プログラミング　10 回目（　　月　　日）　　担当：松田　侑樹

年　　組（　　）

この時間のねらい：**①　関数の作成方法とその利用法を学ぼう。**

②　関数を利用して、高度なプログラミングの作成方法を知ろう。

皆さんは関数と聞くと、どのようなイメージを持ちますか。私の予想としては、数学の問題として、一番解きたくない、（というか、拒否反応が出る）という人が多いのではないでしょうか。

しかし、そんな関数ですが、プログラムの関数に備わっているものは何度も使ったことがあります。

今回の関数は自分で作成するため、さらに分かりづらいかと思います。しっかりと学び、最後の開発時に使用できるようにしましょう。

基本的に関数を考える際には、変数のときに出てきた型を使う場合と、void を使用する場合があります。まずは void から学びましょう。（こちらのほうが利用価値が高いため）

<void　メソッド　>

これから先、関数を作成する際に使用するものは　メソッド　と呼びます。void で作成する関数の特徴は　戻り値　が無いということです。

それなんだよ、と思う人が多いかもしれませんが、あとから詳しく話します。以前、ダウンロードしてもらったサンプルを使用します。

左上の「ファイル」→「サンプル」→「Contributed Examples」→「Getting Started...」

→「09_Functions」→「Ex_09_03」という順番にクリックしていきましょう。

最終的に Ex_09_03 というプログラムが出てきます。実行すると　フクロウ　が出ます。

一羽描くために　13　行を費やしていることが分かります。それでは、もう一羽を隣に描こうと思います。何行使いますか？　26　行になるはずですね。ここまでで分かる通り、フクロウさんを描くにはなかなか頭を使い、さらに打ち込むという、非常に面倒なことをしなくてはいけません。これを簡単にするために、関数を作成してそれを呼び出すことで描けるように改良していきます。（改良したものは Ex_09_05 です。）

7、8 行目にあるものが関数です。owl という関数は Processing 内では存在しません。11 行目以降を見てみると、owl(int x, int y)というようにこの関数を定義しています。引数の部分に数値を入れると、定義の中の同じ文字の中に数値を代入してくれます。今回の場合は 13 行目の関数に代入です。特に変わらないものに関しては値をそのままにして、変更するものについては文字で置く。非常に難しく感じると思いますが、慣れて使えるようにすると、開発がはかどります。見るだけでは学べないので裏面で練習してみましょう。

右のプログラムを打ってみてください。皆さんも知っているであろうキャラクターが描かれます。

```
size(100, 100);
ellipse(width/2, height/2, 40, 40);
fill(255, 0, 0);
ellipse(width/2, height/2, 30, 30);
fill(0);
ellipse(width/2 - 8, height/2, 5, 5);
ellipse(width/2 + 9, height/2, 5, 5);
noFill();
arc(width/2 - 3, height/2 + 2, 10, 10, PI/4, 3*PI/4);
arc(width/2 + 5, height/2 + 2, 10, 10, PI/4, 3*PI/4);
```

それでは練習です。このキャラを 5 個表示してみてください。

表面の内容をしっかりとみればできるはずです。

<データ型の　メソッド　>

変数を宣言するときに使用した型を利用すると、計算結果を返すような関数を利用することができます。計算だけでなく、保存された文字列などを（自動で）選択して文章を作成させるようなこともできます。

では、実際に足し算を行う関数 a を作成してみましょう。右のように整数型の関数 a を宣言してみると最後にコンソール表示される数字は　13　になります。この中で作成した関数は 6～9 行目の内容です。

引数を決めてあげれば、その二つの数について、　和　を求めてくれる関数です。

```
void setup(){
  int d = a(5, 8);
  println(d);
}

int a(int b, int c){
  int sum = b + c;
  return sum;
}
```

また、文字の場合右のようにすると 2 つの文字を”/”で仕切った形でコンソールへの表示を行うことができます。ただし、一番大元の関数 b が文字列型であることを間違えてはいけません。char 型を複数並べれば、文字列型（String 型）になりますから。

```
void setup(){
  println(b('あ', 'い'));
}

String b(char moji1, char moji2){
  String mojiretsu = moji1 + "/" + moji2;
  return mojiretsu;
}
```

以上のような形で、プログラムの関数を作成して、自分がやらなければいけない仕事を少しでもパソコンに割り振って、作業を進めていきましょう。