

卒業論文

ChatGPTによるタイピング練習用ソフト iotype の 開発

関西学院大学理工学部

情報科学科 西谷研究室

27020712 五百尾真翔

2024年2月

概要

開発ターゲットとしてはタイピング練習ソフトを選んだ。西谷が担当するコンピュータ演習では、Touch typing の修得を目的とした「Shunkuntype」というソフトウェアが使用されている。しかし、1年時には Ruby に触れる機会は限られており、環境構築やプログラムについて理解しながら使用することは容易ではない。

本研究では生成 AI である ChatGPT を使用しソフトウェア開発を行うこととする。今まではネット検索が主流であったが、ChatGPT といった生成 AI がその座を占めるようになった。プログラミング開発から見たときに、検索と生成とでは便利さが格段に違う。ただ、ChatGPT を使用したソフトウェア開発には問題点もあり、これらを実際に確かめるべく ChatGPT を使用し開発した。

最終的な目標は、Touch typing を修得するためのタイピング練習ソフト「iotype」を西谷が担当するコンピュータ演習内で導入することである。iotype は、Touch typing の習得をサポートする 3 つのトレーニングメソッドや、トレーニング過程を可視化するグラフ表示機能を提供する。iotype の開発は、先行研究である「Shunkuntype」を参照しながら進め、実装には Python 3 を用いた。開発過程における、ChatGPT の対話内容を第 3 章に記述している。

目次

第1章 序論	4
1.1 研究の背景	4
第2章 方法	6
2.1 ソフトウェアのプラットフォームと利用環境	6
2.2 既存システムの参照 shukuntype	6
2.3 実装言語とライブラリの選定	6
第3章 ChatGPT との対話内容	7
3.1 タイピングソフトの基本機能	7
3.1.1 ChatGPT との開発過程	7
3.1.2 コード解説	10
3.2 keytype(段別タイピングトレーニング)の開発	12
3.3 training の開発	15
3.4 プロジェクトのディレクトリ構造を整理	16
第4章 iotype の仕様	20
4.1 keytype 段別タイピングトレーニング	20
4.2 training	20
4.3 test 段別タイピングテスト	21
4.4 タイピングトレーニング結果のグラフ表示機能	21
第5章 今後の課題	23
第6章 まとめ	24

目次

3.1	最初に提案された GUI タイピングアプリケーション.	7
3.2	タイピングソフト CUI 版.	8
3.3	ファイルを用いたタイピング練習.	8
3.4	カウント機能を追加.	9
3.5	ストップウォッチ失敗.	9
3.6	sentence.	10
3.7	counter.	11
3.8	stopwatch.	11
3.9	iotype の実行例.	12
3.10	ランダムにセンテンスを表示.	13
3.11	1文字ずつ表示.	13
3.12	段別トレーニング.	14
3.13	keytype の実行例.	14
3.14	ChatGPT からの code 提供.	15
3.15	ファイルからの読み込み.	15
3.16	training 失敗例.	16
3.17	training の実行例.	16
3.18	ディレクトリ構造の提案.	17
3.19	iotype の tree の現状ディレクトリー構造.	17
3.20	iotype tree.	19
4.1	key トレーニングの実行例	20
4.2	training の実行例	21
4.3	test の実行例	21

4.4 トレーニング結果のグラフ表示	22
------------------------------	----

第1章 序論

1.1 研究の背景

プログラミングのスキルを向上させていくときに、今まではネット検索が主流であった。ところが2022/11/30にOpenAIによって公開されて以来、ChatGPTがその座を占めるようになった[1]。プログラミング開発から見たときに、検索と生成とでは便利さが格段に違う。従来の検索は、実装言語やアルゴリズムがマッチする参考文献を自身で探し出す必要があるため見つからないことも多く、一定の知識がなければ検討することもできない。また、実際の環境に合わせてコード開発しなければならないため、応用するのが困難であり、エラーメッセージの解読も容易ではない。よって、プログラミング開発において多くの時間を要する。一方で生成AIは、多様なプログラミング言語に対応しており、検索することで解決策まで提供され、環境に合わせたコード開発までを行えるため非常に効果的である。また、提供されたコードを理解できないときには、解説やコメントアウトにて詳細を確認することや、どのような手順で検索・生成をしたか記録されるため見返すことも可能である。そして、初心者が苦戦するエラーメッセージの解読までも生成AIによって解決することができるため、生成AIを使用するユーザーが増加したと考えられる。

ただし、学習を繰り返すことで回答が変わってくることや、意図とは違う結果を返すこと、プログラムコードの妥当性がわからないといった問題もある。これらについて、実際に確かめるべくタイピング練習用ソフトの開発を本研究で行った。

開発ターゲットとするタイピングは、コンピュータスキルを習得する一つのブームとなっている。例えば、丸善京都店のコンピュータコーナーの店頭ディスプレイには、ChatGPT[3]と並んで、Typingのテキストが並べられている。また、西谷が担当しているコンピュータ演習では、Touch typingの修得を目的としたShunkuntype[2]というRubyで開発されたCUIソフトウェアが使われている。使用するユーザーは、情報工学課程所属の1年生である。しかし、1年時にはRubyに触れる機会が少ないため環境構築やプログラムについて

理解しながら使用することは容易ではない。

よって、本研究では西谷の講義で1年生が Touch typing を習得するためのタイピング練習ソフトを Python を用いて開発する。最終的な目標は、本学生徒の効果的なプログラミングスキルの修得に寄与することである。さらに、タイピングソフトのアプリケーション作成を通じて、ChatGPT を使用した開発が、どのような文脈や条件下で有益であり、改善が必要であるかを明確にし、新技術の適用範囲をより広く理解することを目指した。

第2章 方法

2.1 ソフトウェアのプラットフォームと利用環境

本研究で開発するタイピングソフトウェア「iotype」は、PC上のターミナル環境での利用を想定している。これは、プログラミングを習得しはじめた初心者が、shell やコマンドなどの CUI に慣れることを目的としている。

2.2 既存システムの参照 shukuntype

「iotype」の開発においては、先行研究である「shunkuntype」を参照している。shunkuntype は同様にタイピングソフトウェアであり、その特徴や実装を参考にし、iotype の開発に活かしている。

2.3 実装言語とライブラリの選定

iotype の実装は Python 3 を用いた。クロスプラットフォーム対応が強力であり、様々な環境で実行可能なことや情報工学課程のコンピュータ実習にて導入することを想定し選択した。

タイピングの時間制限を実装するために、Python の標準ライブラリである threading を利用した。threading は軽量なスレッドを扱うことができ、非同期処理を実現するために適している。

ソフトウェアのライブラリ管理には pip を使用する。pip は Python パッケージのインストール、アップデート、削除などを効率的に行うことができる。これにより、iotype の依存関係の管理が容易に行え、ユーザーが簡単かつ迅速にソフトウェアを導入・更新・削除できる環境を提供する。

第3章 ChatGPT との対話内容

本章では,ChatGPT との対話となる「入力」と「出力」を時系列に従って示している. 実際の出力ではコードや解説が付いて冗長となるため,要約した内容を示している. 全ての記録は Github 上の docs 内の file に保存している. また,「出力」から次の「入力」までの間に考えた内容を記述している.

3.1 タイピングソフトの基本機能

3.1.1 ChatGPT との開発過程

最初に, タイピングソフトの基本機能の実装を試みた.

入力 これからタイピング練習用ソフトを Python にて開発したい

出力 Tkinter を使用したシンプルなウィンドウで, ランダムな文章が表示され, ユーザーが入力すると正確性が表示される GUI アプリケーションが提案された.

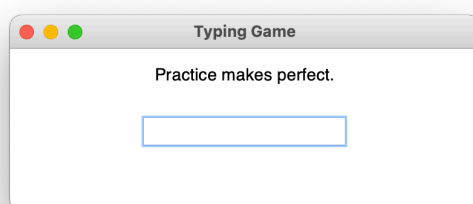


図 3.1: 最初に提案された GUI タイピングアプリケーション.

与えた情報が少なく, CUI で開発をしていきたいが図 3.1 のように, GUI で開発されてしまうため, CUI という仕様を明示する.

入力 CUIで開発したい

出力 図 3.2 のように, CUI アプリケーションが提案された.

```
iioomanato@iioomanato ~/r/g/s/m/C/base (main)> python3 b2.py
Welcome to Typing Game!
Press Enter to start...

Type the following text:

Practice makes perfect.

Your input: abcdefg

Accuracy: 0.00% | Time: 3.84 seconds
```

図 3.2: タイピングソフト CUI 版.

現段階では, タイピングすべき単語がないためタイピング練習用ソフトとして機能していない. そこで, ファイルから読み込みタイピングを行うという仕様を明示する.

入力 ファイルから読み込みタイピングを行うように変更.

出力 テキストファイル (例: text.txt) を用意して, その中の文章を読み込んでゲームが始まる.

```
iioomanato@iioomanato ~/r/g/s/m/C/base (main)> python3 b3_1.py text.txt
Welcome to Typing Game!
Press Enter to start...

Type the following text:

a b c d e

Your input: a b c d e

Accuracy: 90.00% | Time: 6.23 seconds
```

図 3.3: ファイルを用いたタイピング練習.

ChatGPT に従い, text.txt ファイルを作成し実行する. 図 3.3 のように, ファイルから読み込み表示させることが実現された. 現段階では, 正答率とタイピングスピードが結果として返されている. shukuntype を参照し, 正答数を表示するように変更したい.

入力 単語のマッチをとる counter メソッドを追加

出力 単語のマッチを計算する counter メソッドを追加したバージョンのサンプルスクリプトを以下に示します.

```

[ioomanato@ioomanato ~/r/g/s/m/C/base (main) [1]> python3 b4.py text.txt
Welcome to Typing Game!
a b c d e
Type return-key to start.

Type the following text:

a b c d e

Your input: a b c d e

Accuracy: 90.00% | Time: 5.63 seconds | Word Matches: 5

```

図 3.4: カウント機能を追加.

図 3.4 が正答数を表示するように変更した実行例である.

Enter キーを押すとプログラムが終了してしまう. そこで, 制限時間になると強制終了するストップウォッチの機能を追加した.

入力 ターミナル上から制限時間を受け取り制限時間になると強制終了させたい

出力 制限時間を設定して, その時間内にユーザーの入力がない場合に強制終了させるには, time モジュールを使用して経過時間を監視し, 制限時間を超えた場合にプログラムを終了させるように調整できます.

提供されたソースコードを使用した, 制限時間になっても終了しない.

図 3.5 が, 制限時間になっても終了しない時の実行例である.

```

a b c d e
Type return-key to start.

Type the following text:

a b c d e

Your input: a b c d e

Accuracy: 90.00% | Time: 5.31 seconds | Word Matches: 5

Type the following text:

a b c d e

Your input: a b c d e

Accuracy: 90.00% | Time: 10.48 seconds | Word Matches: 5

Type the following text:

a b c d e

Your input: a b c d e

Accuracy: 90.00% | Time: 14.54 seconds | Word Matches: 5

```

図 3.5: ストップウォッチ失敗.

この旨を ChatGPT に提供するが改善されない。ChatGPT が提供したコードと shunkun-type のコードを比較する。ストップウォッチの機能を別スレッドで動かすことで、この問題を解決することができるのではないかと仮定した。

入力 次に1分間を測ってブレイクをかける loopthread を追加

出力 1分間の制限時間内でループする loopthread を追加するサンプルコードを提供

1分経過したのちに、プログラムを終了させることができた。

3.1.2 コード解説

以下にタイピングソフトの基本機能の code を示した。

1. Example sentence

sentence の主要メソッドの `init_proc` を図 3.6 に示した。ファイルからデータを読み込んで表示し、ユーザに指示を表示してから Enter キーが押されるまで待ち、その後の時刻を取得して返す役割を果たす。

```
def init_proc(self, file_name): # 初期化処理
    with open(file_name) as f:
        data = f.readlines() # データファイルを読み込んで行ごとのリストに
        for word in data:
            print(word, end='') # 文章を表示

    print("\nRepeat above sentences. Type return-key to start.") # ユーザに指示を表示
    line = input() # ユーザの入力待ち
    start_time = time.time() # 開始時刻を記録
    return start_time, data # 開始時刻とデータを返す
```

図 3.6: sentence.

`with open(file_name) as f:` ファイルを開くための `with` 文を使用し、`file_name` で指定されたファイルを開く。 `with` 文を使用することで、ファイルの処理が終わった後に自動的にファイルを閉じることが可能に。

`data = f.readlines():` ファイルから行ごとにデータを読み込み、`data` というリストに格納。

このメソッドは、トレーニングの初期化処理を行い、ユーザに指示を表示してから Enter キーが押されるまで待機する。そして、初期時刻とデータを返す。

2. Word Match

図 3.7 に示した counter メソッドは、2 つの文章を単語ごとに比較して、一致した単語の数をカウントする役割を果たす。

```
def counter(self, word1, word2):
    ws1 = word1.split() # 文章を単語に分割
    ws2 = word2.split()
    match_count = 0
    for i in range(min(len(ws1), len(ws2))): # 単語ごとに正答数をカウント
        if ws1[i] == ws2[i]:
            match_count += 1
        self.word_counter = match_count
        # print("Word correct:", self.word_counter) # この行をコメントアウト
    return match_count # 正答数を返す
```

図 3.7: counter.

スペースで区切られた単語ごとのリストを得るため、word1 と word2 の文章を単語に分割し、それぞれ ws1 と ws2 に格納。

単語数が異なる場合にエラーを避けるため、min(len(ws1), len(ws2)) で、ws1 と ws2 の単語数の少ない方を取得。

self.word_counter に現在の正答数を保持。この行はループ内にあるため、各単語ごとに正答数が上書きされ、最終的には最後の単語の正答数が self.word_counter に格納される。

3. Stopwatch

```
def loop_thread(self, sec):
    def time_up(wait_time): # 制限時間経過時の処理
        time.sleep(wait_time)
        print('\a Time up. Type return-key to finish...')
        self.time_flag = False
    threading.Thread(target=time_up, args=(sec,)).start() # 別スレッドで制限時間を実行
```

図 3.8: stopwatch.

図 3.8 に stopwatch のコードの主要部を示した。

threading: モジュールの中で Thread メソッドクラスをインスタンス化によって非同期処理を実現している。

これらを作った後にアプリがスタートする。

time_up: タイムアップ時に実行される内容をメソッドとして定義

time.sleep: 指定された時間待つ

time.flag: 制限時間経過すると False にセットされて以降の入力を受け入れる.

これらより, タイピングソフトに必要な3つの基本機能を有するコードが実装できた.

図 3.9 が実行例である.

```
-----
[ioomanato@ioomanato ~/r/iotype> Python3 iotype2.py test.txt 15
test print open close -----1
Repeat above sentences. Type return-key to start.

test print open close
test print open close -----2

test print open close
test print open close

test print open close
test p Time up. Type return-key to finish.. -----3
rin

Total correct: 9
```

図 3.9: iotype の実行例.

1 でファイルから任意のセンテンスを読み出している.

2 は入力とのマッチを確認している.

3 で別のスレッドで動かしていた時計から割り込みが入り強制終了している.

3.2 keytype(段別タイピングトレーニング)の開発

keytype を開発するにあたり, タイピングソフトの基本機能を実装したソースコードを ChatGPT に学習させる.

ChatGPT を利用し keytype を実装するには, はじめにタイピングソフトの基本機能を実装したソースコードを学習させたい.

入力 タイピングソフトの基本機能を実装したソースコード

出力 このコードは, タイピングのトレーニングを行うためのプログラムのようです...

学習後, キーボード内の文字をランダムに表示させタイピングトレーニングを行うような keytype を開発したい. そのため, この機能の詳細を明示する.

入力 これらのコードを基本とし、キーボード内の文字をランダムに表示させタイピングテストを行うような key.py を開発したい。1 分間テストを行う、キーボード内の文字をランダムに表示

出力 以下は、キーボード内の文字をランダムに表示し、1 分間のタイピングテストを行う Python プログラム key.py の基本的な構造です。

```
iioomanato@iioomanato ~/r/g/s/m/C/key (main)> python3 k1.py
Typing test starting. You have 10 seconds.
RfWapaWoNa1CZnahdYsPdfmXUbSbHVleOUrPzHCg
RfWapaWoNa Time up. Test finished...
I
Incorrect. Try again.

Test finished. Correct count: 0
```

図 3.10: ランダムにセンテンスを表示.

図 3.10 のように、キーボード内の文字をランダムに表示させタイピングトレーニングを行うことが可能となった。しかし、40 文字のランダムな文字列が生成されてしまっている。

これは、意図していた内容ではなかったため、このプログラムの詳細を 1 文字ずつ表示するように変更させた。

入力 key.py の詳細を変更したい。1 文字ずつ表示する

出力 test.py を 1 文字ずつ表示するように変更するためには、文字列を 1 文字ずつ分解して表示するようにコードを修正する必要があります。

図 3.11 が実行した時の実行例である。

```
iioomanato@iioomanato ~/r/g/s/m/C/key (main)> python3 k2.py
Typing test starting. You have 10 seconds.
q q
o o
b b
r r
l l
u Time up. Test finished...
u
Test finished. Correct count: 6
```

図 3.11: 1 文字ずつ表示.

次に、段ごとにタイピング練習できるような仕様に変更したい。

入力 キーボードの段ごとにテストを行うように変更

出力 キーボードの段ごとにテストを行うように変更するには、キーボードの段をリストとして定義し、ランダムに段を選んでからその段からランダムな文字を選択するようにコードを変更します。以下は、その変更を加えたコードです。

図 3.12 が段別トレーニングを実行した時の実行例である。

```
iioomanato@iioomanato ~/r/g/s/m/C/key (main)> python3 k3.py
Typing test starting. You have 60 seconds.
Choose a keyboard row to practice:
1. Top row (qwertyuiop)
2. Middle row (asdfghjkl)
3. Bottom row (zxcvbnm)
4. All (Test all rows)
Enter row number (1-4): 1
y y
i i
r r
p p
```

図 3.12: 段別トレーニング。

最後に、iotype から管理実行させるべくその詳細を明示した。

入力 これを iotype から実行できるようにしたい

出力 「TypingTest」クラスを iotype.py から実行するために、iotype.py スクリプト内で TypingTest クラスをインスタンス化し、実行する必要があります。以下は、iotype.py から 「TypingTest」クラスを実行するためのコードです。

図 3.13 が、keytype の機能を iotype から実行することに成功した。

```
iioomanato@iioomanato ~/r/g/s/m/C/key (main)> python3 Iotype.py 10
Typing test starting. You have 10 seconds.
Choose a keyboard row to practice:
1. Top row (qwertyuiop)
2. Middle row (asdfghjkl)
3. Bottom row (zxcvbnm)
4. All (Test all rows)
Enter row number (1-4): 4
Selected row index: 3
w w
d d
p p
a a
q q
u u
y y
f f
a a
t Time up. Test finished...
Test finished. Correct count: 9
```

図 3.13: keytype の実行例。

3.3 trainingの開発

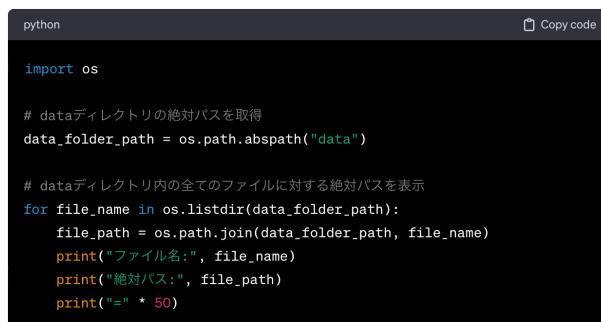
training を開発するにあたり、タイピングソフトの基本機能を実装したソースコードを ChatGPT に提供し学習させる。

ファイルを読み込み、そのテキストをタイプすることでタッチタイピングを習得する training モジュールを開発したい。

入力 これから data フォルダに管理している STEP-.txt ファイルを読み込みタイピングトレーニングを開発したい。Python, 絶対パスを使用

出力 タイピングトレーニングを開発するために、Python を使用して data フォルダ内の STEP-.txt ファイルを読み込む方法を以下に示します。絶対パスを使用しています。

図 3.14 が ChatGPT から提供されたコードである。



```
python Copy code  
  
import os  
  
# dataディレクトリの絶対パスを取得  
data_folder_path = os.path.abspath("data")  
  
# dataディレクトリ内の全てのファイルに対する絶対パスを表示  
for file_name in os.listdir(data_folder_path):  
    file_path = os.path.join(data_folder_path, file_name)  
    print("ファイル名:", file_name)  
    print("絶対パス:", file_path)  
    print("=" * 50)
```

図 3.14: ChatGPT からの code 提供.

ChatGPT の回答を参照し、実行すると図 3.15 のようにファイルからテキストを読み込むことに成功した。

```
iioomanato@iioomanato ~/r/g/s/m/C/t/data (main)> python3 ../t2.py STEP-1.txt  
ファイル名: STEP-1.txt  
ファイル内容:  
frf ded juj kik
```

図 3.15: ファイルからの読み込み.

ファイルから読み込みテキストを表示することができた。

次にファイルからテキストを読み込むコードを用いてトレーニングを行いたい。

入力 ターミナル上でそれぞれのファイルを指定してトレーニングを行いたい

出力 ターミナル上で指定されたファイルを使用してタイピングトレーニングを行うには、以下のような Python スクリプトを作成することができます。このスクリプトは、指定されたファイルを読み込んでその内容を表示するだけの簡単な例です。実際のトレーニングのロジックを追加することで、タイピングトレーニングを行うことができます。

```
iioomanato@iioomanato ~/r/g/s/m/C/t/data (main)> python3 ../t3.py STEP-1.txt
frf ded juj kik
frf ded juj kik
finished.
```

図 3.16: training 失敗例.

図 3.16 のように、実行したところ、全て打ち終えエンタキーを押すとプログラムが終了してしまう。

ターミナル上から制限時間を受け取り終了させることを明示した。

入力 これらを制限時間をターミナル上から受け取りタイピング練習を繰り返すように変更

出力 了解しました。以下は、制限時間をターミナルから受け取り、その制限時間内にできるだけ多くのタイピング練習を行うスクリプトの例です。

```
iioomanato@iioomanato ~/c/g/s/m/C/t/data (main)> python3 ../t4.py STEP-1.txt 10
frf ded juj kik
frf ded juj kik
frf ded juj kik Correct!
frf ded juj kik
frf ded juj kik Correct!
frf ded juj kik Correct!
Time up. Test finished...
fr
frf ded juj kik false!
```

図 3.17: training の実行例.

図 3.17 のように、training を行うことが可能となった。

3.4 プロジェクトのディレクトリ構造を整理

プロジェクトのディレクトリ構造を整理したい

入力 プロジェクトのディレクトリ構造の整理を手伝って

出力 各ディレクトリの役割について説明される。

pip 化するために整理したい

```
plaintext

my_typing_app/
|-- my_typing/
|   |-- __init__.py
|   |-- iotype.py
|   |-- key.py
|   |-- read_file.py
|   |-- training.py
|   |-- typing_test.py
|-- tests/
|   |-- test_iotype.py
|   |-- test_key.py
|   |-- test_read_file.py
|   |-- test_training.py
|   |-- test_typing_test.py
|-- data/
|   |-- STEP-1.txt
|   |-- STEP-2.txt
|   |-- ... (他のデータファイル)
|-- scripts/
|   |-- trans.rb
|   |-- trans2.rb
|-- word.list
|-- setup.py
|-- README.md
|-- requirements.txt
```

図 3.18: ディレクトリ構造の提案.

入力 これらを pip 化するにあたって整理したい

図 3.18 が ChatGPT から提供されたディレクトリ構造である.

次に, 変更手順 (コマンド例) を受け取りディレクトリーの整理を始めようとしたが, 私が提供した環境下ではなく, ChatGPT の仮想した図 3.18 の環境下での変更手順であったため, 次の命令を与えた.

```
You
今はiومانانو@iومانانو ~/r//graduation [127]> tree.

.
├── iotype.py
├── __pycache__
├── iotype.cpython-38.pyc
├── key.cpython-38.pyc
├── read_file.cpython-38.pyc
├── test.cpython-38.pyc
├── test2.cpython-38.pyc
├── training.cpython-38.pyc
├── typing_test.cpython-38.pyc
├── data
│   ├── STEP-1.txt
│   ├── STEP-10.txt
│   ├── STEP-11.txt
│   ├── STEP-12.txt
│   ├── STEP-13.txt
│   ├── STEP-14.txt
│   └── STEP-15.txt
```

図 3.19: iotype の tree の現状ディレクトリー構造.

図 3.19 に示した現状のディレクトリ構造を提供することで, ターミナル上で変更を行うための手順を受け取った.

入力 提案されたディレクトリ構造にするためのコマンドを提供して提供されたディレクトリ構造に変更するべく, ChatGPT から提供されたコマンド操作に従い整理を進めた. その際, ディレクトリではなくファイルとして存在していたもの, そもそも存在しなかったものがあったが, エラー文をその都度 ChatGPT に提供し, 再度コマンド操作を受け取ることで解決に至った.

全てのコマンド操作を終えた結果は図 3.20 の通りである.

```
graduation — ~/r/I/graduation — -fish — 67x47
ioomanato@ioomanato ~/r/I/graduation> tree .
.
├── Iootype_app
│   ├── Ioo_typing
│   │   ├── Iootype.py
│   │   ├── key.py
│   │   ├── test.py
│   │   └── training.py
│   ├── scripts
│   ├── tests
│   │   ├── test_iootype.py
│   │   ├── test_key.py
│   │   ├── test_test.py
│   │   └── test_training.py
│   └── word.list
├── __pycache__
│   ├── Iootype.cpython-38.pyc
│   ├── key.cpython-38.pyc
│   ├── read_file.cpython-38.pyc
│   ├── test.cpython-38.pyc
│   ├── test2.cpython-38.pyc
│   ├── training.cpython-38.pyc
│   └── typing_test.cpython-38.pyc
└── data
    ├── STEP-1.txt
    ├── STEP-10.txt
    ├── STEP-11.txt
    ├── STEP-12.txt
    ├── STEP-13.txt
    ├── STEP-14.txt
    ├── STEP-15.txt
    ├── STEP-16.txt
    ├── STEP-17.txt
    ├── STEP-18.txt
    ├── STEP-19.txt
    ├── STEP-2.txt
    ├── STEP-20.txt
    ├── STEP-21.txt
    ├── STEP-22.txt
    ├── STEP-23.txt
    ├── STEP-24.txt
    ├── STEP-25.txt
    ├── STEP-26.txt
    ├── STEP-27.txt
    ├── STEP-28.txt
    ├── STEP-29.txt
    └── STEP-3.txt
```

☒ 3.20: iootype tree.

第4章 iotypeの仕様

4.1 keytype 段別タイピングトレーニング

「keytype」は、段別のタイピングトレーニングを提供する主要なモジュールだ。トレーニングの進捗や正確性をモニタリングし、ユーザーのスキル向上をサポートする。

```
iomanipato@iomanipato ~/r/I/g/I/Ioo_typing [127]> python3 Iotype.py keytype 10
Typing test starting. You have 10 seconds.
Choose a keyboard row to practice:
1. Top row (qwertyuiop)
2. Middle row (asdfghjkl)
3. Bottom row (zxcvbnm)
4. All (Test all rows)
Enter row number (1-5): 1
e e
o o
q q
r r
e e
q q
y y
o Time up. Test finished...
Test finished. Correct count: 7
Do you want to see the graph? (y/n): █
```

図 4.1: key トレーニングの実行例

4.2 training

「training」モジュールは、ユーザーがトレーニングを行うためにファイルを読み込み、その中に含まれる文字をタイプしていく機能を提供。ユーザーは自身のペースでトレーニングを進め、継続的なスキル向上を促進する。

```
ioomanato@ioomanato ~/r/I/g/I/Ioo_typing [127]> python3 training.py data/STEP-1.txt 10
frf ded juj kik
frf ded juj kik
frf ded juj kik

Correct数: 12

training finished.
```

図 4.2: training の実行例

4.3 test 段別タイピングテスト

「test」モジュールでは、ユーザーがタイピングを行っているかを確認する。ユーザーは1分間のテストを行い、その結果を通じてスキルの定量的な評価が可能だ。これにより、ユーザーの進捗やパフォーマンスの向上を定期的に評価できる。

```
h h
e e
h h
q q
u u
p p
q q
c c
z z
f d
j j
n n
h h
e e
n n
d f
g g
d Time up. Test finished...
d
Test finished. Correct count: 45
```

図 4.3: test の実行例

4.4 タイピングトレーニング結果のグラフ表示機能

ユーザーがタイピングトレーニングを行うたびに、トレーニングの結果がシステムに蓄積される。結果には日付や正答数が含まれる。これにより、ユーザーは自身の進捗を的確に把握できる。

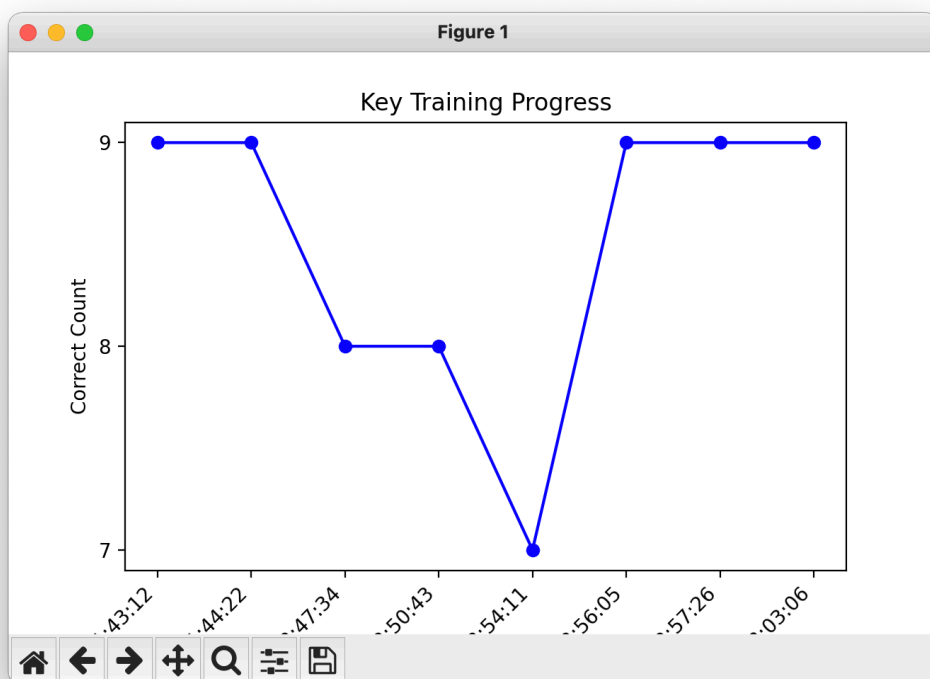


図 4.4: トレーニング結果のグラフ表示

第5章 今後の課題

研究の終了にあたり, iotype を実際に使用することで未検出の問題や改善点が存在する可能性が考えられる. そのため, 未解決の問題の特定と修正を通じて, iotype が問題なく利用できることを確認する必要がある. また, 西谷が担当するコンピュータ演習に iotype を導入するには pip 化が必要である.

現在のスクリプトはコマンドラインから実行されるが, アプリケーションを pip パッケージとして公開することで, ユーザーがより簡単に利用することができる.

アプリケーションの動作を確認するために, 複数のプラットフォーム (Mac, Windows など) でのテストを行うことが必要である.

第6章 まとめ

本研究では, 先行研究である「shunkuntype」を参照し, Python にてタイピング練習用ソフトを開発した. iotype は, 3つのタイピングトレーニングメソッドを提供し, それらの過程を可視化するグラフ表示機能に実装に成功した. これらにより, ユーザーはタイピングトレーニングをより効果的に取り組むことが可能となった.

ChatGPT を利用したソフトウェア開発に取り組んだことで, 以下について確認できた. ChatGPT に期待する動作やコードを得るためには, 仕様やアプローチ方法をより詳細に明示する必要がある. その際に, 知識やコード開発のスキルが必要であるため, ある程度のプログラミングスキルがなければ開発を進めることができない.

謝辞

本研究を進めるにあたり、西谷滋人教授には指導教員として終始熱心なご指導を頂きました。心から感謝いたします。

参考文献

- [1] OpenAI - <https://ja.wikipedia.org/wiki/OpenAI>(accessd on 08 Feb 2024).
- [2] daddygongon, 2016, “Shunkuntype”, <https://github.com/daddygongon/shunkuntype>, (accessed on 24 Jan 2024).
- [3] ChatGPT - <https://chat.openai.com> (accessd on 24 Jan 2024).