D-01

日本語プログラミング言語「Nature」の開発と学校教育での活用可能性の検討

水野 晴斗†　　伊原 彰紀†　　豊田 充崇‡
Haruto Mizuno　　Akinori Ihara　　Michitaka Toyoda

1. はじめに

2020 年から小学校でプログラミング教育が必須化，2021 年から中学校でのプログラミング内容の拡充，2022 年から高等学校で情報の授業が新設，必修化となり，プログラミング教育への関心が高まっている．プログラミングにおける論理的思考を学ぶきっかけとして，C 言語や Java 言語をはじめとするテキストプログラミング言語の前にビジュアルプログラミング言語でプログラミングを学ぶことが良い[1]．ビジュアルプログラミング言語は，特定の命令処理を実行するブロックを組み合わせることでプログラミングすることができる．ビジュアルプログラミングでは，文法エラーは発生せず，直観的にプログラミングを実現しており，学習難易度が低い．ビジュアルプログラミングによって論理的思考に慣れると，C 言語や Java 言語をはじめとするテキストプログラミングに移行する．テキストプログラミングでは，英語による記述，文法誤りによる構文エラーなどが発生するため，ビジュアルプログラミングからの移行の障壁となる[2]．

ビジュアルプログラミングからテキストプログラミングへの移行するステップとして，日本人が母国語とする日本語をベースとするプログラミング言語「なでしこ」の使用が考えられる．なでしこは，老若男女問わず，誰でもプログラミングできる環境を提供することを目的として開発されている言語である[1] [3]．なでしこは，日本語で記述できる言語のため，英語を十分に習得していなくとも記述することができる初心者向けの言語となっている．なでしこはテキストで記述するプログラミング言語と同様に，文法が決まっているため構文エラーが発生する．構文エラーへの対処方法を学ぶために，日本語で記述できるなでしこはプログラミング初学者にとって有用である．しかし，構文エラーに慣れていない初学者にとっては，プログラミングの文法は障壁となる．

本研究では，プログラミングにおける文法に対する障壁を取り除くために，口語表現で記述可能な日本語プログラミング言語「Nature」と言語翻訳ツール「Natopy」を開発した．「Nature」は，なでしこと同様に，日本語をベースとするプログラミング言語であり，英語で記述する必要がない．また，日本語の文法を元に Nature の文法が決まっており，表記ゆれにも対応しているため，日本語話者にとって学習しやすく扱いやすく親しみやすい言語となっている．

「Natopy」は，Nature で記述したプログラムを Python 言語に翻訳をする言語翻訳ツールである．Nature によってテキストプログラミングを学習し，文法の制約のあるテキストプログラミング言語として Python へ移行を容易にすることを目的とする．Nature は，英語が苦手な小中学生でもプログラミングを行うことができるため，ビジュアルプログラミング言語からテキストプログラミング言語への移行の支援し，小中学生は容易に論理的思考を養うことができると示唆する．

続く 2 章では，本研究で開発する Nature の比較対象となるなでしこの特徴を述べる．3 章では Nature の概要・仕様について，4 章では Nature の実装とデモ，Natopy について述べる．5 章では Nature を用いた授業シナリオを述べ，6 章ではまとめを述べる．

2. 「なでしこ」との比較

日本語で記述できるプログラミング言語なでしこは，メモリ管理やポインタの扱いなどの複雑な処理や，変数の型の宣言が不要のため，文法が単純であり，英語が未習熟であったとしても日本語で記述することが可能である．2021 年には教科書にも採用されている．なでしこは，Web に簡易エディタを提供しているため，環境構築を準備することなくプログラミングを始めることができる．ダウンロード版では，機械学習の実装をすることも可能である．また，PHP 上でなでしこを動かすことや，HTML に埋め込むことにより JavaScript の代わりに記述することもできる．なでしこは，GUI アプリケーションの作成も可能なため，実務用としても使用することができる多機能な言語となっている．

本研究が開発する Nature となでしことの違いは，主に次の 2 点である．

(1) 表現方法

なでしこは，プログラミング言語として文法が決められている．一方で，Nature は，日本語の自然言語に基づき開発している．図 1 は，なでしこと Nature でそれぞれ算術演算する実装の比較を示す．

図 1　なでしこと Nature の比較 1
（上:なでしこ，下:Nature）

２－１を表示
2 から 1 を引いて表示

なでしこでは，計算式と日本語が混在した記述となっており，算術計算は演算子を使用することが文法で決められて

[1] なでしこ：http://nadesi.com/

† 和歌山大学システム工学部, Faculty of Systems Engineering, Wakayama University

‡ 和歌山大学教育学部，Faculty of Education, Wakayama University

いる．Nature は日本語による文章での記述となる．

(2) ブロックを用いる文法

なでしこは，if 文などの条件分岐，while 文などの繰り返し処理は，インデントによってブロックが定義されている．日本語の表現では，段落先頭の字下げを除いてインデントを挿入することはない．Nature はインデントを用いることなく，ブロックを 1 文で記述することができるため，より日本語に近い形の簡潔なプログラムとなる．図 2 は，条件分岐を含むプログラム例「a と 1 が等しい時に，『Hello World』と出力する」の比較を示す．なでしこに対して Nature は，より日本語に近い表現で，字下げを用いることなく，ブロックを 1 文で記述できる．

図 2　なでしこと Nature の比較 2

（上:なでしこ，下:Nature）

```
もし、a=1 ならば
　「Hello World」と表示
ここまで
```

```
もし a と 1 が等しければ「Hello World」と表示する。
```

Nature は，自然言語に近い形で記述をすることが可能であるため，日本語を習熟している人にとって親しみやすい文法となっている．特に，異なる単語であっても同一の意味を表す表記ゆれした命令も同じ処理を行う．

ビジュアルプログラミング言語の経験を持つ学習者が，さらに高度な実装のためにテキストプログラム言語を学習する上で，プログラミング特有の文法への理解が障壁となる．自然言語として使用される日本語で記述できる Nature を学校教育に取り入れることにより，プログラミング特有の文法の障壁なくプログラミング的思考を身に付けることができると示唆する．また，Python 言語や JavaScript 言語など，他の言語に移行においても，Nature での記法と他の言語での記法を対応付けて学習することができ，Nature から他のテキストプログラミング言語への移行も可能である．

3. 言語設計

3.1 概要

Nature は，日本語でプログラムを記述できインタプリタ方式の動的型付けプログラミング言語である．口語的な日本語でプログラミングができることを特徴にしており，プログラミング特有の文法を覚える必要がない．ビジュアルプログラミング言語からテキストプログラミング言語への移行を容易にし，プログラミング学習の支援を行うために開発した．自分の考えたことを，日本語の文章で記述するのみでプログラミングができるため，初学者にとって容易に論理的思考を習得できる言語である．

3.2 要求

Nature は，口語的な日本語を用いてプログラミングできる言語を目指しているため，口語的な日本語で発生する次の 4 つの課題に対応するように言語設計した．

(1) 同義語

自然言語において同じ意味，意図でも異なる表現を使用する場合がある．例えば，「A と B を足す」，「A と B を加える」，「A と B の和」は，プログラムとしては同じ命令処理となる．そのような同義語に対応するために，字句解析において，同じ意味を表現する単語を同じトークンとして設定する．

(2) 目的語と補語の入れ替わり

日本語では，目的語と補語の順番が入れ替わっても同じ意味を表す場合がある．特定の助動詞・助詞は，文章中の位置が異なっていても，同じ処理になる可能性がある．例えば，「2 から 1 を引く」と「1 を 2 から引く」は意味的には同じ文章である．しかし，語順は異なっている．このような文章では，Nature は助動詞や助詞の場所によって異なる処理を行う．具体的には，「から」という助詞の場所によって，異なる処理を行っているので，どちらも同じ処理を行うことを可能にしている．その他，「と」「より」に対応している．

(3) ブロックを用いない記法

プログラミング言語において if 文や while 文などのブロックを波括弧やインデントで表現するが，口語的な日本語でその表現は使用しない．Nature では，自然な日本語に近づけるために，インデントを用いずに日本語文を表現し，プログラムを実装することができる．しかし，例外として関数宣言の際は，ブロックを明確にする必要があるので，インデントを用いて記述する．

(4) 目的語の省略

口語的な日本語では，頻繁に目的語が省略される．例えば，「a に 10 を代入して出力する」という文章は「出力する」の目的語「a」を省略している．正確には，「a に 10 を代入して a を出力する」のような文章になる．Nature では，最後に使用した目的語を記録することにより，目的語が省略された文章でも正しく処理を行う．

3.3 仕様

プログラミング言語で使用する命令処理を Nature における実現方法を述べる．

代入：変数は基本的に半角英語を使用することを推奨するが，一般名詞の場合，日本語を使用してもよい．また，変数は代入された時に宣言される仕様となっている．コード例 1 に事例を示す．

コード例 1 ： 代入

```
{変数 1}に{数値|文字列}を代入(する)
{変数 1}に{変数 2}を加える
{変数 1}に{数値}を加える
{変数 1}に{数値}を減らす
```

出力：出力は，「書く」「出力する」「記述する」の 3 種類の記述が可能である．3.2 要求(1)に対応する．コード例 2 に事例を示す．

コード例 2 ： 出力

```
{変数|数値|文字列}を{書く|記述(する)|出力(する)}
```

関数宣言：「関数{関数名}を定義」は，関数行に，半角または全角スペースを行頭に入れることにより，関数宣言のブロックと認識する．Nature では，理解容易なプログラミング言語を目的とするため，ローカル変数の概念が無く，全てグローバル変数で処理する．コード例 3 に事例を示す．

コード例 3 ： 関数宣言

```
関数{関数名}を宣言(する)
　{変数|数値|文字列}を返す
```

四則演算：「{変数 1}から{変数 2}を引く」と「{変数 2}を{変数 1}から引く」は，同じ処理となる．「{変数 1}と{変数 2}の差」は，{変数 1}から{変数 2}を引いた絶対値を算出する．3.2. 要求(2) に対応する．コード例 4 に事例を示す．

コード例 4 ： 四則演算

```
{変数 1}と{変数 2}を足す
{変数 1}と{変数 2}の和
{変数 1}から{変数 2}を引く
{変数 1}と{変数 2}の差
{変数 1}と{変数 2}を掛ける
{変数 1}と{変数 2}の積
{変数 1}から{変数 2}を割る
{変数 2}を{変数 1}から割る
```

比較演算子：四則演算子と同じように，「{変数 1}が{変数 2}より小さければ」と「{変数 2}より{変数 1}が小さければ」は同じ処理となる．また，「{変数 1}が{変数 2}より小さくない」は「{変数 1}>={変数 2}」，「{変数 1}が{変数 2}より大きくない」は「{変数 1}<={変数 2}」を示す．コード例 5 に事例を示す．3. 2. 要求(2)に対応する．

コード例 5 ： 比較演算子

```
{変数 1}と{変数 2}が等しい
{変数 1}と{変数 2}が等しくない
{変数 1}が{変数 2}より大きい
{変数 1}が{変数 2}より大きくない
{変数 1}が{変数 2}より小さい
{変数 1}が{変数 2}より小さくない
```

条件分岐：接続助詞「ば」を用いることで，条件式を記述できる．また，「違えば」を用いると else 文を記述できる．コード例 6 に事例を示す．3. 2. 要求(3)に対応する．

コード例 6 ： 条件分岐

```
{比較式}ば，{処理 1}
違えば，{処理 2}
```

while 文，loop 文：Nature では，for 文の代わりに loop 文がある．loop 文は，指定した回数同じ処理を繰り返す構文である．C 言語と異なり，カウンタ変数が存在しない．「n 回繰り返す」という文章を記述することにより，loop 文が記述できる．コード例 7，コード例 8 に事例を示す．3. 2. 要求(3)に対応する．

コード例 7 ： loop 文

```
{処理 1}を n 回繰り返す
```

コード例 8 ： while 文

```
{比較式}ばずっと、{処理}
```

論理演算子，数学関数：「かつ」は両辺の論理積，「または」は両辺の論理和を表す．コード例 9， コード例 10 に事例を示す．

コード例 9 ：論理演算子

```
{比較式 1}かつ{比較式 2}
{比較式 1}または{比較式 2}
```

コード例 10 ：数学関数

```
{変数 1}の四捨五入
{変数 1}の切り捨て
{変数 1}の切り上げ
```

4. 実装とデモ

4.1 実装

Nature は，日本語を用いて記述されたプログラムを自然言語解析するため，自然言語解析のパッケージが多数公開される Python を用いて実装した．Nature は入力されたプログラムを，janome を用いて形態素解析する．形態素解析した文は，登録されたトークンと同じ単語かどうかを確認するために原形・品詞，条件式を処理するために活用形を使用する．次に，形態素解析された単語の内，処理に不要な語句を排除したのちに構文解析をおこない，命令処理を実行する． 本章では，プログラム例「a と b と c を足して出力する」の処理の事例を使って手順を述べる．

(1) janome で形態素解析を行う．

（{単語} … {原形}，{品詞}，{活用形}）

a … a，記号−文字，なし
と … と，助詞−格助詞，なし
b … a，記号−文字，なし
と … と，助詞−格助詞，なし
c … a，記号−文字，なし
を … を，助詞−格助詞，なし
足し … 足す，動詞−一般，五段−サ行
て … て，助詞−接続助詞，なし
出力 … 出力，名詞−普通，なし
する … する，動詞−非自立可能，終止形−一般

(2) 不必要な語句を取り除く．

プログラム例では，「て」が処理と処理を繋ぐ単語となり，その前後の処理を実行するため，「て」は不必要と判断をする．また，「する」は前の名詞を動詞化するための動詞である．「する」の記述がない場合，つまり名詞単体でも処理を行うため，「する」は不要である．単体では意味をなさない補助動詞や接続助詞は取り除く．

(3) 特定の助詞を処理する．

プログラム例では，「と」が特別な助詞として処理される．「a と b と c」は 5 つの単語から成り立っているが，[a, b, c]という 1 つのブロックに変換をして処理する．「と」以外の特別に処理される助詞には次のようなものがある．

「から」：3. 2 で述べたように，減算・除算演算子は複数の書き方がある．「b を a から引く」は「a−b」を意味する．しかし，減数・被減数の語順が異なる．この場合，減算と被減算を入れ替えて引き算をする反転減算演算子「r−」を用いて処理をする．今回の場合，「b r− a」となり，正しい日本語の意味で処理される．「a から b を引く」は「a−b」を意味するが，減算・被減算の語順が同じなため，特別な処理を行わない．

「より」：比較演算子にも複数の書き方が存在する．「b が a より小さければ」は「a<b」を意味する．オペランドの語順がそれぞれ異なっているが，同じ意味を表す．この場合，オペランドの語順はそのままに，「b>a」に変換して処理を行う．

(4) 構文解析をおこなう．

図 3 のように，日本語の語順のまま構文木を作成する．

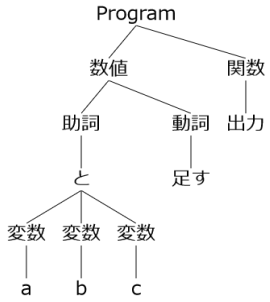


図 3 構文木

(5) 処理

Nature は，「a と b を足したものと c と d を足したものを掛ける」という複雑な記述でも正しく処理するために，スタックを用いて命令処理を実行する．図 4 はスタックを使用した事例を示す．

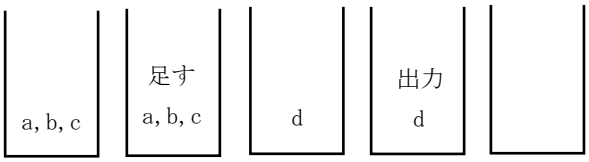


図 4 ： スタックを用いた Nature の処理
（a と b と c を足したものを d とする）

4.2 デモ

本節では，Nature を用いたプログラムの実装事例を示す．

事例 1 ：プログラムと出力 1 ： 関数宣言・if 文・while 文

（上:プログラム，下:出力）

```
関数 sum を宣言する
  a と b を足して返す

a に 3 を代入する
a を 2 で割った余りが 1 ならば、「a は奇数」と書く
b に 1 と a を足して代入する
b が 10 より小さければずっと、b に 1 を加えて出力する
sum を出力する
```

```
a は奇数
5
6
7
8
9
10
13
```

プログラムと出力 1 は次のような順番で処理される．

① 変数 a と変数 b の和を返す関数 sum の宣言．
② 変数 a に 3 を代入．
③ 変数 a の奇数判定．a は 3，つまり奇数であるため「a は奇数」と出力．
④ 変数 b に 1 と a の和，つまり 4 を代入．
⑤ 変数 b が 10 より小さい時，b に 1 を加えて出力するループ処理．b は 4 であったため，5〜10 までの整数を出力が出力．関数 sum を用いて，a と b の和，つまり 13 を出力．

事例 2 : プログラムと出力 2 ： 日本語だけのプログラム

（上:プログラム，下:出力）

```
みかんに 10 を代入する
みかんと 100 の積が 1500 より大きければ、「買えない」と出力し、違えば「買える」と出力する
りんごに 15 を代入する
みかんとりんごを足したものが 20 より小さければ、「バスケットに入る」と出力する
違えば、「バスケットに入らない」と出力する
```

```
買える
バスケットに入らない
```

プログラムと出力 2 は次のように処理される．

① 変数「みかん」に 10 を代入．
② みかんと 100 の積が 1500 より小さいか否かを判定．1500 より大きい場合は「買えない」，そうでない場合は「買える」と出力．
③ 変数「りんご」に 15 を代入．
④ みかんとりんごの和が 20 より小さいか否かを判定．小さい場合は「バスケットに入る」，小さくない場合は「バスケットに入らない」と出力．

Nature は janome に辞書登録されている日本語名詞であれば，変数としても使用することができるので，上のように日本語だけを用いて，プログラムを記述することもできる．

事例 3 : プログラムと出力 3 ： ループの中に条件式

（上:プログラム，下:出力）

```
i に 0 を代入する
i が 5 より小さければずっと、i に 1 を足して、i が偶数ならば「i は偶数です」、違えば「i は奇数です」と出力する
```

```
i は奇数です
i は偶数です
i は奇数です
i は偶数です
i は奇数です
```

プログラムと出力 3 は次のような順番で処理される．

① i に 0 を代入する．
② i が 5 より小さければずっと，i に 1 を足して，i の偶数判定を行う．偶数ならば「i は偶数です」，違えば「i は奇数です」と出力する．

プログラムと出力 3 のように，while 文の中に if 文を記述することが可能である．

4.3 Natopy

Nature から Python に翻訳をする言語翻訳ツールである Natopy の紹介を行う．Natopy は，4.1. 仕様のように Nature の構文木を生成して，Python の構文木に変換をしてから翻訳を行う．また，Python において，条件文やループ文などのインデントを含むコードは，条件文とブロックの部分を分けて変換を行い，最後にインデントによってブロックを表現する．事例 4 と事例 5 は，Natopy を用いて，Nature を Python に翻訳したコードを記す．これらのように，Nature を Python に翻訳することが可能である．Nature を Python に翻訳することにより，Nature からテキストプログラミング言語である Python を対応させて学習することができる．ビジュアルプログラミング言語から Nature，Natopy を用いて Nature からテキストプログラミング言語に移行することが可能である．Nature と Natopy を用いて，プログラミング言語学習の支援を行うことが可能である．

事例 4 : Natopy による翻訳 1

（上:Nature，下:Natopy）

```
「Hello!Natopy!」と出力する
a に 1 を代入する
a が 0 と等しければ、「a は 0 です」と出力する
違えば、「a は 0 ではありません」と出力する
a が 5 より小さければずっと、a に 1 を足して、a を出力
```

```
する
print（ “Hello!Natopy!” ）
a = 1
if a == 0 ：
    print（ “a は 0 です” ）
else ：
    print（ “a は 0 ではありません” ）
while a ＜ 5 ：
    a += 1
    print（ a ）
```

事例 5：Natopy による翻訳 2

（上:Nature，下:Natopy）

```
りんごに 10 を代入する
みかんに 15 を代入する
りんごとみかんを足して出力する
りんごとみかんの和が 30 より小さければ「買える」と出力する
違えば「買えない」と出力する
```

```
りんご = 10
みかん = 15
print（ りんご + みかん ）
if（ りんご + みかん ）＜ 30 ：
    print（ “買える” ）
else ：
    print（ “買えない” ）
```

4.4 今後の課題

Nature は，janome を用いて形態素解析を行い，登録されたトークンのみを処理するように設計している．従って，未登録の単語が入力された場合は処理ができない．教育目的での利用では，児童・生徒が正しい表現の日本語を入力したのにも関わらず，Nature では動作しないことが示唆される．今後は，現在登録していない全ての単語・表現に対応するために，類語辞書を活用することで，より多くの表現に対応することを検討する．

また，字句解析で，同じ意味を表現する単語を同じトークンとして設定している．しかし，そのような単語は多数存在し，本研究では全てを登録できていない．そのため，辞書的に考えられる同義語の全ての単語をトークンとして設定する，または，同義語を検索して同じトークンとして処理をする，などの処理を行うことによって，同義語に対応することを検討する．

5. 授業シナリオ

本章では，Nature を用いた学校教育の授業シナリオを和歌山県教育委員会のきのくに ICT 教育学習指導案を参考に述べる．Nature を用いると，プログラミング的思考をより深められることができると考える．文部科学省は，プログラミング教育を次のように定義している．

> 子供たちに、コンピュータに意図した処理を行うように指示することができるということを体験させながら、将来どのような職業に就くとしても、時代を超えて普遍的に求められる力としての「プログラミング的思考」などを育成するもの（文部科学省　小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）より）

また，プログラミング的思考は次のように定義している．

> 自分が意図する一連の活動を実現するために、どのような動きの組合せが必要であり、一つ一つの動きに対応した記号を、どのように組み合わせたらいいのか、記号の組合せをどのように改善していけば、より意図した活動に近づくのか、といったことを論理的に考えていく力（文部科学省　小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）より）

Nature を用いることにより，日本語を通して，プログラムの 1 つ 1 つがどのような役割を持つのか，そのプログラムをどのように組み合わせると思い通りの動作ができるかなど，プログラミング的思考をより深めることができると考える．表 1 は，きのくに ICT 教育プログラミング教育学習指導案の概要を示す．小学校では，プログラミングに触れ，フローチャートを用いた簡単なロジックの作成をする．中学校では，ロジックの作成に加え，課題発見・解決や社会有用性について学び，また，テキストによるプログラミング言語とビジュアルプログラミング言語の比較やテキストプログラミング言語のメリットについて学ぶ．高等学校では，HTML や CSS，テキストプログラミング言語である JavaScript を用いてアプリケーションの作成をする．これらの過程は，プログラミング的思考を身に付けることを目的としている．しかし，中学校まではビジュアルプログラミング言語，高等学校でテキストプログラミング言語を扱う．テキストプログラミング言語のデメリットとして，初学時の障壁が高い[7]．そのため，テキストプログラミング言語を理解できず，十分にプログラミング的思考を身に付けられない可能性がある．Nature は，ビジュアルプログラミング言語からテキストプログラミング言語への移行を支援する言語である．ビジュアルプログラミング言語からテキストプログラミング言語に移行する前に，Nature を学習することによって，より深いプログラミング的思考を身に付けられる．文部科学省検定済教科書の技術分野の教科書は，東京書籍と開隆堂の 2 社が著作している．東京書籍の教科書では，HTML・JavaScript を用いて防災マップ，ドリトルを用いて校内チャットシステムを制作する．開隆堂の教科書では，C 言語や JavaScript 言語をはじめとする様々な種類の言語の紹介と，ドリトルを用いて図形を描画する．中学校プログラミング教育がビジュアルプログラミング言語とテキストプログラミング言語の移行期間とされている[7]．従って，Nature は中学 2，3 年次に導入することを検討している．

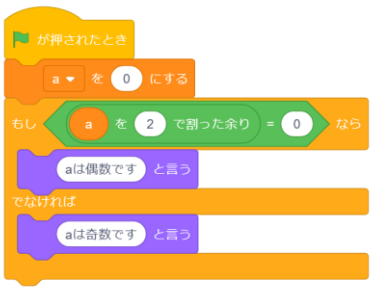
中学 1 年次までは，小学校・中学校プログラミング教育学習指導案のように，ビジュアルプログラミング言語を用いて学習を行う．中学 2 年次では，Nature を用いてプログラムを記述する．3. 3. 仕様で紹介したように，代入や演算，条件分岐，繰り返し処理についての記述を学ぶ．ビジュアルプログラミング言語と Nature のプログラムを対応させることによって，学習を行う．また，ブロックプログラミング言語で記述されたプログラムやフローチャート図を見て，同じ動作をするように Nature のプログラムを記述したり，与えられた問題を解決するプログラムを記述したりする．それらを通して，Nature を用いてロジックを記述する．中学 3 年次では，Nature を通してテキストプログラミング言語を学習する．Nature とテキストプログラミング言語を対応させて学習することにより，テキストプログラミング言語のそれぞれが果たす機能について学習する．また，

Natopy を用いることにより，自分の書いた Nature のプログラムは Python ではどのように記述されるのかを対応させて学習することも可能である．表 2 はブロックプログラムの 1 つであり，実際の教育現場でも使用されている Scratch，Nature，Python の対応したプログラムを示す．

表 1 きのくに ICT 教育プログラミング教育指導案の概要

	主要技術	プログラムの内容
小学校[4]	Studuino	図形の描画，音楽の作成
	Hour of Code	アルゴリズム基礎の学習
	アーテックロボ	センサを用いた制御による自動ドアの制作
	プログル	3 の倍数の判定プログラムの制作，平均値の計算
	フローチャート	図形の区別・水溶液の分類
中学校[5]	Studuino	計算機や計算クイズ 乱数を用いて 2 数の生成と積を計算，入力した数と等しいかの計算
	micro:bit	メッセージングアプリの制作，それを通してメッセージを送受信するための課題発見，手順・アクティビティ図の作成
	EV3・アーテックロボ	掃除ロボットの制作，社会ではどのような機能が求められるかの考察
	JavaScript 言語・Arduino 言語	ブロックプログラミング言語との比較，テキストプログラミング言語のメリットを学ぶ
高等学校[6]	Monaca・HTML・CSS・JavaScript	乱数を用いたおみくじアプリ・連想配列を用いた図鑑アプリの制作

表 2 Scratch・Nature・Python の対応表

Scratch	Nature	Python
	a を 0 にする もし a を 2 で割った余りと 0 が等しいなら、「a は偶数です」と表示する 違えば、「a は奇数です」と表示する	a = 0 if a % 2 == 0 : 　print(“a は偶数です”) else : 　print(“a は奇数です”)
	a を 0 にする a に 1 加えて a を表示を 10 回繰り返す	a = 0 for i in range(10): 　a += 1 　print(a)

Nature を学校教育に導入することによって，より深いプログラミング的思考を身に付けられると考える． Nature を導入することのメリットによって，より親しみやすい言語であることが挙げられる．普段使用している日本語を用いての記述のため，英語が苦手な生徒であっても抵抗なく学習することができる．日本語を用いることにより，より自然な形でプログラミングに触れることができる．以上のような点から，プログラミング授業への Nature の導入によって，プログラミング言語に対する理解を深め，プログラミング的思考を身に付けることが可

能だと考える．また，高等学校でのプログラミング教育で，JavaScript を学ぶ際も中学校でテキストプログラミングを学習しているため，困難を感じることなく，アプリの開発を行うことができるだろう．

6. おわりに

本研究では，口語的な日本語で記述できるプログラミング言語 Nature の開発，Nature から Python に翻訳をするプログラミング言語翻訳ツール Natopy の開発，そして，Nature を用いた学校教育の活用可能性についての提案をした． Nature を中学 2，3 年次のプログラミング教育に導入することにより，生徒のプログラミング的思考がより深められると考える．今後は，Nature を教育に導入することによる効果の調査を行う．

参考文献

［1］ Bau, D., Gray, J., Kelleher, C., Sheldon, J. and Turbak, F.: Learnable Programming: Blocks and Beyond, Communications of the ACM, Vol. 60, No. 6, pp.72-80 (2017).

［2］ Weintrop, D. and Wilensky, U.: Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms, ACM Transactions on Computing Education, Vol.18, No.1, pp.1-25 (2017).

［3］酒徳 峰章，日本語プログラミング言語「なでしこ」，コンピュータ ソフトウェア，Vol. 28，No. 4，pp. 23−28，2011.

［4］和歌山県教育委員会．きのくに ICT 教育 小学校プログラミング教育 学習指導案集．https://www. pref. wakayama. lg. jp/prefg/500100/jyouhouka/d00213120_d/fil/kinokuniICT_es_R40309. pdf

［5］和歌山県教育委員会．きのくに ICT 教育 中学校プログラミング教育 学習指導案【Artech】．https://www. pref. wakayama. lg. jp/prefg/500100/jyouhouka/d00213120_d/fil/kinokuniICT_jhs2104a. pdf

［6］和歌山県教育委員会．きのくに ICT 教育 きのくにＩＣＴ教育 高等学校＜共通教科情報科＞プログラミング教育 学習指導案．https://www. pref. wakayama. lg. jp/prefg/500100/jyouhouka/d00213120_d/fil/kinokuniICT_hs. pdf

［7］小谷 悠介，望月 久稔，ブロック言語とテキスト言語の双方で学習可能な中学生を対象とするプログラミング教材の提案，Vol. 2020−CE−153，No. 16，pp. 2，2020