data
print(filtered_students)
```

```
{'Cierra Vega': (6.2, 71)}
```

Exercise 11 Filter:

Write a Python program to remove all elements from a given list present in another list using lambda.

In [64]:
```
def rem(list1, list2):
    newls = list(filter(lambda x: x not in list2, list1))
    return newls

# Example usage of the rem function
list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list2 = [2, 4, 6, 8]
filtered_list1 = rem(list1, list2)
print(filtered_list1)
```

```
[1, 3, 5, 7, 9, 10]
```

Exercise 12 Reduce:

Write a Python program to calculate the product of a given list of numbers using lambda.

In [66]:
```
import functools

# List of numbers
list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Calculate the product of all elements using functools.reduce() and lambda
product = functools.reduce(lambda a, b: a * b, list1)

print(product)
```

```
3628800
```

Exercise 13 Reduce:

Write a Python program to multiply all the numbers in a given list using lambda.

In [17]:
```
import functools

# List of numbers
l1 = [4, 3, 2, 2, -1, 18]

# Calculate the numbers of all elements using functools.reduce() and lambda
product = functools.reduce(lambda a, b: a * b, l1)

print(product)
```

```
-864
```

Exercise 14 Reduce:

Write a Python program to calculate the average value of the numbers in a given tuple of tuples using lambda.

Original Tuple : ((10, 10, 10), (30, 45, 56), (81, 80, 39), (1, 2, 3))

In [36]:
```
data = ((10, 10, 10), (30, 45, 56), (81, 80, 39), (1, 2, 3))

# Calculate the average of numbers in each group using lambda and map
average_values = tuple(map(lambda x: sum(x) / len(x), zip(*data)))

print("Average value of the numbers:", average_values)
```

```
Average value of the numbers: (30.5, 34.25, 27.0)
```

Exercise 15:

Write a Python program to sort a given mixed list of integers and strings using lambda. Numbers must be sorted before strings.

Original list: [19, 'red', 12, 'green', 'blue', 10, 'white', 'green', 1]

Sort the said mixed list of integers and strings:[1, 10, 12, 19, 'blue', 'green', 'green', 'red', 'white']

In [16]:
```
def categorize_and_sort_list(input_list):
    string_list = []  # Initialize an empty list for strings
    integer_list = []  # Initialize an empty list for integers

    for item in input_list:  # Loop through each item in the input list
        if type(item) == int:  # Check if the item's type is integer
            integer_list.append(item)  # If it's an integer, add to integer_lis
        elif type(item) == str:  # Check if the item's type is string
            string_list.append(item)  # If it's a string, add to string_list

    sorted_integer_list = sorted(integer_list)  # Sort the integer_list
    sorted_string_list = sorted(string_list)  # Sort the string_list

    return sorted_integer_list + sorted_string_list  # Return combined and sor

a = [19, 'red', 12, 'green', 'blue', 10, 'white', 'green', 1]  # Original list
result = categorize_and_sort_list(a)  # Call the function with the list
print(result)  # Print the sorted and combined list
```

```
[1, 10, 12, 19, 'blue', 'green', 'green', 'red', 'white']
```

Exercise 16:

Write a Python program to count the occurrences of items in a given list using lambda.

Original list : [3, 4, 5, 8, 0, 3, 8, 5, 0, 3, 1, 5, 2, 3, 4, 2]

Count the occurrences of the items in the said list : {3: 4, 4: 2, 5: 3, 8: 2, 0: 2, 1: 1, 2: 2}

In [25]:
```
def occurrences_of_items(item):
    list_count = {}  # Create an empty dictionary to store item counts
    for num in item:  # Loop through each item in the input list
        if num in list_count:  # Check if the item is already a key in the dic
            list_count[num] += 1  # If yes, increment its count
        else:
            list_count[num] = 1  # If not, add it as a key with a count of 1
    print(list_count)  # Print the dictionary of item occurrences

a = [3, 4, 5, 8, 0, 3, 8, 5, 0, 3, 1, 5, 2, 3, 4, 2]  # Original list
occurrences_of_items(a)  # Call the function with the list as input
```

```
{3: 4, 4: 2, 5: 3, 8: 2, 0: 2, 1: 1, 2: 2}
```

Exercise 17:

Write a Python program to remove None values from a given list using the lambda function.

Original list: [12, 0, None, 23, None, -55, 234, 89, None, 0, 6, -12]

Remove None value from the said list:[12, 0, 23, -55, 234, 89, 0, 6, -12]

In [30]:
```
my_list = [12, 0, None, 23, None, -55, 234, 89, None, 0, 6, -12]

# Remove multiple occurrences of an item by value
value_to_remove = None
while value_to_remove in my_list:
    my_list.remove(value_to_remove)

print(my_list)  # Output: [1, 2, 4, 5]
```

```
[12, 0, 23, -55, 234, 89, 0, 6, -12]
```

In [31]:
```
my_list = [12, 0, None, 23, None, -55, 234, 89, None, 0, 6, -12]

[item for item in my_list if item != None]
```

Out[31]:
```
[12, 0, 23, -55, 234, 89, 0, 6, -12]
```