福岡県立大学人間社会学部紀要
2022, Vol. 31, No. 1, 103－113

調査報告

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）

石　崎　龍　二*

要旨　福岡県立大学人間社会学部で2020年度に開講されたプログラミングに関する科目「プログラミング概論」の教育効果を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、eラーニング確認テスト等から考察した。

プログラミング言語の基礎的なスキルの習得については、各スキルのいずれ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、受講後でも自己評価が低い項目が複数あり、プログラミングの基礎的なスキルの習得についての教育効果があったと認められるものの、スキルの定着には至ってはおらず、教育方法に課題が多いことがわかった。

授業は同期型と非同期型を組み合わせたオンライン授業で行った。授業形式の良い点として、自分の都合の良い時に閲覧できた点、わかりにくいところは何度も繰り返し閲覧できる学習できた点、周りが気にならず自分のペースで学習を進めることができた点などが挙げられた。問題点として、課題が多い、目や腰が疲れるなど身体的負担、孤独感などが挙げられた。

キーワード　プログラミング教育　オンライン授業　教育効果　eラーニング

1．はじめに

近年、IoTの進展に伴う大規模なビッグデータの利活用が始まっており、AIなど新しいアルゴリズム的手法の活用も活発になってきている。教育現場では、情報通信技術を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ており、高等学校では、学習指導要領改訂[1)]に伴い、2022年度からプログラミング教育を含む「情報Ⅰ」が必修化された。こうした時代の変化に伴い、大学におけるプログラミング教育の見直しが求められている。文部科学省により、「数理・データサイエンス・AI教育プログラム認定制度」が令和２年度から開始され、全国の大学・短期大学・高等専門学校を対象としたデータサイエンス教育も推進されている[2)]。

各大学でも、プログラミング教育の改善に向けた様々な取り組みが行われている。特に文系の学部におけるプログラミング教育には課題が多い。葛西・金澤・大島・末次・渡邉（2022）

＊福岡県立大学人間社会学部・教授

の文系大学生を対象にした調査によると、高等学校までにプログラミングを学習したことがある学生はわずかであり、AIを学ぶための数学やコンピュータに対して不得手の認識が多い等の課題が示されている。道越・奥井・丸野(2019) の文系学部におけるプログラミング教育に関するeラーニングシステムのデータを使った分析によると、プログラミングの授業を終えた段階で習熟度が十分ではなく、プログラミングが難しいというイメージが授業の初回と最終回で変化がみられない等の課題が示されている。

本学では、「統計」「情報」に関する科目を整備し、2016年度入学生から全学横断型教育プログラムとして保健福祉情報教育プログラム(2020年度にデータサイエンス・プログラムへ名称変更) を導入している (柴田・石崎 (2016))。本プログラムでは、保健福祉分野での課題解決に、統計学、情報学の知識やスキルを応用できる力を養成することを目的とし、第１段階として数学、統計学、情報学、情報処理の共通基礎、第２段階として統計学・情報学の専門基礎、第３段階として、統計・情報学の演習により応用力を身につけることとしている。また、本学人間社会学部では、社会調査、データ分析、情報スキルといった専門ツールを取得させるために専門教育に社会調査・情報処理の科目を置いており、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。

「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段階に位置づけられた科目である。プログラミングの基本的な要素、アルゴリズムの基本となる制御構造を理解し、問題に応じて変数の宣言、制御文、関数を適切に使ったプログラミングができることを履修目標として、プログラミングの基本的な要素 (変数、データ型、演算子、配列など)、アルゴリズムの基本となる制御構造 (順次、分岐、反復など)、関数の作り方、ファイル処理などの手法を学習する。

本稿では、2020年度に実施された「プログラミング概論」の教育効果を、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、eラーニング確認テスト結果等から考察した。尚、2020年度は新型コロナウイルス感染拡大防止のため授業は全てオンラインで行った。

２．調査方法

⑴　事前事後調査

調査対象

福岡県立大学人間社会学部で2020年度後期に開講された「プログラミング概論」の受講者45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 (2020 (令和２) 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 (2021 (令和３) 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 (２項目)、Py-

thonのスキルに関するもの（18項目）、JavaScriptのスキルに関するもの（24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（５項目）、授業への要望に関する自由記述（１項目）、以上の全55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）、Pythonスキルに関するもの（18項目）、JavaScriptの知識に関するもの（24項目）、Python及びJavaScript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（３項目）、授業形式や授業全般に関する自由記述（２項目）、以上の全59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	45	45	100
受講後	27	45	60

３．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」の授業で学習するプログラミング言語は、高水準言語であるPythonとスクリプト言語であるJavaScriptである。Pythonは、比較的プログラムが読みやすく、書きやすい言語であり、人工知能や深層学習の分野や、データ解析の分野でよく活用されている言語であることから、学習する言語として選択している。また、JavaScriptは、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、学習する言語として選択している。また、二つの異なる言語を学ぶことで、プログラミング言語の成り立ちを学び、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。

３．１　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ついての受講生の受講前と受講後での自己評価の調査結果を以下に示す。数の四則演算と演算結果の画面への出力については、表２に示す通り、３項目のいずれについても受講後の自己評価が有意に高かった。但し、受講後でも、「実数の多項式の計算」については、「十分できる」又は「ややできる」の割合が受講後でも60％未満と低い。

分岐処理については、表３に示す通り、２項目のいずれについても受講後の自己評価が有意に高かった。

反復処理については、表４に示す通り、２項目のいずれについても受講後の自己評価が有意に高かったが、「十分できる」又は「ややできる」の割合が受講後でも60％未満と低い。

リストの活用については、表５に示す通り、３項目のいずれについても受講後の自己評価が有意に高かった。しかし、受講後でも、３項目とも「あまりできない」又は「全くできない」の回答率が50％を超えている。

ユーザ定義関数の作成については、表６に示す通り、４項目のいずれについても受講後の自己評価が有意に高かった。しかし、受講後でも、４項目とも「あまりできない」又は「全くできない」の回答率が50％を超えている。

関数の活用については、表７に示す通り、２

表２　数の四則演算と演算結果の画面への出力

項目		自己評価	受講前		受講後		
			(人)	(%)	(人)	(%)	
画面へ文字を表示する		十分できる	1	2.2	9	33.3	**
		ややできる	3	6.7	14	51.9	
		あまりできない	9	20.0	4	14.8	
		全くできない	32	71.1	0	0.0	
計算	数の四則演算	十分できる	0	0.0	6	22.2	**
		ややできる	0	0.0	16	59.3	
		あまりできない	7	15.6	5	18.5	
		全くできない	38	84.4	0	0.0	
	実数の多項式の計算	十分できる	0	0.0	3	11.1	**
		ややできる	0	0.0	12	44.4	
		あまりできない	7	15.6	11	40.7	
		全くできない	38	84.4	1	3.7	

受講前後での比較：n.s.：非有意，* :p＜0.05，** :p＜0.01
（フィッシャーの直接確率検定）

表３　分岐処理

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
if文による分岐処理	十分できる	0	0.0	2	7.4	**
	ややできる	0	0.0	16	59.3	
	あまりできない	7	15.6	9	33.3	
	全くできない	38	84.4	0	0.0	
if～elif文による分岐処理	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	17	63.0	
	あまりできない	6	13.3	9	33.3	
	全くできない	39	86.7	0	0.0	

受講前後での比較：n.s.：非有意，* :p＜0.05，** :p＜0.01
（フィッシャーの直接確率検定）

表４　反復処理

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
for文による反復処理	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	15	55.6	
	あまりできない	6	13.3	11	40.7	
	全くできない	39	86.7	0	0.0	
while文による反復処理	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	15	55.6	
	あまりできない	6	13.3	11	40.7	
	全くできない	39	86.7	0	0.0	

受講前後での比較：n.s.：非有意，* :p＜0.05，** :p＜0.01
（フィッシャーの直接確率検定）

表５　リストの活用

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
１次元リストの活用	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	12	44.4	
	あまりできない	7	15.6	14	51.9	
	全くできない	38	84.4	0	0.0	
２次元リストの活用	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	11	40.7	
	あまりできない	8	17.8	15	55.6	
	全くできない	37	82.2	0	0.0	
文字列リストの活用	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	11	40.7	
	あまりできない	7	15.6	15	55.6	
	全くできない	38	84.4	0	0.0	

受講前後での比較：n.s.：非有意，* :p＜0.05，** :p＜0.01
（フィッシャーの直接確率検定）

表６　ユーザ定義関数の作成

項目	自己評価	受講前		受講後		
		(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	2	7.4	**
	ややできる	0	0.0	11	40.7	
	あまりできない	6	13.3	13	48.1	
	全くできない	39	86.7	1	3.7	
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	2	7.4	**
	ややできる	0	0.0	11	40.7	
	あまりできない	7	15.6	13	48.1	
	全くできない	38	84.4	1	3.7	
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	2	7.4	**
	ややできる	0	0.0	9	33.3	
	あまりできない	7	15.6	15	55.6	
	全くできない	38	84.4	1	3.7	
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	2	7.4	**
	ややできる	0	0.0	9	33.3	
	あまりできない	7	15.6	15	55.6	
	全くできない	38	84.4	1	3.7	

受講前後での比較：n.s.：非有意，* :p＜0.05，** :p＜0.01
（フィッシャーの直接確率検定）

表７　関数の活用

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
文字列操作関数の活用	十分できる	0	0.0	0	0.0	**
	ややできる	0	0.0	12	44.4	
	あまりできない	7	15.6	14	51.9	
	全くできない	38	84.4	1	3.7	
数学関数の活用	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	11	40.7	
	あまりできない	7	15.6	14	51.9	
	全くできない	38	84.4	1	3.7	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

項目のいずれについても受講後の自己評価が有意に高かった。しかし、受講後でも、２項目とも「あまりできない」又は「全くできない」の回答率が50％を超えている。

クラスの定義とインスタンス作成については、表８に示す通り、受講後の自己評価が有意に高かった。但し、受講後でも、「十分できる」又は「ややできる」の割合が受講後でも60％未満と低い。

ファイルの入出力関数の活用については、表

表８　クラスの定義とインスタンス作成

項目	自己評価	受講前		受講後		
		(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	1	3.7	**
	ややできる	0	0.0	14	51.9	
	あまりできない	7	15.6	11	40.7	
	全くできない	38	84.4	1	3.7	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

９に示す通り、受講後の自己評価が有意に高かった。しかし、受講後でも「あまりできない」又は「全くできない」の回答率が50％を超えている。

表９　ファイル入出力関数の活用

項目	自己評価	受講前		受講後		
		(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	2	7.4	**
	ややできる	0	0.0	8	29.6	
	あまりできない	7	15.6	15	55.6	
	全くできない	38	84.4	2	7.4	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

３．２　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ついての受講生の受講前と受講後での自己評価の調査結果を以下に示す。変数の宣言については、表10に示す通り、３項目のいずれについても受講後の自己評価が有意に高かった。但し、受講後でも、「配列変数の宣言」については、「十分できる」又は「ややできる」の割合が受講後でも60％未満と低い。

数の四則演算と演算結果の画面への出力については、表11に示す通り、３項目のいずれにつ

表10　変数の宣言

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
数値型変数の宣言	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	16	59.3	
	あまりできない	7	15.6	10	37.0	
	全くできない	38	84.4	0	0.0	
文字型変数の宣言	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	17	63.0	
	あまりできない	7	15.6	9	33.3	
	全くできない	38	84.4	0	0.0	
配列変数の宣言	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	13	48.1	
	あまりできない	6	13.3	12	44.4	
	全くできない	39	86.7	1	3.7	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

表11　数の四則演算と演算結果の画面への出力

項目		自己評価	受講前		受講後		
			(人)	(%)	(人)	(%)	
画面へ文字を表示する		十分できる	0	0.0	6	22.2	**
		ややできる	1	2.2	15	55.6	
		あまりできない	7	15.6	5	18.5	
		全くできない	37	82.2	1	3.7	
計算	数の四則演算	十分できる	0	0.0	3	11.1	**
		ややできる	0	0.0	16	59.3	
		あまりできない	8	17.8	8	29.6	
		全くできない	37	82.2	0	0.0	
	実数の多項式の計算	十分できる	0	0.0	2	7.4	**
		ややできる	0	0.0	15	55.6	
		あまりできない	7	15.6	10	37.0	
		全くできない	38	84.4	0	0.0	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

表12　分岐処理

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
if文による分岐処理	十分できる	0	0.0	2	7.4	**
	ややできる	0	0.0	15	55.6	
	あまりできない	7	15.6	10	37.0	
	全くできない	38	84.4	0	0.0	
if ～ else 文による分岐処理	十分できる	0	0.0	2	7.4	**
	ややできる	1	2.2	16	59.3	
	あまりできない	5	11.1	9	33.3	
	全くできない	39	86.7	0	0.0	
switch文による分岐処理	十分できる	0	0.0	0	0.0	**
	ややできる	0	0.0	13	48.1	
	あまりできない	7	15.6	14	51.9	
	全くできない	38	84.4	0	0.0	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

表13　反復処理

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
for文による反復処理	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	14	51.9	
	あまりできない	6	13.3	12	44.4	
	全くできない	39	86.7	0	0.0	
while文による反復処理	十分できる	0	0.0	0	0.0	**
	ややできる	0	0.0	14	51.9	
	あまりできない	7	15.6	13	48.1	
	全くできない	38	84.4	0	0.0	
do ～ while文による反復処理	十分できる	0	0.0	0	0.0	**
	ややできる	0	0.0	12	44.4	
	あまりできない	7	15.6	14	51.9	
	全くできない	38	84.4	1	3.7	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

いても受講後の自己評価が有意に高かった。

分岐処理については、表12に示す通り、３項目のいずれについても受講後の自己評価が有意に高かった。しかし、「switch文による分岐処理」については、受講後でも「あまりできない」又は「全くできない」の回答率が50％を超えている。

反復処理については、表13に示す通り、３項目のいずれについても受講後の自己評価が有意に高かった。しかし、「for文による反復処理」「while文による反復処理」は、「十分できる」又は「ややできる」の割合が受講後でも60％未満と低い。「do～while文による反復処理」については、受講後でも「あまりできない」又は「全くできない」の回答率が50％を超えている。

配列の活用については、表14に示す通り、３項目のいずれについても受講後の自己評価が有意に高かった。しかし、「１次元配列の活用」については、受講後でも「十分できる」又は「ややできる」の割合が60％と低い。「２次元配列の活用」「文字型配列の活用」については、受講後でも、「あまりできない」又は「全くできない」の回答率が50％を超えている。

ユーザ定義関数の作成については、表15に示す通り、４項目のいずれについても受講後の自己評価が有意に高かった。しかし、受講後でも、

表14　配列の活用

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
1次元配列の活用	十分できる	0	0.0	0	0.0	**
	ややできる	0	0.0	16	59.3	
	あまりできない	7	15.6	11	40.7	
	全くできない	38	84.4	0	0.0	
2次元配列の活用	十分できる	0	0.0	0	0.0	**
	ややできる	1	2.2	13	48.1	
	あまりできない	6	13.3	14	51.9	
	全くできない	38	84.4	0	0.0	
文字型配列の活用	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	12	44.4	
	あまりできない	7	15.6	14	51.9	
	全くできない	38	84.4	0	0.0	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

表15　ユーザ定義関数の作成

項目	自己評価	受講前		受講後		
		(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を作成	十分できる	0	0.0	2	7.4	**
	ややできる	0	0.0	12	44.4	
	あまりできない	7	15.6	12	44.4	
	全くできない	38	84.4	1	3.7	
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	0	0.0	2	7.4	**
	ややできる	0	0.0	13	48.1	
	あまりできない	7	15.6	11	40.7	
	全くできない	38	84.4	1	3.7	
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	0	0.0	2	7.4	**
	ややできる	0	0.0	12	44.4	
	あまりできない	7	15.6	12	44.4	
	全くできない	38	84.4	1	3.7	
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	0	0.0	1	3.7	**
	ややできる	0	0.0	12	44.4	
	あまりできない	7	15.6	13	48.1	
	全くできない	38	84.4	1	3.7	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

「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」「ユーザ定義関数（値を返さない、引数のある関数）を使ったプログラムを作成」「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」については、「十分できる」又は「ややできる」の割合が受講後でも60％未満と低い。「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、「あまりできない」又は「全くできない」の回答率が50％を超えている。

関数の活用については、表16に示す通り、2項目のいずれについても受講後の自己評価が有意に高かった。しかし、受講後でも、2項目とも「あまりできない」又は「全くできない」の

表16　関数の活用

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
文字列操作関数をプログラムで活用	十分できる	0	0.0	0	0.0	**
	ややできる	0	0.0	11	40.7	
	あまりできない	7	15.6	15	55.6	
	全くできない	38	84.4	1	3.7	
数学関数をプログラムで活用	十分できる	0	0.0	1	3.7	**
	ややできる	0	0.0	12	44.4	
	あまりできない	7	15.6	12	44.4	
	全くできない	38	84.4	2	7.4	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

表17　構造体の活用

項目	自己評価	受講前		受講後		
		(人)	(%)	(人)	(%)	
構造体を使ったプログラム作成	十分できる	0	0.0	0	0.0	**
	ややできる	0	0.0	11	40.7	
	あまりできない	7	15.6	14	51.9	
	全くできない	38	84.4	2	7.4	
構造体配列を使ったプログラム作成	十分できる	0	0.0	0	0.0	**
	ややできる	0	0.0	11	40.7	
	あまりできない	6	13.3	14	51.9	
	全くできない	39	86.7	2	7.4	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
（フィッシャーの直接確率検定）

回答率が50％を超えている。

構造体の活用については、表17に示す通り、２項目のいずれについても受講後の自己評価が有意に高かった。しかし、受講後でも、２項目とも「あまりできない」又は「全くできない」の回答率が50％を超えている。

ファイルの入出力関数の活用については、表18に示す通り、受講後の自己評価が有意に高かった。しかし、受講後でも「あまりできない」又は「全くできない」の回答率が50％を超えている。

表18　ファイル入出力関数の活用

項目	自己評価	受講前		受講後		
		(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	1	3.7	**
	ややできる	0	0.0	10	37.0	
	あまりできない	7	15.6	14	51.9	
	全くできない	38	84.4	2	7.4	

受講前後での比較：n.s.：非有意, * :p＜0.05, ** :p＜0.01
(フィッシャーの直接確率検定)

４．ｅラーニング確認テストの達成度から見る学習の定着度

前節まで、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。表19は、各回での確認テストの達成度である。確認テストには何度もトライすることができる。ここでの達成度は、各回での問題を全て正解の場合を100点として、受講生の平均点を算出したものである。Pythonの学習では、１次元・２次元リスト、ユーザ定義関数、文字列操作関数、クラス・メソッド・インスタンス、

表19　eラーニング上の確認テストの達成度

回	授業内容	達成度（点）
1	Python－文字列や数値の入出力	79.1
2	Python－数値データの入力・計算・出力	85.2
3	Python－プログラムの選択処理	76.3
4	Python－プログラムの反復処理	72.8
5	Python－１次元リスト	67.6
6	Python－２次元リスト	66.7
7	Python－文字列のリスト	75.1
8	Python－ユーザ定義関数	62.2
9	Python－文字列操作関数	66.7
10	Python－数学関数	73.3
11	Python－クラス・メソッド・インスタンス、ファイル処理	64.4
12	JavaScript－数値データの入力・計算・出力	65.3
13	JavaScript－プログラムの選択処理	60.0
14	JavaScript－プログラムの反復処理、配列	61.8
15	JavaScript－関数、メニュー、フォーム、ボタンの配置等	56.0

ファイル処理の達成度が70点未満である。JavaScriptの学習では、いずれも達成度70点未満である。授業回数が11回のPythonに比べて、JavaScriptは４回しかなく、学びが定着するには、授業回数が少なかったと考えられる。

５．オンライン授業について

2020年度の授業では、同期型と非同期型の混合形式で行った。同期型での出席が難しい受講生には、非同期型での受講を可能とした。授業は、Zoomを使って同期型で授業を行い、その録画（MP4）を動画配信サイト上に登録した。ｅラーニング上に、講義資料（PDF）をアップロードし、確認テスト、授業アンケート、課題提示等により受講生の理解状況を確認しながら行った。尚、同期型で出席した受講生も、受講後に授業録画、講義資料を見ることができるようにした。また、プログラミング言語Python

については、eラーニング上にインストール手順書をアップロードし、受講生に、Python公式サイトからのPythonパッケージのダウンロードと各自が所有するPCへのインストールを求めた。プログラミング言語JavaScriptについては、ブラウザとエディタがあれば、プログラミングができるため、特に事前の準備を受講生に求めなかった。

このオンライン授業の形式に関する受講生の回答結果を考察する。良かったと思える点として、「録画を含む講義資料を自分の都合の良い時に閲覧でき、学習できた」74.1%（20)、「面接授業に比べて、周りが気にならず、自分のペースで学習を進めることができた」70.4%(19)、「講義資料（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った」59.3%（16）の回答が多かった。

表20　授業形式の良かった点（N＝27）（人）（複数回答）

録画を含む講義資料を自分の都合の良い時に閲覧でき、学習できた	20	74.1%
面接授業に比べて、周りが気にならず、自分のペースで学習を進めることができた	19	70.4%
講義資料（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った	16	59.3%
授業課題が毎回出されることで、課題に取り組むことにより理解が深まった	15	55.6%
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くれていた	15	55.6%
毎回の授業で、確認テストを設けてあったので、理解が深まった	10	37.0%
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ていた	9	33.3%
毎回の授業に掲示板があり、質問ができるようにしてくれていた	5	18.5%
該当するものはない	0	0.0%

表21　授業形式の問題点（N＝27）（人）（複数回答）

課題が多いと感じる	13	48.1%
目や腰が疲れるなど身体的負担	11	40.7%
孤独感（面接授業のように他の学生との交流がないため）	6	22.2%
質問がしにくい	5	18.5%
通信環境がよくない（音声の途切れ、画面が固まるなど）	3	11.1%
パソコンやネットワーク機器の機能が十分でない	3	11.1%
通信費用が気になる	1	3.7%
該当するものはない	2	7.4%

一方、問題点として、「課題が多いと感じる」48.1%（13)、「目や腰が疲れるなど身体的負担」40.7%（11)、「孤独感（面接授業のように他の学生との交流がないため)」22.2%（６）の回答が多かった。

授業形式に関する自由記述では、「リアルタイムで講義をしながら録画も残してくださったので、動画を見返しながら取り組むことができありがたかったです」「個人のペースでできるため、内容自体は難しく感じましたが，理解を深めることができました」等の録画を繰り返し見ることができる利点への記述が多く見られた。一方で、「対面授業のほうがその場で聞けるので良いのではないかと思った」という記述や「あまり聞きなれない言葉が多くあったので、その説明をより丁寧にしてくださるとありがたいと思いました」とする要望もあった。

６．まとめ

本稿では、本学２年次に開講されている「プログラミング概論」の受講生に対して受講後でのプログラミング言語の基礎的なスキルの習得状況等について質問紙調査を実施した。

Pythonの基礎的なスキルの習得については、各スキルのいずれについても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、「実数の多項式の計算」「反復処理」「クラスの定義とインスタンス作成」については、「十分できる」又は「ややできる」の割合が受講後でも60％未満と低い。「リストの活用」「ユーザ定義関数の作成」「関数の活用」「ファイルの入出力関数の活用」については、受講後でも「あまりできない」又は「全くできない」の回答率が50％を超えている。

JavaScriptの基礎的なスキルの習得についても、各スキルのいずれについても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、「配列変数の宣言」「for文による反復処理」「while文による反復処理」「１次元配列の活用」「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」「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」については、「十分できる」又は「ややできる」の割合が受講後でも60％未満と低い。「switch文による分岐処理」「do～while文による反復処理」「２次元配列の活用」「文字型配列の活用」「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」「関数の活用」「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「あまりできない」又は「全くできない」の回答率が50％を超えている。

受講前でのPythonとJavaScriptの基礎的なスキルの習得に対する自己評価から、葛西・金澤・大島・末次・渡邉（2022）の調査結果と同様に、高等学校までにプログラミングを学習した学生は少なかったと推測される。また、受講後での自己評価の低さから、道越・奥井・丸野（2019）の分析結果に見られるように、受講後でも、プログラミングが難しいというイメージは高い状態であると推測される。

eラーニング上の確認テストの達成状況から、Pythonの学習では、１次元・２次元リスト、ユーザ定義関数、クラス・メソッド・インスタンス、ファイル処理の達成度が低い。JavaScriptの学習では、いずれも達成度が低い。Python、JavaScriptの基礎的なスキルの習得に関する自己評価とeラーニング上の確認テストの達成状況を比較すると、習得に課題のある部分が共通していることがわかった。

以上のことから、プログラミングの基礎的なスキルの習得についての教育効果が認められるものの、スキルの定着には至ってはおらず、教育方法に課題が多いことがわかった。

2020年度の授業で行った同期型と非同期型の混合形式の良かった所としては、自分の都合の良い時に閲覧できた点、わかりにくいところは、何度も繰り返し閲覧できる学習できた点、周りが気にならず、自分のペースで学習を進めることができた点などが挙げられた。他方、問題点として、課題の多さ、目や腰が疲れるなど身体的負担、孤独感（面接授業のように他の学生との交流がないため）、などが挙げられた。今後、授業方法として、こうした良かった点を維持し、問題点を改善していく必要がある。

「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段階に位置づけられる授業であり、こうした科目の教育効果を検証することは、教育プログラムの教育効果の検証という点からも重要であり、今後も継続して調査を実施し、指導方法や教育プログラムの改善につなげたい。

謝辞

本研究は福岡県立大学附属研究所研究奨励交付金の助成を受けたものである。

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〉（2022年 6 月 5 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_data-science_ai/00002.htm〉（2022年 6 月 5 日最終閲覧）.

3）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.

4）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.

5）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.