『情報Ⅰ』におけるプログラミング言語指導のアプローチ

Instructional Approaches to Teaching Programming Languages in “Information I”

児玉 満

要旨

高等学校必履修科目「情報Ⅰ」におけるプログラミング教育について、教科書で採用されている複数のプログラミング言語および大学入学共通テストで使用される共通テスト手順記述標準言語を踏まえ、その指導言語のあり方を検討した。初等教育で Scratch を用いて論理的思考の基礎を育成し、高等学校段階では Python と共通テスト手順記述標準言語の記述形式の類似性から、通常の授業を通じて大学入学共通テストへの対応力を養うことが可能である点を示した。一方で、特定言語の習得を目的化することの課題にも触れ、「情報Ⅰ」におけるプログラミングは言語技能ではなく、問題解決力や論理的思考力を育成するための手段として位置づける必要性を考慮した。

キーワード：情報Ⅰ、プログラミング言語、擬似言語、論理的思考

Abstract

This study examines programming education in the compulsory high school subject *Information I*, focusing on the programming languages adopted in authorized textbooks and the Common Test Standard Pseudocode Language used in the National Center Test for University

Admissions. The aim of the study is to clarify an appropriate approach to programming language instruction that aligns with the objectives of the subject and the assessment system.

In primary education, Scratch is commonly used to develop the foundations of logical thinking through visual and intuitive programming. At the high school level, text-based programming is required to deepen students' understanding of algorithms and data utilization. Python is widely adopted for this purpose due to its simple and readable syntax. In addition, Python shares similarities in notation and structure with the Common Test Standard Pseudocode Language, such as indentation-based block representation. This similarity enables students to acquire the skills needed for the university entrance examination through regular classroom instruction, without relying heavily on test-oriented learning.

The study also addresses the limitations of focusing on the mastery of a specific programming language. In *Information I*, programming should be regarded not as an end in itself but as a means of developing problem-solving abilities and logical thinking skills. Therefore, programming languages should be positioned as tools for achieving educational goals rather than as objectives of instruction.

Key words : Information I, Programming Language, Pseudocode, Logical thinking

はじめに

2022年度から実施されている新しい高等学校学習指導要領に基づき、高等学校情報科において「情報Ⅰ」が共通必履修科目として位置づけられた。これは、それまで選択科目として設置されていた「社会と情報」および「情報の科学」の２科目を統合・再編されたものであり、すべての生徒が情報に関する基

礎的な資質・能力を身に付けることを目的としている。この必履修化は、学習指導要領に置いて情報活用能力が「学習の基礎となる資質・能力」と明確に位置づけられたこととも対応している。

必履修化に至った背景として、情報通信技術の高度化と社会への浸透が挙げられる。AI（人工知能）、IoT、ビッグデータなどの技術が産業や行政、日常生活の多様な場面で活用されるようになり、データを収集・分析・活用する能力が不可欠なものとなっている。このような社会状況を踏まえ、情報を主体的に扱い、問題解決に活用できる人材の育成が学校教育に求められるようになった。また、産業界を中心にIT人材の不足が指摘されており、初等中等教育段階から情報技術に関する基礎的理解と実践的能力を育成する必要性が強調されている。

「情報Ⅰ」は、「情報社会の問題解決」「コミュニケーションと情報デザイン」「コンピュータとプログラミング」「情報通信ネットワークとデータの活用」の4つの領域から構成されている。このうち「コンピュータとプログラミング」は、アルゴリズムの考え方やプログラムによる処理の自動化など、情報処理の根幹に関わる内容を扱う領域である。学習指導要領解説に示されている標準的な指導計画を見ると、この領域には他の領域と比較して多くの指導時間が配分されており、またコンピュータを用いた実習の割合も高い。このことから、高等学校情報科においてプログラミング学習が大きな比重を占めている実態が読み取れる。

さらに、大学入学共通テストにおいても、2025年度から「情報」が必履修科目として導入され、多くの大学が「情報Ⅰ」を利用した入試を実施している。大学入学共通テスト「情報」は「情報Ⅰ」で扱われる4領域全てから出題され、その中で「コンピュータとプログラミング」は全体のおよそ1／4を占めている。公開されている試験結果の分析によれば、「コンピュータとプログラミング」に関する問題は、他の領域と比べて平均点得点率が低い傾向にあることが示されている。この得点率の低さについては、プログラムの動作を理解したり、処理の流れや条件分岐などの論理構造を把握したりすることが、受験生にとって負担となっているのではと考えられる。また、高等学校の授業では

Python などの実在するプログラミング言語を用いて学習することが一般的である一方、大学入学共通テストでは「共通テスト手順記述標準言語」と呼ばれる独自の記述形式が用いられている。そのため、授業で習得したプログラミングの知識や技能を、この標準言語に読み替えて理解する必要があり、これが学習上の難しさにつながっているのではないかとも考えられる。

以上のように、「情報Ⅰ」におけるプログラミングは、教育課程上も評価制度上も重要な位置を占めており、その指導のあり方は高等学校教育全体に大きな影響を及ぼしている。これらの制度的背景と現状を踏まえ、高等学校情報科においてどのようなプログラミング言語を用い、どのような方法で学習を進めることが妥当であるのかについて検討する。

プログラミング言語の使用状況

近年の IT 業界においては、「AI の業務実装」「クラウドネイティブ化の深化」「セキュリティ強化」が主要な技術的トレンドとして位置付けられている。また、日本国内においては「2025 年の崖」と呼ばれるレガシーシステム問題があり、この課題への対応として、DX 推進やシステム刷新を目的とした需要も大きくなっている。

AI の業務実装

単に AI ツールを導入することを指すのではなく、AI が生成・処理した情報を業務プロセスの中で活用し、意思決定や次の作業につなげることを意味する。具体的には、AI による分析結果を基に業務判断を行ったり、自動処理されたデータを後続業務に連携させたりするなど、AI を業務の流れに組み込むことが求められる。このような実装により、定型業務の自動化や作業効率の向上、業務品質の平準化が可能となる。

また、AI を活用することで、人手では困難であった大量データの処理や複雑なパターンの抽出が可能となり、業務の最適化が実現されている。さらに、既存業務の改善にとどまらず、新たなサービスや価値の創出につながる事例も

見られる。そのため、AIの業務実装には、技術導入だけでなく、業務設計の見直しや人とAIの役割分担を明確にした運用体制の構築が不可欠であり、AIを継続的に活用するための仕組みづくりが重要となっている。

クラウドネイティブ化の深化

既存システムを単にオンプレミス環境からクラウド環境へ移行することにとどまらず、クラウドが持つ拡張性・柔軟性・自動化といった特性を前提としてシステムや業務を設計・運用する段階へ成熟度を高めることを指す。具体的には、需要変動に応じたリソースの自動拡張や縮小、迅速な環境構築、障害時の回復性向上などが可能となる。

このようなクラウドネイティブなアプローチでは、コンテナ技術やマイクロサービスアーキテクチャの活用により、システムの変更や機能追加を部分的かつ継続的に行うことが可能となる。また、開発から運用までを一体的に管理する体制を整えることで、リリースサイクルの短縮や運用不可の軽減が図られている。結果として、ITシステムは単なる業務支援基盤ではなく、ビジネス価値を継続的に生み出す基盤として位置付けられるようになっている。

セキュリティ強化

リモートワークの普及やクラウド利用の拡大といった利用環境の変化に対応することを主な目的として進められている。従来は、社内ネットワークの内外を分け、内部を信頼する「境界防御」モデルが主流であったが、業務環境の多様化により、この前提が成り立たなくなっている。そのため、すべての通信や利用者を原則として信頼せず、都度検証を行う「ゼロトラスト」を前提とした防御モデルへの移行が進められている。

ゼロトラストに基づくセキュリティでは、利用者認証や端末の状態確認、アクセス権限の最小化などを組み合わせて管理することが重視される。また、AIを活用したサイバー攻撃が高度化・巧妙化している状況を踏まえ、ログの監視や異常検知といった対策の重要性も高まっている。このように、近年のセキュリティ強化は、単一の対策ではなく、複数の技術と運用を組み合わせた包

括的な取り組みとして位置付けられている。

2025 年の崖（レガシーシステム問題）と DX 推進

日本企業において老朽化・複雑化したレガシーシステムの刷新が進まない場合、2025 年以降に年間最大 12 兆円の経済損失が発生し、国際競争力の低下を招く可能性があると経済産業省が「DX レポート」で指摘した問題を指す。多くの企業では、既存システムが長年の改修によってブラックボックス化し、保守運用に多大なコストと人員を要している。この状況は、新技術の導入やデータ活用を困難にし、DX 推進の大きな障害となっている。

この議題を克服するためには、既存システム棚卸しと刷新計画の策定に加え、経営層が主体的に関与する体制の構築が不可欠である。また、IT 人材の育成や確保、DX 推進ガイドラインに基づいた全社的な取り組みが求められている。そのため、DX 推進は「2025 年の崖」を回避するための中核的な対応策として位置づけられている。

このような IT 業界の情勢であることを踏まえると、様々なシステムを開発することとなり、そこで利用されるプログラミング言語は「AI・データ分析」「高速処理・安全性」「Web アプリのモダン化」に強いものが特に求められる。（Table 1）

Table 1　需要の高いプログラミング言語

言語	主な用途・特徴	必要性が高い理由
Python	AI・機械学習、データ分析、スクレイピング	AI 開発のデファクトスタンダード。ライブラリが豊富。
Java	大規模システム、Android アプリ	安定・堅牢で、企業の基幹システムで根強いシェア。
JavaScript／TypeScript	Web フロントエンド	Web 開発に必須。TypeScript は安全性から主流化。
Go	クラウドサービス、高速バックエンド	コンテナ技術との相性が良く、パフォーマンスが高い。
Rust	高速処理、組み込み、安全なシステム開発	パフォーマンスと安全性を両立。
PHP/Ruby	Web サービス開発、スタートアップ	実装速度が速い。

教科書で採用されているプログラミング言語について

実際に高等学校においてはプログラミング言語を用いて学習を行うが、その際に採用するプログラミング言語は様々である。「情報Ⅰ」の教科書で採用しているプログラミング言語は、主に「Python」「VBA」「JavaScript」「Scratch」の4言語であり、その中から教科書ごとに特徴を踏まえ1～2言語を選択して収録をしている。

主な採用言語と特徴について以下に記す。

Python

1991年にGuido van Rossumによって開発されたプログラミング言語である。開発当初は限定的な利用にとどまっていたが、GoogleやInstagramなどの大規模なサービス開発に採用されたことを契機として、広く利用されるようになった。また、Webアプリケーション開発やデータサイエンス分野を中心に、多数のライブラリやフレームワークが整備されたことも普及の要因の1つである。現在主流となっているPython3系は2008年に公開されており、他の主要なプログラミング言語と比べると比較的新しい言語に分類される。近年では機械学習やAI開発の分野においても標準的な言語として用いられている。

言語仕様の特徴として、簡素で読みやすい構文を持つ点が挙げられ、諸学者にも理解しやすい。こうした特性から、教育分野においても教科書や教材への採用例が多い。

VBA（Visual Basic for Applications）

1990年代にMicrosoft社が開発したプログラミング言語であるVisual Basicを基に作られた言語であり、主にMicrosoft Office製品に組み込まれて利用されている。Microsoft Excelでの利用が特に広く知られているが、Word、PowerPoint、Accessなど、Office製品内であれば共通して仕様することができる。VBAを用いることで、表計算や文書作成における繰り返し作

業や、複数手順を要する処理を自動化し、業務の効率化を図ることが可能である。

言語の特徴として、命令文が人間の言葉に近い形式で記述されるため、構文が比較的理解しやすい点が挙げられる。また、Office アプリケーションの機能と密接に連携できることから、業務データを直接操作するプログラムを容易に作成できる。さらに、専用の開発環境を新たに構築する必要がなく、Excel などのアプリケーションがあればすぐに学習・実行できるため、プログラミング初学者にも扱いやすい言語として利用されている。

JavaScript

1995 年に Brendan Eich によって開発されたプログラミング言語である。開発当初は LiveScript と呼ばれていたが、当時広く普及していた Java 言語の人気に乗じて JavaScript に改名した経緯がある。JavaScript は、HTML や CSS と並び、WWW（World Wide Web）を構成する中核技術の 1 つとして位置づけられている。主に Web ページ上で動的な処理や利用者との対話的な機能を実装するために用いられ、ブラウザ上で直接実行される点が特徴である。また、Node.js の登場により、JavaScript は Web サーバ側の処理にも利用されるようになり、フロントエンドとバックエンドの双方で使用可能な言語として活用範囲が拡大した。さらに、JavaScript に型情報などの機能を追加した TypeScript といった拡張言語も開発されている。HTML や CSS と組み合わせて学習・活用できることから、Web 制作や情報デザイン分野においても広く用いられている。

Scratch

非営利団体である Scratch 財団が、MIT（マサチューセッツ工科大学）メディアラボ内の Lifelong Kindergarten Group と共同で開発し、無償で公開しているビジュアルプログラミング言語である。主に 8 歳から 16 歳の利用者を対象として設計されており、プログラミング初学者でも扱いやすい点が特徴である。マウス操作を中心としたインターフェースを採用しているため、キー

ボード操作に不慣れな小学生でも利用しやすい。

Scratch では、命令をブロックとして組み合わせることでプログラムを作成する。構文エラーが発生しにくく、処理の流れや条件分岐、繰り返しといった基本的なプログラムの仕組みを直感的に理解できるよう工夫されている。このような特徴を持つことから、小中学校のプログラミング教育で広く利用されており、その学習の流れを受けて、高校学校の教科書や教材においても Scratch が取り上げられている。

共通テスト手順記述標準言語（DNCL）について

高等学校情報科の必履修科目である「情報Ⅰ」は、「情報に関する科学的な見方・考え方を働かせ、情報技術を活用して問題の発見・解決を行う学習を通して、問題の発見・解決に向けて情報と情報技術を適切かつ効果的に活用し、情報社会に主体的に参画するための資質・能力を育成すること」を目標としている。その具体的内容として、コンピュータの仕組みやデータ活用能力の習得、情報社会と人との関わりへの理解、さらにプログラミングやデータ分析を通じた課題発見・解決能力の育成が挙げられている。この目標に照らせば、教科書で採用されている複数のプログラミング言語のいずれを用いて学習を行なっても、本質的には大きな問題はないといえる。

一方で、大学入学共通テストを受験することを前提とした場合、「共通テスト手順記述標準言語（以下 DNCL とする）」の理解は不可欠となる。DNCL は、大学入学共通テスト「情報Ⅰ」において使用される試験専用の擬似プログラミング言語であり、特定の実在するプログラミング言語の経験の有無によって受験者に有利・不利が生じないよう、公平な評価を行うことを目的として導入されている。

DNCL は、主に情報処理技術者試験で用いられてきた擬似言語を簡素化した独自の疑似コードを基盤としている。文法面では、インデント（字下げ）によって処理のまとまりを表現する点など、記述形式が Python に近い特徴を持つ。また、「もし～ならば」「～の間」といった日本語を交えた表記が用いられ

ており、アルゴリズムの流れを直感的に理解しやすい構成となっている。これらの特徴は、記号や構文の暗記ではなく、処理手順や論理構造そのものを問うことを意図した設計に基づくものである。

DNCL は、旧センター試験における「情報関係基礎」でも使用されており、これを便宜上「旧 DNCL」と呼ぶことがある。2025 年度から実施されている大学入学共通テスト「情報 I 」では、この旧 DNCL を基に文法や表記が見直され、新たに整理されたものが採用されている。これが「新 DNCL」または「DNCL2」と呼ばれているものである。

このように、DNCL は「情報 I 」で学習するプログラミングの考え方を評価するための共通的な表現手段として位置づけられており、特定言語の習熟度ではなく、アルゴリズムに基づく論理的思考力を測定するための重要な役割をになっている。

DNCL2 と Python の比較

DNCL2 と Python は先に述べた通り記述形式が非常に似ていることから、プログラミング言語の習得としては Python を選択することが合理的であるように思われる。

以下、構文ごとに DNCL2 と Python の記述の比較を紹介する。

コメント文

コメント文とはプログラム中に記述する人間向けの注釈やメモ書きに相当するもので、プログラム実行時にはコンピュータに無視され、実行に一切の影響を与えない文字列のこと。DNCL2 と Python はどちらも行頭に「#」をつけて記述する。(Figure 1)

DNCL2
```
# コメント文
```

Python
```
# コメント文
```

Figure 1　コメント文例

変数と代入

変数とは数値や文字列などのデータを一時的に記憶しておくためのメモリ上の領域につけられた名称のことを言う。データの出し入れや変更が可能な「データを入れるための箱」といったイメージのもの。名付けに関してはDNCL2とPythonで大きな違いはないが、DNCL2は「日本語ローマ字を使用し、２ワード以上は「_」で繋ぐ」とされ、Pythonは「全て小文字を使用し、２ワード以上は「_」で区切り、基本的に英単語を用いる」といった命名規則がある。また、配列変数の場合は、DNCL2では「最初の文字を大文字にする」が追加される。

代入とは変数に値（数値や文字列など）を入れることを言い、どちらの言語でも「=」を使用するので相違はない。(Figure 2)

DNCL2	Python
hensu = 3 # 数字の場合	hensu = 3 # 数字の場合
moji = “ABC” # 文字の場合	moji = “ABC” # 文字の場合
Hairetsu = [1, 2, 3, 4, 5] # 配列変数（先頭大文字）	hairetsu = [1, 2, 3, 4, 5] # 配列変数（リスト）
Hairetsu[2] = 3 # 配列の一部の要素を変更する	hairetsu[2] = 3 # 配列（リスト）の一部の要素を変更する

Figure 2　変数・配列・代入文例

表示文

表示文とは、値を出力する際に使用するもので表記はDNCL2とPythonで異なるが構造は同じで「命令（値)」という形になっている。DNCL2では「表示する（表示したい値)」と表記し、Pythonでは「print（表示したい値)」となる。また、複数の値を表示する際はどちらであってもカッコ内を「,」で区切って複数記述をする。(Figure 3)

DNCL2	Python
表示する(“表示しました”) # 「表示しました」と表示	print(“表示しました”) # 「表示しました」と表示
表示する(“合計：”, hensu) # hensuが5の時「合計:5」と表示	print(“合計：”, hensu) # hensuが5の時「合計:5」と表示

Figure 3　表示文例

演算

四則演算

加算、減算、乗算、除算の４つの基本的な計算のことで、これにはそれぞれ「+」、「-」、「*」、「/」の記号（演算子）が使われるが、DNCL2 では割り算の商の場合にのみ「÷」が使われる。（割り算の商とは答えの整数部のこと。３÷２＝１余り１）Python では割り算の商を「//」と記述する。（Figure 4）

DNCL2

```
tasu = 2 + 3 # 加算
hiku = 3 - 2 # 減算
kake = 2 * 3 # 乗算
waru = 3 / 2 # 除算（waruは1.5）
sho = 3 ÷ 2 # 割り算の商（shoは1）
amari = 3 % 2 # 割り算の余り
beki = 2 ** 3 # べき乗（累乗）
```

Python

```
tasu = 2 + 3 # 加算
hiku = 3 - 2 # 減算
kake = 2 * 3 # 乗算
waru = 3 / 2 # 除算（waruは1.5）
sho = 3 // 2 # 商（shoは1）
amari = 3 % 2 # 割り算の余り
beki = 2 ** 3 # べき乗（累乗）
```

Figure４　四則演算文例

比較演算

比較演算とは２つの値や式を比較し「等しい」「等しくない」「大きい」「小さい」などを判定して、その結果を真（true）または偽（false）で返す記号のことで「==」「!=」「>」「<」などと表記する。さらに「≧」「≦」は「>=」「<=」と記述する。（Figure 5）

DNCL2

```
hensu > 3  # hensu は 3 より大きい
(hensu * 2) <= 8  # hensu×2 は 8 以下
“あいうえお” == “あいうえお”  # 等しい
“ABC” != “abc”  # 等しくない
```

Python

```
hensu > 3  # hensu は 3 より大きい
(hensu * 2) <= 8  # hensu×2 は 8 以下
“あいうえお” == “あいうえお”  # 等しい
“ABC” != “abc”  # 等しくない
```

Figure５　比較演算文例

論理演算

論理演算とは真（true）と偽（false）の真偽値（ブール値）を扱い、「A かつ B」「A または B」「A でない」といった表現をするもので「and」「or」「not」で記述をする。これにより複数の条件を組み合わせたり、条件を反転させたりすることができる。（Figure 6）

DNCL2	Python
hensu >= 2 and hensu <=5 hensu % 2 == 0 or hensu < 10 not hensu >= 10	hensu >= 2 and hensu <=5 hensu % 2 == 0 or hensu < 10 not hensu >= 10

Figure 6　論理演算文例

条件分岐

条件分岐とは比較演算等で条件を設定し、その条件が満たされるかどうか（真か偽か）で処理を変える仕組みのもの。記述構造は DNCL2 と Python で同じだが表記が異なる。（Figure 7）

DNCL2	Python

もし hensu < 5 ならば:
  hensu = hensu + 5
そうでなくもし hensu >= 5 ならば:
  hensu = hensu - 5
そうでなければ:
  hensu = 0
	if hensu < 5:
	  hensu = hensu + 5
	elif hensu >= 5:
	  hensu = hensu - 5
	else:
	  hensu = 0

Figure 7　条件分岐文例

繰り返し文

繰り返し文には「for」と「while」があるがDNCL2は日本語で記述となる。

順次繰り返し文（for）

順次繰り返しはカウンタ変数を初期値から終了値まで増分しながら指定された処理を繰り返し実行するもの。DNCL2 では日本語での表記となるため Python とは大きく異なる。（Figure 8）

DNCL2	Python

i を 1 から 10 まで 1 ずつ増やしながら繰り返す:
  gokei = gokei + 1
	for i in range(1, 11, 1):
	  gokei = gokei + i

Figure 8　順次繰り返し文例

条件繰り返し文（while）

条件繰り返しとは、特定の条件が満たされている間、または決まった回数だけ同じ処理を何度も自動で実行するもの。記述構造は同じだが、DNCL2 では

日本語のため Python とは表記が異なる。（Figure 9）

DNCL2
```
i <= 10 の間繰り返す:
  gokei = gokei + 1
  i = i + 1
```

Python
```
while i <= 10:
  gokei = gokei + i
  i = i + 1
```

Figure 9　条件繰り返し文例

外部入力

「キーボードから文字を入力する」などの場合に使用するもので、DNCL2 では「【外部からの入力】」と記述する。（Figure 10）

DNCL2
```
表示する（“文字を入力してください”）
nyuryoku = 【外部からの入力】
```

Python
```
message = input(“文字を入力してください”)
```

Figure 10　外部入力文例

関数定義と呼び出し

関数とは一連の処理をまとめて記述し、それを呼び出して利用するもので DNCL2 では問題文中に関数定義されている。（Figure 11）

DNCL2
```
# 問題文中に関数の記述がある
和を返す（始めの数，終わりの数）・・・引数として「始めの数」と「終わりの数」が与えられ、「始めの数」以上「終わりの数」以下の値の和を返す関数。
例：「始めの数」が 1、「終わりの数」が 5 の場合、1+2+3+4+5 となるので、和を返す（1，5）の値は 15 となる。

wa = 和を返す（1，10）
```

Python
```
def returnSum(start, end):
  sum = (start + end) * (end - start + 1) / 2
  return sum

wa = returnSum(1, 10)
```

Figure 11　関数定義と呼び出し文例

以上のように記述構造は似ているため、授業では Python を使用して実践的なプログラミングを学んでおけば大学入学共通テストのためにプログラミング言語をもう一度最初から学習し直すといった手間は省けるものと考えられる。

「情報Ⅰ」で用いるべきプログラミング言語

高等学校必履修科目「情報Ⅰ」におけるプログラミング言語の選定は、初等・中等教育における学習の連続性、大学入学共通テストへの対応、そして実社会での活用状況を踏まえて検討する必要がある。初等教育段階では、操作が容易で視覚的に理解しやすいScratchを用いることで、順次処理、条件分岐、繰り返しといった論理的思考の基礎を学ぶことが一般的である。Scratchは構文エラーが生じにくく、プログラムの構造を直感的に把握できるため、プログラミング的思考の導入に適した言語として位置づけられている。

中等教育段階、特に高等学校「情報Ⅰ」では、より実用的なプログラミング言語を用いて学習を進めることが求められる。その中でPythonは、簡潔で可読性の高い構文を持ち、アルゴリズムやデータ構造の理解に集中しやすい言語である。加えて、データ分析やAI分野で広く利用されており、標準ライブラリや外部ライブラリが充実していることから「情報Ⅰ」で扱うプログラミングやデータ活用の学習内容と整合性が高い。実際の社会においてもPythonは多くの分野で利用されており、基礎的な習得が将来の学習や進路に無関係となることは少ない。

また、大学進学を視野に入れた場合、大学入学共通テスト「情報Ⅰ」への対応も重要である。この試験では、共通テスト手順記述標準言語（DNCL2）が用いられており、その理解と読解が不可欠となる。DNCL2は特定の実在言語に依存しない疑似言語であるが、インデントによるブロック表現など、記述形式においてPythonとの共通点が多い。そのため、通常の授業でPythonを用いてアルゴリズムや処理の流れを学習しておくことで、DNCL2の理解に多くの時間を要しないという利点がある。

一方で、Pythonの特性や限界についても正しく理解させる必要がある。Pythonは実行速度が比較的遅いことや、大規模かつ高い性能が求められるシステム開発には適さない場合があることが指摘されている。実社会のシステム開発においては、用途や要件に応じて複数のプログラミング言語が使い分けら

れており、特定の言語のみを習得すれば十分であるという考え方は適切ではない。

このように、「情報Ⅰ」で用いるプログラミング言語としては、初等教育でのScratchによる基礎的理解を踏まえ、高等学校ではPythonを用いて実用性と汎用性を意識した学習を行い、合わせてDNCL2への対応力を養う構成が、教育課程全体の連続性と評価制度の両面から合理的であると位置づけられる。

考察

本研究では、高等学校必履修科目「情報Ⅰ」におけるプログラミング教育を巡り、Scratch、Python、VBA、JavaScriptといった教科書採用言語および大学入学共通テストで用いられる共通テスト手順記述標準言語（DNCL2）について整理・検討を行った。これらを踏まえると、「情報Ⅰ」で用いるプログラミング言語は、特定の言語技能の習得を目的とするものではなく、情報に関する科学的な見方・考え方を育成するための手段として位置づける必要があることが改めて明らかとなる。

初等教育段階では、Scratchを用いたプログラミング学習が広く行われており、視覚的かつ直感的な操作を通じて、順次処理、条件分岐、繰り返しといった基本的な制御構造を理解することが可能である。これは、キーボード操作や構文理解に不慣れな学習者にとって、論理的思考の基礎を形成する上で有効な方法である。一方で、高等学校段階では、より抽象度の高いアルゴリズム理解やデータ活用が求められるため、テキストベースのプログラミング言語を用いた学習が不可欠となる。

その点において、Pythonは簡潔で可読性の高い構文を持ち、プログラムの処理内容を理解しやすい言語である。加えて、データ分析やAI分野など、「情報Ⅰ」で扱う内容と親和性の高い活用分野が多いことから、教育内容との整合性が高い言語であるといえる。また、インデントによるブロック表現などの記述形式は、共通テストで使用されるDNCL2とも共通点が多く、通常の授業でPythonを用いてアルゴリズム的思考を身につけておくことで、DNCL2の理

解にも円滑につなげることが可能である。この点は、日常の授業と大学入学共通テストとの接続を考える上で重要な意義を持つ。

一方で、Python の利便性や社会的利用の広さのみを強調することには注意が必要である。Python は実行速度の面で制約があり、大規模かつ高性能が求められるシステム開発には適さない場合がある。実社会においては、用途や目的に応じて複数のプログラミング言語が使い分けられており、単一の言語のみを習得すれば十分であるという考え方は現実的ではない。そのため、「情報Ⅰ」においては、Python を主要な教材言語として扱いつつも、言語にはそれぞれ得意分野と制約があることを理解させ、目的に応じた選択が重要であるという視点を育成することが求められる。

また、大学入学共通テストにおいて DNCL2 が採用されていることは、学校現場におけるプログラミング教育のあり方にも一定の影響を与えている。DNCL2 は特定の実在言語に依存しない擬似言語であり、受験者間の公平性を確保しつつ、アルゴリズムや論理構造の理解を評価することを目的としている。この設計思想は、「情報Ⅰ」が目指す資質・能力と整合的であり、プログラミングを手段として問題解決力を測るという点で意義がある。一方で、実際の授業では実在言語を用いて学習するため、試験言語との表記差をどのように橋渡しするかが課題となる。その点でも、Python と DNCL2 の構造的な類似性を活用した指導は、過度な試験対策に偏ることなく、自然な形で共通テストへの対応力を養う方法として有効である。

以上を踏まえると、「情報Ⅰ」におけるプログラミング言語の選定は、初等教育からの学習の連続性、大学入学共通テストとの接続、そして実社会との関係性を総合的に考慮した上で行う必要がある。特定の言語習得を目的化するのではなく、アルゴリズム的思考やデータ活用能力、情報社会を主体的に生きるための判断力を育成するという教科本来の目的を常に意識することが重要である。その意味において、Python を中心とした指導は現行制度下で合理性を持つ一方、言語の相対化と位置づけを丁寧に行うことが、「情報Ⅰ」におけるプログラミング教育の質を高める上で不可欠であると考えられる。

参考文献

・特定非営利活動法人みんなのコード『2022 年度プログラミング教育・高校「情報Ⅰ」実態調査報告書』（2023 年７月）
・田﨑丈晴『新学習指導要領の改訂のポイントと学習評価（高等学校 情報科（共通教科「情報」））』独立行政法人教職員支援機構（2022 年１月）
・文部科学省『高等学校情報科「情報Ⅰ」教員研修用教材』（2019 年９月）
・文部科学省『高等学校学習指導要領解説「情報編」』（2018 年７月）
・児玉満『プログラミング教育に関する考察』西日本短期大学総合学術研究論集第８号（2018 年３月）