授業改善セミナーワークショップ「1人1台端末を活用したプログラミングの指導方法について」

北海道札幌北高等学校　前田健太朗

0 はじめに

学校のコンピュータでプログラミングの指導をするとき，コンピュータにプログラミングの開発環境を用意する必要があります。しかし，リース契約しているコンピュータにアプリをインストールするのは躊躇しませんか。インターネットを利用してWeb上でプログラミングができる環境を利用すると，その問題を解決できるでしょう。

Web上でプログラミングできる環境はいろいろあります。Bit Arrow(https://bitarrow.eplang.jp/)やScratch(https://scratch.mit.edu/)，Google Colaboratoryなどが教育用としてよく使われるているのだろうと思います。Google Colaboratortを利用するためには，Googleのアカウントが必要になります。道立高校では，北海道教育委員会がGoogle Workspace for Educationを利用できる環境を整えているので，生徒にGoogleのアカウントを交付することができます。作成したファイルはクラウド上に保存されることと，コンピュータにアプリをインストールする必要がないことから，ネット環境とコンピュータがあれば，生徒はいつでもどこでもプログラミングの学習をすることができます。

そこで，今回のワークショップではGoogle Colaboratoryを利用してPythonによるプログラミングを体験しましょう。授業実践報告で述べたように，昨年度の指導の反省から，短時間でプログラミングの基礎を学べる実習例を考えました。

1 Google Colaboratoryの利用

Google Colaboratoryを利用するためには，Googleのアカウントでログインする必要があります。Google Colaboratoryで作成したファイルは，Googleドライブに作成されるColaboratoryのフォルダ内に自動的に保存されます。ファイルを編集中にコンピュータがフリーズしても，直前までの作業が自動的に保存されています。（インターネットへの接続が切れていると自動的に保存されません。）

なお，Google Colaboratoryで扱えるプログラミング言語はPythonです。他の言語を利用したい場合はBit Arrowなどを利用するとよいでしょう。

2 表示するプログラム

はじめに，コンピュータで表示するプログラムを扱います。プログラムを実行した結果が目で見えないと，プログラムを実行して何が起こっているのかわからないので，まずは画面に表示するプログラムが基本になります。Pythonで画面に表示する命令は「print」です。命令の後に続くカッコ内に表示したい文字を入力します。Pythonでは文字を「'」または「""」で囲むというルールもあります。忘れがちなので注意が必要です。また，表示する文字を日本語にすると，プログラミングの命令等は半角英数字，記号で入力する決まりがあることから，半角と全角を混同したエラーが発生しやすいです。特に記号の半角と全角の違いは紛らわしいので，入力になれるまではプログラム例や実習問題に全角文字を使わない方がいいだろうと個人的には考えています。

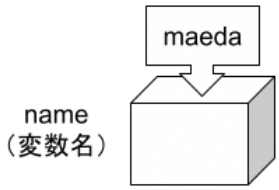
```
print('abc')
```

3 変数を利用するプログラム

　プログラミングの学習でよく使うものに「変数」があります。変数とはプログラミングで扱うデータを記憶する場所のようなものです。

　この変数にデータを記憶するとき，代入という命令を使います。Pythonをはじめ，多くの言語では代入の命令として「=」が使われています。この代入の命令「=」は，右辺の値を左辺の変数名に記憶するという意味になります。変数に文字データを代入するとき，文字データを「'」または「"」で囲むことに注意しましょう。

　なお，変数名はプログラマーが名付けます。記憶させるデータの内容などを変数名にすると，この変数をどのような目的で用意したのかが誰からもわかりやすくなるでしょう。なお，予約語というPythonであらかじめ決められた名前を変数名とすることはできません。



#変数nameに値maedaを記憶するイメージ図

```
name = 'maeda'
print(name)
```

4 数値データを表示するプログラム

　変数に代入するデータが数値データのときは，数値データを「'」や「"」で囲む必要はありません。これらで囲んでしまうと文字データとして扱われます。プログラミングの世界では，数値ではなく数字と区別されることが多いようです。数値は計算できますが，数字は文字扱いなので計算できません。

```
atai = 123
print(atai)
```

5 キーボードから入力したデータを表示するプログラム

　変数に代入するデータを，人がキーボードから自由に入力できるようにします。キーボードから入力する命令は「input」です。カッコの中の文字は，「input」の命令が実行されて入力待ちをしているときに表示される，入力を促すコメントです。どんなデータを入力してもらいたいのかをここに入力するとよいでしょう。

　なお，「input」で入力されたデータは文字扱いされることに注意してください。

```
name = input('input your name')
print('hello',name)
```

6 条件分岐するプログラム

今までのプログラムは，上の行から順番に処理をする順次処理でした。ある目的を達成するための処理の手順をアルゴリズムといいますが，このアルゴリズムには3つの基本構造があり，順次処理の他に分岐処理，反復処理があります。この中の分岐処理とは，条件を設けてその条件を満たしたかどうかで処理が変わるというものです。

次のプログラム例は，あらかじめ決められたパスワード「123456」（※危険なパスワードの指導もする必要がありますね。）を正しく入力されたら「login OK」，パスワードを間違えたら「wrong password」と表示する内容が変わるというものです。

条件の命令は「if」です。条件は比較演算子を用いて記述します。比較演算子の種類は次の表を参照してください。なお，比較演算子の「等しい」が「==」です。数学記号「=」と混同されやすいので注意が必要です。

それから，「if」の行末に「:」を忘れずに入力してください。「:」を入力してエンターキーを押すと，次の行が字下げされます。この字下げが条件を満たしたときの処理の内容を記述していることを意味します。（条件を満たしていないとき，字下げして記述した命令は実行されません。）

条件を満たしていないときに処理があるときは「else:」と記述します。「else」は条件を満たしたときの処理ではないので，字下げをしないでください。（「if」と同じ位置に記述します。）条件を満たしていないときの処理は，再び字下げして記述します。

意味	記号
aはbより大きい	a > b
aはb以上	a >= b
aはb以下	a <= b
aはbより小さい	a < b
aとbは等しい	a == b
aとbは等しくない	a != b

#比較演算子

```
password = '123456'
in_pass = input('input password')
if password == in_pass:
    print('login OK')
else:
    print('wrong password')
```

7 数字データを数値データ（整数）に変換するプログラム

6のプログラムで設定したパスワード「123456」は数字，文字扱いされるデータです。これを数値にしたい場合は，整数なので「int」という命令を利用します。「int」のカッコ内に記述した内容が整数化されます。

```
password = 123456
in_pass = int(input('input password'))
if password == in_pass:
    print('login OK')
else:
    print('wrong password')
```

8 条件によって処理を繰り返すプログラム

コンピュータを使っていると，パスワードの入力を求められることがよくあります。そして，パスワードの入力に失敗して，もう一度入力することもあったのではないでしょうか。そこで，間違ったパスワードを入力すると，再び入力を求められるように繰り返してみましょう。繰り返す処理はアルゴリズムの基本構造の1つである反復処理です。条件によって繰り返すことを指定する場合，命令は「while」になります。今回はパスワードの入力が間違っているときに繰り返すので，パスワードの入力が正しいかどうかを判定するための真偽型の変数loginを用意し，初期値としてFalseを記憶しています。正しいパスワードが入力されたら変数loginに「True」が記憶されて繰り返し処理を抜けます。

```
login = False
password = 123456
while login == False:
    in_pass = int(input('input password'))
    if password == in_pass:
        print('login OK')
        login = True
    else:
        print('wrong password')
```

9 指定した回数だけ処理を繰り返すプログラム

セキュリティを高めるために，パスワードの入力回数が制限されていることはよくあります。そこで，8のプログラムを改造して3回までしかパスワードを入力できないようにします。今度の繰り返し処理は回数の制限があるので，命令は「for」を利用します。「for」の場合，「for」の後ろに変数を用意します。この変数が繰り返している回数をカウントします。

繰り返す回数は「range」を利用して設定します。「range」のカッコの中の数値が1つのとき，その数値が繰り返す回数のカウントは初期値0，カウントする数は1ずつ増えて，カッコの中の数値が終了値となります。Pythonでは，この終了値のときは，「for」の繰り返し処理を行わないので注意が必要です。次のプログラムのように「for i in range(3)」とすると，「for」を処理するときにカウントする変数iの値は最初0，次にこの処理をするときに変数iの値は1，さらに次に処理するときに変数iの値は2，そして次の処理を処理しようとすると変数iの値は3になるので，「for」の繰り返しを抜けて「if login == False:」の処理を行う，という処理の順序になります。

```
login = False
password = 123456
for i in range(3):
    in_pass = int(input('input password'))
    if password == in_pass:
        print('login OK')
        login = True
        break
    else:
        print('wrong password')
if login == False:
    print('blocked your account')
```

10 乱数の利用とサイコロのモデル化

繰り返し処理から一旦離れて，今度は乱数を扱います。乱数とは規則性のない数値の並びです。ほとんどのプログラム言語には乱数を発生させる命令が用意されています。Pythonではrandomモジュールという標準ライブラリの中に乱数を発生させる命令が用意されているので，それを呼び出します。その処理が次のプログラムの1行目の「import」です。なお，今回のプログラムで発生させる乱数は1から6までの整数なので，「randint」という命令を利用します。

```
from random import randint
dice = randint(1, 6)
print(dice)
```

情報の科学の教科書にも，情報Iの教科書にも，モデル化のことが記載されています。現行の学習指導要領解説では，モデル化等は情報の科学においては問題解決の単元で扱われており，アプリやプログラム言語を使って指導することに主眼を置かないと記載されています。新しい学習指導要領解説では，コンピュータとプログラミングの単元に，生徒の状況等に応じてプログラミング，アプリの利用などシミュレーションを行う方法について配慮すると記載されています。このことから，何らかの題材を用いてモデル化の実習をするべきだと個人的には考えます。今回作成したプログラムはサイコロをモデル化したものと考えられるでしょう。

11 サイコロの出目を当てるゲーム

サイコロのモデル（プログラム）ができたら，簡単なゲームをつくってみましょう。今までつくったプログラムを実行しても，それほど楽しいものではありませんが，ゲームをつくろうというと生徒はちょっと期待すると思います。（本当はつまらないゲームですが…。）また，今までは写経のようにプログラムを入力させることばかりでしたが，ここでは今まで学んだ内容を応用することができます。（プログラミングの基礎を学んでいる段階では，どうしても写経のようにプログラムを入力することが中心になってしまいます。）

まず，10のプログラムを改造します。次のプログラムの2行目までが10のプログラムと同じです。（10のプログラムの3行目のサイコロの出目の表示は，後にしないと答えがバレてしまいます。このことを利用して，アルゴリズムの説明をすることも考えられます。）

次に，サイコロの出目を予想してキーボードから入力します。キーボードからの入力の命令は「input」でしたね。また，「input」で入力した値は文字扱いとなります。乱数で発生させた値は数値ですから，文字を数値に変換する命令も必要です。今回扱う数値は1～6の整数ですから「int」を利用しま

す。これが次のプログラムの3行目です。
　乱数を発生させて，さらにキーボードから予想した出目を入力したら，キーボードから入力した値とコンピュータが乱数で発生させたサイコロの出目が一致するかどうかを判定し，一致したら「atari」，そうでなければ「hazure」と表示します。条件分岐ですから「if」を利用します。条件を満たしていないときの処理もありますから，「else」も必要になります。
　最後に，コンピュータが発生させた乱数（サイコロの出目）がいくつだったか表示させましょう。表示する命令の位置は，キーボードからの入力（input）後であればどこでも構いません。ただし，「if」の中に記述すると，あたったときまたははずれたときのみに表示されることになるので，注意が必要です。このことを確認することで，アルゴリズムの重要性を生徒に伝えることができるでしょう。

```
from random import randint
dice = randint(1, 6)
yoso = int(input('dono me ga derukana?'))
if yoso == dice:
    print('atari')
else:
    print('hazure')
print(dice)
```

12 サイコロを10回振る

　今度は，サイコロのモデルを利用してシミュレーションを行います。サイコロの各目の出る確率は同じになると考えるのが一般的ですよね。しかし，本当にそうかどうか試してみましょう。本物のサイコロでシミュレーションするのは大変ですが，プログラムならあっという間にシミュレーションができます。
　まず最初に，サイコロを繰り返し振ってみましょう。繰り返す回数は10回というように指定します。回数を指定した繰り返しには「for」を用いることを既に学習していますから，この実習は演習問題として扱うこともできるでしょう。

```
from random import randint
for i in range(10):
    dice = randint(1, 6)
    print(dice)
```

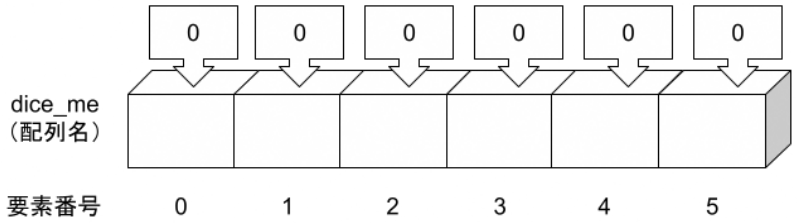
13 サイコロの出目が当たるまで何度もサイコロを振るゲーム（おまけ）

　サイコロを繰り返し振ることができたら，今度は11の出目を予想する問題と組み合わせて，あたるまでサイコロを繰り返し振るプログラムにしてみます。今度の繰り返しは回数ではなく条件でしていしますのでwhile文を利用します。サイコロの出目をあてることも，while文も既に学習した内容ですから，この実習も演習問題として扱うこともできるでしょう。

```
from random import randint
hantei = False
while hantei == False:
    dice = randint(1, 6)
    yoso = int(input('dono me ga derukana?'))
    print(dice)
    if yoso == dice:
        print('atari')
        print('GAME OVER')
        hantei = True
    else:
        print('hazure')
```

14 サイコロモデルのシミュレーション（配列とグラフの描画）

サイコロの出目はどれも同じくらいの確率ででるのかをグラフで視覚化します。1〜6の目がそれぞれ何回出たかを記録するため，配列dice_meを用意しています。配列とは，変数のように値を記憶するものですが，複数の値をまとめて扱えるものです。複数の値をまとめて扱うために，配列は複数の要素で構成されており，各要素を区別するための番号が用意されています。この番号を要素番号といい，0から始まります。そこで，出目が1の回数はdice_me[0]に，出目が2の回数はdice_me[1]に，というように「出目-1」した値を要素番号とします。なお，Pythonでは配列に相当するものをリストといい，要素を追加したり削除したりすることができます。（この資料では配列と表現しています。）



#配列dice_meのイメージ図

グラフの描画するために，matplotlib.pyplotモジュールを利用します。プログラムの2行目の処理が，このモジュールを組み込む処理です。このモジュール名はちょっと長いので，「as plt」と宣言することで，「plt」という別名で扱えます。「plt.bar」という命令が横軸の値と縦軸の回数の値を扱い，「plt.show( )」という命令でグラフを表示しています。

```
from random import randint
import matplotlib.pyplot as plt
dice_me = [0, 0, 0, 0, 0, 0]
for i in range(100):
    dice = randint(1, 6)
    dice_me[dice - 1] = dice_me[dice - 1] + 1

for j in range(6):
    plt.bar(j + 1, dice_me[j])
plt.show()
```

15 線形探索法

ある値を探すアルゴリズムを探索アルゴリズムといいます。たくさんの値の中から特定の値を探すので，値の記憶には配列を利用することが一般的です。配列に記憶されている値から特定の値を探すとき，配列の先頭（要素番号0）から順番（要素番号を1つずつ増やす）に探す方法を線形探索法といいます。

次のプログラムでは，1行目で配列に値を記憶しています。2行目で探す値を入力します。そして，4行目で探す値と配列に記憶している値が等しいかどうかを判定しています。このとき，配列に記憶している値は配列の要素番号0から1つずつ増えるように，「for」で繰り返す回数をカウントする変数iを要素番号として利用します。

```
fruit = ['apple', 'orange', 'grape', 'lemon', 'banana']
want = input('what fruit do you want ?')
for i in range(5):
    if want == fruit[i]:
        print('arimashita')
    else:
        print('arimasen')
```

このプログラムを実行すると，探している値と値を記憶している配列の要素を比較して，「arimashita」か「arimasen」を表示します。ただし，これを繰り返しているので，探している値と，値を記憶している配列の要素番号0から4までを比較することになり，何回も「arimasen」と表示されてしまいます。

16 線形探索法（改造版）

15のプログラムの問題（何回も「arimasen」を表示する）を解決した例が次のプログラムです。真偽型の変数hanteiを用意し，初期値をFalseとします。そして，探索の結果，探している値を見つけたときに変数hanteiの値をTrueに更新し，探索処理が終わった最後にこの値を利用した条件判断を行なって，「arimashita」か「arimasen」のどちらか1つを表示をするようにしています。

```
fruit = ['apple', 'orange', 'grape', 'lemon', 'banana']
hantei = False
want = input('what fruit do you want ?')
for i in range(5):
    if want == fruit[i]:
        hantei = True
        break
    else:
        hantei = False
if hantei == True:
    print('arimashita')
else:
    print('arimasen')
```

17 おわりに

プログラミングの基礎編はここまでです。これから先，二分探索法，ソートの方法の比較，二次元配列などは発展編として扱うのがよいかと思います。まずは基礎がしっかりと理解できるように取り組み，さらに発展編を授業で扱い，最後に演習問題として大学入試等で出題されるようなプログラムを取り組むと，基礎から受験対策まで対応できるのではないかと考えています。ただし，基礎編をできるだけ短い時間で取り組めるようにしても，受験対策を授業だけで行うことは難しいのではないかと思います。授業では実習を取り入れてプログラムをつくる能力を養い，講習等の授業外の時間を設けて受験対策をするとよいのではないかと個人的には考えています。

なお，ある講演で大学入試に関係する方が，大学入試対策のためのプログラミングの指導は望んでいないと話をしていたという記憶があります。大学入試対策を求められることもあるでしょうが，授業では体験を通してできるだけプログラミングの楽しさ，アルゴリズムを考える楽しさ，大切さを生徒に伝えるべきだと私は考えています。