Functions in Python

---

1. **Print a Greeting Message**
   📌 Create a function `greet()` that prints a welcome message.
   📝 *Explanation: Just print something like "Hello, welcome to Python!" inside the function.*

---

2. **Custom Greeting with Name**
   📌 Write a function `say_hello(name)` that prints `Hello, <name>!`
   📝 *Explanation: Take `name` as a parameter and print a personalized greeting.*

---

3. **Print Numbers from 1 to N**
   📌 Create a function `print_numbers(n)` that prints numbers from 1 to `n`.
   📝 *Explanation: Use a `for` loop inside the function to print all numbers from 1 to n.*

---

4. **Check and Print Even or Odd**
   📌 Create a function `check_even_odd(num)` that prints if the number is even or odd.
   📝 *Explanation: A number is even if divisible by 2, otherwise it's odd.*

---

5. **Print the Square of a Number**
   📌 Write a function `print_square(num)` to print the square of the number.
   📝 *Explanation: Square means `num × num`.*

---

6. **Print Sum of First N Natural Numbers**
   📌 Create `sum_natural(n)` that prints the sum of 1 + 2 + 3 + ... + n.
   📝 *Explanation: Add all natural numbers up to `n` using a loop.*

---

7. **Print Multiplication Table**
   📌 Define `print_table(num)` that prints the multiplication table of a number up to 10.
   📝 *Explanation: Like 5×1 = 5, 5×2 = 10, ..., 5×10 = 50.*

---

8. **Check and Print Sign of a Number**
   📌 Write `check_sign(num)` that prints if the number is positive, negative, or zero.
   📝 *Explanation: Use conditions like `if num > 0`, `num < 0`, and `num == 0`.*

---

9. **Sum of Digits**
   📌 Create `sum_digits(num)` that prints the sum of all digits of the number.
   📝 *Explanation: If number is 123, sum = 1 + 2 + 3 = 6.*

---

10. **Check Prime Number**
    📌 Write `check_prime(num)` that prints if a number is prime or not.
    📝 *Explanation: A number is **prime** if it's greater than 1 and has only two factors: 1 and itself. For example, 5 is prime but 6 is not.*
    .

---

11. **Check Voting Eligibility Using Function**
    📌 Create `check_voting(age)` to print if the person can vote.
    📝 *Explanation: In most countries, voting age is 18. Use `if age >= 18`.*

---

12. **Factorial Using Function**
    📌 Define `factorial(n)` that prints the factorial of a number.
    📝 *Explanation: Factorial of 5 is 5×4×3×2×1 = 120. Use a loop.*

---

13. **Palindrome Checker**
    📌 Create `check_palindrome(num)` to print if the number is a palindrome.
    📝 *Explanation: A number is **palindrome** if it reads same backward. Example: 121, 1331.*

---

14. **Fibonacci Series Generator**
    📌 Define `fibonacci(n)` that prints first `n` terms of the Fibonacci sequence.
    📝 *Explanation: Starts with 0, 1. Next term is sum of previous two. → 0, 1, 1, 2, 3, 5...*

---

15. **Check Armstrong Number**
    📌 Create `check_armstrong(num)` to check and print if it's an Armstrong number.
    📝 *Explanation:*
    An **Armstrong number** is a number whose sum of its digits raised to the number of digits equals the number.
    For example: 153 → $1^3 + 5^3 + 3^3 = 153$ ✅

---

16. **Print All Factors of a Number**
    📌 Create `print_factors(n)` that prints all numbers that divide `n`.
    📝 *Explanation: For 12, factors are 1, 2, 3, 4, 6, 12.*

---

17. **Even and Odd Digit Counter**
    📌 Define `count_even_odd_digits(num)` to count and print even and odd digits in the number.
    📝 *Explanation: Go through each digit using `% 10` and check if it's even or odd.*

---

18. **Check Perfect Number**
    📌 Write `check_perfect(num)` to print whether a number is perfect.
    📝 *Explanation:*
    A **perfect number** is a number where the sum of all its **proper divisors** equals the number.
    Example: 6 → 1 + 2 + 3 = 6 ✅

---

19. **Print All Primes in a Range**
    📌 Create `print_primes(start, end)` to print all prime numbers between two numbers.
    📝 *Explanation: Use nested loop — outer for the range, inner for checking prime.*

---

20. **Find and Print LCM**
    📌 Define `find_lcm(a, b)` that prints the Least Common Multiple of two numbers.
    📝 *Explanation: LCM is the smallest number divisible by both. Use a loop from max(a, b).*