生成 AI と Python を活用した業務効率化の取り組み

2026/6/15 生産管理課　髙橋

生成 AI × Python を活用し、運賃計算業務を 20 分→3 分に大幅に効率化しました。
本レポートでは、生成 AI・Python の基本と、実際の業務改善事例、活用する際の実践ポイントをまとめています。

1. 生成 AI（LLM）とは

・ChatGPT などが代表的な LLM（大規模言語モデル）。自然言語で質問し、自然言語で回答を得られる。
・仕組みは文章をトークンに変換し大量データをパターン化して返答するもの。意味を「理解」しているわけではない。
・誤情報を正しいように出力する場合もあるため、最終判断は利用者が行う必要がある。
・現在は LLM 戦国時代。Claude がリードしているが状況は短期間で変わることもある。
・セキュリティ意識も重要：自動更新オフ・不要な拡張機能の削除がセキュリティトレンド。

2. Python とは

・基本ルールが少なく、英語が少しわかればコードが読みやすい、取り組みやすい言語。
・ライブラリ（拡張機能）が豊富で、複雑な処理も簡単に実装できる。

3. 実践例：運賃自動計算システム

従来 20 分程度かかっていた運賃計算を 3 分に短縮。以下の手順で構築：

1. 送り先（都道府県＋市）・荷物寸法・実重量を入力
2. Excel マスタ（距離対応表）を参照し距離 A に変換
3. 換算重量と実重量を比較し大きい方を重量 B として採用
4. 距離 A × 重量 B で運賃表から運賃を出力・メール文面も自動生成

※ 設計・例外対策・検証は人間が行うことが重要。(生成 AI はあくまで優秀な作業員)

4. 初学者向け 実践 3 ポイント

① 前提を言語化してから AI に指示する

ざっくりした指示では AI が勝手に前提を補い意図と違う出力になりがち。要件や条件を整理してから依頼する。

② 途中経過を print()で確認する

AI が生成するコードはエラー時に原因が分かりにくい。処理の区切りで print()を入れ途中経過を表示する。

③ 最終判断は必ず人間が行う

AI に下書き・提案まで作らせ、確認・実行は人間が行う設計にする。自動返信などは大事故のリスクあり。

▶ まとめ

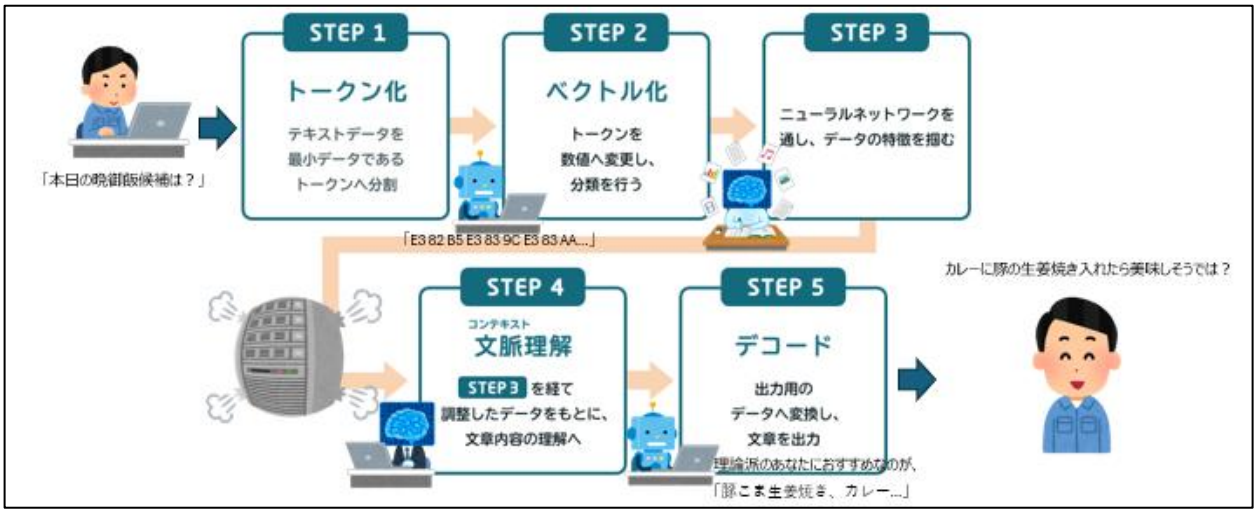
生成 AI と Python の組み合わせで業務処理を効率化できることもある。重要なのは「正確な指示」「途中経過の検証」「最終判断は人間が行う」の 3 点。業務内容を深く理解したうえで、認識合わせしながら一歩ずつ進めることが成功の近道。

最近業務上で生成 AI を利用して Python で色々なことをするようになったので
その経験をベースに生成 AI と Python について書いてみます。
長文になったのでお好みの生成 AI にそのまま要約してもらっても大丈夫な情報だけにしました。
==========
生成 AI について🤖
==========
歌詞を書いたら曲を提供してくれたり、物語を書いたら動画を作ってくれたりと色々な生成 AI がありますが、
おそらく一番慣れ親しんでいる人間の言葉(自然言語)で聞いたら人間の言葉で結果を返してくれる生成 AI を LLM といいます。(*Large Language Model=データ・パターン・計算量が大規模な自然言語モデル)

LLM は Claude や ChatGPT や Gemini や Grok や Copilot あたりがおすすめですが、
自分は会社業務で使用しているので会社指定の *exaBase Claude Haiku Economy(JP)*を使っています。
今はちょうど LLM 戦国時代が到来しており Claude が一歩リードしていますが、他の AI もすぐに学習して進化するので、1 週間後には別の AI が一番になっているかもしれません。
ちなみに最近は Mythos が話題になっている Claude ですが Claude Haiku と Mythos/Fable はナンバリングではなくグレードの違いで、iPhone16 と 17 ではなく、SE と Pro のような違いです。

ただ利用者側は自分含めて学びたてのヒヨコ🐣が多いですが、悪いハッカー側は鷹🦅が多いので、
利用する際は一定のセキュリティ意識も持ってないと最悪社内のシステムが止まりかねません。
最近は自動更新のオフと利用していない拡張機能を削除しておくことがセキュリティトレンドみたいです。
(今後はサイバーセキュリティ関係の AI が伸びる。予感・・・。←既にコンサルが流行っているみたいです＾＾;)

また、生成 AI は日本文化に馴染んで太鼓持ちがうまい部下みたいに振舞うことが多いですが、
実際に生成 AI が意味を理解したり巨大な脳みそがカリフォルニアの地下施設に液漬けにされて思考したりしているわけではなくて、その仕組みは文章を解体してトークンに変換して、大量のデータを圧縮してそれぞれの生成 AI の理念に基づいてパターン化してレビューしているだけなので、利用者側はその大量のデータから必要な情報をすぐに自然言語で受け取れる恩恵を受けている形みたいです。
(もちろん誤ったことを正しいことのように出力することもあるので結局の判断は利用者がしないといけません。)



NEC のコラムより引用(https://www.nec-solutioninnovators.co.jp/sp/contents/column/20240229_llm.html)

ちなみに全世界の人々が問い合わせた情報を処理して瞬時に返しているので、
大量の GPU と電力の消費にフリーライドさせてもらっており、急に有料になっても文句は言えません。

==========
Python について
==========
Python は簡単な基本ルールを覚えて、パソコン操作ができて多少英語を知っていれば使えるので、
他のプログラミングに比べればかなり触りやすいです。
実際のコードも重要な構文が何個かあるので覚えないといけないですが、
日本人がエセ中国語を読むのに似た感覚で、どこで何をしているかは比較的分かりやすいですね。

Python の最大のメリットは iPhone や Android のようにユーザーが多いので拡張機能が多いことです。
Python 本体だけだと、カレーを作るのに色々な香辛料を買い集めて、火を起こして…の作業になりますが、
Python にライブラリ(あとから拡張できるアプリみたいなもの)というものを導入すると、
あとは好きな具を入れて火を入れたら完成の状態まで作れます。
その点、生成 AI はすぐに結果を出してくれるのでファミレスに行ってカレーを注文するようなものですね。

また Python のプログラミングコードを編集する用のソフトも多数あり、
ソフト次第でコードを見やすくしたり、間違っていたら指摘してくれたりなどの便利機能がついてきます。
生成 AI もそうですけど、人間がやりがちなスペルミスや文法ミスなどのしょうもないミス避けの機能が多くあり、
余計な神経をすり減らさずに考えることに集中できてとても好感触です。
(特に要領掴んでない初学者は余計に考えることが増えて大変ですからね。)
ソフトによっては一太郎スマイルと Word くらい差があります。

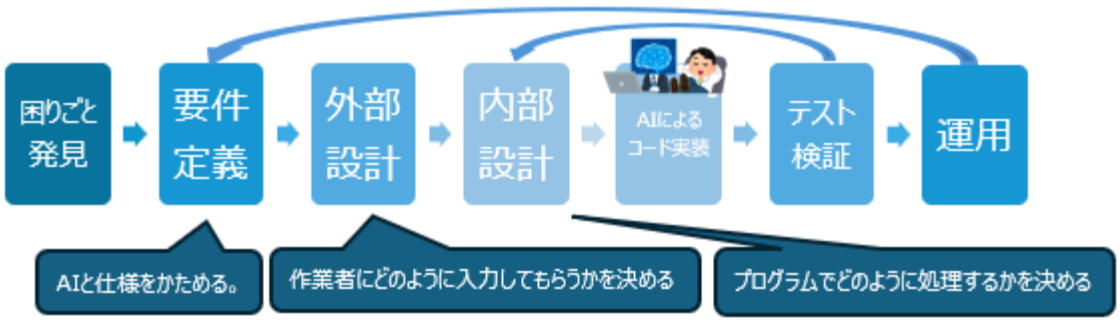
ちなみに自分は CSV のデータをすぐに書き換えるのにメモ帳アプリを使用していたため、
Python のコードを編集するために最初は Windows アプリのメモ帳を使用していましたが、
行数表示がないのと白背景で目が痛くなるので、
Python 編集専用の VisualStudioCode というマイクロソフトのソフトを総務から許可を得て使っています。
(※現行のメモ帳アプリは右上の歯車マークから書式を変更できます。)
*ちなみに e-pack もタイトルバー右クリックでフォントの変更が可能です。Meiryo UI がおすすめです。

複雑なシステムはモジュールファイルを作成して…データフレームを作成して…クラスを…と、ややこしいことが多いですが、初学者は簡単なことからできることを増やしてバルクアップしたほうが近道ですので、今回は簡単な例を用意してどのような手順で作成するかを紹介します。

基本的な Python の要領は色々な人が YouTube やブログなどで解説してくれているので、
自分と似たような価値観の人が出しているものを見て学習するのが良いですね。
自分は 2 時間の動画を 2 倍速でみて分からなかったことは生成 AI に聞いてある程度の要領を掴みました。

==========
生成 AI を利用して Python でシステムを構築する
==========
構想と外部設計までは自分である程度作って AI に校正してもらい、
最終的に構想を AI に伝えてコードに変換してもらい試行錯誤しています。

図に表すと以下の通りです。



生成 AI が何故そんなにすごいのかといいますと、
これまで多くの時間が費やされてきた「繊細なコーディング作業」「エラー発生⇄コード修正の連続」を、
人間は生成 AI で指示+検証+理解するだけでよくなったので時短+質も向上したことです。
例えるなら建設で今まで頑張って古代エジプトのように手作業で作っていたものが、
急に重機や計測器が入ってきて都市建設がとてつもなく進んでいるイメージです。

さらに Claude が頭一つ抜けているのは、場当たり的な回答ではなく、
コードの生成+テストコードの作成やレビュー+問題点の指摘も行うので、
さらに時短+質も上がっているというところですね。
(Claude も新グレード発表で旧グレードの性能は怪しくなっていたので何事も盲信しないことが大事ですね。)

ちなみに入力作業やデータの処理については生成 AI や Python を利用するよりも、
パワークエリとか Excel の備え付けの機能のほうがシンプル早いみたいなことが多いですが、
我々も知らないことが多いと思いますので、「●●という作業を自動化したい」、「出力したいけど何かいい方法はない？」とまずはやりたいことを整理しながらどうやったら実現可能か聞いてみましょう。

また設計や検証の段階で出来る限りのミス避けと例外対策も考えておきましょう。
あらゆる不安を取り除いておくことで今後余計なことを考えなくて済む=作業効率も上がって、
より考えることに集中することが出来ます。(「運用は性善説、設計は性悪説」という標語もありますね。)

＃ ＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝
＃ 実践(今の生成 AI に聞いたらもっと良い返答してくれるかもしれません。)
＃ ＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

物品の運送についての運賃を聞かれることがよくありますが、
大手運送会社でも計算方法がそれぞれ異なるのと、
才数や容積換算重量など計算が面倒な要素があるので、
場所や荷物によってどこを利用すれば良いかは分かっていても、
運賃を求めるとなると計算や調査で 20 分ほどかかります。

概算でもすぐに運賃の情報が欲しいことが多いので、
計算結果をすぐに出したいものです。

そこで計算+調査の工程を書き出してみると…
(ⅰ)荷物の寸法重量を求めて実重量と比較して重い方を採用。
(ⅱ)送り先の市までの下道距離を測る。
(ⅲ)重量×距離の運賃表を参照して運賃を求める。


```python
from geopy.geocoders import Nominatim
from geopy.distance import geodesic
import pandas as pd

# ============================================
# 設定
# ============================================

KAKIHARA_ADDRESS = "広島県福山市〇〇〇〇"

# 運賃表
fare_df = pd.read_excel("運賃表.xlsx")

# ============================================
# 住所 → 座標
# ============================================

def get_lat_lon(address):
```

※運賃システムはお金が絡むので企業秘密は全部伏せています

上記の工程を AI に説明して実装してもらったところ、
① 送り先の住所と荷物の寸法と実重量を入力してもらう
①´ 送り先の住所の不一致や荷物が運送不適合の場合エラーで返す。
② 送り先の住所を座標に変換するライブラリを使用して柿原工業との座標の直線距離を算出します。
②´ 運賃表を参照して(算出した距離)≦(基準距離)となる最小値の距離Ａに変換します。
③ 荷物の寸法から換算重量を求めて実重量と比較し重い方を採用する。
③´ 運賃表を参照して(算出した重量)≦(基準重量)となる最小値の重量Ｂに変換します。
④ 距離Ａと重量Ｂで運賃表を参照して運賃を出力します。

上記の流れで右上図のようにそれっぽいのが完成しましたが、
②の直線距離だと福山市-笠岡市間などの近場はいいですが、
福山市-宮崎市となると実際の陸路よりも少ない距離が表示されてしまいますね。

そのため AI に実際に Google マップで有料道路を使わずに距離を出すことはできない？
ということで、実装するコードを提案してもらいましたが、
Google マップを呼び出すのは有料、代替手段の OSRM は無料だけど有料道路を使用してしまう。
とのことで、別の手段を考えないといけなくなりました。

そこで通勤中に思いつきましたが、
市一覧のリストは官公庁が作っているはずなのでそれを Excel マスタとして落とし込んで、
Excel マスタに各都市との距離を入力しておけば、毎回 Google マップで検索しなくても済むので、
②都道府県+市を入力してもらって距離対応表(運賃表)の Excel を参照して距離 A に変換する。
ということにしました。(距離は空いた時間に埋めていけばいいので完了までに 1 週間かかりました)

上記の工程を AI に説明して再実装してもらったところ、
① 送り先の住所(都道府県+市)と荷物の寸法と実重量を入力してもらう
①´ 送り先の住所の不一致や荷物が運送不適合の場合エラーで返す。
② 距離対応表(運賃表)の Excel を参照して距離 A に変換する。
③ 荷物の寸法から換算重量を求めて実重量と比較し重い方を採用する。
③´ 運賃表を参照して(算出した重量)≦(基準重量)となる最小値の重量 B に変換します。
④ 距離 A と重量 B で運賃表を参照して運賃を出力します。

ここまで作ればやりたいことは 9 割出来ました。

運送会社によって得意不得意な荷姿があって運賃表のコスパが変わるので、
ケース数によって積み方を計算してどれが効率いいのかも確認する必要があります。
そのため、ケース数も入力する項目を設けて、
１パレットあたりの最小～最大積載量を計算してそれぞれの運賃を Excel で出せるようにして、
さらに運送会社に対しても見積確認ですぐ問い合わせできるように、
出力された文章をコピペしてメールすれば OK のフォーマットも作っておきました。

あくまで概算見積の為、
実寸が違ったり、特殊な地域であったり、現地の道路状況を確認したらトラックが進入できなかったり、
送り先がブラックリストに登録されていたり、フォークリフトがなかったりと色々な条件があるので、
業者に問い合わせたり、その関係性から無理を相談したりしないといけない部分もありますが、
どんな料金感になるのかは話を聞きながらでも近い値が出せますので 20 分作業が 3 分に時短となりました。

＃ ＝＝＝＝＝＝＝
＃ まとめ 📝
＃ ＝＝＝＝＝＝＝
他にもメインの業務では客先からのそれぞれ仕様が異なる注文書や内示を、共通のデータとして柿原のシステムに落とし込むために変換するプログラムを作ったり、金型の効率的な配置リストを作ったり、
各社ごとに登録されたカレンダーを抜き出して注文数量を再配置といったことを行っております。

自分がやっている作業を最初から組み立てるのは進めやすいですが、
システムのつながりや作業内容をよく理解していないと作業を進めることは難しいので、
結局は人と相談して実際に現場をみて軌道修正しながら進めていくのが大事ですね。

以上、ちょっと長くなりましたがほぼ生成 AI に書いてもらいました。
RPA にしても VBA にしても Excel にしても結局は非技術的な部分も大事なのと、
最初は皆よくわからんという状態から始めていますので謙虚に一歩ずつ頑張っていきましょう。

あとは生成 AI で Python を触ってみたい人向けに学習段階で知っておけばよかったことを書いておきます。

① 破綻しないような前提を作る・方法を選ぶ。
これは生成 AI でプログラムしようとした人の 80%ぐらいは最初に引っかかると思いますが、
ざっくりと「運賃自動計算システムを作って」という指示だけでも、生成 AI は場当たり的に前提を勝手に黙って補って、それらしいプログラムを作ってしまうので、利用者の頭の中にある前提とは異なるプログラムが完成してしまいます。その結果として「思った動きをしない」「例外が考慮されてない」「一部データでエラーを出してしまう」「後から修正が増えてしまって結果的に時間がかかり、作り直すことになる」「中途半端な処理だけで終わってしまう」「エラーさえ出てこずに終了する」といった状況に陥りがちです。
生成 AI は優秀なのは間違いないですが、良い結果を得るためには指示者自身が前提条件を言語化して整理してから指示することが近道ですね。

別件ですが、web からそのまま情報を引っ張ってくる(=スクレイピング)のも破綻しやすく、
SNS の釣果情報を時短で確認するために要約してくれるシステムを作ろうとしましたが、たまにアップデートや気まぐれで体裁が変わるのでめちゃくちゃ苦労しています(RSS+API なら破綻しにくいと言われていますが…)

結局は「前提で破綻していないか確認する」「すぐに動くものよりも一年後も動いているか」が大事ですね。

② 扱っているデータの途中経過が正しいか確認する
生成 AI の作るシステムは動かすと「処理します…→処理完了！」みたいなきれいなコードを書きがちですが、これだとエラーになってしまった時に「処理します…→どこかでエラーです！」みたいなことになりがちでそのままだと保守が難しい状態になってしまいます。

Python では「print(▲▲)」で▲▲に文字情報や変数を入れて、処理の途中でその変数が何の情報を持っているか確認することが出来ますが、現在の生成 AI の書くコードは完成系を追求するあまり途中経過の確認処理がほとんど入っていないことがあるため、動かしてエラーが出てしまってもどこで間違えているか分からないことが多く、サイゼリヤの間違い探しの最後の 1 個くらい難易度が高いこともあります。
初心者の頃は「生成 AI が全部作ってしまったからどこを見ればいいかわからない。」「それらしいところを直しても全然うまくいかない」という壁にぶつかって、時間とやる気を失いがちです。

print で途中経過を処理の区切りで出すように生成 AI にコードを作ってもらって、エラーが出た時に生成 AI にエラー文を丸投げすると、すぐにエラーを見つけて解決策まで提案してくれるので素晴らしいです。

③ 人が最終的に判断するように導く
生成 AI を使うときに一番言われていますが、生成 AI にアイディア出しや作業をやってもらうのはいいけど、判断は最終的には人が行うようにするということです。身近な例だとみんなの人生相談機になっている ChatGPT でも相談して回答通りにしたら大変なことになってしまったということがあったと思います。

業務例で挙げるとメールがきたら自動返信だといずれ大事故になりかねないですね。
そのため、AI に下書きまで作ってもらってあとは人が確認して送信できるようにしておけばいいということです。

何が言いたいかといいますと、
エラーが出た時に丸投げするのはいいですが、何が原因で何を修正したかを認知していないと途中で何をしていたかわからなくなって破綻してしまうので、謙虚に逐一内容を確認したほうが良いということです。

▶まとめ
かなり長くなって大変申し訳ないですが、社外に漏れてもいいようにそのままお好みの生成 AI に投げて要約しても大丈夫な文章にしておきました。ピンと来なかったところは「○○○○←例えばどういうこと？」とかで生成 AI に聞いてみてください ^ ^ ;
この内容も半年後には少し違う内容になっている可能性があるので恐ろしいですね😥