自動化について

『Pythonではじめるデータラングリングについて』より

タスクが明確でしっかり定義されており、結果が簡単に出せるものなら自動化は簡単
そうでなくても、タスクの一部だけでも自動化は可能
結果が明確で定期的に発生するタスクはどれも自動化すべき

自動化が適切でない場合

- タスクの発生頻度が稀で、非常に複雑な場合は自分で行ったほうが良い
- タスクの結果が成功しているかどうかを簡単に判断できないもの
- 適切な方法を判断するために人間が介在していなければならないもの
- 失敗が許されない場合

自動化が適切かどうかわからないときには、定期的に行っている何か小さなタスクを自動化してみて、それがどうなるか見てみるとよい。

自動化のためのステップ

自動化は次の事項をドキュメント化するところから始めるとやりやすい

- このタスクをいつ始めなければならないか
- このタスクには締め切り、投入時間の上限があるか。あるなら、いつまでに仕上げる必要があるか
- このタスクのために必要な入力は何か
- このタスクの成功、あるいは部分的な成功とは
- このタスクが失敗したら、何が起こるか
- このタスクが作り出す、または提供するものは何か。それは誰に対してか、またどのようにして行うのか
- タスクが終わった後何が起こるのか

これらのうち５つ以上に答えられれれば自動化の余地がある。そうでなければもっと調査を重ね、作業を明確化してから始めたほうが良い。

自動化のための基本的な手順

1. 問題を明らかにし、小さな仕事に分割する
2. 個々のサブタスクが入力として何を必要とし、何をする必要があり、完了とされるためには何が必要かを正確に記述する
3. どのようにすればそれらの入力が入手でき、タスクを実行しなければならないのはいつなのかをはっきりさせる
4. タスクのコーディングに取りかかり、実際のデータかサンプルデータを使ってテストする
5. タスクとスクリプトをクリーンアップし、ドキュメントを追加する
6. エラーのデバッグ、処理の成功に重点を置いてロギング機能を追加する
7. コードをリポジトリにサブミットし、手作業でテストする。必要に応じて変更を加える
8. 手作業を自動化された作業に置き換えてスクリプトを自動化に向けて準備する
9. タスクの自動化が始まったら、ログとアラートに注意し、エラーやバグを修正する。またテストとドキュメントをアップデートする
10. ログをどれくらいの頻度でチェックしてエラーを探すかについて長期的なプランを立てる