自分にとってのTDD(テスト駆動開発)

テスト駆動開発 (TDD) とは

1. **テストを書く**: まず、実装したい機能に対するテストケースを作成します。この時点では、まだその機能は実装されていません。
2. **テストを実行**: 作成したテストを実行します。この時点では、テストは失敗するはずです。
3. **コードを書く**: テストを通過するための最小限のコードを実装します。
4. **テストを再実行**: 実装したコードがテストを通過するか確認します。
5. **リファクタリング**: コードを整理し、必要に応じて改善します。この時点でもテストは通過する必要があります。
6. **繰り返す**: 新しい機能や修正が必要な場合は、再びテストを書き、同じサイクルを繰り返します。

類似の手法と比較して

- 開発者テスト --> テストを先に書くか後に書くかは決まっていない
- テストファースト --> テストを先に書くが誰が書くかは決まっていない
- 自動テスト --> テストを自動化する

TDDのメリット

- **品質の向上**: テストが先にあるため、バグの早期発見と修正が可能です。
- **設計の明確化**: テストを書くことで、機能の要件が明確になり、設計が洗練されます。
- **リファクタリングの容易さ**: テストがあることで、コードの変更や改善が安全に行えます。
- **ドキュメンテーション**: テストケースは、コードの使用方法や期待される動作を示すドキュメントとして機能します。

TDDとAI

人間のレビューが必要なため，人間がボトルネックになる --> テストがあると振る舞いが正しいことが保証される

- AIと並走してテストを作って，テストが通るように実装してもらう
- そもそもATにテストも書いてもらって実装もやってもらう

IT以外でのTDD

- とりあえず，やってみようが自分は多い
  - 目に見えた成果が出にくい
  - 賢者は歴史に学び，愚者は経験に学ぶ
- ゴールを決めてから小さく動くことを教えてくれた
- AIを使って情報を集めやすい時代なので，ゴールを決めるほうが成果が出しやすいかも