به نام خدا

تمرین‌های سری 1

Practice 1: Define a **cell** containing the names of **5** students. Then, using a **loop**, check which names have more than **5 letters** and **print them**.

```
names = {'Ali', 'Fatemeh', 'Mohammad', 'Sara', 'Reza'};
```

Practice 2: Write a program that generates a **random number between 1 and 100**, then uses a **`while` loop** to try to guess the number. At each step, **display** the difference between the guessed number and the actual number.

Practice 3: Write a **function** called **even_avg** that takes a numeric array as input and returns the **average of the even numbers only**.

```
A = [3, 6, 8, 5, 10, 13, 2];
```

Practice 4: Write a **function** called **`is_prime(n)`** that checks whether the input **number `n` is a prime number or not**. Then, write a script that prints all prime numbers between **1 and 100**.

Practice 5: Suppose the grades of 6 students are given as `[17, 15, 19, 12, 14, 20]`, and their names are stored as `{'Ali','Sara','Reza','Mina','Hamed','Zahra'}`. Draw a **bar chart** and use a special color (e.g., gold) for the grade 20.

Practice 6: The **hyperbolic function** is defined as follows:

$$y(x) = a.\cosh\left(\frac{x}{a}\right)$$

where cosh is a hyperbolic function. For $a = 1$, plot the curve in the range [−5,5] and use a color of your choice for the plot.

Practice 7: Generate **random three-dimensional data** and create a **3D scatter plot**. Then, color the data points with different colors based on one of the components.

Practice 8: Write a **`while` loop** that sums **odd numbers** and stops when the sum **reaches or exceeds 1000**. Then, display the number of odd numbers that have been summed.

Practice 9: **The Fibonacci sequence is defined as follows:**

$$F(0) = 0,\ F(1) = 1,\ F(n) = F(n-1) + F(n-2);\ \text{for } n \geq 2$$

Calculate the Fibonacci sequence up to $n = 15$ using a `for` loop and display it.

Practice 10: **Merge Sort** is a **divide-and-conquer sorting algorithm** that first divides the array into smaller sub-arrays and then recursively sorts these sub-arrays. Finally, it merges them together to form the final sorted array.

Write a **function** that **sorts an array using the Merge Sort algorithm.**

```
A = [38, 27, 43, 3, 9, 82, 10];
```

Practice 11: The **Newton-Raphson method** is an iterative root-finding algorithm that uses the function and its derivative to converge to a root. Starting from an initial guess x_0, the method updates the estimate using the formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{\acute{f}(x_n)}$$

The iterations continue until a stopping condition is met, typically when the absolute value of the function becomes sufficiently small.

Solve the equation

$$f(x) = x^3 - 2x^2 + 4x - 8$$

using the Newton-Raphson method. Use an **initial approximation of 2** and a stopping condition of $f(x) < 10^{-6}$.

Practice 12: The **central difference method** is a numerical technique for approximating derivatives. It estimates the derivative of a function $f(x)$ at a point x using values of the function slightly before and after the point:

$$\acute{f}(x) \approx \frac{f(x+h) - f(x-h)}{2h},$$

This method is more accurate than forward or backward differences, with an error term proportional to h^2.

Use the central difference formula with $h = 0.01$ to approximate the derivative of $f(x) = \ln(x)$

at $x = 2$, and compare the result with the exact value $\acute{f}(2) = \frac{1}{2}$. Compute the relative error.

Practice 13: Design a scenario in **MATLAB** for each of the following **types of errors** and provide **a solution** for fixing it**:**

1. **Syntax Error**
2. **Runtime Error**
3. **Logical Error**

Practice 14: Review the following codes and identify and fix all the errors present

```
% محاسبه مجموع اعداد اول کمتر از یک عدد مشخص
n = input('Enter a positive integer: ');
sum = 1

for i = 1:n
    isPrime = true;
    for j = 2:i-1
        if mod(i,j) == 0
            isPrime = false;
    end

    if isPrime = true
        sum = sum + i;
    end
end

disp("The sum of prime numbers less than n is: ", sum)
```

```
% یافتن میانگین نمرات و بالاترین نمره
names = {'Ali', 'Sara', 'Reza', 'Mina', 'Hamed'};
grades = [17, 19, 14, 18];

total = 0;
maxGrade = 0;
for i = 1:length[names]
    total = total + grades(i);
    if grade(i) > maxGrade
        maxGrade = grade(i);
        topStudent = names(i);
```

```
    end
end

avg = total / length(grades;
disp("Average grade is: " + avg);
disp("Top student is: " + topStudent + " with grade " + maxGrade);
```

```
% حل یک معادله عددی با استفاده از روش نیوتن-رافسون
f = @(x) x^3 - 2*x^2 + 4*x - 8;
df = @(x) 3*x^2 - 2*x + 4;

x0 = 2;  % مقدار اولیه
tolerance = 1e-6;
maxIter = 50;
x = x0;

for i = 1:maxIter
    x = x - f(x) / df(x);
    if abs(f(x)) > tolerance
        break;
    end
end

disp('Root found: ');
disp(x);
```