プログラミング言語と日本語の相互変換サイトの作成

廣瀬　太知

目　次

――プログラミング言語と日本語の相互変換サイトの作成――

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」ことに則って、私は卒業論文のテーマを「プログラミング言語と日本語の相互変換サイトの作成」とすることにした。

本校の講義である「プログラミング演習」は、受講者が実際に PHP などの言語を用いてプログラムを組みながらプログラミングの流れや組み方を学んでいくという講義である。

しかし、受講者のほとんどはプログラミングの構文の意味を真に理解せずに、何がどうなっているのか分からないまま「おまじない的」にプログラムを書いて講義を進めている傾向にあることに気がついた。事実、私自身もプログラミングを学び始めた際には、コードの各行が何をしている部分なのか理解しないまま、「英語で書かれた意味不明なコードを教科書通りに書き連ねる」といった講義の進め方をしていた。これではプログラムの手法を真に理解しているとは言えず、教科書に書かれたサンプルコードを動かすことはできてもエラーの対処やプログラムを自由に組むなどの実践的な力は全く身につかないと考えられる。

このような受講者が多い原因は、そのほとんどが英語や記号で構成されているというプログラミング言語の特徴にあると考えられる。この特徴によりプログラムの構造が理解しにくくなり、プログラミングの流れや組み方を理解する前の段階で「言語の壁」と対峙することになってしまう。

そこで、我々に馴染み深い「日本語」を用いてプログラムを組めるアプリを作成することにより、プログラミング言語を学び始めた人が最初に対峙することになる「言語の壁」を払拭することができれば、プログ

ラムの各行が何の処理を行っているのかがひと目で分かるようになり、より円滑にプログラミングの流れについて学べるような「初心者の足掛かり」となる教材ができるはずだと考えた。このため、プログラミング言語と日本語の相互変換サイトの作成を卒業論文のテーマにしようと考えたのである。

2　類似ソフトについて

これまでにも日本語でプログラミングが出来るソフトはいくつか開発されている。Mind(1)、ひまわり(2)、なでしこ(3)、TTSneo(4)などがその代表格として挙げられる。

しかし、これらの日本語プログラミングソフトは、「日本語で書ける新しいプログラミング言語を作る」といったコンセプトで制作されており、かつての英語で書く一般的なプログラミング言語のような動作をどこまで日本語のプログラミング言語で再現できるかということに力を注いでいる印象がある。そのためか、これらの日本語プログラミング言語は自然言語とは程遠い言語構成になっており、プログラミング初心者にとっては不向きであると言える。また、これらの言語は他のプログラミング言語との比較ができず、日本語ですべてが完結してしまうため、Python などの他のプログラミング言語に移行しようとする際には言語仕様を一から学び直さなければならないといった問題点などが挙げられる。

これに対し、以前福田先生が制作していた教育用言語に Wython というものがある。Wython はプログラミング初心者のためだけに作られた教育用言語であり、「日本語をプログラミング言語として解釈してそのままコンパイルする」のではなく、「日本語で書かれたプログラム

を Python のスクリプトに変換する」といった特徴をもっている。この Wython は、我々が普段話すような自然な日本語でのプログラミングを可能にすることで、プログラミングの流れなどを理解しやすくして Python などの本格的な言語を学ぶための土台作りとして役立てるために制作されており、私が制作しようとしているアプリケーションとコンセプトが一致している。

Wython は、福田ゼミ生により度々改良されてきており、現在最も新しいバージョンは、2008 年に福田ゼミを卒業した山口あゆみさんが作成したものである。しかし、現時点での Wython には「変換できる日本語の言い回しがかなり限られている」、「変換前の日本語と変換後のプログラミング言語の比較がしにくい」、「実行するためには Python の実行環境が必要である」、「エラーが頻発し、最後まで動かない場合がある」などといった、いくつかの問題点がある。そのため、現状の仕様では「初心者のための足掛かり」としてはうまく機能しない可能性があり、改善の必要があると考えた。そこで、Wython のコンセプトを踏襲した上でこれらの問題点を解決した、より初心者にやさしい新たなアプリケーションを制作することにしたのである。

3　構想

上記の理由から、Wython のような日本語からプログラミング言語への変換を可能にするアプリケーションを制作しようと考えた。なお、今回制作するアプリケーションの名前は、「日本語を PHP あるいは Python へ変換する」という特徴に則って『J2P(Japanese to PHP/Python)』とすることにした。

では、具体的に J2P をどのような仕様にするのかを前述の Wython

の特徴とその改善案に則って述べようと思う。

(1) 言語仕様

まず、Wython の 1 つ目の特徴は、様々な言い回しに対応することを目標とする言語仕様である。日本語は「違うならば」「そうでないならば」「異なるならば」など、同じ意味でも無数の言い回しがあるが、このような我々が普段話しているような自然言語について、正規表現を用いて様々な言い回しに対応することで、ある程度の表現の差異は吸収させ、正しく目的の構文が出力できるようにすることを目標にしている。

しかし、現状の Wython は「様々な言い回しに対応する」には程遠く、リファレンスで決められた日本語の言い回し通りにプログラムを書かないとエラーが出て正しく動作しないという問題があった。そこで、J2P ではリファレンスを見なくても自然にプログラムを書ける程度にはこれを改良し、本当の意味で「様々な言い回しに対応する言語仕様」を目指したいと考えた。

(2) 実行環境

2 つ目の特徴は、Wython が CUI ベースのソフトウェアであるという点である。現状の Wython は、テキストファイルに書き込んだ日本語プログラムをコマンドプロンプト上で Wython.py に引数として取り込ませると内部で変換処理がなされて変換結果のファイルを新しく出力するといった仕様である。この動作には Python を使用しているため、Wython を実行するためにはコンピュータに Python を導入しなければならず、自宅などで気軽に予習することができないという問題がある。また、CUI は高度なプログラミングをする際には適してはいるが、

Unix の基本的なコマンドを新たに覚えた上でコマンドプロンプト上においてすべての操作を行わなければならず、初心者には不向きであると言える。

そこで、J2P は Unix に馴染みの無い初心者でも簡単に扱えるようにするために、Web サーバー上で動く cgi アプリケーションという形で制作することにした。これにより、利用者は面倒な Python の導入や CUI コマンドを学習する前にプログラミングの学習に専念することができる。

また、cgi アプリケーションにすることより、PC がなくても Web 環境さえ整っていれば iPad やスマートフォンなどを用いて学習が行えるという点も大きな利点となるだろう。

(3) 教材としての仕様

Wython の 3 つ目の特徴は、その教材用ソフトとしての仕様である。現状の Wython は、変換前の日本語プログラムのテキストファイルと、変換後のテキストファイルをそれぞれテキストエディタなどに表示して左右に並べることにより、両者の違いを見比べながら対応する箇所を確認し、プログラミングの流れを把握するといったものである。

実装されている教育機能はこれと日本語プログラムの書き方が記されたリファレンスのみなので、全くの初心者の場合そもそも何から手をつければいいか分からないといった状況になると考えられる。また、その比較の方法に起因して、どことどこが対応しているのかがわかりづらく、Python などのプログラミング言語の学習へ移行する際への足掛かりとするための教材としては不十分であると考えられる。

以上の点を踏まえて、J2P では変換機能だけに力を入れるのではな

く、教材としての機能も大幅に強化し、初心者の足掛かりとして実用的なものを制作したいと考えた。具体的には、変換前と変換後のテキストをブラウザの同一画面上に並べて表示できるようにして、両者の各行を同期させてどことどこが対応しているのかを直感的に理解できるシステムを作りたいと考えた。

また、Wython のように、利用者に初めから自力で日本語プログラムを書いてもらうのではなく、J2P のチュートリアルを兼ねた初歩的プログラミング講座のページを作成し、利用者がそこに記された基本的な日本語プログラム文を J2P で実行しながら学んでいくことで、プログラミングのおおまかな流れと J2P の使い方を同時にマスターできるという仕組みを実装したいと考えた。

(4) 使用制限

Wython では変換範囲を「プログラミング演習」にて使用する構文のみに留めている。これは、Wython が日本語プログラミング言語を目指すのではなく、あくまでもプログラミング初心者用の足掛かり的教材言語を目指しており、ある程度プログラミングの組み方を理解できたら、後は徐々に Python などの本格的なプログラミング言語の学習にシフトしていってもらうといったコンセプトであるからだ。

J2P でもこの考えは踏襲し、変換範囲は「プログラミング演習」にて使用する構文のみに留めることにした。また、今回は Web 上で実行できるアプリケーションを目指すため、上記に加えて Web 上で動く関数のみを変換対象とする。よって、J2P で変換対象となる具体的な構文は、以下の通りとする。

=(代入処理)　+=(追加処理)　計算式　論理式　if　else if

else　while　brake　raw_input(Python 限定)　strlen　date　date_default_timezone_set　改行　コメント　インデント

(5) 変換先言語

現在、Wython は日本語から Python への変換にしか対応していないが、本年度より「プログラミング演習」の講義では Python ではなく PHP を使用することになったため、J2P では Python への変換もサポートしつつも、メインの変換先言語は PHP とすることにした。これに伴い、変換先言語の Python と PHP の切り替え機能を実装したいと考えた。

また、J2P の補助機能として PHP から日本語への変換機能も実装しようと考えた。これにより、教科書などに書かれている例文で意味がよくわからない部分があった場合でも、J2P で日本語に変換する事により即座にその意味を理解することができるだろう。

(6) 仕様と目標のまとめ

以上の仕様と目標をまとめると、

1. 様々な言い回しに対応することを目標にする
2. Web 上で実行可能なアプリケーションにする
3. 教材機能を実装する
4. 変換対象は「プログラミング演習」にて使用する構文のうち、Web 上で実行できるものに限る。
5. 変換先言語は PHP をメインとするが、Python などの他の言語もサポートする。
6. PHP から日本語に変換できる機能も実装する。

となる。以上の目標を、正規表現や CSS、JavaScript などを駆使することで実装していきたい。

4　インターフェイスについて

これから J2P を作成するにあたって、まず初めに利用者が日本語プログラムを入力するための画面を作成した。この部分は利用者が最も多く目にする部分であるため、どこに何があるが分かりやすい構造にする必要がある。そこで、J2P のインターフェイスを (資料　図 1) のようなレイアウトにて作成した。

ここでは、入力フォーム (資料　図 1-1) の背景色を灰色に設定する事により、変換結果表示エリア (資料　図 1-2) との区別化を図っている。変換結果表示エリアには左側に入力した日本語プログラムが、右側には変換後のプログラムが表示されるようになっている。

更に、プログラム実行領域 (資料　図 1-3) も搭載した。これにより、利用者は変換したプログラムが動作するか We ｂ上ですぐに確認できるようになっている。これに伴いセキュリティ関連の問題も発生したが、その対策については後ほど記そうと思う。

また、(資料　図 1-4) のプルダウンメニューにおいて言語を選ぶことにより、変換元言語と変換先言語を設定することができる。現時点では、変換元言語に「日本語」と「PHP」、変換先言語には「日本語」「PHP」「Python」が選択可能になっている。

5　変換プログラムについて

J2P では、利用者が入力した文章の各行において、変換のための一連の処理を一行毎に繰り返し、上の行から最後の行までを順番に変換し

ていく仕組みになっている。

その際の J2P における変換プログラムの仕組みについては、大別して 3 つのセクションに分類することが出来る。1 つ目は、入力されたデータの中から変換対象の構文を見つけ出すための正規表現を設定する「正規表現セクション」。2 つ目は、見つけ出した変換対象文を、実際に PHP などのプログラミング文に変換する「変換セクション」。3 つ目は、表示されたデータを表示する「表示セクション」である。

これらのセクションを連動して動かすことで、日本語をプログラミング言語に変換して表示することが出来る。具体的な例として、「○○を表示する」という文章を「print ○○;」に変換するための J2P のコードが資料編 (資料　図 2) である。

では、上記の例に則って、各セクションにおける処理の解説を以下に述べようと思う。

(1) 正規表現セクション

正規表現セクションでの目的は、利用者が入力したデータの中から、変換対象となる文字列を見つけ出すための正規表現を設定することである。ここで設定した正規表現を、次の変換セクションにおいて参照することで、日本語をプログラム文に変換することができるようになる。

ここで注意しなければならないのは、利用者がどのような言い回しを使っても目的の変換結果が得られるような正規表現を定義することである。

（資料　図 2-1）は print 文における正規表現であるが、print 文を日本語で表すとするならば「○○を表示する」となる。これにマッチングする文字列を探すために、ここでは 2 種類の正規表現を定義している。

1 番目の正規表現では、「○○を表示する」の○○の部分が計算に使う記号、若しくは英数字の場合に使用する正規表現である。この正規表現により、「1+4 を表示する」や「$test を表示する」といった記述に対応している。また、接続詞について「と」と「を」のどちらでもマッチするようにし、更には print 文の部分に該当する所を「プリント」及び「表示」とすることで、「何らかの値を表示する」という日本語の表現について考えられるもの全てにマッチングするように定義している。

また、「○○を表示する」と書いた際の○○の部分が値ではなく文字列、つまり括弧やクォーテーションに囲われたものだった場合については、2 番目の正規表現定義において括弧若しくはクォーテーションに囲われた全ての文字列にマッチングするように設定し、日本語などの表示にも対応している。

このような配慮を、print 文だけではなく J2P が変換に対応する全ての構文に施す事により、細かい言い回しの違いは全て吸収できるようにするという目標を達成している。

(2) 変換セクション

変換セクションは、正規表現セクションで設定したものとマッチングした部分があれば、対応する日本語のプログラム文と変換後のプログラム文をそれぞれ別々の変数に格納するという流れで動いている。

(資料　図 2-2) の print 文の例では、正規表現セクションで設定した「文字列かどうか」の判定を行っている。まず、文字列の判定をして、マッチしたならば文字列用の変換プログラムを走らせた後に phan という変数に 1 を代入して文字列にマッチングしたことを記している。もし、phan が 0 のまま print 文にマッチングしたならば、その文は値を

表示しようとしているということなので値用の変換プログラムを実行する、という処理を行っている。

(3) 表示セクション

ここでは変換セクションにおいて得られた結果を、利用者にわかりやすい形で表示することを目的としている。そのため、変換前の日本語プログラム文と、変換後のプログラム文を並べて表示することにより、可読性を向上させる事にした。

また、変換前のデータと変換後のデータの各単語の上にマウスカーソルを乗せることにより、両者の対応する部分の背景色が変わり、どことどこが対応しているのかがはっきりと分かるようにした (資料　図 3)。尚、この背景色については、赤＝文字列、紫＝条件分岐、緑＝論理式、青＝代入処理、ピンク＝関数、グレー＝空白として、区別するようにした。

この動作を実現するためには Javascript が必要であり、変換セクションにおいて変換前のデータと変換後のデータを格納する際に、それぞれ対応する部分に Script のタグを挿入するように修正 (資料　図 4) することで、両者のマウスオーバーによる背景色連動を可能にしている。

(4) 工夫した点

このような流れで J2P は日本語をプログラミング文に変換して表示している。ここでは print 文を例に説明したが、J2P では print 文以外の全ての構文においても、考えられる限りの日本語での言い回しなどを考慮し、正規表現をうまく設定しながら「様々な言い回しに対応する」ように組み上げている。

しかし、日本語の言い回しはあまりにも膨大なので、上記の「文字列かどうかの判定」のように、どうしても正規表現だけでは吸収しきれない言い回しなどの問題が出現してしまう。これについては変換セクションにおいてその問題を吸収できるよう上手く処理をすることにした。ここでは正規表現だけでは吸収できなかった問題のなかでも対処が難しかったものと、その解決方法について述べていこうと思う。

(i) エラー処理

まず初めに必要だったのが、エラー処理である。人間が入力するデータにミスが起こるのは当たり前のことで、このミスに対応できずに途中でアプリが止まってしまう事態になれば、それは「様々な言い回しに対応する」プログラムであるとは言えないだろう。そこで、J2P ではエラーの処理に特に力を注ぐことにした。具体的には、変換用セクションの最後に、変換対象の行の中に正規表現とマッチングする部分があるのかを判別し、マッチングしていなかった場合のみ、その行をコメント化して出力するようにしている。

(ii) 代入処理

代入を日本語で表現しようとすると、「○○に△△を入れる」「△△を○○に代入」など、意味は同じでも値の順番が前後する可能性があることがわかる。この差を吸収するために、J2P では「○○ (に｜へ)」と「△△ (を｜と)」をサーチして、「に｜へ」及びその直前にヒットした文字○○を常に左側に配置し、「を｜と」及びその直前にヒットした文字△△を常に右側に配置している。これにより、「△△を○○に代入」と書かれていても、自動的に「○○に△△を代入」となるようにして記述の順番の差を吸収しており、どちらの言い回しを使ったとしても目的の動作が行われるように工夫している。

更に、「△△を代入する」の△△の部分が値ではなく、「12 足す 1 を $keisan に代入」のような計算式である場合も想定し、△△の部分が計算式かどうかもサーチした上で、もし計算式ならば代入処理ルーチンの内部で計算式用のものに条件分岐をして、「$keisan = 12+1;」のように変換できるよう対応している。

（iii）print 文

print 文では、文字列を表示する場合と、変数や値を表示する場合の二種類が考えられる。この二種類を判別するために、表示対象となる文字列の前後に括弧があれば文字列、なければ変数として扱うことにしている。そのため、変数においては「＄」が付かないアルファベットのみの文字列でも変数と認識するようになっている。しかし、Python においては$が付かない英数字も変数として認識するのだが、PHP においては$が付かない変数を使ったまま動かそうとすると当然エラーを吐いてしまう。これについては、「変数の頭には$を付ける」という習慣を利用者に学ばせるという意図も込めて、利用者に手動で変数の頭に$を付けてもらう事を J2P の仕様とすることにした。

（iv）if 文・While 文・論理式

if 文、及び論理式については、上記の他の構文とは比較にならないほど様々な言い回しや書き方が存在し、実装に大変苦労した部分であった。具体的には、「もし、○○が△△と同じならば」と「もし、○○が△△ならば」そして、「もし、○○＝＝△△ならば」などに加えて、○○の部分が文字列の場合と△△の部分が文字列の場合、そのどちらでもない場合などと、同じ結果を示すものでも複数の書き方が存在し、且つそれらが「同じ、同じでない、以上、以下、より大きい、より小さい」などの全ての論理式において適応される。更に、これらから派生する「も

し、○○が△△と同じならば、繰り返す」という While 文にも対応するとなると、その組み合わせは実に 100 通りを超えることになる。

これらの言い回しに対応するために、まずは「もし」や「そうでなくもし」などの条件式の部分のみにマッチングするように正規表現を設定した。論理式の部分の言い回しは多種多様だが、その外の部分は、if 文と else if 文、else 文 While 文の 4 通りしかないためである。そして、この 4 通りの条件式について、「もし、{ 論理式 } ならば」や「そうでなくもし、{ 論理式 } ならば」などといった論理式の部分を後からはめ込むといった方法を取ることで、その工程数を削減した。

論理式の部分は、「△△が□□と同じ」という日本語での指定方法と「△△＝□□」という記号での指定方法の両方に対応するために、まずは「同じ、異なる、以上、以下」などの日本語での論理式指定方法かどうかを判別し、もしそうならば日本語用の処理群に移動し、そうでなければ記号用の処理群に移動する機構を実装した。

また、「＝＝」と「！＝」の論理式に限り、「○○＝＝△△」の△△の部分が文字列の可能性があるので、これにも対処した。その方法としては、△△に該当する部分の一番初めの文字が括弧の開始部分であり、尚且つ△△の一番最後の文字が閉じ括弧ならば、△△が文字列であることを示す変数「mhan」に 1 を代入し、これを変換の際に判別して「○○＝＝ ”△△ ”」と、文字列用の変換結果を返すようにした。また、本来望ましくないのだが、○○の部分が文字列である可能性も捨てきれないので、こちらの部分にも△△と同じ方法で対処し、変換の際に、どちらも文字列の場合、○○のみが文字列の場合、△△のみが文字列な場合、どちらも文字列でない場合と、処理を分岐している。

その他日本語用の「以上、以下、より大きい、より小さい」などの論

理式については、それぞれ上から順番に判別し、マッチしたものを変換するといった方法をとった。これらについては、○○と△△のどちらもが文字列ではないはずなため、「＝＝」で実装したような文字列判定機構は実装していない。

また、「＝＝」についてはひとつ特殊な日本語の言い回しが存在し、これについては上記の論理式とは別に処理を実装することにした。その言い回しとは、「もし、○○が△△ならば」というものである。この言い回しでは、「○○と△△が同じ場合」を表したいものだと、我々人間は理解することができるが、実際の所「同じ、以上、以下」などの最も肝心なところが省略されているため、論理式に該当する日本語をサーチする方法ではどうしても判別ができない。「ならば」の部分をサーチするにしても、「ならば」は「違うならば」「大きいならば」など、他の言い回しでも使用する可能性があるので区別化できず、うまくいかないのだ。しかし、この特殊な言い回しは幸いにも「もし、○○が△△ならば」の 1 通りだけなので、この言い回し専用の正規表現を定義し条件分岐処理の最後でマッチングさせることで解決することができた。

変換した後はこれらの論理式の部分を格納する変数「ronri」に代入し、条件式の部分に挿入している。また、論理式の部分が最初から記号だった場合については、論理式の部分の記号を半角英数字に変換した後に変数「ronri」に代入し、条件式の部分の挿入する方法を取った。

尚、条件分岐の変換時の仕様について、条件分岐にマッチングすれば「if (○○) {」のように始め中括弧までが自動で挿入されるようになっているが、処理を終了する際に必要な終わり中括弧までは自動で入力されないため、利用者が自ら入力する必要がある。これは、条件分岐式において「条件文　(論理式) { 実行したい処理 }」という、実行したい部分

は中括弧で囲うという習慣をつけてもらうために、J2P の仕様とすることにした。

しかし、開始中括弧が自動で挿入され、終わり中括弧が自動で挿入されないという仕様では利用者が中括弧を全く使用せずに変換を行ってプログラムを実行しようとした際には、閉じ中括弧が欠如していることが原因でエラーが出るだろう。これを回避するために、変換セクションの最後の部分で中括弧の数をカウントし、始め中括弧と終わり中括弧の数が一致しなかった場合は「{ } の数が一致しません」と表示し、プログラム実行領域を非表示にすることで、利用者に括弧の数が一致していない事を知らせる機能を搭載した (資料　図 5)。これにより、利用者は中括弧の数をしっかりと合わせる習慣がつき、初心者がよく引き起こす「括弧の不一致」によるエラーについて耐性ができると考えられる。

(5) プログラミング言語から日本語への変換

プログラミング言語は日本語とは違い、言い回しは数種類しか存在しない。よって、プログラミング言語から日本語への変換は正規表現の設定のみで可能なものが多く、比較的容易に実装することができた。ここでは、変換元プルダウンメニューにおいて日本語が設定されていた場合は日本語用の変換ルーチンへ、そうでなければプログラミング用の変換ルーチンが動くシステムになっている。

(6) セキュリティ上の対策

J2P には変換したプログラムを Web 上で実行できるといった機能があるが、これは非常に危険な処理でもある。もし、利用者が悪意あるプログラムを Web 上で実行させようとした場合、こちらが予期できない

様々な被害を被ってしまう可能性があるからだ。J2P ではこうした被害を最小限に抑えるために、危険を及ぼす可能性のある関数は使用できないように対策を施した。具体的な方法としては、変換結果を表示する直前に、「英数字（」という文字列をサーチし、ヒットした場合はプログラム実行領域を非表示にするというものである。

しかし、これでは文字列の中身についても危険関数除去が働いてしまい、「”strlen()”と表示する」のように、関数を文字列として表示させたい場合にも、エラーが出るようになってしまう。

そこで、危険関数除去部分には、ホワイトリストを設定できるようにした。これにより、J2P 側で指定した関数以外の関数を除去することが可能となる。現在は、プログラミング演習にて使う関数のみをホワイトリスト化している。今後、ホワイトリストに登録している関数以外のものを使用する必要が出てきた場合、危険なものではないかを十分吟味した上で、関数をホワイトリストに追加するといいだろう。

尚、利用者が悪意なく使用できない関数を使用してしまった場合を考えて、どの部分が危険関数と認識されたのかが分かるシステムも搭載している（資料　図 6)。

6　教材システムについて

教材システムについては、利用者が一番初めに J2P を触った際に、何をすれば良いかわからないといった状況を回避するために、チュートリアルの意味も込めて制作した。これに伴い、チュートリアル用の文章と J2P 本体のプログラムは同じウィンドウ内に収まっていたほうが分かりやすくなると考え、インターフェイスに改良を加えた。（資料　図 7）

右メニュー（資料　図 7-1）をクリックすると、jQuery を用いたアニ

メーションと共に、クリックしたメニュー内容に応じた内容物が右側から現れるようになっている（資料　図 8)。この際、変換システム部分のサイズは 50 %に縮小されるようになっており、画面上の半分が変換システム、右側が教材システムといったレイアウトになるようにしている。また、メニュー部分を再度クリックすることにより、教材システム部分はいつでも格納できるようになっている。

メニューの項目には「Top」「Help」「Lesson」の 3 種類を用意している。ここでは、この 3 種類について述べようと思う

(1) Top メニュー

Top メニューでは、J2P の使い方とメニューの説明を書いている（資料　図 8)。J2P に初めて触る人はここを見ることにより、J2P で何ができるか、どう使うかをひと通り知ることができるようになっている。

(2) Help メニュー

ここでは、J2P で使える構文の一覧、及びリファレンスを見ることができる（資料　図 9)。J2P はこういったリファレンスを見なくとも動作するように設計しているが、万が一変換できない言い回しを使用する利用者が居た場合には、このリファレンスを参照することで、どのような人でも変換が行えるように対処している。

(3) Lesson メニュー

Lesson は、チュートリアルを兼ねたプログラミング初歩講座である（資料　図 10)。「文字の表示方法」、「変数の使い方」、「条件分岐の方法」、「繰り返し文の使い方」、「関数の使い方」の 5 ステップに分かれており、上から順番に履修していくことで、プログラミングの基礎と J2P

の使い方をマスターできるようになっている。

各ステップにおいては、黒い背景色のボックスの部分に J2P の日本語構文例を表示している。利用者はこれを左側の変換プログラム部分に書き込み変換することで、各ステップに応じた結果を得られるようになっている。こうして、lesson の文章を読みながら J2P を使って利用者自らが日本語でプログラミングをしていくことで、利用者がプログラミングの流れについて十分に理解できるように配慮している。

(4) 工夫した点

(i) Cookie

右側の教材を見ながら左側の変換プログラムを動かしていくといった性質上、変換の度にページが更新され、ページやスクロール位置が一番初めの状態に戻されてしまう。このままだと、利用者は変換ボタンを押す度に今まで見ていたページを開き直さなければならず、スムーズな学習ができない。そこで、Cookie を利用して、変換の際に利用者が最後に見ていたページを保存し、更新後、保存した Cookie に基づいてページを元の状態に開き直すという仕組みを実装した。

(ii) Step

「Lesson」メニューを開いた時のみ、教材システム上部に Step 数が表示されるようになっている (資料　図 10-1)。この部分をクリックすることにより、各ステップに対応した部分に自動的にスクロールして、現在表示しているステップの色が黒くなるようになっている (資料　図 11)。

また、マウススクロールにて下に読み進めた場合にも、現在表示しているステップの位置が黒くなり、どこを読んでいるのかが分かるように

なっている。

尚、このようなページ内リンクやスクロール関連の処理も jQuery を用いて行っている。その理由として、変換プログラム本体では「overflow:hidden」を用いてスクロールバーを表示させない仕様にしているのだが、これを用いていると通常のページ内リンクではリンク位置がズレてしまうという問題が発生したためである。これを回避するために、各ステップの位置を jQuery の関数を用いて取得し、各変数に代入した後に上記の Cookie を適用したページ状態にするなどとして運用している。こうすることで、どのようなウィンドウサイズの場合でもページ内リンクの位置がずれるという問題は起こらなくなる。

しかし、ページ更新後にウィンドウサイズを変更した場合や、PC 実行速度などの問題により、稀に本来のステップの位置とは違う場所にアンカーが設置される状態になることがある。これについては、プログラムの処理的に解決する方法が見つからなかった。よって、こうした不具合が起こった場合は利用者に変換ボタンを押してもらうことで正しい位置にアンカーが設置されるので、この動作を利用者に促すための注意書きを「Help」に記載している。

7　アンケートに基づく評価

上記のように J2P を制作していったが、ここでは他の人が J2P を使用した際にどう感じるかを知るためにアンケートを実施した。アンケートの項目は、「インターフェイスについて」「変換プログラムについて」「『Lesson』と『Help』について」の 3 つに分け、その中で各質問を行った。実施対象者は、4 回生 3 人、3 回生 4 人、その他 3 人の計 10 人である。

(1) インターフェイスについて

ここでは、「サイトデザインは見やすいか」「操作方法はわかりやすいか」という二つの点について質問した。

(i) サイトデザインは見やすいか

ここでは、10 人中 10 人全員から「シンプルで見やすい」などといった好意的な意見を頂くことができた。

しかし、その中の 2 人からは「メニューがどこにあるか一瞬わからなかった」などといった、メニューの視認性関連の問題点を指摘された。この点を踏まえて、J2P を初めて開いた際には TOP メニューが既に開いた状態にしておき、J2P の使い方が書かれた画面を初めに読んでもらうことで、解決した。

(ii) 操作方法はわかりやすいか

10 人中 8 人から「直感的でわかりやすい」などといった評価を得ることができた。しかし、中には「日本語を入力するエリアがどこかわかりにくかった」といった意見もあったので、Top 画面における J2P 使用方法の該当する部分を強調することで対処した。

(2) 変換プログラムについて

ここでは、「変換プログラムにエラーはでなかったか」「変換は思い通りにできたか」という 2 点について質問した。

(i) 変換プログラムにエラーはでなかったか

10 人全員から、変換プログラム自体にはエラーはなかったといった回答を得ることができた。

しかし、10 人中 4 人から、「稀に変換ボタンを押した際に、テキストエリアが初期化されデータも送れない」という意見を頂いた。これにつ

いて、確認するために実際に操作を行ってみると、確かに稀にだがこの現象が起こることを確認した。しかし、何度かテストを行った結果「大谷大学内の PC で、Web 上にアップロードした J2P を実行」した際にのみこの現象が起こることが判明した。「自宅の PC から Web 上の J2P を実行」した際や、「大谷大学内の PC で PC 内の仮想サーバーに設置した J2P を実行」した際にはこの現象は現れなかったのである。

上記の理由から、この現象は大谷大学のサーバーの設定に何か問題があるため発生する不具合であると推測した。これを解決するためには、大谷大学のサーバー設定自体を見なおさなければならず、私の手の及ぶ範囲ではないので、放置せざるを得ない状態となってしまった。現状では PC 内部に J2P を設置して実行することが唯一の対処法である。しかし、この現象は毎回発生するわけではないので、ある程度の初期化に目をつぶれば、Web 上で J2P を実行することも可能であると言える。

(ii) 変換は思い通りにできたか

「もし、○○ではない場合」といった処理を行いたい際に、「もし、○○じゃない場合」という表現があり、アンケート実施当時はこれをサポートしていなかったため、うまくいかなかったという意見を 1 人から頂いた。こちらについては、既に「〜じゃない」といった言い回しをしても動作するように修正している。

残りの 9 人からは「思い通りに変換できた」といった回答を得ることができた。更に、「$や { } を全角で記入した際に自動的に半角に変換できて便利だった」という意見も頂くことができた。

(3) 『Lesson』と『Help』について

ここでは、「説明はわかりやすいですか」という 1 点について質問した。

(i) 説明はわかりやすいですか

10 人中 10 人全員から「わかりやすい」といった回答を得ることができた。更に、「結果と自分の結果を見比べることができたので、間違いが明白だった」、「実行内容と結果が色分けされているのでわかりやすかった」などといった意見を頂くことができた。

8 総評

変換プログラムについては、当初の Wython のプログラムをほぼ全て書き換える形になってしまったが、アンケートの結果から「様々な言い回しに対応する」という目標や、その他に目標として掲げていた 5 つの項目についても、全て達成できたと考えている。

教材プログラムについては、ページ内リンク関連の不安は残るものの、「利用者が本格的なプログラミングを学び始める前の足掛かりとして実用的なものを作る」という目標はほぼ達成できたと考えている。アンケートの結果も上々で、J2P を利用することで、現状の「意味が分からない英文をコピー&ペーストしてプログラムを作る」という学生を減らすことができるだろう。

9 今後の課題

変換システムについては未だにテスターの総数が少なく、今回の「～じゃない場合」のように予期せぬ言い回しを使用する利用者は他にも大勢いると考えられるため、実際に運用しながら更なる調整を実施してい

く必要性があるように感じた。

また、今回は「日本語から Python と PHP への変換」と「PHP から日本語への変換」は実装することができたが、Python と PHP 以外のプログラミング言語への変換や Python から PHP への変換などと、プログラミング言語間の変換も実装することができれば「初心者の足掛かり」だけには留まらない様々な層のプログラマーに役立つものになる可能性を秘めていると感じた。

10　おわりに

今回、「人の役に立つものを作る」というテーマで J2P を制作した。制作にあたっては「初心者にわかりやすいものを作る」という事を重要視していたが、わかりやすいものを作ろうとすればするほど制作者側の苦労は大きくなっていくことを痛感した。しかし、その苦労を乗り越え、限りなくわかりやすいツールを作る事こそが、我々人文情報学科が目標に掲げている「文系と理系の橋渡しをする」ということなのだと思う。

「今後の課題」に記した通り、J2P には未だ多くの可能性が残されている。今後、この J2P が改良され続けて、より良いアプリケーションになって多くの人の役に立つものとなってくれることを切に願う。

注

（1）http://www.scripts-lab.co.jp/mind/whatsmind.html

（2）http://kujirahand.com/himawari/

（3）http://nadesi.com/top/

（4）http://tts.utopiat.net

文献表

大谷大学　人文情報学科『正規表現オンライン教材』
http://tibet.que.ne.jp/otani/RE/index.html

semooh.jp『jQuery 日本語リファレンス』
http://semooh.jp/jquery/

The PHP Group『PHP マニュアル』
http://www.php.net/manual/ja/

Python Software Foundation『Python 言語リファレンス』
http://docs.python.jp/2.7/reference/index.html

HTMQ『HTML クイックリファレンス』
http://www.htmq.com

banban『WEB for beginner 作成支援』
http://www.scollabo.com/banban/index.html