Ruby等を用いたプログラミング教室の手引

亀谷千香子

令和元年〇月〇日

目 次

第1章　手引作成の目的

この手引の作成は平成 30 年度から東北公益文科大学 (以下本学) の文部科学省「私立大学ブランディング事業」の 1 つで行った小学 5、6 年生向けプログラミング教室の「Ruby てらこった」の活動を基準としている。実際に Ruby などを用いたプログラミング教室を企画し、運営していくのに役立つ手引をとして利用してもらうことを目的としている。そのために、企画するうえでの計画の立て方、プログラミング教室を通して身につけてほしい力、小学生に教える範囲の決め方などを述べている。運営では、教えるために必要な役割、授業の準備、教えるための方法を述べている。また、**??**章では Ruby てらこったの活動をふまえての企画、運営を提案として 1 例を挙げている。

第2章　プログラミング教室の企画

プログラミング教室の企画では、主にプログラミング教室を行うために必要なこと、計画の立て方などについて明らかにしていく。必要なことについては活動をする人の知識や能力について述べている。計画の立て方では、役割の分担や日程、小学生に教える範囲の決め方について述べている。

2.1　活動を行うために必要なこと

プログラミング教室の活動を行う時に必要で必要なこととして「知識」「教える能力」「発想力」「時間管理能力」「伝える力」4つが必要であるのではないかと考える。その理由としては、以下の通りである。

2.2　知識

本学では2年次に1年間必修科目の基礎プログラミングでプログラミング言語のRubyを用いたプログラミングを学ぶ。1年間で学ぶ内容は以下の内容である[?]。プログラミング教室で実際に教える時は小学生の質問に対して分かりやすく教えることができるのが前提になっている。そのために、ただ内容を理解しているだけではなく、自分なりに考えて教えられるほど理解していなければならない。

計算システムの基本操作と概念

基本的なコマンド操作・Unixについて学ぶ。

Rubyの基礎

プログラミング言語のRubyについて・Rubyプログラミングの作成方法を学ぶ。

変数・演算子・制御構造

変数とは「値」を入れる「箱」のようなものである。変数を使い方学ぶ。

演算子ではRubyで足し算、引き算、掛け算、割り算などの数値計算を行うための表現方法を学ぶ。

制御構造とはプログラムは通常上から下へ一直線に解釈実行される流れを変える方法を学ぶ。

メソッド

メソッドとはあらかじめ決められた(複雑な)処理を行なってくれるものについて学ぶ。

配列

配列は複数の値をまとめて保持するもので多くのデータを処理する方法を学ぶ。

パターンマッチング【正規表現】

正規表現はなにかを検索するときに、そのデータに含まれている文字列のパターンを汎用的に指定する方法を学ぶ。

ファイルの入出力

ファイルの入出力は Ruby プログラム起動時に引数を何も指定しないと端末 (キーボード) から、引数にファイルを指定した場合はそのファイルから 1 行ずつ内容を読み込む方法を学ぶ。

計算機の内部構造

2 進数、16 進数、文字コードについて学ぶ。

ハッシュ

ハッシュは別名連想配列とも言われ、任意の値に任意の値を結び付ける方法を学ぶ。

ハッシュと配列を組み合わせた繰り返し処理

多くの数のデータを処理できるプログラムは有効であり、Ruby では配列を利用することで、多くのデータを処理することができる方法を学ぶ。

再帰

ハノイの塔のような再帰処理の方法を学ぶ。

CGI

CGI とは HTML 文書内にあるデータ入力窓とそれらを受け渡すスクリプト 名を書いておき、ボタンを押すとそのスクリプトに入力値が 渡るようなしくみを学ぶ。

2.3　教える能力

小学生の一人一人理解している部分、キーボードの操作の速度などが違うため、それぞれに合わせて教える必要がある。小学生とコミュニケーションを取りながら分かりやすく教えることは難しいことである。Ruby てらこったに参加している学生は、小学生と関わること、教えることが比較的に得意な学生が参加している。例えば、教職課程を履修している学生、学習支援サークルとして小中学生に学習を教えている学生、2 年次の必修の基礎プログラミングの教員アシスタントをしている学生などである。

2.4　発想力

発想力では、どうしたら小学生がプログラムを楽しく、分かりやすく学んでもらうことができるのかをイメージしてアイディアを出す必要がある。例えば、誰かが授業で使う資料を見せてきた時にこうしたらもっと良くなるなど意見を 1 つ出すだけでも分かりやすさというのが変わってくる。また、小学生の質問に対してどうしたら分かりやすく伝わるかなど考える必要がある。

2.5　時間管理能力

業務で割り振られたものを完璧に締め切りまでで完成させるためには、どのくらいの準備時間がかかるのかということを意識しなければならない。締め切りで提出すればいいという意識で作業をすると良い物ができない。そのために、ただ作業をするのではなく、他のメンバーから確認してもらい改善、修正する時間を作るということが必要になってくる。作成物の修正の時間のことを考えて余裕をもって作成をしなければならなので逆算をして物事を考えてくる必要がある。また、作業以外にも他にしなければならないことがあると考えられるため時間の管理というのを意識していくことが大切になる。

2.6 伝える力

ここで使う伝える力とは、正確に相手に伝えるということを意味している。どの業務も自分一人だけで作業できることは少ない。そのため、メンバーで協力、分担しながら作業を進めていくことのほうが多い。そのためどの作業がどこまでできているのかなど具体的にメンバーに伝えるということは大切である。

2.7 プログラミング教室を行う目的と身につけたい力

プログラミング教室を行う目的とプログラミング教室を通して身につけたい力について述べていく。

2.7.1 Ruby てらこったと活動の目的

昨年度から大学のブランディング事業の一つの地域資源を活用する人材育成の研究としてプログラミング言語の Ruby を用いたプログラミング教室の企画・運営を行ってきている [?]。目的は、庄内地方を中心とした地域の若者達に情報技術を教え、情報社会を生き抜くために必要な力を身につけていくことである。対象としては小学 5、6 年生を対象としている。その理由としては、キーボードので入力、操作の関係からアルファベットを習い終わっていることや小学生のうちからプログラミング言語 Ruby を用いた本格的なプログラミングを学ぶ機会を提供したいと考えているためである。

授業は全 5 回行い大学生が教える。また、小学生 1 人に対して大学生が 1 人でサポートできるような体制になっている。授業内容は比較的に簡単な内容であり、授業を通して簡単なゲームプログラムを作成できるように内容を組み立てた。

2.7.2 授業を通して身につけたい力

小学生のプログラミング教育で身につけたい力として「知識及び技能」、「思考力、判断力、表現力等」、「学びに向かう力、人間性等」が述べられている [?]。その 3 つの力に追加して Ruby てらこったで身につけてほしい力についても述べる。

知識及び技能
: 生活でコンピュータが活用されていくことや、問題の解決には必要な手順があること

思考力、判断力、表現力等
: 発達の段階に即してプログラミング的思考力を育成すること

学びに向かう力、人間性等
: 発達段階に即して、コンピュータの働き、よりよい人生や社会づくりに生かそうとする態度を涵養すること

この項目に加え Ruby てらこったで身につけたい力は以下の通りである。

工夫する力
: 授業で使うサンプルのプログラムを自分で考えて、工夫をして、造し他の人とは違うプログラムをつくれるようになること

伝える力

伝える力としては、2 つの意味がある。1 つ目は、自分の作成したプログラムの発表を通して工夫し点、頑張った点など伝えられるようになれること

2 つ目は、周りの人が作成したプログラムの発表を聞いて感想など伝えられるようになること

2.7.3　教える範囲

教える範囲としては小学生が比較的に理解しやすいところであり、例えばクイズ、くじ引きのゲームを自分の力で考えられるようにするために必要なものを教える範囲とした。

出力メソッド (print, printf, puts)

アスキーアートのように入力したものをそのまま出力させるために必要と判断した。

文章処理メソッド (gets, chomp)

キーボードに打ち込んだ値を文字列として取得するゲームプログラミングを作るときに必要と判断した。

制御構造 (while, if, elsif, else)

レジスタープログラムなどの入力したものを繰り返しの処理を行うときに必要と判断した。また、クイズの結果の判定や条件で繰り返しを行うために使うため必要であると判断した。

配列及び乱数 (srand, rand)

2 つを組み合わせてクイズ問題を用意して乱数で選ばせたり、ジャンケンの手の内をランダムに出したりするために使うので必要と判断した。

sleep 関数 (sleep)

プログラムを時間を指定して一時停止することができるので小学生が楽しむことができるのではないかと判断した。

2.8　授業・日程

小学生に教える順番、授業時間の設定、開催日時について述べる。

2.8.1　教える順番と教える時間

教える順番は、基本的な操作から徐々に難しい内容に設定している。また、5 回目で自由に作成し、発表できるようなプログラムを作成できるように組み立てた。

教える時間は 1 回の授業で 2 時間としている。小学生が集中して授業ができるように大体 45 分から 50 分ぐらい授業をしたら約 15 分位の長めの休憩を入れるようにした。

2.8.2　各小学校のとの調整

地域の小学生に参加してもらうため各小学校に何をするのか、申し込み方法、問い合わせ先などが書いてあるチラシなどを配布する。

表 2.1: 授業内容

回数	内容	
1 回目	基本操作	画面出力 自己紹介プログラム
2 回目	ループ	繰り返し処理のプログラム
3 回目	配列	データ処理のプログラム
4 回目	条件分岐	条件のあるプログラム
5 回目	まとめ	習ったことを応用してプログラムを作成

2.8.3 開催日程

開催日程としては、小学生がプログラミング教室に参加しやすくするために学校が基本休みである土曜日、日曜日にした。また、全 5 回の授業に参加してほしいことから学校行事の地区運動会、習い事の大会などが被らないように開始日を決めた。

第3章　運営

企画した内容を実際にどのような流れで運営していくのかを運営方法、必要な道具、授業準備の点から述べていく。

3.1　運営方法

運営では、主な役割、業務内容、ミーティング方法について述べる。

3.1.1　役割関係の明確化

活動をする時に、スムーズに作業ができるように役割を明確にする必要がある。Rubyてらこったでの主な役割は以下のようなものである。

リーダー

リーダーの役割は全体的に計画、作業の進行状況などを把握、ミーティングを仕切ることをする。全体の計画では、いつまでに何をするのかを明確にする。明確にしたらグループ全員で内容を吟味する時間、内容修正する時間などを含め逆算し、作業の締め切りを決める。その時に、締め切りを決めてそのままにするのではなく作成を他のんだ人に対して、締め切りの何日か前に中間成果物を見せてもらうようにするなどして作業の進み具合を把握する。

ミーティングミーティングを仕切るときはあらかじめ何を話し合うのか決めておき、あらかじめメンバーに伝える。作成した成果物を用意してに対して全員からどうしたらもっと良くなるのかを意見をもらう時間にする。修正したものをいつまでに共有するか決める。また、ミーティングミーティングでは、業務の割り振りなどもする。業務の割り振りでは、メンバーの話を聞き、それをふまえて的確に割り振る必要あがある。

授業担当

授業担当者は、授業に必要な教材を作成し、授業をする。まずは、どの範囲を教えるのか決める。そのあとに、教える範囲を分かりやすく教えるための授業用のスライド、サンプルのプログラム、授業のタイムテーブルを用意する。詳しくは**??**で述べる。一人で作成すると分かりやすく教えているつもりでも相手には分かりにくい部分などは気づきにくいので、多くの人から意見をもらったり、確認をしてもらうことが大切になる。意見をもらうことで、客観的にまとめることができる。

各授業のアシスタント

各授業のアシスタントは、小学生が分からない所を説明、操作の補助などスムーズにプログラミングができるようにサポートすることがメインの業務になる。そのため、授業の内容をあらかじめ理解しておく必要がある。授業の内容が分からなく、質問してくる小学生がいるので、授業とは違う表現方法で教えなければいけないことがある。そのため多くの場面を予想して準備をしておかなければならない。操作の

補助では、キーボード操作が遅い小学生に対しては、どこにあるかを言葉で教えたり、押したいキーボードの近くを指でさして教えたりしてサポートする。

タイムキーパー

タイムキーパーは、各授業のタイムテーブルに合わせて授業担当者に時間を伝える。そのため、授業の進み具合など確認して授業が円滑に進むようにサポートする。時間に余裕がある時は、プログラム作成の時間や発表時間などにするなど休憩中に打ち合わせする。反対に、時間が思ったよりも足りない時は、変更する部分など提案してなるべく時間内で授業が終わるように考えて動く必要がある。

連絡係

連絡係は参加する保護者の方にメールで連絡を取ることをする。メールで連絡をするので誰が読んでも正確に伝わる内容を速くお知らせすることを意識して業務をする必要がある。また、メールで質問が来た時も対応をする。

3.2　必要な道具

授業で使う道具は以下のようなものを用意した。

表 3.1: 必要な道具一覧

アイテム	用途
パソコン	小学生の操作用、授業用、スライド用のため使用
USB	作成したプログラムを記録するため使用
教科書	授業の内容を確認すため使用
ノート	必要なことを書くために使用
ホワイトボード	プログラムを作成する前に使用
マグネット	プログラムの作成する前に使用
アルファベット対応表	キーボードのアルファベットを確認のため使用

3.2.1　パソコン

パソコンは小学生用、授業用、スライド用の 3 種類使用した。小学生用と授業用の OS は NetBSD である。

- 小学生用

  授業中のプログラムを作成、Web ページからプログラムをダウンロードするために使用

- 授業用

  小学生の使用しているパソコンと同じであるためどのように操作するのか小学生に分かりやすくするため使用

- スライド用

  スライド用を用意することで画面の切り替えすることができるためスムーズに内容の説明ができる

3.2.2 USB

学習システムをインストールしておき、小学生が自宅の PC でも引き続き学習ができるようにした。

3.2.3 教科書

授業内容に合わせて、学生らで作成した。オリジナルのイラストや分かりやすいイメージ図などを取り入れ、興味が沸くような内容にすることを心がけた。また、語句の説明なども小学生でも分かるようにした。内容は以下のようなものである。

プログラミングとは
プログラミングとは何か、Ruby とは何かを説明する

ファイル、ディレクトリ
ファイルとディレクトリの説明をする

小学生でも分かるようにファイルは、絵を書く時の紙でディレクトリは紙をしまっておく場所というようにした

print, puts, gets について
画面出力の方法、キーボードから入力させる方法をまとめた

while について
ループについて学ぶ

配列、乱数
配列を使ってランダムに取り出す方法をまとめた

if, elsif, else について
条件分岐をさせる方法、if, elsif, else の違いについて図を用いて分かりやすくした

プログラムを工夫するためのメソッドについて
プログラムを出力させるときに文字が変化したり、時間をおいて表示させたりするものをまとめた

コマンド集
プログラムの作成時などに覚えておくと便利なコマンドをまとめて書いておき、分からない時にすぐに確認できるようにした

3.2.4 ノート

初回の授業で配布し、小学生が学んだことや気づいたこと、ポイントなどをこまめに書き込めるように用意した。授業内では、ノートに書く場面があれば先生が指示を促し小学生がノートを書く時間を設けた。

3.2.5 ホワイトボード

ホワイトボードにプログラムを書いてから PC にプログラムを打ち込んだ方がわかりやすいと考え、用意した。また、小学生がホワイトボードに書きながらプログラムを組んで行くことで理解を深める目的で使った。

3.2.6　マグネット

その授業の内容で勉強する新しいことや大切なものをマグネットにした。それらを貼るだけでよいようにし、記憶しやすくするために作った。これらを授業内でホワイトボードに貼りながら作業を進めていった。例を挙げると、「 while 」と「 end 」などのセットで扱うものに関しては、セットで先にマグネットをホワイトボードに貼り付けてからプログラムを組み立てるなどの工夫をした。あらかじめ用意されているものを使用したため、間違いの減少や作業時間の削減にも繋がった。

3.2.7　アルファベット対応表

小学 5、6 年生はアルファベットを習い終えているが大文字小文字を理解するために作成し配布した。キーボードでは大文字表記になっているため「 l(エル) 」と「 i(アイ) 」や「 h(エイチ) 」と「 n(エヌ) 」などが間違いやすかった。

3.3　授業準備

授業の準備では授業を行うまでの流れを述べていく。

3.3.1　内容の吟味

内容の吟味では授業での内容をミーティングなどで話し合い確認する。教える内容では、小学生でも分かりやすく学ぶことができるものを中心にしている。そのため、小学生にとって内容が難しすぎないか、変数などのイメージ図はどうするかなど話し合う。1 回分の授業で教える内容が多くなり過ぎないか、教えるプログラムは分かりやすいか、プログラム作成の時間が十分に取れるかなどを考える。教える内容が多くなりすぎないかという基準については模擬授業をしてみたり、各授業での小学生の理解度を見たりして内容を決めた。プログラムについては??で述べる。

3.3.2　教える体制

教える体制としては授業の担当者 1 人、タイムキーパーが 1 人、授業アシスタントは参加する小学生の人数と同じようになるようにした。1 対 1 で教える体制にしたのは、授業に追いつけない部分をサポートできるようにするためである。授業が進む速度に対して、小学生が理解する速度に差が発生することがあった。その際には、随時授業の合間に時間をとり、小学生が理解する速度に合わせて授業を展開していく形を心がけた。

3.3.3　使うプログラム決め

授業用サンプルのプログラムは授業の内容のポイントを抑えたものにした。

1 回目が画面出力の print,puts を使ったプログラムを作成している。このプログラムでは、print と puts の違いを理解してもらうために両方使用した。画面に出力される ””部分の文字を変更する。速く操作ができる小学生には print または put を増やして好きな文字を書き込ませて変えるようにした。

```
#!/usr/koeki/bin/ruby
# -*- coding: utf-8 -*-

print "名前\n"
print "学校名\n"
puts "好きな食べ物"
puts "好きな動物"
```

2回目は変数、whileのループをつかったプログラムをしている。gets.chompを最初に追加して所持金を入力させたり、最後の合計に1.08をかけて消費税込みの金額をだしたりなど変えることができる。

```
#!/usr/bin/env ruby
# -*- coding: utf-8 -*-

puts "スーパーのレジ買ったものの値段を入れてね。(100 円→100 と入力しよう)"
memo = 0

while true
print "値段は? (終わりたい時は q を押してね)"
nedan = gets.chomp

if nedan == "q"
break
end

memo += nedan.to_i
printf("今の合計は %d 円だよ \n",memo)
end
printf("今回の合計は %d 円です。お買い上げありがとうございました! \n",memo)
%\end{verbatim}
```

3回目は配列と乱数を組み込んだプログラムをサンプルにした。この配列の部分をグー、チョキ、パーに変えるとジャンケンプログラムに変えることができる。

```
#!/usr/bin/env ruby
# -*- coding: utf-8 -*-

kuji = ["1等","2等","3等","4等","ハズレ"]
print("なにがでるかな～？")

puts("....")

sleep(1)
srand()
nani =rand(5)
printf("あなたは%sでした！\n",kuji[nani])
%\end{verbatim}
```

4回目は条件分岐のifをつかいクイズプログラムをサンプルにした。クイズの問題文を変えるだけではなく選択肢を増やしたり、回答によって違うものを画面出力できるように変えることができる。

```
#!/uer/bin/env ruby
# coding: utf-8

puts"今の元号はなんでしょうか？"

puts"1:平成"
puts"2:令和"
puts"3:昭和"

print"答えを入力:"
kotae = gets.to_i

sleep(1)

if kotae == 1 then
  puts"惜しい！ハズレだよ！"
elsif kotae == 2 then
```

```
  puts"正解だよ！読み方はわかるかな？？"
elsif kotae == 3 then
  puts"ザンネン！ハズレだよ！"
else
  puts"その番号はないよ！！"
end
%\end{verbatim}
```

5回目は今まで学んだことをいれたプログラムをサンプルにした。

```
#!/usr/bin/env ruby

puts"コンピュータとじゃんけんをしよう!"
puts"5回中3回勝てればクリアだよ!"
sleep(3)
puts""
print"それじゃ GAME START\n"
puts""

win = 0
kaisuu = 1
janken = ["だしてないよー", "グー", "チョキ", "パー"]

while kaisuu <= 5
  sleep(2)
  puts""
  printf("%d回目!(あなたの勝利数:%d)\n", kaisuu, win)
  puts""
  print"じゃんけん!(グーなら「1」チョキなら「2」パーなら「3」を押してね): "
  while true
    you = gets.to_i
    if you >= 4
        you = 0
    end
    com = rand(3) + 1
    printf("ぽん!(あなたは「%s」でコンピュータは「%s」)\n",
 janken[you], janken[com])
    sleep(1)
    if (you==1&&com==2) || (you==2&&com==3) || (you==3&&com==1)
      puts"あなたの勝ち!"
      win += 1
      kaisuu += 1
      break
    elsif (you==1&&com==3) || (you==2&&com==1) || (you==3&&com==2)
      puts"あなたの負け!"
      kaisuu += 1
      break
    elsif you == com
      print"あいこで(グーなら「1」チョキなら「2」パーなら「3」を押してね): "
      redo
    else
      puts"なにもだしてないからあなたの負け!"
      kaisuu += 1
      break
    end
  end
end

sleep(3)
printf("あなたの勝利数は「%d 回」\n", win)
sleep(3)

if win >= 3
  puts"おめでとう!"
  sleep(2)
  puts"+--------------------------------------------------+"
  system 'banner YOU WIN!!'
  puts"+--------------------------------------------------+"
else
  puts"残念..."
  sleep(2)
  puts"+--------------------------------------------------+\n"
  system 'banner GAME OVER'
  puts"+--------------------------------------------------+\n"
```

```
end
```

参加した小学生が作成したプログラムは以下のようなものである。このプログラムは latter という配列の中に入っている”グラードン”,”カイオーガ”,”レシラム”,”キュレム”の 4 つをどのくらいの時間で入力できるかというプログラムである。

```
#!/usr/koeki/bin/ruby
# -*- coding:utf-8 -*-

letter = ["グラードン","カイオーガ","レシラム","キュレム"]

print("どれだけ早くポケモンを打てるかな\t")
printf("%s\n", letter)

start = Time.now.to_i

i = 0
while i < letter.length
  printf("\v%s:", letter[i])
  input = gets.chomp
  if input == letter[i]
    stop = Time.now.to_i
    STDERR.print("\t正解\n")
    i += 1
  end
end

printf("かかった時間 %d 秒!\n", stop - start)
%\end{verbatim}
```

3.3.4 スライド

授業で使うスライドはあくまでも授業の副教材であることを意識する。スライドは、イメージ図、重要な部分だけにする。するとポイントなどが小学生にも分かりやすく伝えることができる。

3.3.5 模擬授業

模擬授業では授業担当者が実際の授業を想定して行う。主に見るポイントとして以下のようなものを確認する

- 教え方が分かりやすいか (プログラムの説明など)
- ノートを取る部分の確認
- スライドが見やすいか (スライドの配色、文字が多すぎないか)
- 時間配分 (プログラムの作成時間があるか、時間が無くなったらどこを省いて説明するか)

3.3.6 他の機関との調整

大学から教育委員会に連絡をとり、教育委員会から各小学校に資料を配布してもらった。各小学校の先生から小学生に資料を配布してもらった。

第4章　授業

授業では当日の準備作業、授業の流れ、プログラムの発表方法について述べていく。

4.1　当日の準備・確認事項

当日の準備は、備品の準備からから始める。ホワイトボード、ペン、マグネット、パソコンの設置などである。パソコンの設置では、ネットワークが繋がるかを試す。その後、授業のおおまかな流れを確認する。

4.2　授業の流れ

授業の流れは以下の通りである。

1. 前回の授業の復習

   前回の授業で学んだ内容のポイントを復習する。

2. タイピング練習

   タイピング練習では、授業で使う英単語を 3 回ずつ練習する時間をとる。速く入力が終わった人には、5 回など入力する回数を増やすようにした。

3. 内容の説明

   内容の説明では、授業で新しく学ぶことを説明する。重要なポイントは、メモをとる時間を作る。

4. サンプルプログラム

   サンプルプログラムは、新しく学ぶ内容を使う。最初にどのようなプログラムなのか実行させる時間にする。

5. サンプルプログラムの説明

   4 のサンプルプログラムがどうしてそのような動きをするのか説明する時間をつくる。

6. 休憩

   長めの休憩をとる。休憩の間には、小学生とコミュニケーションを取る。

7. プログラム作成

   プログラムの作成では、学んだ内容を使い自分で考えて作成してもらう。作成の時は、ホワイトボードにマグネットを貼ったり、ペンで書いたりしてからパソコンに入力する。イメージ通りにプログラムが作れない時は、アシスタントがサポート、説明をする。サポート、説明で重要なのは、考えてもらい理解してもらうということだ。

8. 作成したプログラム発表

   作成したプログラムは、発表する時間をつくる。発表方法はみんなの前で作成したプログラムを実行し発表する方法と他の人が作成したプログラムを実行してみる方法の 2 つである。次の**??**で詳しく述べる。

4.3　プログラム発表について

授業で作成したプログラムを発表する機会をつくった。その理由としては、自分の作成したものを発表し伝える力を身につけてほしいということや、他の人の発表、プログラムの良さなど学んで欲しいと考えたからである。授業で行った発表方法は 2 つある。その方法は以下の通りである。

みんなの前で発表する

みんなの前で発表する。最初に作成したプログラムの難しかった点、頑張った所などをを話してもらう。プロジェクターに映し、作成プログラム実行をして発表する方法である。発表する時に何を話せばいいのか困ってしまう小学生もいるのであらかじめ話してほしい項目を提示すると困った様子が見られなかった。

他の人のプログラムを実行する

他の人のプログラムを実行して作品を見る方法では、実際に実行できるので工夫している部分など気づくことが多い。実行してみた感想などお互いに伝えることができる。

4.4　授業後のミーティング

授業後に約 30 分ほどのミーティングをする。まずはじめに、授業の担当者から良かった点、悪かった点を先に話してもらう。悪かった点についてはメンバーから改善策、意見をもらう。そのあとに、授業のアシスタントから参加した小学生の様子、理解度についての意見をもらう。その後に次回までの授業の改善点、対応策、準備しなければならないことなどを話し合う。

第5章　授業を受講した小学生の感想

1回目から5回目まで授業後にアンケート (授業で楽しかったこと、難しかったこと) をとった。以下の表はそのまとめである。

表 5.1: 楽しかったこと

回数	内容
1回目	パソコンの操作、コマンド操作、自己紹介プログラムを発表したこと、大学生と話しをしたこと
2回目	キーボードの操作、みんなが作成したプログラムを実行すること、大学生と話をしたこと
3回目	みんなが作成したプログラムを実行すること
4回目	プログラムの作成、大学生と話しをしたこと
5回目	オリジナルのプログラムを作成したこと

表 5.2: 難しかったこと

回数	内容
1回目	パソコンの操作、自己紹介
2回目	break、ループが難しかった、プログラムを改造すること
3回目	乱数をでランダムに取り出すこと、自由にプログラミングを作ること
4回目	if,elsif,elese の使い方が難しかった、プログラムを改造すること
5回目	変数が多くなりすぎて難しかった、オリジナルのプログラムを作成すること

5.1　楽しいと感じる部分

楽しいと感じた部分として発表が楽しかったという意見が多かった。発表を聞きたり、他の参加者のプログラムを動かしてみたりするのが新たな発見があり楽しかったのだと思う。

また、大学生と話しをするのが楽しかったという意見があった。プログラムをつくる時や休憩の時など多くの場面で話す機会があったのは良かったと考える。

5.2　難しいと感じる部分

操作の部分ではキーボードの操作が難しいと感想に書いていた。パソコンをあまり使わない小学生にとってキーボードを使い入力するには、どこに入力したい文字、数字があるのか探しながら打つので思ったよりも時間がかかるため難しく感じたのだと考える。

プログラムの内容では変数の部分のが難しかったという感想が多かった。2回目から5回目にかけて難しいプログラムを作成していくとに、変数が徐々に増えていく。そのため、何をどの変数にしたか確認しながらプログラムを作成するため難しく感じたのだと考える。

第6章　教えるための要点

この章では主に実際に小学生にプログラミングを教えるための注意点をまとめている。

6.1　教える上で注意すること

授業担当者と授業アシスタントの両方の視点から述べていく。

6.1.1　授業担当者

授業担当者は、小学生に教えるということで内容を噛み砕いて教えることが重要になってくる。授業は基本的に操作など全員一緒に進んでいくため常に小学生の顔を見るようにする。すると操作が終わっているのか、まだ作業の途中なのかなど確認することができる。進み具合をみて、かなり遅れている場合はアシスタントに頼んでサポートしてもらう。授業中は、ただ話すのではなく、常に問いかけたり、手を上げてもらったりと反応を見るようにする。話を聞く、ノートを取る、ホワイトボードに書き込む、プログラムを作成するなど指示をしっかり出して、何をするのか明確にすることが大切である。

6.1.2　授業アシスタント者

授業アシスタント者は操作、授業で理解できなかった部分の説明など個人に合わせてサポートしていく。教える小学生に対して目線を合わせて教えていくことが大切である。難しい内容でも分かりやすい言葉を使って教える。プログラム作成の時は、全てサポートするのではなく考えてもらう時間を大切にしていく必要がある。操作が遅くて焦ってしまう小学生がいた場合は、声をかけて、ゆっくりでも大丈夫なことを伝えて、確実に操作してもらうようにする。

6.1.3　小学生の理解度に合わせる

一人一人授業の内容の理解度、操作速度などは違う。そのため一人ひとりに合わせた教え方をしなければならない。

- 理解が早い小学生

  理解が早い小学生には、サンプルのプログラムを改造してもらうということをして何もしていない時間をなくすようにした。最初にどの部分を変えると良いのかヒントを出しながら少しずつ教えていった。

- 操作が遅い小学生

  操作が遅いと追いついいかなければと焦ってしまう小学生がいる。そのときは、どこまでできているのか確認し、話しかけながら操作のサポートをする必要がある。

6.2　その他

その他として授業が計画通りにいかなかった場合、参加者が来れなくなった場合、パソコンが調子悪くなった場合の対応を述べる。

6.2.1　授業の内容が予定より進まなかった場合

どこのを中心に教えるのかということを考える。説明に時間がかかりすぎてしまったときは、次の授業でどのぐらいの時間で説明するかのと打ち合わせをして教える範囲を全て 5 回で終わらせるようにする。

6.2.2　参加者が来れなくなった場合

全 5 回の授業で 1 つどこかの回の授業に参加できなかった場合は次回の授業の 30 分前に来てもらい簡単な授業を受けてもらい、その後次の授業に参加してもらった。すると、次の授業においていかれることなく参加してもらうことができた。

6.2.3　利用しているパソコンの調子が悪くなった時

インターネットを使い Web ページを見る時に無線 lAN ではうまく繋がらない場合があった。そのときは、有線で接続をした。次の授業からは予備のパソコンを使い対応をした。

第7章　実際に行うための一例

Ruby てらこったの活動のを通して良かった部分、悪かった部分があった。そのことを踏まえて企画、運営を円滑に行うための提案をする。プログラミング教室を企画から開催するまでの大まかな流れ、内容をまとめる。詳しい部分は前の章を参考にすると良い。

7.1　企画

企画ではこのような流れで役割、教える範囲などを決めた。

1. 活動するメンバーを集める

   活動をするにあたってどのようなメンバーが必要か考え集める。

2. 役割を決める 1

   役割では、リーダーや必要な業務の責任者を決める

3. プログラム教室の目的を明確にする

   プログラミング教室を通して小学生に何を学んで欲しいのか、身につけた力などを考える。

4. 教える範囲を決める

   教える範囲は小学生でも理解しやすい内容、最終的に小学生にどのようなプログラムを自分で考え作成できるようになってもらうのかを決める。

5. 役割を決める 2

   各回の授業担当者とアシスタントなどを決める

6. 授業のについて決める

   教える範囲をどの順番で教えていくか考える。それぞれの授業で教えるイメージ図についてなどを決める。

7. 日程について決める

   日程は全 5 回の授業であるため小学生が参加しやすい日程を考えて組み立てる。

8. 資料の配布について決める資料を作成したら配布する範囲、方法など決める。

7.2　運営

運営の流れは以下の通りである。

1. 役割決め

   運営が始まると外部の人と連絡が取ることが多くなるため連絡をして日程調整などをする役割が必要になってくる。

2. 必要な道具の決定

   授業をするときにパソコン以外にもどのような道具があったら良いか考え作成をする。Ruby てらこったの場合は、プログラムを作成前にホワイトボードにプログラムを書いてもらうことでプログラムを作成するときのミスを防ぐように工夫した。

3. 教える体制を決める

   小学生の分かりやすく教えるための体制 (アシスタントの人数など) を決める。

4. 内容の吟味

   内容の吟味では授業担当者が考えてきた授業の構成、教える内容、タイムテーブルなどを話し合い決める。

5. サンプルプログラムを決める

   サンプルプログラムは、学ぶ内容を含み改造しやすいものにする。

6. 模擬授業

   模擬授業では、実際の授業を想定して、作成したタイムテーブルをもとに授業をする。小学生役の人が授業を体験し改善点などの意見を出しあう。模擬授業は上手くいくまで何回も繰り返す。

7.3　広報関係提案

広報関係では大学から教育員会に資料を持っていた。その後教育委員会から各小学校へポスターを配布した。ポスターにはどのようなことをするのか、開催日時、申し込み方法、締め切り、お問い合わせの連絡先など必要な情報を簡潔にまとめる。

7.4　ミーティング

ミーティングでは情報共有が大切になってくる。ミーティングで何かをはじめて準備するのでなはく、あらかじめ作成したものに意見をもらったり改善策を練ったりと全員で決めたり話すことだけに絞ることで効率的に進めることができる。

7.5　申し込み方法の提案

申し込み方法で情報、確認が確実なのは、申し込みを Web で行い、参加者が決定後に電話でやり取りするのが良い方法だと考える。その理由としては、Web でおこなうことで必要な情報を残すことができる。また、参加する人に電話をすることで Web 申し込みの内容を直接確認することができ、迅速かつ臨機応変に対応できるからだ。

7.6　授業

授業では、分かりやすく丁寧ということを意識する。小学生の反応、操作の進み具合などを確認して教える速度を調整する。アシスタントは、小学生が途中で困らないように声をかけながらサポートする。

7.7 授業後のミーティング

授業後のミーティングは、授業担当者、アシスタント、タイムキーパーなど全員が授業を振り返る時間にする。悪かった部分に対しては、改善策を考えるようにする。

関連図書

[1] 広瀬雄二.「Ruby プログラミング基礎講座」, 技術評論社,2006,p312

[2] 文部科学省. ” 小学校プログラミング教育の手引 (第二版)”. http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/11/06/1403162_02_1.pdf, (参照日 2019-5-24).

[3] 大石桃菜 佐々木大器 山口円馨 .” 東北公益文科大学における小学生向けプログラミング教室「Ruby てらこった」の取り組み”. 文部科学省 私立大学研究ブランディング事業 日本遺産を誇る山形県庄内地方を基盤とした地域文化とＩＴ技術の融合による伝承環境研究の展開（平成 29 年度〜平成 33 年度)p50-54 .