**Assignment 1**

create an infographic illustrating the test-driven development (TDD) process, highlights step like writing tests before code. benefits such as bugs reductions and how it fosters a software reliability

**Test-Driven Development (TDD)**



1. **Write Test Cases**

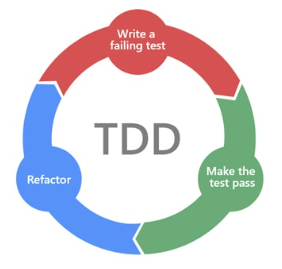
* Write tests before writing any code.
* Describe what you want the code to do.
* Tests act like a roadmap for building the software.

1. **Write the Minimum Code**

* Write the simplest code to make the test pass.
* Keep it straightforward and functional.
* Follow the "Red, Green, Refactor" method.

1. **Refactor Code**

* Improve code without changing its behavior.
* Make sure all tests still pass.
* Remove any unnecessary repetition and make the design cleaner.



* **Benefits of TDD**
* **Bug Reduction**: Find and fix bugs early.
* **Improved Software Reliability**: Keep the software working as expected by constantly checking it against the requirements.
* **Collaboration**
* Work together with other developers and stakeholders.
* Boost trust in the code you're building.
* **Time and Cost Savings**
* Spend less time fixing bugs.
* Save money by catching issues sooner rather than later.
* **Continuous Integration**
* Fit smoothly into continuous integration systems.
* Make sure your changes don't break anything already working.

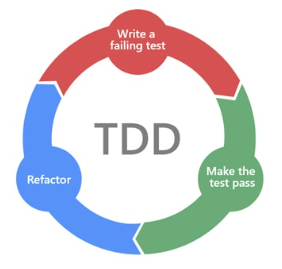
**Conclusion**

Test-Driven Development means better software, more confidence in your work, and quicker delivery of features.

**Assignment 2**

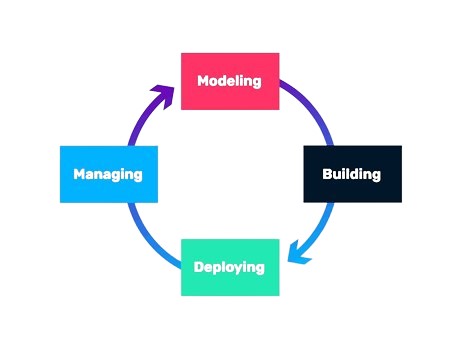
Produce a comparative infographic of TDD, BDD, and FDD methodologies. Illustrate their unique approaches, benefits, and suitability for different software development contexts. Use visuals to enhance understanding**.**

1. **Test-Driven Development (TDD)**



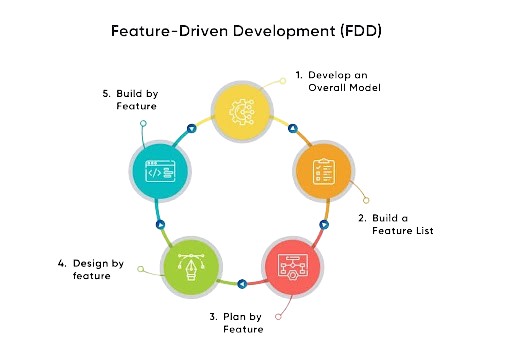
* **Approach**
* Write tests before writing code.
* Focus on what the code should do.
* Develop in small, manageable steps.
* **Benefits**
* Find bugs early.
* Easier to fix bugs.
* Write cleaner, more organized code.
* **Suitability**
* Good for smaller projects.
* Works well in flexible, fast-paced environments.
* Developers take the lead.

1. **Behavior-Driven Development (BDD)**



* **Approach**
* Focus on what the software should do for users.
* Use everyday language to describe behaviors.
* Everyone involved in the project collaborates.
* **Benefits**
* Better communication among team members.
* Clearer understanding of what needs to be done.
* Clients and stakeholders are actively involved.
* **Suitability**
* Great for complex projects with many moving parts.
* Ideal for projects with lots of stakeholders.
* Places emphasis on meeting client needs.

1. **Feature-Driven Development (FDD)**



* **Approach**
* Break the project into smaller features.
* Design and plan each feature carefully.
* Develop features one at a time.
* **Benefits**
* Easily track progress by feature.
* Efficient use of resources.
* Keeps development on track and organized.
* **Suitability**
* Best for large projects with lots of features.
* Projects that require detailed planning.
* Works well with diverse teams and skill sets.

**Conclusion**

Choose the methodology that matches your project's size, complexity, and team dynamics. Each approach offers unique advantages and can be adapted to fit different software development situations.