1.Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.

Examination- Driven Advancement (TDD) Refine

Compose Examination Instances:

Prior to creating any kind of code, designers develop automated examination situations that specify the preferred habits of the software application.

Examinations are contacted match the performance that requires to be executed.

Run Tests (Red):

Perform the examination collection. At first, all examinations must fall short, shown by a "" red"" condition.

This failing validates that the capability is not yet executed, as anticipated.

Create Code (Green):.

Compose the minimum quantity of code essential to make the stopping working examinations pass.

The objective is to compose simply adequate code to please the demands specified by the examinations.

Refactor (Blue):.

When the examinations pass, improve the code to enhance its style, readability plus efficiency.

Refactoring guarantees that the code continues to be tidy as well as maintainable without altering its habits.

Advantages of Test-Driven Development (TDD).

Insect Reduction:.

By composing examinations prior to code programmers capture pests early in the growth procedure.

Automated examinations offer continual comments, protecting against regressions along with making sure that brand-new functions do not present unanticipated actions.

Boosted Software Reliability:.

TDD brings about even more reputable software program by motivating programmers to concentrate on creating little, modular, and also well-tested code.

The thorough screening procedure makes sure that each element of the software program acts as anticipated under various problems.

Boosted Confidence:.

TDD enhances designers' self-confidence in their code by giving a safeguard of automated examinations.

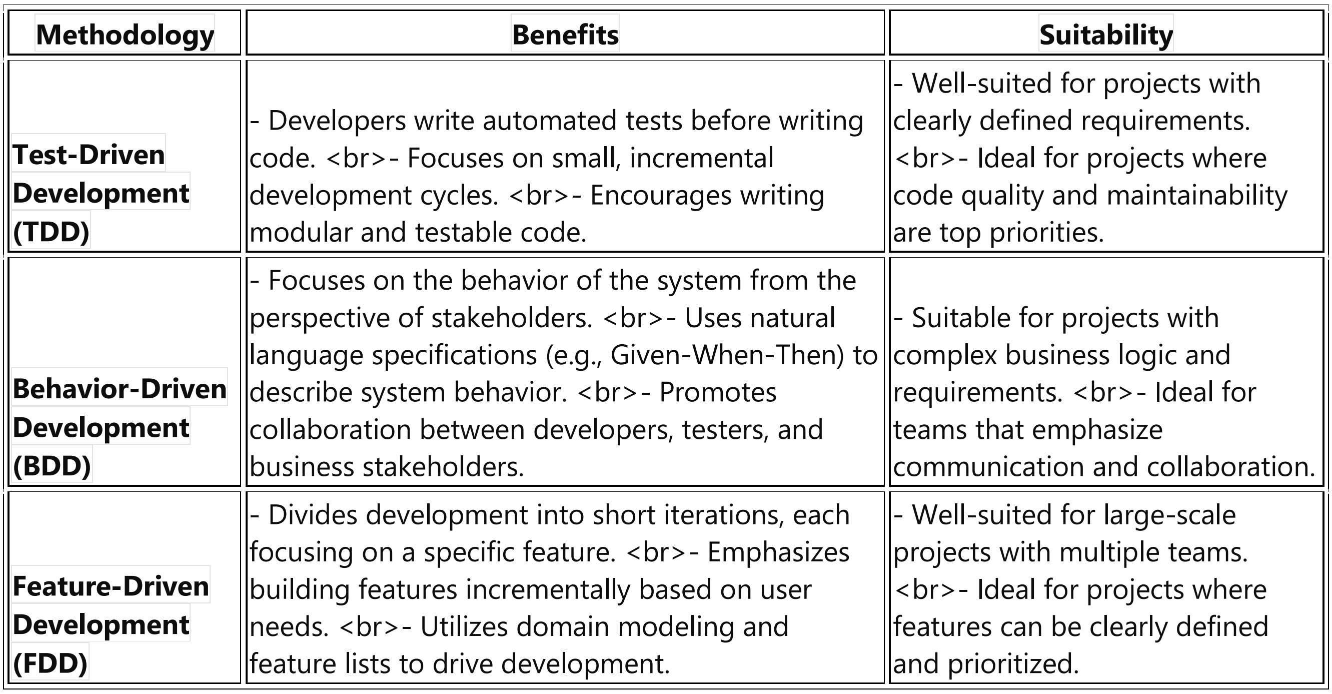
Constant assimilation plus automated screening pipes better confirm the integrity of the codebase.

Much Better Design plus Maintainability:.

TDD motivates much better software program style by advertising loosened combining, high communication plus adherence to develop concepts.

The step-by-step nature of TDD permits designers to improve code with self-confidence bringing about a lot more maintainable codebase gradually.".

2. Produce a comparative infographic of TDD, BDD, and FDD methodologies. Illustrate their unique approaches, benefits, and suitability for different software development contexts. Use visuals to enhance understanding.



Methodology	Benefits	Suitability
Test-Driven Development (TDD)	- Developers write automated tests before writing code. - Focuses on small, incremental development cycles. - Encourages writing modular and testable code.	- Well-suited for projects with clearly defined requirements. - Ideal for projects where code quality and maintainability are top priorities.
Behavior-Driven Development (BDD)	- Focuses on the behavior of the system from the perspective of stakeholders. - Uses natural language specifications (e.g., Given-When-Then) to describe system behavior. - Promotes collaboration between developers, testers, and business stakeholders.	- Suitable for projects with complex business logic and requirements. - Ideal for teams that emphasize communication and collaboration.
Feature-Driven Development (FDD)	- Divides development into short iterations, each focusing on a specific feature. - Emphasizes building features incrementally based on user needs. - Utilizes domain modeling and feature lists to drive development.	- Well-suited for large-scale projects with multiple teams. - Ideal for projects where features can be clearly defined and prioritized.

Key Differences:

Focus: TDD focuses on writing tests before code, BDD emphasizes behavior-driven specifications, and FDD concentrates on building features incrementally.

Collaboration: BDD encourages collaboration between developers, testers, and business stakeholders, while TDD and FDD primarily involve developers.

Scope: TDD and BDD focus on testing and behavior, respectively, while FDD concentrates on delivering features based on user needs.

Suitability: TDD is suitable for projects with well-defined requirements, BDD is suitable for projects with complex business logic, and FDD is suitable for large-scale projects with multiple teams.