Python 風日本語プログラムの作成について

山口あゆみ

目　次

1　序論 1

1　はじめに 1

2　制作計画 2

3　対象者 3

4　目標 3

5　類似ソフトについて 5

2　本論 6

1　制作条件 6

2　制作過程 7

3　具体的な機能 10

4　使用方法 10

5　問題点および解決の過程 11

6　工夫した点 13

7　wython との比較 14

8　使用制限 15

3　結論 16

1　自己評価 16

2　動作テスト 16

3　改善点 19

4　今後の課題と結論 20

——Python 風日本語プログラムの作成について——

1　序論

(1) はじめに

人文情報学科の福田ゼミのテーマは、「人の役に立つものを作る」である。この「人の役に立つものを作る」ことに則り、私は卒業論文のテーマを「Python 風日本語プログラムの作成について」とすることにした。

福田ゼミの必須科目の 1 つである「プログラミング演習 1 」はスクリプト言語 Python によってテキストの処理を主体にしたプログラムを実際に組んで、プログラムの基礎を学習する授業である。プログラムの多くは、日本人があまり馴染みのないアルファベットや数字、英単語で組まれることが多くプログラムの構造やプログラムの文法がいまひとつ理解しにくいと考えている。

実際プログラミングを始めたくても始められない人の中には、アルファベットや数字の羅列で何がどうなっているのかわからないと言っている人も多い。私自信も、3 回生の時に受講したプログラミング演習 1 でプログラミングを学び始めたときにプログラムの構造がわからず苦労した経験を持っている。このような経験を持つ人は決して少なくは無いと考えられる。

そこで、Python などプログラミング言語を学び始めた人がプログラムの流れを簡単に理解できるような間を取り持つアプリケーションがあれば便利だと感じた。人文情報学科の福田洋一先生は以前より Python 導入のため日本語のプログラムを Python スクリプトに変換する wython というソフトウェアを作っていたと聞いた。そこで、wython の開発のことが書かれている先生のブログ(1)を読み、

そこで、日本語でプログラムの手順を組み立てるのに適したプロ

グラミング言語を作り、その基本的な概念や考え方を、特定の言語の制約に煩わされずに理解できるようにしたら、プログラミングの初心者の入門用言語として、とても役に立つのではないか。(http://yfukuda.blog.so-net.ne.jp/archive/c30073-5 2005-01-05)

という趣旨に賛同した。事実、日本人にとって使い慣れた母国語である日本語によってプログラムを組むことができたら、プログラムを学び始める難関になっているアルファベットの羅列で何がどうなっているかわからないということが解消されるはずである。このため、日本語プログラミングを卒業論文のテーマにしようと考えたのである。

(2) 制作計画

具体的には、「日本語で書かれたプログラムを Python スクリプトに変換する」というアプリケーションを作る予定である。このアプリケーションは日本語でプログラムを書き、その後すぐにプログラムを実行するというものではなく、日本語で書かれたプログラムを Python スクリプトにそのまま置き換えていくというものである。

つまり日本語で書かれたプログラムを翻訳するといった翻訳機の役割を持つものを作ろうと考えている。アプリケーションが翻訳機の性格を持つ事によって、自分が書いたプログラムがどういったものになったか確認することができ、Python スクリプトでプログラムを作るときの混乱を軽減させるのに役に立つだろう。

また Python スクリプトに変換していく際、日本語で書かれたプログラムのファイルに直接上書きするのではなく、別にファイルを作り、そこに Python スクリプトに変換したものが出力されるようにする予定で

ある。

別のファイルに出力させる理由は、もとの日本語で書かれたプログラムとできあがった Python スクリプトが書かれたファイルを並べて見比べることができ、プログラムの流れがどのようになっているのかという理解がより深まるのではないかとの意図からである。

つまり Python スクリプトをどう書いていったら良いかという勉強のために制作するのである。

(3) 対象者

対象者はプログラム、特に Python を学び始めた人とする。このアプリケーションは日本語でプログラムを組み、即座に実用的なプログラムを作るということを第一の目的としている訳ではなく、プログラムの具体的な流れを理解してもらうことを第一の目的としている。このような理由から、ある程度プログラミングを理解している人には使う必要のないアプリケーションであると位置づけている。よって、このアプリケーションはプログラムの流れが分からないという理由で前へ進めなくなってしまう機会の最も多いプログラムを学び始めた人を対象としている。

(4) 目標

「だれがどんな使い方をしても動くようにする」を基本に目標を立てている。具体的な目標は、

1. プログラミング演習 1 で使われる文法や関数を変換できるようにする
2. どんな言い回しをされても対処できるようにする

3. 手軽に使えるようにする

の 3 つである。この 3 つの目標に重点を置いて作りたいと考えている。

まず、1 つ目の目標は、プログラミング演習 1 で使われる文法や関数を変換できるようにすることについてである。プログラミング演習 1 は Python の基礎的なことを学ぶ授業である。このアプリケーションはプログラミングを学び始めた人を対象にしている。よって制作者は、基礎的な文法をカバーできれば良いと考えている。その他の理由としては、多くの関数や構文を処理しようとするとプログラム自体も複雑になる可能性があるからである。よって、制作者はプログラミング演習 1 で使われる基本構文や関数を重点的に変換できるようにすることを目標としている。

次に、2 つ目の目標は、どんな言い回しをされても対処できるようにすることについてである。制作者は、日本語の特徴に関する理由からこのような目標を立てたのである。日本語は 1 つのことを表すことでも多様な言い回しができる。たとえば「質問」という言葉に対して、「質問する」や「尋ねる」といった類義語が存在する。同じことを意味する言葉でも使用者によってまったく異なった言い方がされるので、どんな表し方をされても同じように動くようにしたいと考えている。

最後に、3 つ目の目標である手軽に使えるようにすることを制作者が目標に立てたのは、いくら良いアプリケーションであっても使い勝手が悪ければそれは使えないアプリケーションであるという考えからである。たとえば自動車の場合日本でこれだけ普及するまでの間、操作のしにくいミッションから操作が簡単なオートマチックの出現、ハンドルの重さの改善など多くの開発や改良がなされ、今では免許取りたての人や老人でも自動車に乗れるように簡略化が進み、多くの人に使われるよう

になった。

これと同様に、使い方が複雑であったり使い勝手が悪かったりした場合、良いソフトウェアであっても使われなくなってしまうと制作者は感じている。よって制作者は、誰でも手軽に使えるようにしようと考えた。

(5) 類似ソフトについて

現在、日本語でプログラムを組めるアプリケーションは様々に存在している。ひまわり、なでしこ、言霊、Mind、TTSneo、日本語といったものがその代表格として挙げられるだろう。これらの日本語プログラミングソフトは、日本語でプログラムを組み直接実用的なプログラムを作り出すことができるものである。

こういったソフトウェアはそのソフトウェア自身が独自の便利な命令を備えていることが大半であり、後に Python などでプログラムを作ろうとしたときにその便利な命令が逆に障害になる可能性がある。

これに対し、wython は、日本語プログラミング言語で書かれたプログラムを Python のスクリプトに変換するものである。よって、ひまわり、なでしこ等の日本語プログラミングソフトとは異なる。現在作ろうとしているものは日本語でプログラムを組み、それを Python スクリプトに置き換えるというものなので、wython の機能と同じものである。

では、なぜ同じ機能を持つ物があるのに作ろうと考えたかというと、2 つ理由が挙げられる。1 つ目の理由は、トラブルが発生したらしく wython が失われてしまったからである。2 つ目の理由としては、wython は制作者である福田先生によって新しく作り直されていたが、

実際使うにあたって書き方についての制約が多く、言葉の自由度が少なかったからである。以上の理由から新しく作ろうと考えたのである。

2　本論

(1) 制作条件

制作に使うスクリプト言語は、Python を使うことにした。これは、置き換える対象が Python だからである。それ以外の理由として、Python はプラットホームの多くをサポートしており、環境に依存しないコードを書くことができるからである。

また Python は、テキスト処理やインターネット関連など各種処理を行う数多くのいろいろな拡張モジュールが配布されており、拡張機能を簡単に使うことができるからである。実際、このプログラムのメインとなる正規表現は Python の拡張機能の 1 つを使っている。

これらの理由により、このプログラムの制作で使用するスクリプト言語は Python を使用することにした。

制作は主に Mac OSX と Windows を使用した。Mac OSX では Emacs と Jedit を使い、Windows では EmEditor を使用した。これは学校での制作環境と自宅での制作環境が異なるためである。Mac OSX で Emacs と Jedit を使用したのは、print や import などの命令文や正規表現、コメントアウトをしてあるところをそれぞれ別の色で色分けして表示させるため、大変見やすく使い勝手が良いからである。

Windows で EmEditor を使用したのも同様の理由で、このソフトも正規表現やプログラムの命令、コメントアウトをしてあるところを色分けして表示するので、非常に制作が行いやすくなるからである。

その他の理由として、Mac OSX で使用した Jedit と Windows で使用した EmEditor は、OS によって異なる改行コードや文字エンコードを大変容易に変更できるからである。

これらの理由により Mac OSX では Emacs と Jedit を使用し、Windows では EmEditor を使用したのである。

(2) 制作過程

制作を開始するに当たって、はじめに Python の勉強を行った。Python は一度必須科目である「プログラミング演習 1」で Python を学んでいたが、この授業では Python の基本のみを学んだのであって、応用のことは学べていなかった。さらにプログラミングの決まりごとなども詳しくわかっていなかった。そのため制作者は、このプログラミングの決まりごとも学びたいと考え、もう一度 Python を勉強したのである。

Python の勉強は『Python で学ぶプログラム作法』で勉強することにした。この本は、「Python」という言語を使用してプログラミングのことを解説しているが Python だけの解説書ではなく初心者向けのプログラミング解説書である。プログラムの概念や変数、条件の分岐といったプログラムの基本要素から再帰呼び出し、正規表現、デバッグなどプログラミングの重要な要素を解説している。

この本は初心者向けの入門書としては比較的難しく、徹底的にわかりやすい解説を行っておらず、コメントの書き方や変数の名前の付け方といった初歩的なことからプログラマーが押さえておくべきことを解説すると共に、プログラミングにおける重要な技法や作法が書かれている。

具体的なコードを使用して解説しており、初歩的な内容から高度な内容、そしてゲームなどの応用プログラムの開発にいたるまで説明されている。

単純に Python のことを解説している本ではなく、プログラマーが知っておくべき作法について書かれている。このような理由からこの本を選び Pyhton の勉強をすることにした。

次に、変換する際に使う日本語を決めていくことにした。具体的な決定方法は、3 回生のときに受講した「プログラミング演習 1」で作っていた Python スクリプトにコメントをつけるような感覚で日本語に置き換えていき、書き出した。こうして書き出したものを一度、表示処理をあらわすものなら表示文、代入処理をあらわすものなら代入文と同種類の処理を表すものをまとめた。

この他にも、日本語プログラミングソフトのひまわりやなでしこで使われている日本語も参考にさせてもらい言葉を集めた。そして、処理を行うことに使われている日本語の中で存在し、共通して使われている言葉を探し使用する日本語を決定していった。

次に、上記の制作過程から考えて実際に制作をするための方法を決定した。このアプリケーションは日本語で書かれたプログラムの命令を検出し、その検出したものを Python スクリプトに変換するものである。日本語で書かれたプログラムの命令を検出し、Python スクリプトに変換することに 1 番適しているのが正規表現である。

そこで、新たに正規表現を学ぶことにした。勉強は、インターネットで公開されている正規表現のチュートリアルを使うことにした。勉強に使った正規表現のチュートリアルは、

1. 正規表現 HOWTO [(2)]

2. Python ライブラリリファレンスの 4.2 re – 正規表現操作[3]
3. お気楽 Python プログラミング入門の第 4 回 正規表現とジェネレータ[4]

の 3 つである。この 3 つのチュートリアルは、それぞれはじめに正規表現について説明しその後、Python のモジュールの 1 つである re モジュールを使用した実践的な使い方を解説したものである。これらのチュートリアルは、勉強の際にも利用したが、制作中でもリファレンスとして活用させてもらった。

勉強は、上記のインターネットで公開されている正規表現のチュートリアル以外にも秀和システムが出版している『正規表現 ハンディリファレンス』を使うことにした。この『正規表現 ハンディリファレンス』は、インターネット上で公開されている正規表現のチュートリアル同様、まず正規表現について説明し、その後、egrep、gawk、emacs、perl、JavaScript、ruby といった各ツールで利用できるメタキャラクタを個々に分けて解説している。さらに実際に表示される画面の画像もありより理解しやすくなっている。

最後に、決定した使用日本語に基づいて色々な言い回しにマッチできるような正規表現を決定し、制作に取りかかった。

制作した順番は、はじめに基本動作となる数の処理、代入文、表示文を作った。1 番はじめに基本動作を作ったのは、この基本動作がないと重要となる動作チェックができないためである。その後、制御文である if 文 while 文、インポート、リスト関係、ファイル操作関係、関数と作っていった。

(3) 具体的な機能

制作するアプリケーションは、プログラミング演習 1 で使われる文法や関数を変換できるようにするというものである。よって、プログラミング演習 1 で使われている文法や関数が変換対象となる。この変換が可能な文法や関数は、代入文、表示文、加算、除算、乗算、割り算等の基本動作が主である。

その他に制御文である if 文と while 文や拡張機能を読み込むインポート文も使用可能である。リスト関係としては、リストの最初から何番目か、あるいは最初から最後まで、または途中から途中までといったものを使うことができる。ファイル操作関係では、ファイルのオープン、クローズ、書き込み、読み込みを行うことができる。

関数では、長さを数えることができる len()、数、文字列をそれぞれ浮動小数点数、整数、文字列に変換する、float()、int()、str() を変換可能とした。これ以外にも、使用者に質問や入力を促す関数である raw_input() も使用可能である。

また、プログラミング演習 1 で、使用されるモジュールである sys の関数である argv も変換することができる。これらがこのアプリケーションで使用可能な文法や関数である。

(4) 使用方法

このアプリケーションを使うには、使用するコンピュータにあらかじめ Python をインストールしておく必要がある。これをふまえた上で、変換対象となるファイルとこのアプリケーションを同じディレクトリに入れおき、Python のインストールされているコンピュータのターミナ

ル上で動かすことになる。

はじめに、ターミナル上で変換対象となるファイルとこのアプリケーションの入っているディレクトリに移動しておく。

次に、「python 本アプリケーション名 変換対象となるファイル名」と入力し実行する。そうすると、アプリケーションは出力されるファイル名を聞いてくるので使用者に任意のファイル名を入力してもらう。この時、拡張子を書く必要はなく、使用者はただ任意のファイル名を入力するだけでよい。

最後に出力が成功したら、このアプリケーションと変換対象となるファイルがあるディレクトリ内に使用者が任意につけたファイル名 +.py のファイルが作成されているはずである。この作成されたファイルはターミナル上で動かすと、エラーが出ない限りプログラムとして実行できる。

(5) 問題点および解決の過程

このプログラムを制作している途中様々な問題が発生した。そこで、問題が発生した点とその問題の解決方法を述べていくことにする。問題が発生したものは

1. 文字列と変数の区別
2. 代入文
3. print 文

である。

１つ目の問題は、文字列と変数の区別である。文字列は、平仮名や片仮名の場合はすぐに文字列と判断できるが、英単語や数字は文字列では

なく変数と区別してしまう可能性がある。そこで、文字列を表す部分には使用者にシングルクォーテーションマークやダブルクォーテーションマーク、鉤括弧などで文字列であるということを示してもらうことにした。こうすることによって文字列か変数かを区別することができるようになった。

ちなみに鉤括弧で文字列を示してもらった場合、鉤括弧のままプログラムを実行しようとするとエラーがでてしまうので鉤括弧は自動的にダブルクォーテーションマークに置き換えられるようにした。

２つ目の問題は代入文で起こったものである。この代入文の処理で起こった問題は主に、言い回しの違いや関数を使ったときに起こるものである。代入文は多くのことに使われる基本的な文である。代入文は、数字や文字列を代入するものから、計算式を代入するもの、関数によってある処理をされたものを代入するなど多様な使い方ができる。代入文のこれら全ての使い方に対応するために代入文の処理は、全ての文字や記号に対応させるようにした。

３つ目の問題は、print 文である。print 文もまた、変数の表示や文字列の表示など使える範囲が広い命令である。そこでこの print 文も代入文と同様に全ての文字や記号に対応させるようにした。

また、表示することを示す命令である print は、改行することも表している。そこで、日本語で書かれたプログラムの中で改行するという命令が書かれていた場合でも print と置き換えることにした。しかし、使用者によっては改行したいときにわざわざ「改行する」や「改行」と書かない人も多い。そこで、空白行を検出したら自動的に print と表示するようにした。

(6) 工夫した点

プログラムを制作するにあたり特に気をつけたことは、目標の 1 つにも挙げてある「どんな言い回しをされても対処できるようにする」を実行できるように工夫することである。同じことをあらわすものであっても使う人によって全く違った表現の仕方をすることは日常生活でも頻繁に起こっている。

例えば、代入文で「a = 0」という記述があったとする。これは日本語にすると「a に 0 を代入」となる。しかし、この言い回しの他にも、「a に 0 を代入する」といったように「代入」の後に「する」とつける使い方をする使用者がいるかもしれない。その他にも「代入」を「入れる」とする使用者もいるだろう。

また、この「a = 0」という記述を「a に 0 を代入」とせずに「0 を a に代入」とする使用者もいる可能性がある。このような 2 つの場合に対処するために、代入文の処理をする正規表現を資料編ソースコードの 25 行、26 行のように 2 つに分けることにした。こうしておけば「a に 0 を代入」とする場合と「0 を a に代入」とする場合の両方に対応することができる。

こういった言い回しの違いにできるだけ対処できるように特に気を配り制作を行った。このように言い回しの違いに対処できるように制作したのは、使用者にできる限り自由な表現をしてもらいたいとの考えからである。

使用者による違いは、言い回しの違いの他にも、書き方の違いもある。書き方の違いは、文章を書くことに置いて数字を全角で書くのと半角で書くということもあてはまる。全角数字と半角数字では表す意味が変わらないので、全角で書かれた数字は全て半角の数字に置き換えるよ

うにした。

また、その他の工夫として、出力ファイル名は自分で決められるようにした。福田先生が以前制作した wython で設定されていた出力ファイル名は、読み込んだファイル名＋.py であった。たしかにこうしておけばいちいちファイル名を考える必要がなくなり便利である。

しかし、他のファイル名をつけたいと思ったときには、出力したファイル名を再び名前を変えることになる。こういった場合には不便であると考えられる。そこで、このアプリケーションは読み込んだ後に、出力ファイル名を自分で決定してもらうことにした。

これら以外にも、出力した Python スクリプトの先頭に「`#!/usr/bin/env python`」、次の行に「`#-*-coding:utf-8-*-`」と書き込まれるようにしておいた。こうしておくことによって、変換後のプログラムファイルが手直しせずそのままターミナル上で実行できるようにした。

(7) wython との比較

先に述べたように、wython とこのアプリケーションの異なる点は、出力ファイル名を自分で決定してもらった点である。しかし、それ以外にも wython と異なる点が存在する。それは日本語プログラムの命令文の言い回し方の自由度をさらに向上させた点である。

例えば wython では、「print a」のような記述を日本語にしたとき「a を表示」では変換されず、「a を表示する」としなければ変換されなかった。

wython ではこのように言い方が決められてしまい、この言い方を

知っていなければ使うことはできない。これは、使いやすいとは言えないだろう。そこで、日本語命令の言い回しをさらに自由に行えるようにした。

(8) 使用制限

なるべくどんな言い回し、使い方にも対処できることを目標にこのアプリケーションを制作していたが、それでも使用に制限をする必要が出てきた。そこで、使用に関しての制限を設けることにした。

1 つ目の制限はプログラミング演習 1 で使用する構文や関数以外は使えない物とする。これは当初の計画の 1 つでもある。プログラミング演習 1 で出てきた関数や構文以外のものは日本語で書いても変換されることはない。

2 つ目の制限は関数定義で関数を定義するときの関数名で日本語を使えないというものである。これは、Python で関数名が日本語であった時のことがサポートされていないからである。よって日本語で関数名をつけた場合にはエラーが発生してしまいプログラムが実行されない。

3 つ目の制限は文字列を表すときは必ず鉤括弧を文字列の前後につけなければならないというものである。使用者によっては文字列の前後に鉤括弧をつけない人もいるかもしれない。しかし、このプログラムの仕様では、変数名や数字と文字列を区別するために文字列の前後に鉤括弧をつける必要がある。

4 つ目の制限は if 文や while 文などの制御文や関数定義に関するものである。if 文や while 文、関数定義は実行処理内容を書くときに字下げを行うが、このアプリケーションでは自動で字下げがされない。よっ

て、if 文や while 文など制御文や関数定義の中で実行処理内容を書くときは、使用者自ら字下げを行う必要がある。

3　結論

(1) 自己評価

当初の目標であった、3 回生で学ぶ基本的なレベルのものは 1 部を除いてほぼ網羅できたと考えている。1 部のものとは、シングルクォーテーションマーク 3 つで括る、改行された通りに表示させる処理と文字列を区切り文字で区切りリストに返す処理をする関数の split() のことである。この 2 つは授業で使用されることが少なかったので置き換えはしないことにした。

これらの 1 部のものは変換されないが、それでもプログラミング演習 1 で学ぶ際に作られる基本的なプログラムは、日本語でプログラムを組むことができる。

(2) 動作テスト

人のアプリケーションの使い方は千差万別である。このアプリケーションの目標である「誰がどのように使っても同じように使えるようにする」という目標を達成させ、より完成度の高い物を作るためにはできるだけ多くの人に使ってもらい多くの情報を集める必要がある。

そこで、福田ゼミ 3 回生のゼミ生の 4 人にこのアプリケーションを実際に使ってもらい使用した感想などをアンケート形式にして答えてもらうことにした。ただし、4 人中 2 人がアプリケーション起動の際になん

らかのトラブルが発生して動かすことができなかった。このため、アンケートの結果は動かすことができた 2 人のものを見ることにし、動かすことのできなかった 2 人のものは参考とした。

アンケートはこのアプリケーションの使いやすさ、必要性があるかどうかの 2 つの設問を 3 段階評価で評価してもらい、その他にも良かった点、悪かった点および改善すべき点、その他の意見を自由に回答してもらった。

まず、1 つ目の「このアプリケーションは使いやすいか」という設問に関して、全員が「普通」と回答した。可もなく不可もなしといった状況である。使いにくいとの答えが返ってこなかったのは良かったが、使いやすさを目標に掲げていたため、このアンケート結果を真摯受け止めたい。

次に 2 つ目の「このアプリケーションは必要かどうか」という設問に関しては、全員が「必要である」と回答した。この結果は、このアプリケーションが有用であると評価してもらえたと考えている。

次に 3 つ目の「良かった点」という設問に関しては、以下のような答が返ってきた。

- 3 回生の最初の時期等、プログラミングの概念の無い人に、プログラミングの考え方を教えるのに良いと思う
- プログラムを作るためではなく、プログラムを解説するのに使えるかも知れない

2 つの回答ではどちらも日本語でプログラムを作るためではなく、プログラムの解説に使えるとしている。この回答については当初の狙いであるプログラムの流れを理解するのに役立てるという目的が完全に反映された物として受け止めている。

次に 4 つ目の「悪かった点」という設問に関しては、以下のような答えが返ってきた。

・足し算がうまくできない
・「以上」「以下」と表示させたいのに「>=」「<=」と表示されてしまう。
・かけ算等複雑な計算式を代入するときに時間が掛かる
・動作が遅い

この回答については、こちらのプログラムミスを指摘された結果となった。ここで指摘された問題点は主に 2 種類に分けられる。

1 つは記述ミスである。プラスとマイナスの判定が逆になっていたため動かなかったのである。また、他に自身が想定していなかった言い回しをされたために動かなかったところもあった。

2 つ目は動作が遅いという物である。これは、置き換えるときに別の置き換えの文を読み込んでしまったために遅くなったと考えられる。ここで指摘された点については修正した。

最後に 5 つ目の「その他の意見」という設問では、

・文法などを解説した別紙のマニュアルがほしい。

という意見が出た。この意見については、制作者もある程度までの使い方を解説したマニュアルは必要だと考えた。実際、初めてこのアプリケーションを使用者使うときは、どう使って良いかわからず戸惑ってしまうだろう。そこで簡単な導入と基本的な使い方の解説を書いたマニュアルを作り、アプリケーションと同梱することにした。

3 回生へのアンケートの他、先生の意見もいただいた。その意見とは、出力ファイル名は使用者に入力してもらうより読み込んだファイル名にした方が混乱が少ないため、出力ファイル名は読み込んだファイル

にした方が良いというものと、変換後、出力ファイルがすぐに実行された方が結果がわかりやすいというものであった。

(3) 改善点

以上アンケートと自己評価、それから先生の意見を取り入れた結果、アプリケーションを見直す必要が出てきた。見直しが必要な点をまとめると

- 足し算等の計算を間違いなく行えるようにする
- 素早く変換されるようにする
- 文章の最後に句点があっても、Python スクリプトの末尾に表示されないようにする
- 文字列「以上」「以下」を print 文で使うと「>=」「<=」に置き換えられるため置き換えられなくする
- 出力されるファイル名は、読み込んだファイル名 +.py にする
- 変換が完了した後、すぐに変換後のプログラムが実行されるようにする
- アプリケーションが動かなかったのでこれを動かせるようにする

のようになった。ただし、この見直しが必要な点のうちの最後、アプリケーションが動かなかった点については、後日あらためて同じ状況を作り出し動作テストを行ったところ問題なく動いた。そのため修正する必要は無いと考えている。それ以外の修正が必要な箇所を重点的にプログラムを修正していくことにする。

まず、1 番多く目立ったプログラムの記述ミスを修正することにした。これによって誤変換や変換不能箇所を減らすことができた。動作が遅い

点についてもこのプログラムの修正によって直すことができた。

次に、人によっては文章の最後に句点をつけるということがこのテストによって明らかになったので、文の末尾に句点が付きそうな箇所全てに句点も検出できるように変更した。

次に、文字列「以上」「以下」の変換ミスについて修正することにした。文字列「以上」「以下」の変換ミスが起こった原因は、単純に置き換えていたからである。そこを変更し、print 文で「>=」「<=」と置き換えられないようにした。

次に、先生の意見を取り入れ、はじめに制作していたような出力ファイル名を任意で決定してもらうことを止め、出力ファイル名を読み込んだファイル名とすることにした。

最後に、これも先生の意見を取り入れ、変換が完了した後、すぐに変換した成果がわかるようにプログラムを実行するように改善した。

このアプリケーションは、1 回で複数ファイルの処理をすることができない。このことについては、特にできなくても現段階では問題がないので今回は割愛することにした。

(4) 今後の課題と結論

動作テストをしてもらった結果、多くの問題点の指摘や様々な意見をもらうことが出来た。特に目立ったプログラムの記述ミスについては、プログラムを組む時に 2 度とこのようなことのないように心がけたい。

それ以外でも、この動作テストの結果分かったことは、人の言葉の言い回しは種々さまざまな違いがあることが分かった。まさかこんな言い方はしないだろうと思っていたことでも、事実その言い回しをする人は

いた。これ以外にも全く予期しない言葉の表し方をする人もいたので、はじめに立てた目標の 1 つである、どんな言い回しでも対応できるようにするということがどれほど困難なことか思い知る結果となった。より広く対応するためには、より多く言葉の表し方を知る必要があると感じた。しかし、それだけでは完全に言葉を網羅することはできないだろう。

これらのことから、幅広い人の役に立つものを作ることは非常に困難であると考えさせられた。しかし、ごく少数の人であっても、作った物を使ってもらえ、有用だと感じてもらえるのならば、それは人の役に立つものであるということを知ることができた。

注

（1）http://yfukuda.blog.so-net.ne.jp/archive/c30073-1

（2）http://www.python.jp/Zope/articles/tips/regex_howto

（3）http://php.atpedia.jp/python/ja/lib/module-re.html

（4）http://www.geocities.jp/m_hiroi/light/python04.html

文献表

クジラ飛行机（山本峰章）

2004 『日本語でかんたんプログラミング！「ひまわり」で学ぶアプリケーション作成』毎日コミュニケーションズ

有限会社ジェイ・シー・エヌ

2001 『正規表現 ハンディリファレンス』秀和システム

Alan Gauld(著) 松葉素子 (訳)

2001 『Python で学ぶプログラム作法』ピアソン・エデュケーション