福岡県立大学人間社会学部紀要
2023, Vol. 32, No. 1, 21－35

原著

福岡県立大学人間社会学部における「プログラミング概論」の教育効果（2022年度）
－PythonとJavaScriptの基礎的なスキル習得と授業形式の比較－

石　崎　龍　二*

要旨　福岡県立大学人間社会学部で2022年（令和４年）度に開講された「プログラミング概論」について、プログラミング言語の基礎的なスキルの習得状況に関する質問紙調査やeラーニング確認テストなどの観点から教育効果を考察した。

調査の結果、Python、JavaScriptの基礎的なスキルに関しては、受講前に比べて受講後での受講生の自己評価が有意に高く、授業での教育効果が認められた。さらに、対面形式で行った2022年（令和４年）度の授業は、オンライン形式で行った2020年（令和２年）度の授業と比較して、受講後での受講生の自己評価が高かったことがわかった。これは、「わからない時はその場で質問できる」「授業のペースに合わせて学習できる」といった対面形式の利点に加え、録画を後から視聴できるようにしたことが要因として考えられる。ただし、受講後でも自己評価が低い項目が複数あり、授業内容や課題の難易度を適切に調整することやタイピングが苦手な受講生の入力時間を考慮した授業進行の工夫が必要であることが示唆された。

キーワード　プログラミング教育、対面授業、オンライン授業、教育効果、ｅラーニング

1　研究背景と目的

近年、IoTの進展に伴い、大量のビッグデータが蓄積され、その利活用が進んでいる。また、AI技術などを利用した新しいアルゴリズム手法も盛んに開発されている。このような時代において、教育現場でも情報通信技術を手段として主体的に活用できる技能の習得が求められるようになってきた。文部科学省では、児童生徒向けの１人１台学習者用端末と高速大容量の通信ネットワークを一体的に整備するGIGAスクール構想が進められており、高等学校でも、学習指導要領改訂[1)]に伴い、2022年（令和４年）度の入学生から「情報Ⅰ」が必修科目となり、プログラミング教育を必ず受けることになった。

このような変化に合わせて、大学においてもプログラミング教育の見直しが必要となってい

＊福岡県立大学人間社会学部・教授

る。2020年（令和２年）度から文部科学省による「数理・データサイエンス・AI教育プログラム認定制度」が開始され、全国の大学・短期大学・高等専門学校を対象として、データサイエンス教育が推進されている[2)]。当大学では、データサイエンス・プログラムを全学横断型教育プログラムとして導入しており[3)]、保健福祉分野における課題解決に必要な統計学や情報学の知識やスキルを修得し、それらを応用する力を養成することを目的としている。第１段階では、数学、統計学、情報学、情報処理の基礎を学び、第２段階では、統計学や情報学の専門基礎を身につけ、最後に第３段階では、統計学・情報学の演習により応用力を磨くことを目指している。また、当大学人間社会学部では、社会調査やデータ分析、情報処理などの専門分野に必要な知識やスキルを身につけるために、社会調査や情報処理の科目を設置している。所定の単位を取得すれば、上級情報処理士や社会調査士の資格が取得できる。

2020年（令和２年）度から、小学校におけるプログラミング教育の必修化が始まった。これにより、2028年（令和10年）度以降に大学に入学する学生たちは、小学校５年生や６年生からこの教育を受けた世代となる。大井・堀江（2023）によると、高等学校での情報科におけるPythonを用いたプログラミングの授業の実践において、いくつかの課題があるものの、生徒の提出物や試験の結果からプログラミングの学習にそれほど困難さはみられなかったとしている[4)]。このように大学入学前でのプログラミング教育の強化を受けて、大学ではどのようなプログラミング教育を提供するべきかという課題が生じている。特に、文系の学部においては、プログラミング教育における課題が多いことが調査により明らかとなっている。葛西・金澤・大島・末次・渡邉（2022）が実施した調査によれば、文系大学生の多くが高等学校を卒業するまでにプログラミングを学習しておらず、AIを学ぶための数学やコンピュータに対する苦手意識があることが指摘されている[5)]。齋藤・隈本・池田・平山（2022）は、文系学部におけるPythonを用いたプログラミングとデータサイエンス導入に関する意識調査を実施し、学生の興味関心は高く、知識の重要性についても理解は得られているものの、プログラミングやデータサイエンスのスキルを身につけることに積極的な回答は得られなかったとしている[6)]。

このような文系大学生の状況を考慮した上で、文系学部におけるプログラミング教育をはじめとするデータサイエンス教育の進め方について、どのような方法が適切かを考える必要がある。金子（2022）は、高等学校と大学で、Pythonを使用したプログラミングの授業を実施し、授業前後での学習者の興味・関心や学習意欲の変化について分析した。授業実施により、プログラミングへの「興味」が高まった点、授業を行う前の「興味」「必要」が高ければ学習「意欲」も高くなることを実証し、プログラミングの学習者には「興味」「必要」「意欲」のいずれかを高める必要性を指摘している[7)]。

プログラミングの学習への興味や関心を高めるには、学習者が学んだ内容を理解できることが不可欠である。内容を理解できなければ、プログラミングの学習への「興味」「必要」「意欲」にはつながりにくい。道越・奥井・丸野（2019）は、文系学部におけるプログラミング教育の教育効果に関して、ｅラーニングシステムの学習履歴データから教育効果を分析した結果、プログラミングの授業を終えた段階でも、習熟度が

不十分であり、初回と最終回でプログラミングの難しさに対する認識に変化が見られなかったという課題を示している[8)]。当大学でも、人間社会学部で２年次に開講している「プログラミング概論」の教育効果に関する質問紙調査を、2020年（令和２年）度に実施し、プログラミングの基礎的なスキルの習得に関しては教育効果が認められるものの、スキルの定着までには至ってはおらず、教育方法に改善の余地があることを示した[9)]。

文系学部におけるプログラミング教育の教育効果を向上させるためには、プログラミング言語の学習において、具体的にどの部分に難点があるのかを明確にし、教育方法の改善点を見い出すことが重要である。そこで本稿では、2022年（令和４年）度に本学人間社会学部で実施した「プログラミング概論」の授業に関する質問紙調査やｅラーニング確認テスト等の結果を分析し、具体的な難点がどこにあるのか、さらにどのように教育方法を改善すれば良いのかについて検討した。2022年（令和４年）度は授業をすべて対面形式で行ったが、2020年（令和２年）度は新型コロナウイルス感染拡大防止のためすべての授業をオンライン形式で行っていた。そのため、このような授業形式の違いによる教育効果の比較も行った。

2　方法

本研究では、以下の質問紙調査を行った。

⑴　事前事後調査

調査対象

福岡県立大学人間社会学部で2022年（令和４年）度後期に開講された「プログラミング概論」の受講者52名。

調査方法

「プログラミング概論」の授業時に、ｅラーニングシステムを使って質問紙調査を実施した（ｅラーニングシステム上には、個人を特定する情報は記録されない）。

調査時期

調査は２回実施した。１回目は、「プログラミング概論」の初回の授業開始時（2022（令和４）年10月）、２回目は、「プログラミング概論」の最終回の授業終了時（2023（令和５）年２月）に実施した。

調査項目

受講前の調査項目は、所属に関するもの（２項目）、資格取得に関するもの（２項目）、他の科目の履修状況に関するもの（６項目）、プログラミングの知識に関するもの（９項目）、JavaScriptのスキルに関するもの（24項目）、Pythonのスキルに関するもの（18項目）、プログラミング全般のスキルに関するもの（１項目）、PCの利用状況に関するもの（６項目）、授業への要望に関する自由記述（１項目）、以上の全69項目である。

受講後の調査項目は、所属に関するもの（２項目）、資格取得に関するもの（２項目）、他の科目の履修状況に関するもの（２項目）、JavaScriptのスキルに関するもの（24項目）、Pythonのスキルに関するもの（18項目）、JavaScript及びPythonのスキルに関するもの（１項目）、PCの利用状況に関するもの（５項目）、授業形式に関するもの（４項目）、自由記述（３項目）、以上の全61項目である。

これらの調査項目のうち、受講前と受講後で、所属に関するもの（２項目）、資格取得に関するもの（２項目）、JavaScriptのスキルに関するもの（24項目）、Pythonのスキルに関するもの（18項目）、PCの利用状況に関するもの（５項目）が同じ項目である。

回答者の内訳

調査回答者は表１の通りである。

表１　受講前後の調査の回答者数

	回答者数（人）	受講者数（人）	回答率（%）
受講前	48	52	92.3
受講後	29	52	55.8

３　調査結果

３．１　プログラミング言語の基礎的なスキルの習得

「プログラミング概論」は、データサイエンス・プログラムの第２段階に位置づく科目で、プログラミングに必要な基本的な要素やアルゴリズムの基本となる制御構造を理解し、学生が問題に応じて適切に変数の宣言や制御文、関数を使ったプログラミングができるようにすることを履修目標としている。代表的なプログラミング言語を例にして、変数、データ型、演算子、配列などのプログラミングの基本的な要素や、順次、分岐、反復などのアルゴリズムの基本となる制御構造、関数の作成方法、ファイル処理などの手法を学習することとしている。

当該授業で取り扱うプログラミング言語は、PythonとJavaScriptである。Pythonは高水準言語であり、プログラムが比較的読みやすく、書きやすいプログラミング言語であり、数値計算、Web開発、データ解析、人工知能における機械学習など、幅広い分野で活用されているため、学習言語として選択している。また、JavaScriptはスクリプト言語であり、Webサイトに動的な処理を実装できる言語であり、比較的学びやすいことから、Web開発おいて基本的なプログラミングスキルを習得するための学習言語として選択している。さらに、異なるプログラミング言語を学ぶことにより、プログラミング言語の概念、構造や機能についてより深く理解できるようになり、新たなプログラミング言語を学習する力にもつながると考えている。2022年４月から高等学校で使用される「情報Ⅰ」の教科書でも、PythonとJavaScriptが数多く取り扱われている。

受講前でのプログラミング言語の学習経験については、表２に示す通り、最も多かったのはJavaであったが、その割合は10.4%（5）でしかなかった。プログラミング言語を学んでないと回答した割合は、「わからない」を含めて実に79.2%（38）であった。金澤ら（2022）が報告した結果と同様に、当該授業で初めてプログラミング言語を学ぶ受講生が多いことがわかる。

表２　受講前でのプログラミング言語の学習経験（N＝48）（複数回答）

プログラミング言語	回答者数（人）	割合（%）
Java	5	10.4
HTML/CSS	4	8.3
JavaScript	3	6.3
C	1	2.1
Ruby	1	2.1
R	1	2.1
Python	0	0.0
その他の言語	0	0.0
わからない	10	20.8
学んだことはない	28	58.3

３．２　Pythonの基礎的なスキルの習得

Pythonの基礎的なスキルの習得に関する調査について、受講前と受講後での自己評価の変

化を以下に示す。数の四則演算と演算結果の画面への出力については、表３に示す通り、受講前と比べて受講後の自己評価が３項目とも有意に高いことがわかった。ここでは、「十分できる」を４、「ややできる」を３、「あまりできない」を２、「全くできない」を１とし、Wilcoxonの順位和検定の片側検定を行った。以降の検定も全て同様に行った。

表３　数の四則演算と演算結果の画面への出力プログラムの作成

項目		自己評価	受講前（N＝48）		受講後（N＝29）		
			（人）	（%）	（人）	（%）	
画面へ文字を表示するプログラム		十分できる	0	0.0	17	58.6	**
		ややできる	0	0.0	10	34.5	
		あまりできない	1	2.1	2	6.9	
		全くできない	47	97.9	0	0.0	
計算のプログラム	数の四則演算	十分できる	1	2.1	17	58.6	**
		ややできる	0	0.0	7	24.1	
		あまりできない	3	6.3	5	17.2	
		全くできない	44	91.7	0	0.0	
	実数の多項式の計算	十分できる	0	0.0	9	31.0	**
		ややできる	1	2.1	13	44.8	
		あまりできない	2	4.2	7	24.1	
		全くできない	45	93.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

分岐処理のプログラムについても、表４に示す通り、受講前と比べて受講後の自己評価が２項目とも有意に高いことがわかった。

表４　分岐処理のプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
if文による分岐処理	十分できる	0	0.0	13	44.8	**
	ややできる	1	2.1	13	44.8	
	あまりできない	3	6.3	3	10.3	
	全くできない	44	91.7	0	0.0	
if ～ elif 文による分岐処理	十分できる	0	0.0	13	44.8	**
	ややできる	1	2.1	12	41.4	
	あまりできない	2	4.2	4	13.8	
	全くできない	45	93.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

反復処理のプログラムについても、表５に示す通り、受講前と比べて受講後の自己評価が２項目とも有意に高いことがわかった。

表５　反復処理のプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
for文による反復処理のプログラム	十分できる	0	0.0	4	13.8	**
	ややできる	0	0.0	17	58.6	
	あまりできない	2	4.2	8	27.6	
	全くできない	46	95.8	0	0.0	
while文による反復処理のプログラム	十分できる	0	0.0	3	10.3	**
	ややできる	1	2.1	18	62.1	
	あまりできない	2	4.2	8	27.6	
	全くできない	45	93.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

リストの活用についても、表６に示す通り、受講前と比べて受講後の自己評価が３項目とも有意に高いことがわかった。しかし、「２次元リストの活用」については、受講後でも、「十分できる」又は「ややできる」の回答率が55.2％と低い。

表６　リストを活用したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
１次元リストの活用	十分できる	0	0.0	6	20.7	**
	ややできる	1	2.1	14	48.3	
	あまりできない	2	4.2	9	31.0	
	全くできない	45	93.8	0	0.0	
２次元リストの活用	十分できる	0	0.0	4	13.8	**
	ややできる	1	2.1	12	41.4	
	あまりできない	1	2.1	13	44.8	
	全くできない	46	95.8	0	0.0	
文字列リストの活用	十分できる	0	0.0	4	13.8	**
	ややできる	1	2.1	16	55.2	
	あまりできない	1	2.1	9	31.0	
	全くできない	46	95.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

ユーザ定義関数の作成についても、表７に示す通り、受講前と比べて受講後の自己評価が４項目とも有意に高いことがわかった。しかし、「リストの引き渡しを行うユーザ定義関数を使ったプログラム作成」については、受講後でも、「十分できる」又は「ややできる」の回答率が58.6％と低い。

表７　ユーザ定義関数を活用したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
ユーザ定義関数（値を返さない、引数のない関数）を使ったプログラム作成	十分できる	0	0.0	6	20.7	**
	ややできる	1	2.1	17	58.6	
	あまりできない	1	2.1	5	17.2	
	全くできない	46	95.8	1	3.4	
ユーザ定義関数（値を返さない、引数のある関数（参照による呼び出し））を使ったプログラム作成	十分できる	0	0.0	6	20.7	**
	ややできる	0	0.0	15	51.7	
	あまりできない	2	4.2	8	27.6	
	全くできない	46	95.8	0	0.0	
ユーザ定義関数（値を返し、引数のある関数）を使ったプログラム作成	十分できる	0	0.0	8	27.6	**
	ややできる	0	0.0	13	44.8	
	あまりできない	2	4.2	8	27.6	
	全くできない	46	95.8	0	0.0	
リストの引き渡しを行うユーザ定義関数を使ったプログラム作成	十分できる	0	0.0	4	13.8	**
	ややできる	0	0.0	13	44.8	
	あまりできない	2	4.2	12	41.4	
	全くできない	46	95.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

関数の活用についても、表８に示す通り、受講前と比べて受講後の自己評価が２項目とも有意に高いことがわかった。しかし、「文字列操作関数の活用」については、受講後でも、「十分できる」又は「ややできる」の回答率が58.6％と低い。

表８　関数を活用したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
文字列操作関数の活用	十分できる	0	0.0	2	6.9	**
	ややできる	0	0.0	15	51.7	
	あまりできない	2	4.2	12	41.4	
	全くできない	46	95.8	0	0.0	
数学関数の活用	十分できる	0	0.0	5	17.2	**
	ややできる	0	0.0	14	48.3	
	あまりできない	2	4.2	10	34.5	
	全くできない	46	95.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

「クラスの定義とインスタンス作成」についても、表９に示す通り、受講後の自己評価の方が有意に高いことがわかった。しかし、受講後でも、「十分できる」又は「ややできる」の回答率は受講後でも37.9％と極めて低い。

表９　クラスの定義とインスタンスを作成するプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
クラスを定義し、インスタンスを作成	十分できる	0	0.0	3	10.3	**
	ややできる	0	0.0	8	27.6	
	あまりできない	2	4.2	18	62.1	
	全くできない	46	95.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

「ファイルの入出力関数の活用」についても、表10に示す通り、受講後の自己評価の方が有意に高いことがわかった。しかし、受講後でも、「十分できる」又は「ややできる」の回答率が55.2％と低い。

表10　ファイル入出力関数を活用したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
ファイル入出力関数を使ったプログラム作成	十分できる	0	0.0	3	10.3	**
	ややできる	0	0.0	13	44.8	
	あまりできない	1	2.1	13	44.8	
	全くできない	47	97.9	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

３．３　JavaScriptの基礎的なスキルの習得

JavaScriptの基礎的なスキルの習得に関する調査について、受講前と受講後での自己評価の変化を以下に示す。変数の宣言については、表 11に示す通り、受講前と比べて受講後の自己評価が３項目とも有意に高いことがわかった。

表11　変数を宣言したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
数値型変数の宣言	十分できる	0	0.0	3	10.3	**
	ややできる	1	2.1	21	72.4	
	あまりできない	3	6.3	5	17.2	
	全くできない	44	91.7	0	0.0	
文字型変数の宣言	十分できる	0	0.0	5	17.2	**
	ややできる	0	0.0	19	65.5	
	あまりできない	4	8.3	5	17.2	
	全くできない	44	91.7	0	0.0	
配列変数の宣言	十分できる	0	0.0	1	3.4	**
	ややできる	0	0.0	18	62.1	
	あまりできない	3	6.3	10	34.5	
	全くできない	45	93.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

数の四則演算と演算結果の画面への出力についても、表12に示す通り、受講前と比べて受講後の自己評価が３項目とも有意に高いことがわかった。

表12　数の四則演算と演算結果の画面への出力プログラムの作成

項目		自己評価	受講前（N＝48）		受講後（N＝29）		
			（人）	（%）	（人）	（%）	
画面への文字を表示するプログラム		十分できる	0	0.0	18	62.1	**
		ややできる	2	4.2	7	24.1	
		あまりできない	1	2.1	4	13.8	
		全くできない	45	93.8	0	0.0	
計算のプログラム	数の四則演算	十分できる	0	0.0	16	55.2	**
		ややできる	1	2.1	7	24.1	
		あまりできない	5	10.4	6	20.7	
		全くできない	42	87.5	0	0.0	
	実数の多項式の計算	十分できる	0	0.0	10	34.5	**
		ややできる	0	0.0	12	41.4	
		あまりできない	3	6.3	7	24.1	
		全くできない	45	93.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

分岐処理のプログラムについても、表13に示す通り、受講前と比べて受講後の自己評価が３項目とも有意に高いことがわかった。しかし、「switch文による分岐処理のプログラム」については、受講後でも、「十分できる」又は「ややできる」の回答率が51.7％と低い。

表13　分岐処理のプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
if文による分岐処理	十分できる	0	0.0	10	34.5	**
	ややできる	1	2.1	13	44.8	
	あまりできない	3	6.3	6	20.7	
	全くできない	44	91.7	0	0.0	
if～else文による分岐処理	十分できる	0	0.0	10	34.5	**
	ややできる	0	0.0	13	44.8	
	あまりできない	4	8.3	6	20.7	
	全くできない	44	91.7	0	0.0	
switch文による分岐処理	十分できる	0	0.0	4	13.8	**
	ややできる	0	0.0	11	37.9	
	あまりできない	2	4.2	14	48.3	
	全くできない	46	95.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

反復処理のプログラムについても、表14に示す通り、受講前と比べて受講後の自己評価が３項目とも有意に高いことがわかった。

表14　反復処理のプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
for文による反復処理のプログラム	十分できる	0	0.0	2	6.9	**
	ややできる	0	0.0	19	65.5	
	あまりできない	2	4.2	8	27.6	
	全くできない	46	95.8	0	0.0	
while文による反復処理のプログラム	十分できる	0	0.0	3	10.3	**
	ややできる	0	0.0	16	55.2	
	あまりできない	2	4.2	10	34.5	
	全くできない	46	95.8	0	0.0	
do～while文による反復処理のプログラム	十分できる	0	0.0	2	6.9	**
	ややできる	0	0.0	16	55.2	
	あまりできない	2	4.2	11	37.9	
	全くできない	46	95.8	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

配列の活用についても、表15に示す通り、受講前と比べて受講後の自己評価が３項目とも有意に高いことがわかった。しかし、３項目とも、受講後でも、「十分できる」又は「ややできる」の回答率が60％未満と低い。

表15　配列を活用したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
１次元配列の活用	十分できる	0	0.0	3	10.3	**
	ややできる	0	0.0	14	48.3	
	あまりできない	2	4.2	12	41.4	
	全くできない	46	95.8	0	0.0	
２次元配列の活用	十分できる	0	0.0	2	6.9	**
	ややできる	0	0.0	15	51.7	
	あまりできない	2	4.2	12	41.4	
	全くできない	46	95.8	0	0.0	
文字型配列の活用	十分できる	0	0.0	3	10.3	**
	ややできる	0	0.0	12	41.4	
	あまりできない	1	2.1	11	37.9	
	全くできない	47	97.9	3	10.3	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

ユーザ定義関数の作成についても、表16に示す通り、受講前と比べて受講後の自己評価が４項目とも有意に高いことがわかった。しかし、「配列の引き渡しを行うユーザ定義関数を使ったプログラムを作成」については、受講後でも、「十分できる」又は「ややできる」の回答率が44.8％と低い。

表16　ユーザ定義関数を活用したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
ユーザ定義関数（値を返さない、引数のない関数）を使ったプログラム作成	十分できる	0	0.0	9	31.0	**
	ややできる	0	0.0	13	44.8	
	あまりできない	1	2.1	7	24.1	
	全くできない	47	97.9	0	0.0	
ユーザ定義関数（値を返さない、引数のある関数（参照による呼び出し））を使ったプログラム作成	十分できる	0	0.0	7	24.1	**
	ややできる	0	0.0	16	55.2	
	あまりできない	1	2.1	6	20.7	
	全くできない	47	97.9	0	0.0	
ユーザ定義関数（値を返し、引数のある関数）を使ったプログラム作成	十分できる	0	0.0	6	20.7	**
	ややできる	0	0.0	14	48.3	
	あまりできない	1	2.1	9	31.0	
	全くできない	47	97.9	0	0.0	
配列の引き渡しを行うユーザ定義関数を使ったプログラム作成	十分できる	0	0.0	2	6.9	**
	ややできる	0	0.0	11	37.9	
	あまりできない	1	2.1	16	55.2	
	全くできない	47	97.9	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

関数の活用についても、表17に示す通り、受講前と比べて受講後の自己評価が２項目とも有意に高いことがわかった。しかし、２項目とも、受講後でも、「十分できる」又は「ややできる」の回答率が60％未満と低い。

表17　関数を活用したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
文字列操作関数の活用	十分できる	0	0.0	1	3.4	**
	ややできる	0	0.0	13	44.8	
	あまりできない	1	2.1	15	51.7	
	全くできない	47	97.9	0	0.0	
数学関数の活用	十分できる	0	0.0	6	20.7	**
	ややできる	0	0.0	11	37.9	
	あまりできない	1	2.1	12	41.4	
	全くできない	47	97.9	0	0.0	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

構造体の活用についても、表18に示す通り、受講前と比べて受講後の自己評価が２項目とも有意に高いことがわかった。しかし、受講後でも、２項目とも「十分できる」又は「ややできる」の回答率が40％未満と極めて低い。

表18　構造体を活用したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
構造体の活用	十分できる	0	0.0	1	3.4	**
	ややできる	0	0.0	8	27.6	
	あまりできない	1	2.1	17	58.6	
	全くできない	47	97.9	3	10.3	
構造体配列の活用	十分できる	0	0.0	1	3.4	**
	ややできる	0	0.0	9	31.0	
	あまりできない	2	4.2	16	55.2	
	全くできない	46	95.8	3	10.3	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

ファイルの入出力関数の活用についても、表19に示す通り、受講前と比べて受講後の自己評価が有意に高いことがわかった。しかし、受講後でも「十分できる」又は「ややできる」の回答率が50％未満と低い。

表19　ファイル入出力関数を活用したプログラムの作成

項目	自己評価	受講前（N＝48）		受講後（N＝29）		
		（人）	（%）	（人）	（%）	
ファイル入出力関数を使ったプログラムを作成	十分できる	0	0.0	3	10.3	**
	ややできる	0	0.0	11	37.9	
	あまりできない	2	4.2	13	44.8	
	全くできない	46	95.8	2	6.9	

受講前後での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

3．4　eラーニングの確認テストから見る学習の定着度

前節までは、PythonとJavaScriptのプログラミングスキルに関する自己評価の受講前後での変化について考察した。本節では、eラーニング上の確認テストの達成状況について考察する。確認テストは、プログラミング言語に関する専門用語の理解度を確認するためのテストである。出題は選択処理におけるif文の使い方、反復処理におけるfor文の使い方、リストや配列の活用などプログラミング言語に関する基礎的な知識を確認するための選択形式の問題を課した。表20は、各回での確認テストの達成度を示す。確認テストには複数回トライすることができる。達成度は、各回の問題を全て正解の場合を100点として、その割合を受講生の平均点として算出したものである。Pythonの学習においては2次元リストの達成度が相対的に低く、JavaScriptの学習においては選択処理（分岐）の達成度が相対的に低い傾向がみられる。ただし、確認テスト全体の平均点が、2020年（令和2年）度の68.8点から大幅に向上し、93.3点に達している。これはプログラミング言語の専用用語に対する理解度が向上していることを示唆しており、良い結果と言える。

表20　eラーニング上の確認テストの達成度（N＝52）

回	授業内容	達成度（点）
1	プログラミングの概要	97.2
2	基本的なデータ表現、Python－簡単なデータの入出力	99.1
3	Python－数値データの入力・計算・出力、選択処理（分岐）	93.2
4	Python－選択処理（分岐）	98.0
5	Python－反復処理（繰り返し）	100.0
6	Python－1次元リスト	97.3
7	Python－2次元リスト	90.2
8	Python－文字列のリスト、関数	95.8
9	Python－関数の作り方	98.7
10	Python－文字列操作関数	95.0
11	Python－数学関数	93.6
12	Python－クラス、ファイル処理、JavaScript－データの入出力	97.1
13	JavaScript－選択処理（分岐）	85.8
14	JavaScript－反復処理（繰り返し）、配列、関数	98.8
15	JavaScript－メニュー、フォーム、ボタンの配置等	92.6

4　授業形式の違いによる教育効果の比較

2022年（令和4年）度においては、原則として対面形式で情報処理教室での授業を行い、授業の録画を後からeラーニングから視聴できるようにした。一方、2020年（令和2年）度においては、Zoomを使用して同期型のオンライン形式で授業を行い、後から録画を視聴できるようにした。これらの授業形式の違いが教育効果に与えた影響を検証するため、授業終了時の受講生の自己評価を比較した。

Pythonの基礎的なスキルの習得については、2022年（令和4年）度の方が18項目中17項目で「十分できる」又は「ややできる」の回答率が高く、11項目で統計的に有意に高い結果となった（表21）。JavaScriptの基礎的なスキルの習得についても、2022年（令和4年）度の方が24項目中20項目で「十分できる」又は「ややできる」の回答率が高く、9項目で統計的に有意に高い結果となった（表22）。

表21　受講後のPythonの基礎的なスキルの習得（2020年度［n＝27］と2022年度［n＝29］）

項目		自己評価	2020年度		2022年度		
			(人)	(%)	(人)	(%)	
画面へ文字を表示するプログラム		十分できる	9	33.3	17	58.6	*
		ややできる	14	51.9	10	34.5	
		あまりできない	4	14.8	2	6.9	
		全くできない	0	0.0	0	0.0	
計算のプログラム	数の四則演算	十分できる	6	22.2	17	58.6	*
		ややできる	16	59.3	7	24.1	
		あまりできない	5	18.5	5	17.2	
		全くできない	0	0.0	0	0.0	
	実数の多項式の計算	十分できる	3	11.1	9	31.0	*
		ややできる	12	44.4	13	44.8	
		あまりできない	11	40.7	7	24.1	
		全くできない	1	3.7	0	0.0	
分岐処理のプログラム	if文による分岐処理	十分できる	2	7.4	13	44.8	**
		ややできる	16	59.3	13	44.8	
		あまりできない	9	33.3	3	10.3	
		全くできない	0	0.0	0	0.0	
	if～elif文による分岐処理	十分できる	1	3.7	13	44.8	**
		ややできる	17	63.0	12	41.4	
		あまりできない	9	33.3	4	13.8	
		全くできない	0	0.0	0	0.0	
反復処理のプログラム	for文による反復処理のプログラム	十分できる	1	3.7	4	13.8	n.s
		ややできる	15	55.6	17	58.6	
		あまりできない	11	40.7	8	27.6	
		全くできない	0	0.0	0	0.0	
	while文による反復処理のプログラム	十分できる	1	3.7	3	10.3	n.s
		ややできる	15	55.6	18	62.1	
		あまりできない	11	40.7	8	27.6	
		全くできない	0	0.0	0	0.0	
リストの活用	１次元リストの活用	十分できる	1	3.7	6	20.7	*
		ややできる	12	44.4	14	48.3	
		あまりできない	14	51.9	9	31.0	
		全くできない	0	0.0	0	0.0	
	２次元リストの活用	十分できる	1	3.7	4	13.8	n.s
		ややできる	11	40.7	12	41.4	
		あまりできない	15	55.6	13	44.8	
		全くできない	0	0.0	0	0.0	
	文字列リストのをプログラムで活用	十分できる	1	3.7	4	13.8	*
		ややできる	11	40.7	16	55.2	
		あまりできない	15	55.6	9	31.0	
		全くできない	0	0.0	0	0.0	
ユーザ定義関数の作成	ユーザ定義関数（値を返さない、引数のない関数）を使ったプログラム作成	十分できる	2	7.4	6	20.7	**
		ややできる	11	40.7	17	58.6	
		あまりできない	13	48.1	5	17.2	
		全くできない	1	3.7	1	3.4	
	ユーザ定義関数（値を返さない、引数のある関数（参照による呼び出し））を使ったプログラム作成	十分できる	2	7.4	6	20.7	*
		ややできる	11	40.7	15	51.7	
		あまりできない	13	48.1	8	27.6	
		全くできない	1	3.7	0	0.0	
	ユーザ定義関数（値を返し、引数のある関数）を使ったプログラム作成	十分できる	2	7.4	8	27.6	**
		ややできる	9	33.3	13	44.8	
		あまりできない	15	55.6	8	27.6	
		全くできない	1	3.7	0	0.0	
	リストの引き渡しを行うユーザ定義関数を使ったプログラム作成	十分できる	2	7.4	4	13.8	n.s
		ややできる	9	33.3	13	44.8	
		あまりできない	15	55.6	12	41.4	
		全くできない	1	3.7	0	0.0	
文字列操作関数の活用		十分できる	0	0.0	2	6.9	n.s
		ややできる	12	44.4	15	51.7	
		あまりできない	14	51.9	12	41.4	
		全くできない	1	3.7	0	0.0	
数学関数の活用		十分できる	1	3.7	5	17.2	*
		ややできる	11	40.7	14	48.3	
		あまりできない	14	51.9	10	34.5	
		全くできない	1	3.7	0	0.0	
クラスを定義し、インスタンスを作成		十分できる	1	3.7	3	10.3	n.s
		ややできる	14	51.9	8	27.6	
		あまりできない	11	40.7	18	62.1	
		全くできない	1	3.7	0	0.0	
ファイル入出力関数を使ったプログラム作成		十分できる	2	7.4	3	10.3	n.s
		ややできる	8	29.6	13	44.8	
		あまりできない	15	55.6	13	44.8	
		全くできない	2	7.4	0	0.0	

2020年度と2022年度での比較：n.s.：非有意, *：$p<0.05$, **：$p<0.01$（Wilcoxon順位和検定（片側検定））

表22　受講後のJavaScriptの基礎的なスキルの習得（2020年度［n＝27］と2022年度［n＝29］）

項目		自己評価	2020年度		2022年度		
			（人）	（%）	（人）	（%）	
変数の宣言	数値型変数の宣言	十分できる	1	3.7	3	10.3	*
		ややできる	16	59.3	21	72.4	
		あまりできない	10	37.0	5	17.2	
		全くできない	0	0.0	0	0.0	
	文字型変数の宣言	十分できる	1	3.7	5	17.2	*
		ややできる	17	63.0	19	65.5	
		あまりできない	9	33.3	5	17.2	
		全くできない	0	0.0	0	0.0	
	配列変数の宣言	十分できる	1	3.7	1	3.4	n.s
		ややできる	13	48.1	18	62.1	
		あまりできない	12	44.4	10	34.5	
		全くできない	1	3.7	0	0.0	
画面への文字を表示するプログラム		十分できる	6	22.2	18	62.1	**
		ややできる	15	55.6	7	24.1	
		あまりできない	5	18.5	4	13.8	
		全くできない	1	3.7	0	0.0	
計算のプログラム	数の四則演算を行うプログラム	十分できる	3	11.1	16	55.2	**
		ややできる	16	59.3	7	24.1	
		あまりできない	8	29.6	6	20.7	
		全くできない	0	0.0	0	0.0	
	実数の多項式の計算をするプログラム	十分できる	2	7.4	10	34.5	*
		ややできる	15	55.6	12	41.4	
		あまりできない	10	37.0	7	24.1	
		全くできない	0	0.0	0	0.0	
分岐処理のプログラム	if文による分岐処理のプログラム	十分できる	2	7.4	10	34.5	*
		ややできる	15	55.6	13	44.8	
		あまりできない	10	37.0	6	20.7	
		全くできない	0	0.0	0	0.0	
	if～else文による分岐処理のプログラム	十分できる	2	7.4	10	34.5	*
		ややできる	16	59.3	13	44.8	
		あまりできない	9	33.3	6	20.7	
		全くできない	0	0.0	0	0.0	
	switch文による分岐処理のプログラム	十分できる	0	0.0	4	13.8	n.s
		ややできる	13	48.1	11	37.9	
		あまりできない	14	51.9	14	48.3	
		全くできない	0	0.0	0	0.0	
反復処理のプログラム	for文による反復処理のプログラム	十分できる	1	3.7	2	6.9	n.s
		ややできる	14	51.9	19	65.5	
		あまりできない	12	44.4	8	27.6	
		全くできない	0	0.0	0	0.0	
	while文による反復処理のプログラム	十分できる	0	0.0	3	10.3	n.s
		ややできる	14	51.9	16	55.2	
		あまりできない	13	48.1	10	34.5	
		全くできない	0	0.0	0	0.0	
	do～while文による反復処理のプログラム	十分できる	0	0.0	2	6.9	n.s
		ややできる	12	44.4	16	55.2	
		あまりできない	14	51.9	11	37.9	
		全くできない	1	3.7	0	0.0	
配列の活用	１次元配列をプログラム	十分できる	0	0.0	3	10.3	n.s
		ややできる	16	59.3	14	48.3	
		あまりできない	11	40.7	12	41.4	
		全くできない	0	0.0	0	0.0	
	２次元配列をプログラム	十分できる	0	0.0	2	6.9	n.s
		ややできる	13	48.1	15	51.7	
		あまりできない	14	51.9	12	41.4	
		全くできない	0	0.0	0	0.0	
	文字型配列をプログラムで活用	十分できる	1	3.7	3	10.3	n.s
		ややできる	12	44.4	12	41.4	
		あまりできない	14	51.9	11	37.9	
		全くできない	0	0.0	3	10.3	
ユーザ定義関数の作成	ユーザ定義関数（値を返さない、引数のない関数）を使ったプログラムを作成	十分できる	2	7.4	9	31.0	**
		ややできる	12	44.4	13	44.8	
		あまりできない	12	44.4	7	24.1	
		全くできない	1	3.7	0	0.0	
	ユーザ定義関数（値を返さない、引数のある関数（参照による呼び出し））を使ったプログラムを作成	十分できる	2	7.4	7	24.1	*
		ややできる	13	48.1	16	55.2	
		あまりできない	11	40.7	6	20.7	
		全くできない	1	3.7	0	0.0	
	ユーザ定義関数（値を返し、引数のある関数）を使ったプログラムを作成	十分できる	2	7.4	6	20.7	n.s
		ややできる	12	44.4	14	48.3	
		あまりできない	12	44.4	9	31.0	
		全くできない	1	3.7	0	0.0	
	配列の引き渡しを行うユーザ定義関数を使ったプログラムを作成	十分できる	1	3.7	2	6.9	n.s
		ややできる	12	44.4	11	37.9	
		あまりできない	13	48.1	16	55.2	
		全くできない	1	3.7	0	0.0	
文字列操作関数をプログラムで活用		十分できる	0	0.0	1	3.4	n.s
		ややできる	11	40.7	13	44.8	
		あまりできない	15	55.6	15	51.7	
		全くできない	1	3.7	0	0.0	
数学関数をプログラムで活用		十分できる	1	3.7	6	20.7	n.s
		ややできる	12	44.4	11	37.9	
		あまりできない	12	44.4	12	41.4	
		全くできない	2	7.4	0	0.0	
構造体の活用	構造体を使ったプログラムを作成	十分できる	0	0.0	1	3.4	n.s
		ややできる	11	40.7	8	27.6	
		あまりできない	14	51.9	17	58.6	
		全くできない	2	7.4	3	10.3	
	構造体配列を使ったプログラムを作成	十分できる	0	0.0	1	3.4	n.s
		ややできる	11	40.7	9	31.0	
		あまりできない	14	51.9	16	55.2	
		全くできない	2	7.4	3	10.3	
ファイル入出力関数を使ったプログラムを作成		十分できる	1	3.7	3	10.3	n.s
		ややできる	10	37.0	11	37.9	
		あまりできない	14	51.9	13	44.8	
		全くできない	2	7.4	2	6.9	

2020年度と2022年度での比較：n.s.：非有意，*：p＜0.05，**：p＜0.01（Wilcoxon順位和検定（片側検定））

対面形式で授業を行った2022年（令和４年）度の授業が、同期型のオンライン形式で授業を行った2020年（令和２年）度よりも受講生の自己評価が高くなった要因を考察する。2022年（令和４年）度の授業形式で良かった点として、「対面形式だったので、わからない時はその場で質問できる」という回答が65.5％（19）で最も多く、次いで「対面形式だったので、オンライン形式に比べて授業のペースに合わせて学習できる」という回答が41.4％（12）、「録画を含む講義資料を自分の都合の良い時に閲覧でき、学習できる」という回答も同じく41.4％（12）となっていた。これらの結果から、対面形式の授業においては、質問がしやすいことや授業ペースに合わせて学習できることが良かったと言える。また、オンライン形式の授業の利点である、授業の録画を後からｅラーニングから視聴できるようにしたことも良かったと考えられる。

表23　授業形式の良かった点（N＝29）（人）（複数回答）

対面形式だったので、わからない時はその場で質問できる。	19	65.5％
対面形式だったので、オンライン形式に比べて授業のペースに合わせて学習できる。	12	41.4％
録画を含む講義資料を自分の都合の良い時に閲覧でき、学習できる。	12	41.4％
毎回の授業で、確認テストを設けてあったので、理解が深まる。	9	31.1％
毎回の授業で、授業アンケートがあり、理解度の確認をしてくれる。	8	27.6％
授業資料（録画）のわかりにくいところは、止めたり、何度も繰り返し閲覧できるので理解が深まる。	6	20.7％
授業レポートが毎回出されることで、授業に熱心に取り組むことにより理解が深まる。	5	17.2％
毎回の授業で、授業アンケートがあり、わかりにくいところの質問ができるようにしている。	5	17.2％
該当するものはない。	1	3.4％

一方、今回の授業において最も多く挙げられた問題点は、「課題が難しい」という回答で、割合は51.7％（15）であった。また、「授業のペースが早くてついていけない」が20.7％（6）、「授業の難易度が高い」という回答が17.2％（5）と続いた。これらは授業の改善点として今後、特に注意すべき点と言える。

表24　授業の問題点（N＝29）（人）（複数回答）

課題が難しい	15	51.7％
授業のペースが早くてついていけない	6	20.7％
授業の難易度が高い	5	17.2％
確認テストの量が多い	3	10.3％
課題が多いと感じる	2	6.9％
質問がしにくい	0	0.0％
該当するものはない	5	17.2％

自由記述によると、授業で身についていないと感じるコンピュータのスキルとして、意外なことに複数の受講生が「タイピングが遅い」ということを挙げていた。山本・新村（2021）によると、2010年代前半のスマートフォンの普及率の急上昇に伴い、大学入学前にパソコンを使用する機会が減り、大学新入生のパソコンスキルが2000年代よりも低下したことが指摘されている[10)]。「授業速度を遅くして欲しかったです」「長い例題を打ち込んでいるとき、説明のタイミングが重なってしまい、理解が追いつかない時があった」といった記述も多数あったため、タイピングが苦手な受講生の入力時間を考慮しながら、授業を進めることが重要であることがわかった。

5　考察と結論

本稿では、当大学人間社会学部２年次に開講されている「プログラミング概論」の受講生に対して、プログラミング言語のスキルの習得状況等について実施した質問紙調査とｅラーニング確認テストの結果をもとに、プログラミングの学習における具体的な難点や、教育方法の改善点について考察した。

Pythonの基礎的なスキルの習得については、受講前に比べて受講後の自己評価が有意に

高く、授業での教育効果が認められた。ただし、「２次元リストの活用」「リストの引き渡しを行うユーザ定義関数を使ったプログラム作成」「文字列操作関数の活用」「ファイルの入出力関数の活用」といった項目については、受講後でも「十分できる」又は「ややできる」と自己評価する受講生の割合が60％未満となっていた。また、「クラスの定義とインスタンス作成」については、受講後でも非常に低い結果で、37.9％にとどまった。

JavaScriptの基礎的なスキルの習得についても、受講前に比べて受講後の自己評価が有意に高く、授業での教育効果が認められた。しかし、「switch文による分岐処理のプログラム」「配列の活用」「数学関数をプログラムで活用」といった項目については、受講後でも「十分できる」又は「ややできる」と自己評価する受講生の割合が60％未満となっていた。また、「配列の引き渡しを行うユーザ定義関数を使ったプログラムを作成」「文字列操作関数をプログラムで活用」「構造体の活用」「ファイルの入出力関数の活用」については、受講後でも50％に達していない。

このように、PythonとJavaScript両方において、受講者が十分なスキルを習得できなかった共通の課題は、リストや配列、ユーザ定義関数、ファイルの入出力関数の活用であった。当該授業の受講前の段階で、プログラミングを初めて学習する受講生が多かったため、これらの項目は初心者にとって難解であったと考えられる。解説や演習をよりわかりやすくする工夫が必要であることがわかった。

2022年（令和４年）度と2020年（令和２年）度を比較すると、2022年（令和４年）度の方が受講後の受講生の自己評価が高くなった。2022年（令和４年）度では、質問がしやすいことや授業ペースに合わせて学習できる対面形式の良さに加え、オンライン形式の利点である授業の録画を後から視聴できるようにしたことなど、授業形式において教育効果を高める要素があったと考えられる。また、2022年（令和４年）度のｅラーニングの確認テストの平均点も2020年（令和２年）度から大幅に向上していた。以上のような点から受講生の自己評価の向上につながったのではないかと考えられる。

しかし、2022年（令和４年）度の授業では、受講生から「課題が難しい」「授業の難易度が高い」という問題点が指摘された。この問題に対応するため、授業内容や課題の難易度の調整が必要であることがわかった。また、受講生の中には自身の「タイピングが遅い」と感じる者も多く、授業速度を遅くして欲しいとする要望もあった。さらに、当大学人間社会学部において大学入学時にパソコンをほとんど利用していない学生の割合が約42％にも上ることが、柴田（2022）による調査[11]で示されており、山本・新村（2021）による大学入学前にパソコンを使用する機会が減っているという指摘とも一致する結果となっている。したがって、一定数の受講生がタイピングが苦手であることが予想されるため、入力時間に余裕をもった授業進行が必要である。

以上のように、プログラミングの基礎的なスキルの習得には教育効果が認められるものの、スキルの十分な定着には至っておらず、改善が必要な点が見出された。

葛西ら（2022）による報告に加え、東（2019）による首都圏の文系学部に所属する大学生を対象とした質問紙調査でも、大学入学前にプログラミング言語を学んだ学生は少ないことが報告

されている。東（2019）は、プログラミング言語を学んでいない学生は、特に、テキスト（コード）を記述するプログラミングでは、スペルミスによりプログラムが動かなかったり、英語に苦手意識がある場合などにより、途中で挫折する可能性があることが懸念されることから、ビジュアル型プログラミング言語の利用を提案している[12)]。視覚的なパーツの組合せによりプログラミングができ、英語表現の文法を覚える必要がないため、学生たちの学習意欲の向上にもつながるとしている。また、山本・新村（2021）は、計算や文字列処理に偏らせないように興味を持たせる題材を取り扱い、ビジュアル型プログラミング言語を用いて変数や演算、順次・選択・条件の制御処理といったプログラミングの基本も教育の中に盛り込む教育事例を紹介している。鞆・上條・増田（2022）は、学習者が動作を目視で確認でき体感的に学べるビジュアルプログラミング言語やロボットを用いたプログラミング学習を提案している[13)]。他方で、章・山本（2023）は、文系学部の学生に不足しがちなデータサイエンスの学習の基礎となる数学、情報、統計の知識の必要性を主張し、Pythonプログラミングでこうした基礎知識を復習できるオンデマンド型教材を提案している[14)]。このように、文系学部のプログラミング教育の改善には多角的なアプローチが求められている。

「プログラミング概論」は、データサイエンス・プログラムの第２段階の専門基礎に位置づけられる科目である。このような科目において基礎力がしっかり身につけられなければ、次の段階の応用力の育成につながらない。したがって、基礎科目の教育効果を評価することは、教育プログラム全体の教育効果を確認するためにも重要であり、今後も調査を継続して実施し、指導方法や教育プログラムの改善に取り組んでいきたい。

謝辞

本研究は、福岡県立大学附属研究所研究奨励交付金「横断型教育プログラム開発研究」の助成を受けたものです。

参考文献

1）文部科学省，「高等学校学習指導要領（平成30年告示）解説 情報編（平成30年７月）」＜https://www.mext.go.jp/content/1407073_11_1_2.pdf＞（2023年５月７日最終閲覧）.

2）文部科学省，数理・データサイエンス・AI教育プログラム認定制度（リテラシーレベル）＜https://www.mext.go.jp/a_menu/koutou/suuri_datascience_ai/00002.htm＞（2023年5月7日 最 終 閲覧）.

3）柴田雅博・石崎龍二（2016）「保健福祉系大学における全学横断型での統計・情報教育拡充への取り組み」情報処理学会研究報告 コンピュータと教育（CE），Vol.2016-CE-134，No.23，pp.1-5.

4）大井良知・堀江光代（2023）「高等学校情報科におけるプログラミング教育の方法-プログラミング言語Pythonを用いて-」『大阪教育大学附属高等学校池田校舎研究部研究紀要』，55，pp.71-75.

5）葛西正裕・金澤小夜子・大島典子・末次新市・渡邉隆俊（2022）「文系AI人材教育に対する調査研究」『経済研究所所報』，第２号，pp.50-78.

6）齋藤朗宏・隈本覚・池田欽一・平山克己（2022）「Pythonを用いたプログラミング並びにデータサイエンス導入カリキュラム実施の試み」『北九州市立大

学商経論集』，57（1・2・3・4合併号），pp.33-42.

7）金子壽一（2022）「プログラミングに対する学習者の意識の変化について：Pythonを使用した授業の前後での変化」『至誠館大学研究紀要』，9巻，pp.1-12.

8）道越秀吾・奥井亜紗子・丸野由希（2019）「アンケート調査とｅラーニングシステムによるプログラミング教育の効果の評価」『京都女子大学現代社会研究』，第21号，pp.85-99.

9）石崎龍二（2022）「福岡県立大学人間社会学部におけるプログラミング教育の教育効果（2020年度）」『福岡県立大学人間社会学部紀要』，Vol.31，No.1，pp.103-113.

10）山本広志・新村太郎（2021）「大学のプログラミング入門教育へのScratch導入実践と、ドローン活用の試行」熊本学園大学論集『総合科学』，第27巻，第1号，通巻51号，pp.51-58.

11）柴田雅博（2023）「福岡県立大学人間社会学部における初年次情報リテラシー教育の効果（2022年度）」『福岡県立大学人間社会学部紀要』，Vol.31，No.2，pp.73-84.

12）東るみ子（2019）「超スマート社会に向けたプログラミング教育の現状と課題～大学生を対象としたプログラミング教育の実践を通して～」『商学研究』，第35号，pp.57-80.

13）鞆大輔・上條英樹・増田浩之（2022）「プログラミング学習におけるリメディアル教育手法の考察―学生の ICT に対する意識調査を基に―」『商経学叢』，68 (3)，pp.191-210.

14）章志華・山本敏幸（2023）「Pythonプログラミングで学ぶデータサイエンスのための高校数学基礎に関する オンデマンド教材の構築」『教育総合研究叢書』，16，pp.109-123.