擬似コードを使ったPyhon入門用教材の作成

竹内 いずみ

2006年1月11日

目　次

1　序論

(1) はじめに

我々福田ゼミのテーマは「人の役に立つものを作る」であり、主に人文情報学生にとって役に立つツールを作成することが目的である。

私の制作においての主題は、は「プログラミング初心者のためのPython の教材の作成」である。これは『プログラムはどのようにして構築されていくのか』を実践的に教えていくことでそこからプログラミングに慣れ、Python への理解を深めることを目標とした教材である。そのため、Python をどう使ったらプログラムが出来上がっていくのか、またプログラムを作るため Python で用意されている関数をどう使い組み立てていけばよいかといったアルゴリズムを、実際の Python のコードを用いることで理解させていき、プログラムに慣れさせることを目指している。

その具体的手段として擬似コードを用いる。

私がこの論題に取り組もうと考えたきっかけは、私自身が全くのプログラム初心者であったために授業で習うプログラミングになかなか馴染めず、プログラミングが苦手だと感じていたためだ。

プログラミングを学んで私が感じたことだが、いくら Python という教材に適したプログラミング言語があっても、私のようなプログラム自体に全く馴染みがない初心者にとっては、苦手意識も手伝って積極的に学ぼうという姿勢が萎んでいく。

また Python でプログラミングの基礎を学び終えても、いざ Python を使って何かしらプログラムを作ろうとしても、『Python で何を作れ

るか』という応用が考えつかない。そのため Python のプログラムの例を載せた教材で勉強しようにも、現在日本語訳されている Python の教材自体少なく、さらにそのプログラム例が少ないため勉強が困難だった。そこで Python の大きいプログラムが実際にどのように書かれているのかを見せプログラムを細かく解説した教材を作成したら、私のようにプログラムが苦手だと感じている人にとって役に立つ教材となると考え、制作することにした。しかし、ただ大きなプログラムを載せただけでは、初心者には解りづらいだけで学習しにくいといった問題が出てくると考えられた。その問題を解消するため、プログラムについての擬似コードを見せることを思いついた。擬似コードは以前授業で使用した際、プログラミングが苦手だった私でも解りやすいと感じることができた。そこから、初心者用の教材に適していると考え、実際に教材に用いることにした。

また、何故 Python がこんなにも『苦手』だと感じるのか、擬似コードを用いることで理解しやすくなったと感じた理由は何かと考えた結果、プログラムを組み立てる際、一番苦労したのがプログラムの構成を考えることだったため、私が学習する上でアルゴリズムからプログラムを考えていくことができなかったということと、プログラミング自体に慣れていないという 2 つの問題が考えられた。

しかしいざアルゴリズムを学ぼうとしても、アルゴリズムを教える方法としてプログラミング言語を用いる教材のその大半が、C 言語を用いたものであり、Python を例として使用した教材は見つけることができなかった。また先に述べたように、プログラムに慣れるために Python コードを参照しようとしても Python の教材やそこに書かれたようなコード例は少ないため、慣れるための学習がなかなかできない。

よって、この２つの問題を重視した教材を作れば、初心者にとって解りやすく学習できる教材が作成できるのではないかと考えた。さらにPython は、その様々な特性や経緯から教材として使用するのに適しているといわれている。この Python の教材を作ることで、プログラミングを習ったけれども途中で躓いた人やプログラミングを習うのが初めての人が、プログラムに親しみ理解する足がかりとなることができるのではないだろうか。

その上で、この教材は人文情報学生を対象とし、中でも Python を学ぶプログラミング初心者を対象とした。そのためこの教材に使用した Python は、日本語環境用インストーラー (Win32) の 2.3.4 バージョンを使用するようにし、初心者が Python を動かせることを前提とした。(1)

(2) Python とは

Python はプログラミング言語の中でも、Java や Perl と同じようなオブジェクト指向プログラミング言語に属している。この「オブジェクト指向」とは比較的新しい概念のもと作られたプログラミング言語であり、これまでの C 言語などのような手続き指向の言語より人間に理解しやすいよう作られている。これはオブジェクト指向言語が私たちの身の回りにあるような「モノ」、つまりオブジェクトを中心に考えられているからである。そのためこのオブジェクト指向言語は、現実社会にあるようなものを、プログラムとして表現することが可能だと言われている。逆に言えば、プログラムを現実社会のものに即して作ることができるということだ。つまりオブジェクト指向言語は、身の回りのものをモデリングできることから人間にとって理解しやすいという特徴があるの

だ。また、新たに「モノ」を作る拡張性の高さやその再利用ができるということもオブジェクト指向言語の特徴である。

Python は Guido van Rossum 氏が作成したプログラミング言語で、van Rossum 氏が好んで見ていた BBC テレビの「空飛ぶモンティ・パイソン」から名づけられた。

この Python の大きな特徴は、オブジェクト指向言語としてのしっかりした文法と独自の字下げによる制御構成を持つことで、さらに人間にとって書きやすく、また見やすいプログラム言語となっていることだ。そのうえコンパイルなどが必要なく、手間をかけることなく実行できるという特徴や、さらに全て無償で入手できるだけでなくその配布も自由にでき、多くの OS 上で動作可能という特徴も持っている。

これらの特徴が、一般に Python がプログラム初心者の教育用に最適であると言われる所以であり、また私が擬似コードに直しやすいプログラミング言語であると考える理由でもある。

しかしこういった多くの利点があっても、日本では Python というプログラミング言語の立場はマイナーなものである。これは和訳された解説書が少ない、Python の日本語対応版がリリースされたのが 2003 年とごく最近であったこと、などが理由として考えられる。一方欧米では、Python は Perl に次ぐ人気となっており Python は主流言語の 1 つとして数えられるほどである。

(3) 擬似コードとは

プログラミング言語を用いてプログラムを作る際、『問題の分析』→『用いられるデータの収集』→『アルゴリズムの構築』→『プログラミン

グ言語での記述』という流れをたどる。

例えばごく単純な足し算を行うプログラムを作成する場合でも、まずこのプログラムは１つの数字特定の数まで加算していく問題である、と分析する。

次にこの足し算を行うには１つの数字と、足していくもう１つの数字と、あと加算される限度数が必要だと分かる。これら材料となるデータは加算対象の「1」と加算していく「1」、加算限度数の「10」となる。

計算方法はこのデータをしようしていくことになる。まず加算対象の「1」の入った変数に加算していく「1」を足す。この結果を変数に代入しておき、それを書き出す、という動作を結果が「10」になるまで繰り返すようにする。つまり加算対象の入った変数が 10 より小さいあいだ、結果を書き出すプログラムとなる。この処理の流れがプログラムのアルゴリズムである。

そしてこのアルゴリズムに当て嵌まるプログラミング言語は「while～：」というループの構文を使用したらいいので、その構文に当て嵌めて実際にプログラムを作成していくことになる。

このように、アルゴリズムとはある特定の目的を達成するための手順を明確に述べたもののことであり、この手順を実行すれば誰にでも同じ結果が返される手段のことである。

一方擬似コードとは、この「アルゴリズムの構築」手順をプログラミング言語に起こす前に、より理解しやすい自然語で記述した命令と、それに簡単な構文規則を持ったせたもののことである。

擬似コードは、その人ごとに作成されるプログラムコードが全く同じではないように、記述する人ごとにそのコードもまた違う。一般的に、

擬似コードの構文規則は使用するプログラミング言語に準じている。以下の Python と C 言語を例にとったプログラムと擬似コードの 2 つの表を比べて分かるとおり、プログラミング言語の中には自然語で記述するには不自然な構文などがある。そのため完全に構文に沿って記述することはできず、また公式に定義された擬似コードの構文も存在しないことから、それらは記述する作成者ごとに違ったものとなると考えられる。

しかしこの擬似コードの不統一性は、自然語で書かれるがために、作成した擬似コードの命令と構文規則が見る人に共通して理解されてさえいれば誰にでもその擬似コードを読むことが可能である、という拡張性の高さに繋がる。

つまり Python のように構文規則が自然語に近く単純なものであれば、C言語のように処理を記号でまとめることで簡略化したものより擬似コードに直すとき自然な言葉に直しやすくなる。ここから、擬似コードを大勢に理解しやすくするには、プログラムコードと自然語どちらにも偏りすぎないような明確な構成規則と単純な構文を備えたものだと考えられる。

Python コードに対する擬似コードの例

Python コード	擬似コード
a = 1	変数 a に 1 を代入
while a < 10：	変数 a が 10 より小さいあいだ：
a += 1	変数 a に 1 を足し代入し直す
print a	変数 a を書き出す

C 言語のコードに対する擬似コードの例

C コード	擬似コード

int a = 1；
while (a ＜ 10) {
　　a++；
　　printf(“%d”, a)；
 	
}
	整数の変数 a に 1 を代入
	(変数 a が 10 より小さい) あいだ {
	　　変数 a に 1 を足し代入し直す
	　　(整数の変数 a を文字列フォーマットで) 書き出す
	}

(4) wython とは

擬似コードは自然語で書かれるために、使用する範囲のみの人が読めればよくため厳密に定義されるものではなかった。

一方、擬似コードをプログラミング言語として使用する『wython』が開発されている。これはプログラミング言語であるため、普通の擬似コードとは違いより厳密にコードが定義されている。

wython は人文情報学科の講師である福田洋一先生が、２００６年現在開発されているプログラミング初心者のための入門用言語である。

そのため特定の言語に煩わされることなく、初心者に基本概念や考え方を理解させるために日本語のプログラミング言語となっている。しかしプログラミング言語といっても wython は日本語で書かれたコードを Python に置換してくものので、正確にはコンパイラではなく翻訳ソフトであるとされている。

この翻訳には、Python の各関数などに対して定義された擬似コードを使用しそれに沿って置換していくことで行うが、その定義外の日本語は無視されるようになっている。つまり分かりやすいよう冗長な説明を付け加えたとしても、そのまま必要なプログラムだけ実行され、不必要

な説明は Python コードに変換されることはない。そしてこのプログラムの実行に不必要な説明は、プログラムを理解するための擬似コードとして利用することができるようになっている。

さらに wython は、Python のコードを翻訳しているためスムーズに wython から Python へと移行することができるという利点がある。それによりプログラミング言語の教材としてまず wython を使用し、そこから Python の教育へと発展していけることになるということが期待される。

現在は、問題を擬似コードで記載し、そこから翻訳のための wython コードを決めてるところであり、実際に試験的な wython のプログラムも作成段階に入っているそうである。

福田先生はサイト『チベットとコンピュータな日々』[(2)]で wython の制作過程を公開されておられる。そこで今後はより日本語を柔軟にし、できるだけ覚えることを少なくした日本語の規則を作成すること、Python コードに、wython のコードを注記として書き出すようにすること、また Perl や Ruby のコード書き出すことなどが開発予定としてあげられている。

2　計画

(1) 制作についての構想

まず教材を作るにあたり、初心者にも分かりやすく解説してくことを目標にした。そのため教材は問題と問題の捉え方と解説と擬似コード、その答えとなるプログラム例で 1 つの大きなプログラムの説明ができるような構成にしようと考えた。そのためには、まず解説で「何故その関

数を使うのか」「その関数は実際どんな動きをするのか」「その結果どうなるのか」といったプログラムの内容を、プログラミング用語をあまり使用せずに明記していくことにした。同じように、問題の捉え方では、問題の考え方と何を目的にした問題なのかを明確に提示するようにする。ここで使用するツールを限定していき、対象者が後で示すプログラム例に辿り着くようにしていくことを目的とする。

また、記載するプログラム例はできるだけ大きい規模のものを実際に自分で作成したいと考えた。これは私自身のプログラミングの勉強のためと、実際に作成したプログラムを説明する際、「そのプログラムは何を目的に作成したか」を簡潔に述べられるようにするためである。このプログラム例を作成する規模は、授業で学習した範囲を目標とし簡単な演算からインターネットのページへの操作までとした。問題数は 20 問を目安として作成しておき、そこから問題に適しているもののみを教材に使用することにした。

またプログラム例の擬似コードも記載していくが、それは作成しようと考えているプログラム例の完成と共に構文規則を決めていきたいと考えた。つまり擬似コードの作成手順としては、まずプログラム例を作成し、そこからその擬似コードが自然語として解りやすく且つ簡潔なプログラムコードとなるよう構文規則を決定し、それに沿って擬似コードを作成していくことになる。

以上の構想から、教材を完成させるには Python のプログラム知識はもちろん、解説のためにアルゴリズムを理解することも必要である。

またごくごく簡単なプログラム例から難しいプログラム例まで、段階を踏んだプログラム例を、目安とした数よりできるだけ多く作成する必要がある。そのためにはプログラムで何ができるか、またどんなプログ

ラムがあったら便利か、他の様々なプログラムを例として見て案を出していくようにする。

(2) 計画を絞り込むときの視点

教材に記載するプログラム例は、Python で使用する一般的なステートメントや関数を使う。

その中で解説するステートメントや関数は、組み込み関数、条件分岐、ループ、頻繁に使用するモジュールの関数までとし、主に授業で使用したものなどごく基本的な働きをするものにする。オブジェクト指向言語の利点の 1 つであるクラスであるが、授業では習わなかったためと初心者に教えるにはあまりに難しい内容であるため、この教材では使わないことにする。

またプログラム例は、各問いごとに新しいステートメントや関数を入れていくようにし、作成したプログラム中の関数が重複するものは小さいプログラム例の方を切り捨てるようにする。

さらに実際に教材として見る場合、パソコンの画面越しに見るより紙として見た方が利用者の負担にならないと判断し、テキストとして作る。

この教材を使用する対象者に必要なプログラミング知識は、Python の授業で習う「変数の代入」が理解できること、作成したプログラムを実行できること、基本的な関数の使い方を知っていることとする。つまり人文情報学生の 1 学年前期の授業が終了している、もしくはその程度の知識を持つ人を対象とする。

1 学年の前期で習うおおよその授業は、Python の使用環境の整備と、変数についての学習から始まり、条件判断や繰り返し処理、ファイルの

操作を学習し、また組み込み関数については文字列に関するもの、数値に関するものである。

(3) 作業計画

実際に作成に入るにあたり、立てた作業計画は以下の通りである。

1. 授業で習った範囲の Python を勉強する
2. 資料作成
   (a) 見本となる Python の、できるだけ大きいプログラムを 20 個以上作成する
   (b) そのプログラム例の擬似コードを作成
   (c) そのプログラム例についての説明を書く
   (d) 教材としてまとめ、書き込んでいく
   (e) 教材を完成させる
3. 資料を確認し、さらに改良していく
4. 資料となる教材の最終確認をし、提出

資料で重要としたい、プログラム例を作り、その擬似コードを書いていき、その説明をつけるという過程である。この 3 つを改良していくことにできるだけ時間をかけるようにする。

3　制作物について

(1) 資料の構成

作成した資料は『擬似コードを使った Python 入門用教材』であり、B5 用紙に全 59 ページの教材となっている。

構成は、学ぶ上で重要なツールごとに問題を振り分け、『条件分岐』から『付録』までの 13 章 12 問編成となっている。章の内容は、「問題」「問題把握」「解説」「擬似コード」「プログラム」の 5 節編成となっており、これで 1 問を解説していくようにしている。

さらに教材の冒頭の 1 章『はじめに』でこの教材の目的と対象を説明した。

本編は、『条件分岐』を使った 2 章の問題 1 から始まる。問題 1 では、対話的な入力をする関数とオブジェクトの型、さらにここで使用される組み込み関数と文字列フォーマットの説明に加え条件分岐と簡単なブール値の説明を解説中で行っている。また説明されたステートメントや関数は次の問題からも使用されるようになっている。

3 章では『ループ』を中心にプログラムが作成されており、乱数生成のモジュールと while ステートメント、for ステートメントの説明がされている。ここで for ステートメントはプログラム中で使用されなかったが、while ステートメントと同じ繰り返し処理を行うステートメントであったのでここで説明することにした。実際にプログラム中で使用されたのは次の問題 3 からとなっている。

4 章では『ファイルの操作』を主としたプログラムであり、sys モジュールの引数を受け取るコマンドや、os モジュール、そしてファイルの操作用の関数と for ループを使用したファイルの操作を説明してい

る。ここで使用した sys モジュールの引数の操作は後の問題で頻繁に使用することになる。

5 章ではこれまでの問題を発展させた『ファイルの操作の応用』であり、難易度の高いプログラムとなっている。ここではこれまでの復習も兼ね、引数と条件判定、for ループを使用したファイルの操作を使用し、それについて解説してある。さらに for ループ内での加算方法を説明し、より複雑なファイル操作を解説している

6 章も『ファイルの操作の応用 2』であるが、これはさらに難易度を上げている。ここでは文字列メソッドの説明を加えてある。

次の 7 章からは『関数作成』を主としたプログラムとなっている。関数作成は難しいと考えることが多いので、ごく簡単なプログラム例で複雑な解説を避けた。ここでは関数作成と、複数行に渡って書き出す場合の書き方の説明のみとした。

8 章はもう少し複雑なプログラムを使用した『関数作成の応用』だが、前回に比べ難易度は低いままとした。

次の 9 章は『ディクショナリ』を主とした問題であるが、より難易度を上げたプログラム例を使用している。ここではディクショナリの説明と文字列メソッドの説明をしている。

10 章では『ディクショナリに似たモジュール』をごく簡単なプログラム例から解説している。これは前回のディクショナリのプログラムが難しかったため、初心者が飽きないようにするためと、ディクショナリは次の問題でも使用するがそこでは新たなモジュールを説明するので、ここで慣れさせておく必要があったためである。

次は『正規表現』を中心にした 11 章である。ここでは正規表現とそのモジュールから使用した関数の解説をしている。さらに新たなメソッ

ドの説明も加えている。

本編最後の 12 章は『例外処理』のプログラムを使用し、例外処理と正規表現の新たな関数を解説している。

教材の最後には 13 章の『付録』として Python のツール集を載せた。これは教材のプログラムで使用したステートメントや関数以外でも、頻繁に使用するであろうツールをまとめてある。内容はオブジェクトの型から演算やブール値、リストとディクショナリの組み込み関数、文字列メソッドとファイル処理や条件判定、繰り返し処理に関数作成と正規表現に例外処理、さらにその他のよく使用するモジュールとその関数を簡単な説明付きで表記した。

(2) 擬似コードの定義

擬似コードは福田先生の『wython』を参考に、論文用に簡単な構成と規則を定めたものを使用した。新たに作成した擬似コードの構成は、プログラムに基づいたインデントと、関数やステートメントに基づいた文法の組み合わせとなっている。

以下は作成したステートメントと関数の定義である。

ステートメント・関数	説明
str(〜) int(〜) float(〜)	〜を文字列に直す 〜を整数に直す 〜を実数に直す
if 〜: elif 〜： else：	もし〜ならば もしくは〜ならば それ以外は
while 〜： for 〜 in 〜： break	〜であるあいだ 〜に、〜を、1 行ずつ繰り返し開いて代入 無条件でループから抜ける
def 〜(〜)： return	関数名〜変数名〜で関数作成 結果を返す
input(〜) raw_input(〜)	〜に対し対話的に入力された答えを整数で代入 〜に対し対話的に入力された答えを文字列で代入
print “〜” print ”’…%d…”’　%　〜	“〜” を書き出す ”’…%d…”’ の % 以下を (順番に)〜に 置換して書き出す
> 〜 < 〜 >= 〜 <= 〜	〜より大きい 〜より小さい 〜以上 〜以下

==	同じ
!= ～	～ではない

これ以外にも資料では組み込み関数やモジュールの関数を使用しているが、その場合は上の表に書かれたステートメントや関数の定義を優先させ、そこに関数の説明をできるだけ同じ言葉を使用して当て嵌めるようにした。また、変数に代入されているオブジェクトの型を明記し、何を目的とした変数なのか判るようにした。

擬似コードとして定義したものについては、その形式を必ず用いるようにするが、それ以外の関数や変数については自然語で簡潔に説明を加えるなどして拡張した。

(3) 制作に於いての問題点と解決策

教材を作成していく中で、完成までにいくつか問題があった。

その第 1 として、そもそも制作者本人である私自身が、Python の理解が浅いことが挙げられる。これは Python の勉強をしていく他ないが、ただ教材を読み、与えられた問題を解いていくだけでなく、教材で使用するための問題を考え出しそれについて試行錯誤していったことがプログラムの理解に繋がったと、完成していくプログラムを見て実感した。

しかし Python のプログラムを考え出すことは当初考えていたよりずっと難しく、Python の教材などで問題例を参照しようにもそのプログラム例の少なさや、あったとしてもそのプログラム例の規模の小ささから参考資料の不足に悩まされた。そのため、その時点ではまだプログラムで具体的に何ができるのか考えつかなかった。そのため解決策として、Python の教材だけでなく別のプログラミング言語である Perl の教

材などに記載されている問題も参照して、そこから作成したいプログラムを考えていった。

また福田先生のプログラム演習の授業で使われたプログラムは、規模も大きく関数なども多く使用されていたため、プログラム例としては最適だった。よって、プログラム問題を参考に、変数名やプログラムを改良するなどして使用することにした。しかし 2 章の条件分岐と 9 章のディクショナリ、11 章の正規表現の 3 つのプログラムは、あまり改良を加えることなく載せることになってしまった。

第 2 に、作成した複数のプログラムで使用した関数が重複しているため解説でどれを優先させるか、また資料に載せる際のプログラムの順番などはどうするか混乱してしまうという問題があった。

それを解消するため、各プログラム例について「ファイル名」「問題内容」「使用する関数」をメモとしてまとめ、そこからさらに規模が小さく、簡単な関数を使用しているプログラム順にまとめ直した。一度このメモを基にプログラム例を取捨選択した。選択基準は、目標とした範囲のステートメントや関数が使用されているものを優先し、その中でもできるだけ大きいプログラムであることと、プログラム内で関数を多く使用しているものを残すようにした。

そのプログラムの解説は、教材の中で新しく関数やステートメントが使用された時に詳しく説明するようにし、次に使われた際はごく簡単な説明を付け加えるのみにした。

第 3 に、擬似コードも複数のプログラムに渡り書いていくうち、擬似コードの表現がそのプログラムごとにまちまちになり、全体で見ると逆に解りにくくなってしまった問題がある。これは自然語で記述する際、より解りやすいよう大幅に拡張したためであり、この時点では擬似コー

ドの規則を細かく決定していなかったためだった。

そのため、プログラムの構文を決定しているステートメントだけでなく、よく使う関数についても新たに規則を作ることにした。その際、厳密に定義しすぎて擬似コードの『拡張性の高さ』という特性を無くしてしまわないように、できるだけ自然な日本語として受け取れるように心がけた。

こうした擬似コードの定義の例として、変数の型を挙げる。Python では変数の宣言が必要ないが、変数に代入したデータのオブジェクトの型がそのまま変数に影響する。しかしこれは変数名だけではそれが何を現す変数なのかはっきりと理解し難い場合、代入したオブジェクトの型を説明として加えることでその変数が何を表すのか解りやすくするというメリットとなった。よって、擬似コードでは変数が解り難い場合などにオブジェクトの型を変数名の前に付け加えることにした。

また第 4 の問題では、「問題把握」での説明もまた全体的にまとまりが悪く、解りにくかったというものがある。これは、その「問題把握」の内容を「プログラム実行時の入力方法」「特に注意する点」「簡単な実行処理の流れ」の 3 つに分けて説明するようにした。

そのため、解説ではより解りやすいようこの「簡単な実行処理の流れ」に従い、より詳しく処理の流れを解説していき、どの点の解説をしているのか分かるようにした。

これらの問題を回避するために、複数の教材やインターネットの辞典を参照した。

参考にした資料は、全てプログラミング初心者を対象にしたものである。よって、これらは基礎から分かりやすく記述していくことを重視しており、私自身の勉強に使用できるだけでなく、作成する資料の参考に

することもできると考え、これらの資料を使うことにした。

教材としてはまず [Mark Luts・David Archer 2004] で Python を学び直し、それから [福田洋一 2004] の人文情報学演習の授業で配布された資料を参照した。

擬似コードについては、[アンク 2003] やインターネットの用語辞典、福田先生のサイト『チベットとコンピュータな日々』を参考にして定義した。

問題についても、同じく [福田 洋一 2004] の授業での配布プリントから用いたり、また [Randal L Schwartz 1998] などを参考にした。

(4) 資料の作成を通して分かったこと

資料の作成を通して、改めてアルゴリズムの重要性が解った。

擬似コードについて調べる過程で、アルゴリズムについて調べていた際、私がプログラミングを習って解らないと感じたことのほとんどがこのアルゴリズムだったことに気がついた。始めは私はただ漠然と「プログラミングに慣れる」ことがその理解に繋がると考えていたが実は、これは「プログラミングのアルゴリズムに慣れる」ことだったのだと気づくことができた。そのため資料の解説がすべてプログラムの処理の流れに沿っているのも、アルゴリズムに沿うことで「プログラミングのアルゴリズムに慣れる」ことを狙ったためである。

また、Python を擬似コードで書く際、インデントや構文が明確であるという Python の特徴が、擬似コードに書き直すことにも適していることを実感した。

擬似コードを学ぶため C 言語の擬似コードを載せているサイト(3)な

ども閲覧したが、そのプログラムと擬似コードを見比べると、初心者の立場から率直に見て Python のコードを用いたほうが解りやすいと感じた。

さらにある程度のアルゴリズムを理解していれば、Python は構文規則をほとんど覚える必要がない。そのためプログラムを作成しようとした場合、ただプログラムの材料となるツールやデータを集めて組み立てていくだけで、大まかなプログラムができてしまう、という考えに至った。そこから、Python は初心者用の教材には適しているという実感を得た。

また、以上のような Python のプログラムの構成に対する理解だけでなく、プログラムの作成についても分かったことがある。

プログラムを作成する際に集めるツールは、教材やマニュアルなどから探すことになるが、同じような働きをするツールはたくさんある。この当て嵌めていくツールを選ぶの時は、できるだけ使い慣れているものを使用したほうがいいと考えた。

それは、私は最初出来るだけたくさんの関数を使用しようとしたり、初めて見る関数などを使用してみようとしていた。しかししばらくしてプログラムを見てみると、同じような働きをする別の関数が何なのか分からなくなったりして変に複雑なプログラムとして見てしまうようになったためである。よって、まず自分がよく使う表現を使用していくことのほうがプログラムを作成しやすく解りやすいことに気づいた。

(5) アンケートより

資料を作成していく途中で、一度資料についてのアンケートを人文情報学科の第 3 学年ゼミ生に記入してもらった。アンケートでは、分かり

やすかった点・逆に分かりにくかった点、改良して欲しい点、感想・意見の記述を求めた。

分かりやすかった点では、問題の意図が書かれていたこと、使用する関数の説明があったこと、擬似コードと実際のプログラム例が書かれていたことが挙げられていた。

分かりにくかった点では、解説が長すぎて、初心者には読みづらいという意見が出た。このため解説をより簡潔にし、処理の流れに沿った説明と冗長な部分を削ることで初心者に飽きられない、自分が今どこを学んでいるのか分かる文にした。

さらに改良して欲しい点では、擬似コードとプログラム例を比較しやすいように別ページにしたほうがいいという意見があった。確かにこの時点で作成していた資料は、ただ書き連ねられていただけで非常に見難かったので、この意見の通り擬似コードとプログラム例が比較しやすいように新しいページの先頭から始まるようにした。

このアンケート評価から、アルゴリズムに則った解説をしていくことが、私だけでなく他の人文情報学生にも「解りやすい」と感じられることが分かり非常に参考になった。

今回のアンケート結果から、『初心者にも解りやすいような教材作成』という基本に立ち返る切っ掛けも得ることができ、如何に解りやすい文章や資料構成をしていくかといったことを念頭に捉えながら制作を進めた。

4　結論

(1) 最初の計画との違い

最初の計画では、記載するプログラムの全てを自分で考え出すこととなっていた。しかし当初考えていたよりも作成できたプログラムの数が少なかったため、福田先生の授業で使用されたプログラムを使うことにした。しかしそれでもプログラム数が少なく、また重複するものが多数あったため、それらを省いたプログラム数は結局 16 個だった。また実際に使用したプログラムは 12 個と非常に少ない数になってしまった。

また、プログラム範囲はインターネットのページの操作まで、としたが、このインターネットのページの操作は省き、例外処理までとした。これは、再帰表現を使用した難易度の高いプログラムだったが、プログラムが上手く動作できず、できたとしてもなぜそこで動作したのかの解説ができなかったためであり、私自身の勉強不足が原因である。

さらに当初構想では資料の解説部は、使用したツールの目的と動作の説明、返される結果を解説していくとしていた。しかし実際はツールの説明はその動作と返される結果を説明したのみであり、主に解説されたのはプログラムの処理の流れに沿ったプログラム内容であった。これは、制作途中でツールの解説のみに終始するのではなく、プログラムに慣れるにはアルゴリズムに沿って解説していくほうが解りやすく感じられると考えたためである。

同じように問題把握でも、最初はただ問題の捉え方を解説し、学ばせたい方向へ誘導することが目的であった。それを「問題に対してどう考えていけばよいか」という方法を提示することにした。これも解説の時と同じように対象者をアルゴリズムに沿ってプログラムを考えさせるた

めであり、また問題の意図が明確に提示することができると考えたからである。

また作業計画では、一度資料を完成させたあと論文に取り組み、そこからさらに資料のほうを改良していく、となっていた。しかし実際は、資料が完成せずに期限が迫ってきてしまったため、資料を作成しながら論文を書き始めることになった。さらに資料があまりはかどらず、その結果論文の進行も滞り、改良を重ねることが期日的に困難となってしまった。これは、プログラム例の作成段階で手間取り、大幅な時間を割いてしまったことがその要因だった。プログラム例がある程度揃わなければ擬似コードも作成できず、期日に追われることとなってしまった。

(2) 今後の改良点

資料を作るにあたり、今回勉強しやすいように紙を使用したテキストの形で作成したが、オンラインで作成することも考えた。それは、オンラインだと関数やステートメントの解説が別にでき、また分からない関数の説明についてすぐ呼び出せるという機能を付けることが可能なためだ。そのためオンラインだと解説がより簡潔になり、解りやすくなるのではないかと考えた。オンラインだと擬似コードと Python コードを、工夫により見比べやすいように表示することもできる。

しかしこれには、改良しなければならない点が多々ある。

まず資料に記載したプログラムのいくつかは、あまり実用性もなく興味ももたれないような内容である。そういったプログラムを、より実用性のある、興味をもたれるプログラムとして作成し直す必要がある。また、より解りやすく簡潔なプログラムに作成し直す必要がある。

さらに解説では、より初心者を意識し関数の使い方や特徴などを簡潔

にまとめていきたいと考えている。内容も人文情報学生を対象とし、ある程度の予備知識を求めたものでなく、全くのプログラム初心者が気負いなく学べていける優しい内容の教材としたい。

また使用する擬似コードは、より自然語に近く、理解しやすい日本語になるよう改良していきたいと考えている。さらに使用する変数やオブジェクトの型もひと目見て分かるように色づけするなどして改良したいと考えている。

(3) 自己評価

制作を終えての自己評価であるが、アルゴリズムからプログラムを実践的に解説していくことはできたであろう。アンケートでは実際に、それが理解へと繋がったと捉えられる意見もあった。しかしこの教材が、本当のプログラム初心者に理解されるかは疑問である。それというのも資料ではある程度のプログラミング用語を使用してしまっているし、またプログラム例自体も教材としてよいものとは言えないからだ。解説も曖昧な点や説明不足ではないかと感じられる点が残るなどあまり満足のいく結果ではない。

しかし、この制作を完成させるにあたり、プログラミングに於いてアルゴリズムが基礎となっていた点、そのプログラムの処理の流れを捉えた解説ができた点、さらにそれを発展させた擬似コードを、プログラミングに対する理解を深める材料とすることができたという点で資料の目的を達成できたと思う。この 3 点を段階的に示していくことで、利用者にプログラムをアルゴリズムから考える力をつけさせ、さらにプログラムを実践的に見ることで Python のプログラムにも慣れることができる教材とすることができたのではないか。

またこれら 3 点はいずれも、Python の構文が人間にとって読みやすいよう作成されたという特徴に沿っている。そのためこの教材は、当初考えていたよりもオブジェクト指向プログラミング言語である Python の特徴を活かしたものとすることができた。

プログラミング言語はそれだけで初心者にとって難解なイメージを持たせるものである。私の制作ではその難解さを和らげることはできていない。もしこのイメージを払拭できるようさらに改良を加えていくことができれば、本当に『人の役に立つもの』ができるのではないだろうか。

注

（1）Python の日本語公式サイト『Python Japan User's Group』
http://www.python.jp/Zope/

（2）http://blog.so-net.ne.jp/yfukuda/

（3）『目指せプログラマー！』http://www5c.biglobe.ne.jp/ ecb/index.html

文献表

Mark Lutz・David Ascher(著)・夏目大 (訳)
2004 『初めての Python 第２版』オライリー・ジャパン

福田 洋一
2004 人文情報学授業・『プログラミング演習』配布資料

アンク
2003 『アルゴリズムの絵本』 翔泳社

Randal L Schwartz(著)・(訳)
1998 『初めての Perl』 オライリー・ジャパン