PYTHON LAB INTERNALS

1. Create a function called outer_function that takes two parameters, a and b. Within this function, define an inner function called inner_function that returns the sum of a and b.

In outer_function, add 5 to the result from inner_function and return this final value to the caller.

```
def outer_function(a,b):
    def inner_function():
        return a+b
    return inner_function()+5
print(outer_function(3,4))
```

#output : 12

2. Define two Python functions to determine the largest of three numbers.

- **Create a helper function that takes two numbers and returns the larger one.**
- **Create a main function that takes three numbers, uses the helper function to compare values, and returns the largest of the three.**

```
def max_of_two(a,b):
    return a if a>b else b
def max_of_three(x,y,z):
    return max_of_two(x,max_of_two(y,z))
print("Largest of three number is:",max_of_three(32,12,74))
```

#output : Largest of three number is: 74

3. Create two functions, sum_of_numbers() and product_of_numbers(), each using Python's *args to accept a variable number of numeric arguments.

- **sum_of_numbers() should return the total of all numbers passed in.**
- **multiply_numbers() should return the product of all numbers passed in.**

- **For example, sum_of_numbers(1, 2, 3, 4) should return 10 and multiply_numbers(1, 2, 3, 4) should return 24."**

```
def sum_of_numbers(*args):
    return sum(args)
def prod_of_numbers(*args):
    from math import prod
    return prod(args)
print(sum_of_numbers(1,2,4,5))
print(prod_of_numbers(11,31))
```

#output:

```
12
341
```

4. Define a Python recursive function to print the Fibonacci series up to n_terms.

```
def fibonacci(n, a=0, b=1):
    if n > 0:
        print(a, end="\n")
        fibonacci(n - 1, b, a + b)
fibonacci(7)
```

# Output: 0 1 1 2 3 5 8

5. Write a Python program that allows the user to choose between computing a factorial or printing a Fibonacci series (without recursion).

```
def factorial(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
```

```
def fibonacci(n):
    a, b = 0, 1
    for i in range(n):
        print(a, end=" ")
        a, b = b, a + b
    print()

while True:
    choice = input("\n1: Factorial\n2: Fibonacci\n3: Exit\n Choose (1/2/3): ")

    if choice == '1':
        num = int(input("Enter a number: "))
        print(f"Factorial: {factorial(num)}")
    elif choice == '2':
        num = int(input("Enter terms: "))
        fibonacci(num)
    elif choice == '3':
        break
    else:
        print("Invalid choice! Try again.")
```

6. Write a menu-driven Python program that lets the user check if a number is even/odd or prime.

```
def is_even_odd(n):
    return "Even" if n % 2 == 0 else "Odd"


def is_prime(n):
    if n < 2:
        return "Not Prime"
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return "Not Prime"
    return "Prime"


while True:
    print("\n1: Check Even/Odd\n 2: Check Prime\n 3: Exit\n")
    choice = input("Choose (1/2/3): ")
    if choice == '1':
        num = int(input("Enter a number: "))
        print(f"The number {num} is {is_even_odd(num)}.")
    elif choice == '2':
        num = int(input("Enter a number: "))
        print(f"The number {num} is {is_prime(num)}.")
    elif choice == '3':
        print("Exiting the program.")
        break
    else:
        print("Invalid choice!")
```

7. Write a Python program that allows the user to reverse a number or reverse a string. reverse a number without converting it into a string also check if the given number is a palindrome.

```
def reverse_number(n):
    rev = 0
    while n > 0:
        rev = rev * 10 + n % 10
        n //= 10
    return rev

while (choice := input("\n1: Reverse Number\n 2: Reverse String\n 3: Check Palindrome\n 4: Exit\nChoose: ")) != '4':
    if choice in ['1', '3']:
        num = int(input("Enter a number: "))
        rev = reverse_number(num)
        print(f"Reversed: {rev}")
        if choice == '3':
            print(f"Palindrome: {num == rev}")
    elif choice == '2':
        print(f"Reversed: {input('Enter a string: ')[::-1]}")
    else:
        print("Invalid choice!")
```

8. Write a menu-driven Python program that displays the following patterns:

```
i)   * * * *          ii) *
     * * *                * *
     * *                  * * *
     *                    * * * *
```

```
def pattern1(n):
    for i in range(n, 0, -1):
        print("* " * i)
```

```
def pattern2(n):
    for i in range(1, n + 1):
        print("* " * i)

while True:
    choice = input("\n1:Decreasing order\n 2:Increasing order\n 3:Exit\n choose(1,2,3):")
    if choice=='1':
        num=int(input("Enter a size:"))
        pattern1(num)
    elif choice=='2':
        num =int(input("Enter a size:"))
        pattern2(num)
    elif choice=='3':
        break
    else:
        print("invalid choice")
```

9. WAP to read roll number and marks of n students and create a dictionary from it having roll numbers as keys.

```
n = int(input("Enter the number of students: "))
student_data = {}
for _ in range(n):
    roll_no = input("\nEnter Roll Number: ")
    marks = float(input("Enter Marks: "))
    student_data[roll_no] = marks
print("\nStudent Records:")
print(student_data)
```

10. Write a python program that accepts a string and calculate the number of uppercase, lowercase, digits and special characters.

```
s = input("Enter a string: ")


counts = {
   "Uppercase": sum(c.isupper() for c in s),
   "Lowercase": sum(c.islower() for c in s),
   "Digits": sum(c.isdigit() for c in s),
   "Special Characters": sum(not c.isalnum() for c in s)
}


print("\nCharacter Counts:")
for k, v in counts.items():
   print(f"{k}: {v}")
```

11. Write a Python program that demonstrates the use of five different list methods. Your program should:

i. **Create a list and allow the user to add elements using the append() method.**
ii. **Insert an element at a specific position using the insert() method.**
iii. **Remove a specific element from the list using the remove() method.**
iv. **Sort the list in ascending order using the sort() method.**

Display the index of any element in the list using the index() method.

```
my_list = []


# Append elements
for _ in range(int(input("How many elements to add? "))):
   my_list.append(input("Enter element: "))
print("List:", my_list)
```

```
# Insert an element
my_list.insert(int(input("Insert at position: ")), input("Enter element: "))
print("After insertion:", my_list)

# Remove an element
elem = input("Enter element to remove: ")
if elem in my_list:
    my_list.remove(elem)
print("After removal:", my_list)

# Sort the list
my_list.sort()
print("Sorted list:", my_list)

# Find index of an element
elem = input("Enter element to find index: ")
print(f"Index of {elem}:", my_list.index(elem) if elem in my_list else "Not found")
```

12. Write a Python program that demonstrates the following:

i. **Create and check the shape of an array**
ii. **Convert a 1D array of 12 elements into a 3x4 matrix.**
iii. **Convert a 2D or 3D array into a 1D array**
iv. **Extract a subarray using slicing**
v. **Extract every alternate element from a given array**

```
import numpy as np
arr = np.arange(12)  # Create 1D array
print("Original Array:", arr)
print("Array Shape:", arr.shape)


matrix = arr.reshape(3, 4)  # Convert to 3x4 matrix

print("\n3x4 Matrix:\n", matrix)


print("\nFlattened Array:", matrix.flatten())  # Convert back to 1D


print("\nSubarray (first 2 rows, 3 cols):\n", matrix[:2, :3])  # Extract subarray


print("\nAlternate Elements:", arr[::2])  # Extract alternate elements
```

Write regular expressions to validate the following inputs:

i. **Email Address – Ensure it follows the standard email format (e.g., user@example.com).**
ii. **Date – Match a date in the format DD/MM/YYYY or MM-DD-YYYY.**

```
import re


def validate_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return bool(re.match(pattern, email))


def validate_date(date):
    pattern = r'^(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/\d{4}$|^(0[1-9]|1[0-2])-(0[1-9]|[12][0-
9]|3[01])-\d{4}$'
    return bool(re.match(pattern, date))
```

```
# Test cases
print(validate_email("user@example.com"))  #  True
print(validate_email("invalid@.com"))           # False
print(validate_date("25/12/2025"))                # True
print(validate_date("12-25-2025"))                # True
print(validate_date("31-04-2024"))                # Matches but doesn't check real months
```

Write regular expressions to validate the following inputs:

i. **URL – Validate a URL that starts with http:// or https:// and includes a domain name.**
ii. **Phone Number – Validate a phone number that may optionally contain two dashes (e.g., 123-456-7890 or 1234567890).**

```
import re


def is_valid_url(url):
   return bool(re.match(r'^https?://[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}(/.*)?$', url))


def is_valid_phone(phone):
   return bool(re.match(r'^\d{3}-?\d{3}-?\d{4}$', phone))


# Test Cases
print(is_valid_url("https://example.com"))   # True
print(is_valid_url("ftp://invalid.com"))       #  False


print(is_valid_phone("123-456-7890"))       #  True
print(is_valid_phone("1234567890"))         # True
print(is_valid_phone("123-45-6789"))        # False
```