文系情報学科3年次のプログラミング授業における多言語教育の一事例

多言語修得の必要性と意義に焦点を当てて

A Trial of Junior Programming Class in Humanities College
- Focusing on the Necessity and Sense of Multilingual Acquisition-

上西秀和
Hidekazu Kaminishi

江戸川大学
Edogawa University

情報化社会の進展に伴う情報リテラシー教育の深化やIT技術者の不足などを背景とし，プログラミング教育の必要性が高まっている．より深いコンピュータの理解のためには，複数のプログラミング言語を学習することが欠かせない．本稿では，江戸川大学メディアコミュニケーション学部情報文化学科の第3学年後期開講科目「プログラミング4」において，さまざまなプログラミング言語について取り扱い，情報科学的な考え方の習得を図った授業事例を紹介する．また，授業の事後アンケートにより，主観的にこの目標が達成されたかを評価したため，報告する．

事後アンケートにより，情報科学的な考え方のうち，主に計算機論的思考についてはある程度達成されたと推測された．しかしながらアーキテクチャの理解は一部達成にとどまったとみられた．

各言語のコーディングスキルの習得については，短時間の授業時間であったことを鑑みると，ある程度達成されたといえる．また，本授業を実施したことによる副次的な効果として，受講者が各言語の開発環境を持つに至り，授業外活用でこれらの環境が受講者の学習・開発活動に役立つ可能性が示唆された．

キーワード：プログラミング教育，情報専門教育，文系大学，コンピュータの科学的な理解

1. はじめに

1.1 背景

高度に情報化が進展した社会におけるIT人材の必要性は高い．IT人材としては，プログラマ，システムエンジニア等が含まれるが，これらの職種において必要とされる人材は常に不足しているのが実態である．経済産業省・みずほ情報総研の調査(2019)では，2030年には国内で最大79万人のIT人材が不足すると予想している．そのような中で，IT人材の育成が急務であるのは明らかであり，大学教育が担う役割は大きいと考えられる．

筆者らはこれまでに，IT人材の不足を補うためには，情報学を専門に学習した(**情報専門教育**を受けた)人材だけでは不十分であると予測し，**一般情報処理教育**におけるプログラミング言語教育を充実させるための試みを行ってきた(Kaminishi & Murota 2011；上西・室田 2014； 上西 2014)．情報専門教育と一般情報処理教育の関係を**図1**に示す．上西(2014)では「情報専門教育」を受けた人材よりも「一般情報教育」に着目し(**図1**の下半分に相当)，その教育の重要性を取り上げている．ここで「情報専門教育を受け

2024年1月22日受付　2024年2月26日受理

た」人材としては，主に「理系」分野からの情報専門教育を受けることを想定(**図1**の右上に相当)していた．

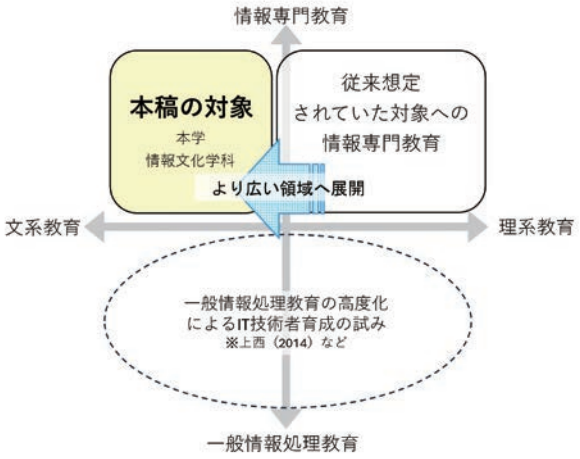

図1　情報専門教育と一般情報処理教育の関係

ところで，「情報科学」「計算機科学」あるいは「Computer Science」という言葉があるように，もともとの情報学教育はScienceからのアプローチ(あるいは情報科学の歴史的経緯からMathematicsからのアプローチ)が一般的であったと考えられていたであろう．一方で，現在では，文系・理系の区別なくコンピュータを活用するのは当たり前であるし，Science/Mathematicsに必ずしも明るくない学生であっても情報専門教育を受ける意義や必要性がある．本稿が対象とする「江戸川大学　メディアコミュニケーション学部情報文化学科」(以下，本学本学科と記す)では(**図1**の左上)，文系学部において，「文系の」情報専門教育を行っている．先の上西(2014)の考え方からみると「情報専門教育」を受ける学生の範囲が拡張したといえる．

ところで，「一般情報処理教育」の現状に視点を当てると，文部科学省では，「デジタル時代の『読み・書き・そろばん』である『数理・データサイエンス・AI』の基礎などの必要な力を全ての国民が育み，あらゆる分野で人材が活躍する環境を構築する必要」があるとして，数理・データサイエンス・AI教育プログラム(MDASH)を制定している．このプログラムでは，大学等の学生一般の情報リテラシーが情報化社会に対応できるように全体の底上げが図られる一方で，必ずしもプログラミング教育に特化しているカリキュラムとなっているわけではない．情報科学への理解を一定程度促す可能性はあるが，特にプログラマのようなIT技術者を育成する教育としてはMDASH認定相当のカリキュラムの内容だけでは不十分な可能性がある．より直接的にいえば，コーディング力そのものを教育しているわけではないともいえる．

「一般プログラミング教育」として，布施(2018)に紹介されている取り組みがあるが，こちらは本文中にも言及されている北海道大学等での教育を前提にしていると考えられ，一般プログラミング教育としては比較的高度な内容を取り扱っている．これは上西(2014)のようなアプローチの延長にあると考えられ，それ自身が意義深いことではある．しかしながら，北海道大学などの国立大学の学生に対し，本学では入学試験科目の設定などから，理系分野に対する理解が弱い傾向がある．そのため，一般プログラミング教育として理系知識を前提とした取り組みをそのまま本学の教育のカリキュラムに当てはめるのは困難が伴う．例えば，

図2　情報文化学科のカリキュラム図と「プログラミング4」授業の位置づけ
図中の☆印がプログラミング4に相当するが，情報教職の科目ではないことに注意

布施(2018)で引用されている取り組み(布施・岡部・中西2017)ではCPU の動作原理の理解に関わる授業実践が一般情報処理教育として行われているが，これはある程度計算に慣れ親しんだ学生でないと興味・関心が持ちにくく，本学においては全学的な一般情報処理教育としてよりはむしろ選択授業としての情報専門教育として扱うべき内容と考えられる．

このような状況に鑑みて，本稿では，「文系大学における」，より専門教育的な観点からのプログラミング教育の検討を取り扱う．

プログラミング教育を行う上で，まずは文法や基本的な構文などの理解，コードの読み書きができるようになることが重要である．その一方で，「よいコードを書く，よい問題解決を図る」ためにはこのような「表層的な」知識の習得だけでは不十分であり，「情報科学的」なコンピュータの理解が必要である．世の中には多数のプログラミング言語(以下，単に言語と呼ぶ)が存在しており，それぞれにさまざまな目的があって存在している．よいプログラマとなるためには，ある言語を深く習得することも必要であるが，かといってある言語だけを習得しているだけでは不十分と考えられる．複数の言語を活用できるようになることで，それぞれの言語の長所・短所を理解できること，ある言語の得意なアルゴリズムを理解して他の言語に応用できるようになることなどが想定される．ソフトウェア開発者・プログラミング言語開発者であるクジラ飛行机(2020)によれば，複数のプログラミング言語が存在する理由として，以下を挙げている．

- 最適な道具：言語によって得意分野・苦手分野があるため最適な道後を選択する必要がある
- プログラミング言語の進化：新しい言語そのものが技術を刷新し，言語が使われていく中でその言語では解決できない問題が見つかる
- プログラミング言語の多様性：IT 分野では独占が起こらず複数の選択肢が並列することがある(例としてスマートフォン向けOS)．これらは独自のバックグラウンドや特徴をもち，それぞれの利用者が多様性を維持している

そのような事情を紹介したうえで，クジラ飛行机(2020)は複数のプログラミング言語を学ぶ意義を説明している．布施(2018)においても，「複数のプログラミング言語を広く浅く学び，プログラミングの表現の共通性と，プログラミング言語による特性を比較し理解すること(中略)は，プログラミング行為を概念的に理解し，問題を抽象化する能力の育成につながるのではなかろうか」と予測している．本稿では，このような状況に鑑みて，ある言語をある程度理解した学生が，多言語を学びながらコンピュータの情報科学的理解を促すことを図る．

1.2 目的

Science/Mathematicsに必ずしも明るくない学生(ここでは「文系」学生とする)に対して行う情報専門教育のうち，第3学年後期の開講科目「プログラミング4」においてさまざまなプログラミング言語について取り扱い，情報科学的な考え方の習得を図る．事後アンケートにより，主観的にこの目標が達成されたか評価する．

2. 授業の位置づけと内容

2.1 当該学科のカリキュラム概要

本稿で対象とする「江戸川大学メディアコミュニケーション学部情報文化学科」(以下，情報文化学科)は，「情報」と「国際」を2つの柱にしたカリキュラムで学び，ICT(情報通信技術)と英語を媒介としたコミュニケーション力を育成する学科である(江戸川大学 2024)．学科内で3コース(情報デザイン，情報システム，国際コミュニケーション)が用意されているが，学生はそのどれかに所属するわけではなく，自身の興味のある授業を選択しやすいように

目安として設定されているものである．例として，本稿で対象とする「プログラミング4」の授業は，情報システムコースでは指定科目(指定学年での履修が強く推奨される)，情報デザインコースでは選択科目とされている．

2.2 「プログラミング4」の概要と位置づけ

「プログラミング4」の授業は，同学科の3年後期科目として開講される科目である．カリキュラム改変や担当者の不在に伴い，前年度(2022年度)は開講されず，2023年度からは新しい内容で開講されることとなった．そのため，

表1 「プログラミング4」授業の内容
※( )のついている項目は本稿の範囲外とする

回	テーマ	概要
1	ガイダンス	各言語の実行環境をインストール
2	Python	既習であるPythonの復習
3-4	C/C++	手続き型言語Cと，OOP対応のC++
5-6	Racket	Lisp系の関数型言語
7-8	PHP	サーバサイドスクリプティングと，DBや外部プログラムとの連携
(9-10)	(UML)	(オブジェクト指向の考え方とUML)
11-12	JavaScript	非クラスベースオブジェクト指向言語，JSONとAPIの活用
(13-14)	(Mini Project-BasedLearning)	(簡単な教育用アプリの開発)

筆者が本年度授業に向けて一から授業設計することとなった．（※カリキュラム改変と関連して，2021年度入学生までは，「オブジェクト指向プログラミングII」の科目名での受講となる．）**図2**は本学のカリキュラムを模式的に示したものであり，図上での本授業が位置するおおよその位置を星印で示したものである．この図が示すように，本科目はプログラミング系科目で最後に受講する科目の1つであり，応用的な内容を含むことが想定されている．同学科ではプログラミング言語教育の柱としてPythonを利用することとしており，科目で必要に応じてJavaScript，R，Unity等を学習している状況である．

このような中で，さまざまな言語を学習したいという要望が学生からあった．また，就職活動を考慮するとさまざまな言語に触れている学生のほうが企業へのエントリーが容易という背景がある．別の背景としては，（筆者の主観ではあるが）学生がこれまでに受講してきたカリキュラムでは，プログラミングを通してその機能や活用方法については学んでいるものの，コンピュータそのものへの深い理解が進んでいないように見受けられた．具体的な例を挙げると，計算論（ここでは主に，プログラムの意味論や計算量など（渡辺・米﨑 1997，p.206））やアーキテクチャに関する理解である．このうち計算量については，筆者が担当した同年度前期の「プログラミング3」において期末課題として再帰の計算量に関するレポートを課したところ，多くの学生から「このような内容は初めてである」「プログラムの内容は既習であるが，計算時間について考察し文書化した経験は無かった」「これほど動作に時間のかかるプログラムを実行したのは初めてである」との声が上がった．このような経緯から，改めて計算論やアーキテクチャを含む情報科学的内容の理解を促す必要があると判断されたため，本「プログラミング4」の講義の内容として，これらを取り扱うこととした．

授業の具体的内容を**表1**に示す．（なお，前述のように，もともとは「オブジェクト指向プログラミングII」として開講されている経緯や，プログラミング3の授業との兼ね合いから，UMLやMini Project-Based Learningの内容も取り扱っているが，本稿では多言語教育に触れることが主であるので，本稿の範囲外とする．）次章では，これらの各項目について説明する．

3. 各授業内容の詳細と目的，実施状況

3.1 ガイダンス

ガイダンスとして授業の概要を説明するとともに，学生に各言語の開発環境のインストール作業を行ってもらった．

本学では，学生が大学から個人用ラップトップPCを支給されている．この授業を含むコンピュータを用いた演習はこのPCを用いて行う．そのため，演習教室の共用端末を利用する場合と比較して，開発環境を学生個人が各自で整備しなければならないがというデメリットがあるが，逆にメリットとして，授業外でこの開発環境を利用することも容易である．（このメリットについての参考となる事例は，考察の章にて後述する）

開発環境としてインストールしたものを**表2**に示す．プログラミング言語を学習する上で最初のハードルは，開発環境を整えることである．そのため，受講者が確実に開発環境を整えることができるよう，これらの説明は学生の達成状況を確認しながら慎重に進めた．

なお，開発環境のインストールにはトラブルがつきものであること，通信環境（学内Wi-Fi）の制約から，一部は時間内にインストールが終わらず，次回以降の演習の空き時間でインストールしてもらうこととなった．参考までに，以下にインストール時に起こったトラブルを含めて状況を記す．

表2　各言語の開発環境

Python	VSCode
C/C++	MinGW+VSCode
Racket	DrRacket
PHP	VSCode+XAMPP
JavaScript	VSCode（+Google Chrome）

VSCode（Visual Studio Code，マイクロソフト 2024）は，Microsoft社が無償で提供する無償のコードエディタである．学生はこれまでのプログラミング3の授業のPythonのコーディング等で利用してきている．2015年リリースの比較的新しいコードエディタではあるが，Pythonだけでなく，プラグインの追加により多くの言語の開発に利用することができる．**表2**にも示している通り，Racket以外の開発にVSCodeを活用している．（Racketでは，VSCodeのプラグインが用意されてはいるが，DrRacketによる対話的な利用のほうが環境の整備・開発作業ともに容易であるため，VSCodeは利用していない）

MinGWは，ほとんどの学生においてインストーラによるインストールが行えず，手動でファイルをダウンロードすることとなった．しかしながら，このファイルは7zip形式で保存されて配布されているため，別途7zipを展開する用のアーカイバをインストールせねばならず，手間であった．またMinGWのファイルサイズは100MBを超えるので，学生が一斉にダウンロードすると教室の学内Wi-Fiの通信帯域を圧迫し，時間を要する原因ともなった．

DrRacketはWindowsのwingetコマンドラインツールからインストールでき，インストール作業自体は比較的容易であるが，インストール時に200MB程度の通信が発生するため，MinGW同様，一斉ダウンロードする際には注意が必要である．

XAMPPも同様にファイルサイズが大きいため，インストールに時間がかかる．そのため，XAMPPは多くの学生

が授業間などの時間や，自宅でのインストール作業を行うこととなった．

3.2 Python

複数のプログラミング言語を学ぶこととその意義について説明したのち，Pythonの基本的な書き方について復習をした．前者では，開発場面ごとに言語の向き・不向きがあることや，チャーチ＝チューリングのテーゼ等について説明した．後者では，変数宣言，標準入出力，配列などのデータ構造，条件分岐やループなどの制御構造，関数，クラスとメソッドなどについて復習した．これらは言語の特徴が表れやすい部分であり，Pythonを他言語と比較するときに参照されやすい事項である．そのため，多くの学生はこれらを十分に理解していると思われるが，改めて説明を行った．

演習問題として，復習の意味を込めてプログラミング教育ではよく知られた問題であるFizzBuzz問題(**表3**)を出題した．この問題は標準出力，条件分岐，繰り返し処理などの理解確認に適している，なお，他言語の演習問題でもFizzBuzz問題を出題している．

表3　FizzBuzz問題と解答例（Python）

0から100までの数を順に表示しなさい．ただし，3の倍数のときはFizz，5の倍数のときはBuzz，その両方を満たすときはFizzBuzzと表示すること．

```
for i in range(0, 101):
    if i % 15 == 0:
        print("FizzBuzz")
    elif i % 3 == 0:
        print("Fizz")
    elif i % 5 == 0:
        print("Buzz")
    else:
        print(i)
```

3.3 C/C++

主にC言語について取り上げ，補足的にC++を取り上げた．本項で主に取り上げる概念は，ポインタとメモリについてであり，この目的に対してはC++の内容は必ずしも必要ではない．しかしながら受講者の中にはゲーム開発会社への就職を希望している者が多くおり，コンシューマゲームの開発には++が欠かせないとの現役開発者の意見もあり，C++も取り扱った．

まず，C言語の基本的な構文(stdio.hのインクルードやmain関数など)やPython同様の変数宣言，標準入出力，配列などのデータ構造，条件分岐やループなどの制御構造，関数について説明した．加えて，C言語は型に厳格な言語であることから，型の概念についても改めて説明した．次に文字列操作，ポインタと配列およびメモリの概念，構造体について学習した．よく知られる通り，ポインタの概念はC言語習得の難関のひとつであるため，学生の到達度を考慮して深く取り扱うことは難しかった．しかしながら，配列とポインタ，メモリの概念に触れることでコンピュータの深い理解につながるのではないかと考えている．

C++の項目では，C言語との標準入出力の違い，クラス宣言のしかたについて学習した．特に，Pythonとの違いとして，「アクセス修飾子」についても説明した．Pythonはインスタンス内の属性値へのアクセス制限が弱い設計となっている．一方，C++ではよりアクセス修飾子により厳密なアクセス制限が設定できることを説明した．

また演習問題として，プログラミング3の授業で既習のコード(クラス宣言を用いたコード)をC++に書き換える問題も出題した．

3.4 Racket(Lisp)

関数型言語の例として，Lispの方言であるRacketを取り扱った．かつて，Racketのもとになったschemeの教育向け実行系としてDrSchemeが活用されていたが，現在同環境はDrRacketとして後継版が提供されている．DrRacketは，Racketやいくつかの言語を扱うことができる実行系である．本授業ではこれを活用した．

Lisp自体は古くから存在し，かつては人工知能研究等に活用されてきたが，近年ではそれほど注目されていないのが現状である．それにもかかわらず本授業で取り扱った目的として，「主要なプログラムの構成やデータの構造を学び，それを言語の基礎となる言語学的機能に関係づけるのに優れた媒体とする独特の特徴を持っている」(サスマン・エイブルソン・サスマン・和田 2000)ことを説明した．

Racketの授業では，以下の項目を扱った．

・四則演算と前置記法
・変数の宣言define
・関数の定義
・変数の束縛とスコープ
・条件分岐
・再帰による繰り返し処理
・ハノイの棟の実装(動作するコードを提示しながら解説)
・ペアとリスト

関数型言語であるRacketの記法は，現在主流の手続き型/オブジェクト指向型言語の記法と大きく異なっているため，学生の戸惑いが大きかった．後述するアンケートでもその様子がうかがえる． なお，時間の都合上，lambda式，map，代入，GUIなどの内容に触れられなかったことが今後の授業実施の課題である．(なお，lambda式については後のJavaScriptで多少触れた．)

Racketの授業を行う上での障害として，ドキュメントの少なさ(特に和文)が挙げられる．DrRacket添付のヘルプは充実しているが，それ以外のサンプルコード等が少ない．Schemeであれば，先に引用したサスマンら(2000)の教科書(MITの教科書として知られる．SICPとも呼ばれる本)

が利用可能であるが，RacketではSchemeの一部のコードに互換性が無いことに注意が必要である．

3.5 PHP

PHPの項目では，基本的な言語の文法に加えて
・Webサーバ構築（XAMPP）
・HTML表示（サーバを介して）とGET/POSTメソッド
・ファイル操作とデータベース連携
・他言語で書かれたプログラムとの連携
について学んだ．ファイル操作とデータベース連携はPHPの重要な機能の一部であり，コンピュータの科学的な理解というよりはWebを構築するシステムの理解につながるような内容である．

さらに，PHPではシステムコールにより他言語で書かれたプログラムをサーバ上で動かして結果を取り出すような連携が容易である．このような他言語との連携の例として，先のRacketの授業で紹介した「ハノイの塔」のプログラムをWebブラウザ上で動作させる例を紹介した．

3.6 JavaScript

Webブラウザのフロントエンドで利用されている言語である．利用者数世界一（SlashData 2021）といわれる言語であるが，現在主流のクラスベースオブジェクト指向言語と異なる実装でオブジェクト指向を実現しており，「関数型言語的で非クラスベースなオブジェクト指向」とも評される．標準規格としてのECMAScript2015以降は糖衣構文としてクラスを利用可能になっているが，あくまで内部では非クラスベースな実装となっている．

授業ではこのような特徴を伝えつつ，
・無名関数の活用
・関数の宣言（function文とアロー式）
・配列のデータ構造
・オブジェクトの利用と関数
・コンストラクタ
について説明した．

次に，JavaScriptのオブジェクト記法が元となっているJSON（JavaScript Object Notation）についても学んだ．JSONは構造化データを表現するための標準のテキストベースの形式として，XMLなどと並びWeb APIとのデータのやり取りなどで広く利用されている．授業ではJavaScriptのオブジェクトとJSONの関係に注意しながら取り扱った．

さらに，Web APIの活用についても学んだ．ここではAPIとのデータのやり取りにJSONを活用した．

4. 受講生情報とアンケートの結果

当該年度の受講生情報を**表4**に示す．3年生後期の授業であることや，4年生の出席者（最低1回出席した人とする）がいたため，単位数が十分な学生にとっては履修放棄となってしまいやすい状況にある．そのため履修放棄者が散見されたが，逆に最後まで履修している学生は（学習内容に興味のある「内的」なものか，単位修得が必要であるという「外的」なものかはさておき）モチベーションの高い学生といえよう．なお，本稿の執筆時点では最終課題の締切前であるため，最終的な単位の修得者数は不明である．

表4　プログラミング4　履修者情報

出席者数（履修放棄者含む．最低1回出席した学生数）	37
履修登録者数 （うち3年生：25　4年生：6）	31
履修放棄者数	6
平均課題提出者数（第1 ～ 12回）	22.9

最終授業日に授業を評価するアンケートを行った．これは，大学が公式に実施するアンケートとは別のものである．このアンケートは任意回答とし，匿名による調査であること，成績とは一切関係ないことを説明の上で回答を依頼した．当日の授業出席者26名中15名から回答があった．アンケートでは
・事前に学習（履修）していた項目
・本授業により習得した各言語に対して，
-学生による主観的評価
-各言語で学べたことや今後学んでいきたいこと（自由記述）
-授業の改善点（自由記述）
・授業全体を通しての主観的評価
を問うた．以下，それぞれの項目について説明する．

4.1 事前に学習・履修していた項目

本授業履修前の学生の学習・履修状況についての質問である．**図3**より，授業開始前に経験のある言語PythonとJavaScriptの経験が多いことが挙げられる．Lisp系言語，PHPについては経験者なしであった．また，グラフには含んでいないが，JavaやC#の経験者が若干名あった．また，**図4**（これまでに履修してきた授業）より，「プログラミング3」「Web開発系の授業」を含めた関連科目の多くを履修していることがわかる．Pythonはプログラミング3を含むプログラミング言語系授業での主要言語としている．また，Web開発にJavaScriptは必須であるので学生はこれらの授業でJavaScriptを学習したと考えられる．

まとめると，プログラミング4で扱う言語を含め，それ以外の言語についてはあまり学習する機会が無かったことが伺える．また，本講義の受講者はプログラミングを含む情報系科目を多く履修していることが確認された．

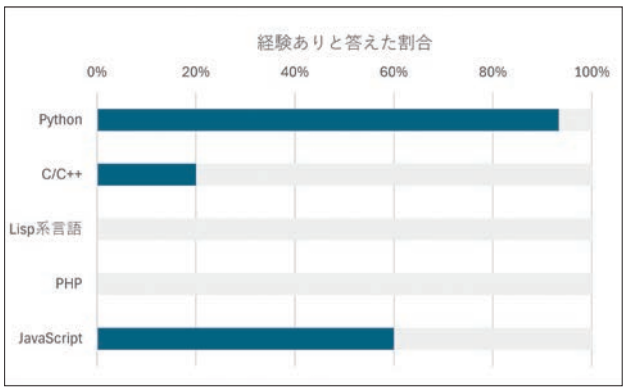

図3　授業開始前に経験のある言語（複数回答可）

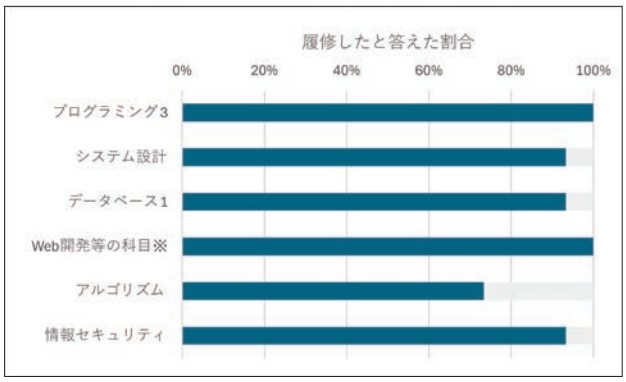

図4　これまでに履修してきた授業（複数回答可）
（※Webサイト制作/Webアプリ開発/Webサービス開発のいずれかの受講を指す）

4.2　各言語の授業の評価

各言語に対して，
・学生の主観的評価アンケート
・自由記述のアンケート
を行った．

前者では，各言語のプログラミングに対して，授業の前後での理解度を0～6点の7段階評価（0：まったく理解できなかった～6：深く理解できた）で主観的に評価してもらった．アンケートの結果を**図5**に示す．7段階評価の平均点と標準偏差，Wilcoxonの符号付順位検定の結果（*:p<.05, **:p<.01）を示した．

後者では，「学べたことや今後学んでいきたいこと」「今後の授業の改善点や意見等」を自由記述してもらった．次項の各言語の結果では「学べたことや今後学んでいきたいこと」について主に述べ，「今後の改善点や意見等」においては節の最後で述べる．

4.2.1　Python

受講前で3.40点，受講後で4.13点と，全体を通して高得点である．授業で取り扱った項目はほぼ復習項目であるが，受講後のスコアが高まっている（p<.01）．

自由記述の主なものとしては
・辞書型の活用の仕方を深く学び，これによって検索できる簡単なデータベースのようなものを作ることができた
・Pythonの良いところ，悪いところがきちんと理解できた
との意見があった．

4.2.2　C/C++

受講前・受講後のスコアの平均点がそれぞれ1.33，2.40点となっており，1.07点の上昇がみられた（p<.01）．多くの学生にとってC/C++は初めてであったようであるが，自由記述などによれば一部の学生はUnityのコーディングで触れた経験があった模様である．

自由記述としては，
・大学入学時から聞いたことがあったが，難しいとも聞いていたのでその入門を本授業で学ぶことができた
・今後はGUIを学んでいきたい
との意見があった．

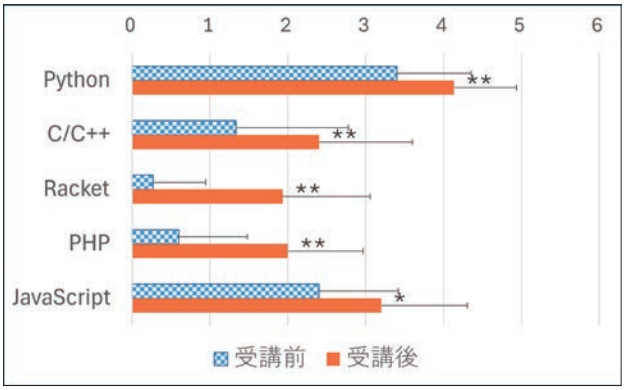

図5　学生の主観的到達度評価（*:p<.05，:p<.01）**

4.2.3　Racket

多くの学生がこれまでにあまり触れてこなかったであろう関数型言語である．受講前・受講後のスコアの平均点がそれぞれ0.27，1.93点となっており，1.67点の上昇がみられた（p<.01）．受講前の理解度が低く，受講後の理解度の伸びは大きい（5言語でもっとも大きい）が，全体的には低スコアである．

自由記述が多かったことも本言語の回答の特徴で，
・最初は理解不能だったところ，詰まっていたところが分かってからは課題の範囲ではそこそこできるようになった
・独特な書き方のプログラミングであったが個人的には中々面白かったので，この授業がなかったら知ることが無かったかもしれないと思うととても有意義だった
という前向きな意見の一方で
・最初は全く知らなかった．触れたけど凄く難しくてよくわからなかった．

・個人的に一番理解できていないプログラム言語，必要とされる場合はともかく今後は触れないかも

というように，否定的な意見も見受けられた．

4.2.4　PHP

受講前・受講後のスコアの平均点がそれぞれ0.60，2.00点となっており，1.40点の上昇がみられた(p<.01)．

自由記述としては，

・今後はウェブサイトを作っていく予定なので，後々勉強したい

・Web運用する際に気を付けることについて学びたい

との意見があった．

4.2.5　JavaScript

受講前・受講後のスコアの平均点がそれぞれ2.40，3.20点となっており，0.80点の上昇であった(p<.05)．学習状況のアンケートによれば事前にWeb開発系の授業等で学習していた学生が多く，受講前，受講後とも平均点がPythonに次いで高くなったことがそれを裏付けているといえる．その分点数の伸びは比較的小さかった．一方で自由記述に

・オブジェクト"||"の記法は知らなかった

とあるように，JavaScriptの記法については「プログラミング言語」の授業として取り扱うべき内容を含めることができたのではないかと判断される．特に「JavaScriptのオブジェクトの記法」の理解は，JavaScriptのコードを書く際だけでなく，JSONやAPIを利活用する際に重要な概念でもあるので，積極的に取り扱うべき内容であると考えられる．

4.2.5　今後の授業の改善点や意見等

各言語で自由記述してもらったが，どの言語でもおおむね同様の意見があったため本節でまとめて述べる．

・プログラム言語における基本的な部分のみやったので，何ができるかよくわからない

という意見については，学生の「何ができるか」を先に知りたいというニーズの強さが表れていると推測される．確かに，「何ができるか」を先に知ることは学習のモチベーションを維持する上で重要な要素である．その点を強調して教えることは今後の改善点であるが，筆者としては「何かができるようになるためにどのように考えるか」を重視して教えていたこともあり，認識のずれがあったかもしれない．

・FizzBuzz以外にももう少し作ってみたほうがいい

という，課題に対する意見があった．課題の難易度を高めすぎることによるドロップアウトを防ぎつつ，基本的な文法への理解を深める意図で出題したが，一部の学生には不足があったと考えられる．今後は基本的な文法問題だけでなく，より応用的で発展的な課題も同時に出題することを考えたい．

4.3　授業全体を通しての主観的評価

授業全体の主観的評価アンケート(0～6の7段階評価，0：まったくそう思わない～6：深くそう思う)の平均点を**図6**に示す．帰無仮説：中央値3と異なるとはいえない・対立仮設：中央値3と有意に異なる　としたWilcoxonの1標本検定の結果も示した(*: p<.05，**: p<.01)．いずれも平均点は3より大きいが，同検定で有意差がみられたのは問2，3，5，6であった．

5．考察

表5に，本研究の目的に対応した項目を含む各項目の達成度を表で記した．達成度はアンケート結果をもとに4段階(◎：よく達成できている　○：ある程度達成できてい

図6　授業全体の主観的評価アンケート（*:p<.05，:p<.01）**

る　△：一部達成できているが改善が必要　×：達成できていない）で判断している．以下，それぞれの項目の考察について記す．

5.1　言語スキルの習得に関する考察

それぞれの言語において，授業前後で理解度のスコアが上昇しており，すべてで有意差が出ている．このことから，授業が「各言語の文法等の習得」に対して，学生の理解度に正の影響を与えたと推察される．一方で，多くの学生が既習の言語（Python，JavaScript．以下，既習言語と記す）とそれ以外の言語（未習言語）では，平均値が大きく異なっている．授業がオムニバス的であることの限界として，1つの言語に深く触れられない（各項目で平均しておおよそ100分授業2コマが目安）ことが挙げられる．過去のカリキュラムや他の授業（例としてプログラミング3）の授業との制約などからUMLなどの項目も取り扱わなければならなかったことの影響もあった．来年度以降はUMLをプログラミング3での学習項目に移行予定であるため，この項目については改善を図りたいと考えている．

既習言語への学習内容の反映については，本授業前後で既習言語のスコアに有意な（ただし，未習言語と比べては小幅な）上昇がみられたことから，ある程度は達成できたと考えられる．ただし．より厳密に評価するならば，長期的な視点で，例えば学生のプログラミングによる制作物による評価などを含んだ，形成的・総括的評価が必要であろう．

5.2　情報科学的理解に関する考察

「計算論的理解」の一部として，主観的評価アンケート（**図6**）の問3.プログラミングで問題解決を図る場合にいろいろな手段があること（計算の意味論）について理解したかを問うた．この結果，学生のスコアが有意に中央値3点を超えていたことから本目的はおおむね達成できたと考えらえる．ただし，試験等による客観的評価がなされているわけではないことに留意しなければならない．（補足として，先に「計算論」の例として「計算の意味論」「計算量」を挙げたが，このうち「計算量」については別の授業（プログラミング3．前述）で扱い，本授業では扱っていないので問うていない．）同アンケート問2の「ある言語の得意なアルゴリズムを理解して，他の言語に応用できるようになったと思った」が有意に中央値3点を超えていたことも，学生の計算論的理解の裏付けているのではなかろうかと推測される．

「アーキテクチャへの理解」であるが，この項目は主にC/C++の授業で説明することを想定し，実際にポインタに関しては図を使って説明した．具体的にはポインタやメモリ管理の項目で理解を促すことを検討していた．ただし，学生の理解度の問題や時間の制約からこれらを深く取り扱うことができなかったことは改善すべき点である．実際に，本項目に対応する主観的評価アンケート（**図6**）の問4.では，中央値3点との有意差はみられなかった．そのためこの目的に対しては，授業の改善が必要と思われる．

5.3　その他項目に関する考察

主観的評価アンケート（**図6**）の問5.「複数のプログラミング言語の開発環境を整えられたことは役立つ」問6.「本授業で複数のプログラミング言語を学べたことは有意義だった」との項目で，中央値3点との有意差がみられた．

このうち，前者に関しては特筆すべき事例があった．本授業のある受講生が，他の授業（システム開発プロジェクト）において，本授業で取り扱った開発・実行環境（PHP，XAMPP）を用いて問題解決を図った事例があった．当該学生によれば，本授業で環境を整備したので，他授業での問題解決に活用できるのではないかと考えたとのことであった．問題解決を図るためのツールとして，複数の言語を扱えるようにすることの有効性を示唆する事例であった．このことから，授業を実施するために副次的に行われた「開発環境の整備」については想定以上の成果を上げたと考えられ，「よく達成できている」とした．

なお，授業への満足感については，前述の問6.「本授業で複数のプログラミング言語を学べたことは有意義だった」で中央値3点との有意差がみられた一方で問7.「本授業の受講を来年度以降の学生に薦めたいと思う」に対して有意差がみられなかったことから（ただし，分散が他項目より若干多いため個人差があると思われる），「おおむね達成した」との判断に至った．

表5　各項目の達成度

大分類	項目	評価
言語スキルの習得	各言語の文法等の習得	○
	（Pythonを含めた）既習言語への学習内容の反映	○
情報科学的理解	計算論的理解	○
	アーキテクチャの理解	△
その他	開発環境の整備	◎
	授業への満足感	○

◎：よく達成できている　○：ある程度達成できている
△：一部達成できているが改善が必要　×：達成できていない（該当なし）

6. まとめと今後の課題

本稿では，「文系」情報学科学生に対する第3学年後期の開講科目「プログラミング4」において，さまざまなプログラミング言語を教えた事例について報告した．事後アンケートにより，情報科学的な考え方の習得を図り，主に計算機論的思考についてはある程度達成されたと推測される．しかしながらアーキテクチャの理解は一部達成にとどまったとみられ，改善が必要と判断された．

また，各言語の習得については短い時間の取り扱いながらもある程度達成され，既習言語のスキルに対してプラスに働くのではないかと予想された．副次的な効果として，受講者が各言語の開発環境を持つに至り，他の授業でこの開発環境が役立った事例もあった．

本稿では授業を主観的に評価したが，学生の成果物当は評価の対象外となっている．また，比較実験的に調査したわけでもない．今後は学生の成果物を含めた授業の評価が必要になると考えられる．

また，現時点で，アンケート結果からいくつかの課題が挙がっている．次年度以降，カリキュラムの微修正などを通して改善を図っていきたいと考えている．

謝辞

本稿の対象とする「プログラミング4」授業において，日本事務機株式会社より資料提示のためのツール「BOOK MARRY」（日本事務機 2024）の無償モニターサービスの提供を受けた．

参考文献

みずほ情報総研（2019）平成30 年度我が国におけるデータ駆動型社会に係る基盤整備（IT　人材等育成支援のための調査分析事業）－ IT 人材需給に関する調査 －調査報告書

布施 泉（2018）大学の一般教育としてのプログラミング教育，システム／制御／情報62（7），266-271

布施 泉・岡部成玄・中西通雄（2017）大学初年次の一般情報教育におけるCPU シミュレータを用いたプログラム作成によるコンピュータ動作の教育，高等教育ジャーナル：高等教育と生涯学習，24，97-105

Kaminishi，H. & Murota，M.（2011）Development and Evaluation of a New Presentation Software Program（CodEx）for Teaching Programming Code，*Research and Practice in Technology Enhanced Learning*. 6（2），83-106

上西秀和・室田真男（2014）Webプログラミング言語教育用HTMLプレゼンテーション”CodEx”のためのスライド作成・編集ソフトの開発と評価，教育システム情報学会誌，31（2），pp.172-184

上西秀和（2014）Web プログラミング言語教育のためのプレゼンテーションシステムの開発と有効性の評価，東京工業大学　博士論文

クジラ飛行机（2020）プログラミング言語大全，技術評論社

江戸川大学（2024）学校法人江戸川学園 江戸川大学 | 情報文化学科，https://www.edogawa-u.ac.jp/colleges/d_informatics/（参照日2024年1月10日）

渡辺　治・米崎直樹.（1997）計算論入門　計算の基本原理理解のために，日本評論社

マイクロソフト.（2024）Visual Studio Code – コード エディタ ー，https://azure.microsoft.com/ja-jp/products/visual-studio-code（参照日2024年1月30日）

サスマン・エイブルソン・サスマン・和田英一（訳）（2000）計算機プログラムの構造と解釈　第二版，ピアソン，

SlashData（2021）State of the Developer Nation 21st Edition.

日本事務機.（2024）BOOK MARRY ¦ ネオシリウス，https://www.njc.co.jp/neocilius/bm/（参照日 2024年1月24日）