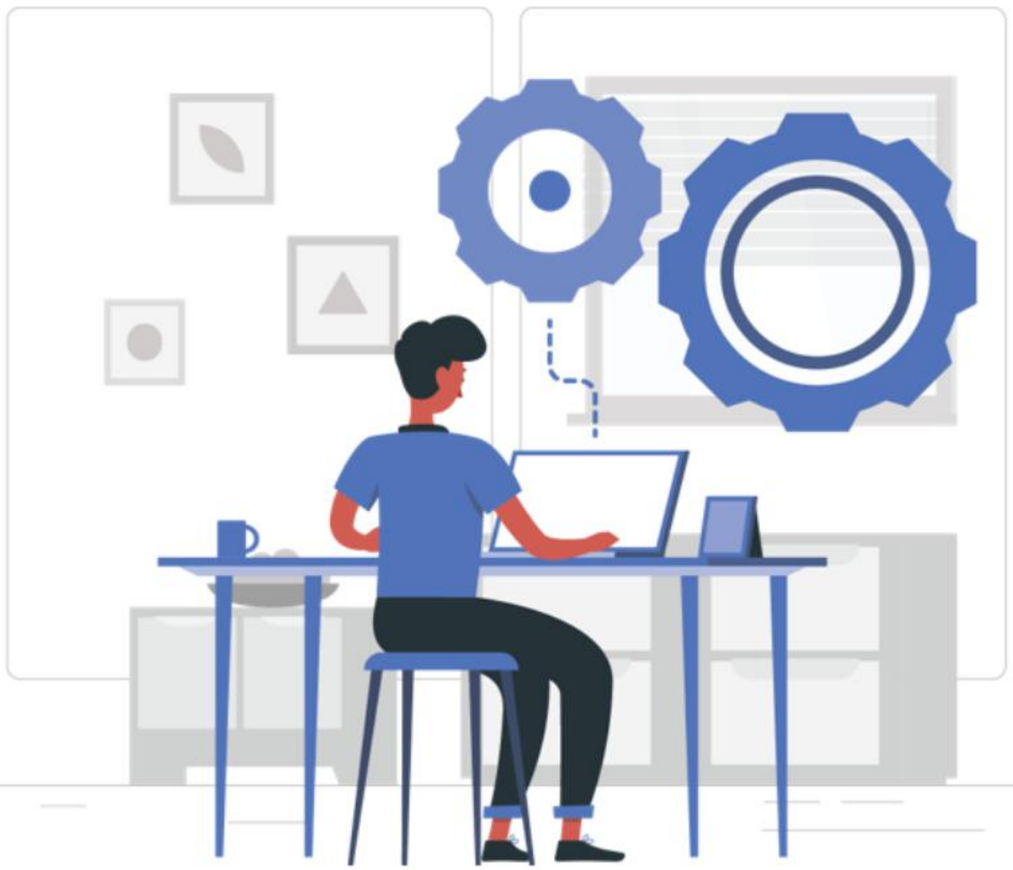
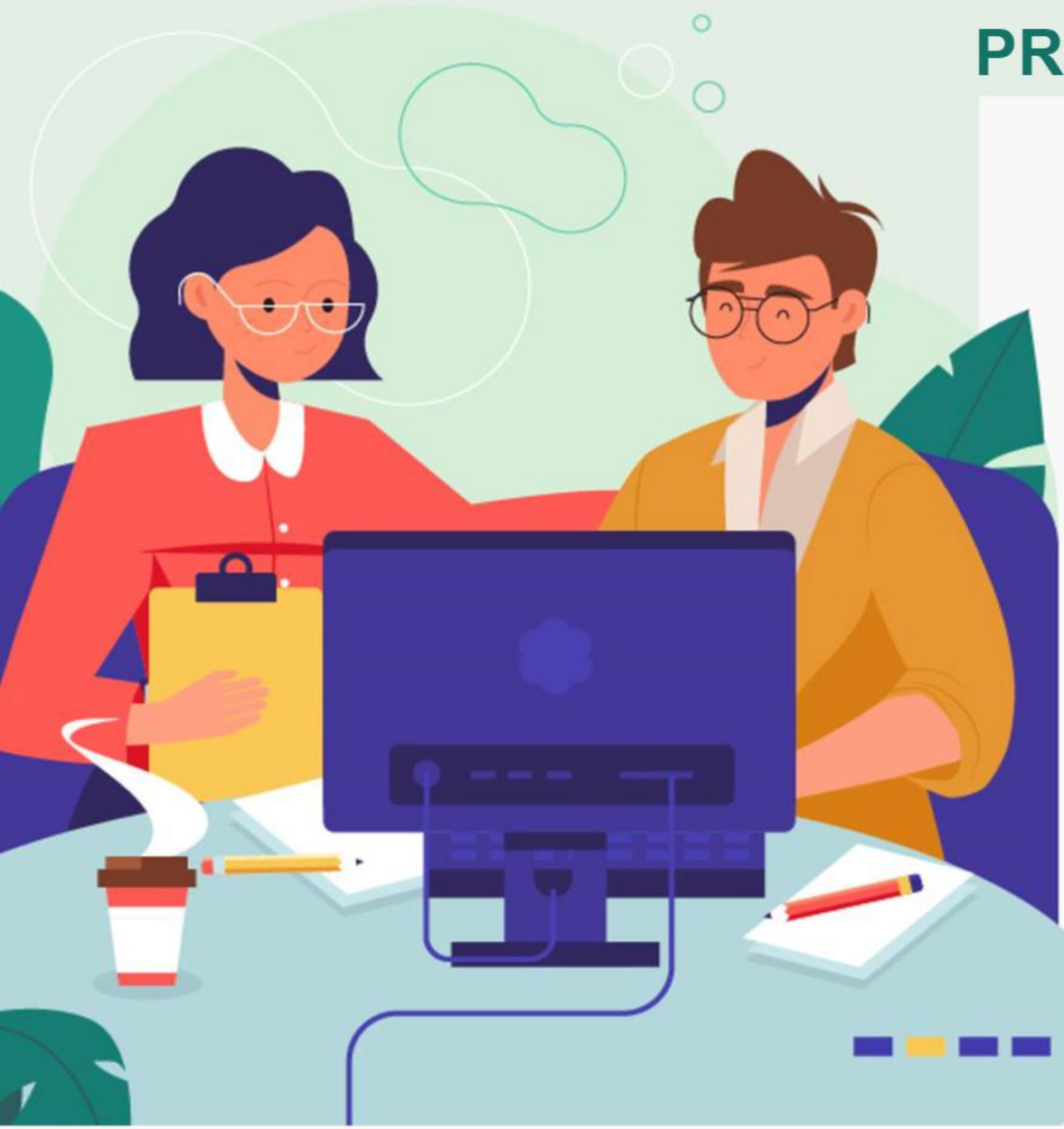


Practice
Implement Modular Programming Using Functions

Practices

- Practice 1: Calculate area
- Practice 2: Calculate weighted score



PRACTICE

Practice 1: Calculate Area

Ron has hens, ducks and cows in his barn. He wants to renovate his barn. He thinks that creating fences and keeping the animals separate will help him feed the animals more easily.

For this, he plans to divide the area into three parts. He wishes to have a square area for the hens, a circular area for the ducks, and a rectangular area for the cows.

Write a program to calculate the area of each fence using JavaScript functions.

Note: Tasks to complete this practice are given in the upcoming slide.

Tasks

- Open the file **p1-submission.js** located inside the **p1-calculate-area** folder to create the solution.
- Write the logic for the following functions:

Function	Purpose	Parameter	Formula	Parameter Type	Return Value
calculateAreaOfSquare()	Calculate the area of a square	side	side*side	Integer	area
calculateAreaOfCircle()	Calculate the area of a circle	radius	3.14*radius*radius	Floating point	area
calculateAreaOfRectangle()	Calculate the area of a rectangle	length and breadth	length*breadth	Floating point	area

- Call the above functions globally and display the output.

PRACTICE

Practice 2: Calculate Weighted Score

The placement team of an engineering college needs to select eligible candidates for campus placement. To do so, they require total weighted score of the students.

Write a program to calculate the total weighted score of a student using JavaScript function.

Note: Tasks to complete this practice are given in the upcoming slide.

Tasks

- Open the file **p2-submission.js** located inside **p2-weighted-score** folder to create the solution.
- Write a program that calculates and displays the total weighted score of a student.

Tasks (cont'd)

- Create the function `calculateTotalWeightedScore()` that takes 5 parameters (marks) of different assessments.
- Inside the function, perform the following steps:
  - Step-1: Initialize 5 variables for storing different weightage.
    - The weightages assigned to each assessment and the corresponding variable names to assign individual weightage are listed below:

Assessment Type	Weightage	Variables For Assigning Weightage
Assignments	10	assignmentWeight
Projects	35	projectWeight
Quiz	10	quizWeight
Mid Term Evaluation	15	midTermWeight
Final exams	30	finalExamWeight

Tasks (cont'd)

- Step-2: Calculate the weighted score of every assessment and store them in their respective variables. For example, calculate the weighted score for assignments using the formula given below.

```
assignmentWeightedScore = (assignmentWeight/100) * assignmentScore
```

- Step-3: Sum all the weighted scores to get the total weighted score of the student and store it in a variable.

```
overallWeightedScore = assignmentWeightedScore + projectWeightedScore +
quizWeightedScore + midTermWeightedScore +   finalExamWeightedScore
```

- Step–4: Display the individual weighted scores and the total weighted score.

Tasks (cont'd)

- Globally, perform the following steps:
  - Initialize the 5 variables given below in the table for assigning the marks.

Assessment Type	Marks (out of 100)	Variables For Assigning Marks
Assignments	97	assignmentScore
Projects	82	projectScore
Quiz	60	quizScore
Mid Term Evaluation	75	midTermScore
Final exams	80	finalExamScore

  - Call the `calculateTotalWeightedScore()` function by passing these variables as parameters.

Tasks (cont'd)

- The final output should display the weighted score in the format shown below.

```
Assessment Type  Weighted score
Assignments      9.7
Projects         28.7
Quizzes          6
Mid Term         11.25
Final Exam       24

The Total Weighted Score is : 79.65
```