
関数
基礎ユニット
PROCESSING FRIENDS

前回の内容：座標変換

座標変換

現在の座標軸の状態を基準に
描画が行われる

座標軸を移動させる

- translate
- rotate
- scale

座標軸

図形や文字列を表示する時の基準

```
text("Anegasaki",100,height/2);
```

デフォルトの座標軸

座標軸を15度回転

今日の内容

関数

関数

特定の処理を行う受付窓口

内部の細かい処理内容を隠す（隠蔽）

プログラム(パンの作成関数)

あんぱん

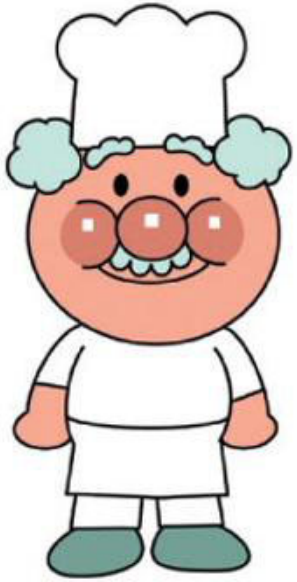
カレーパン

成果物(パン)

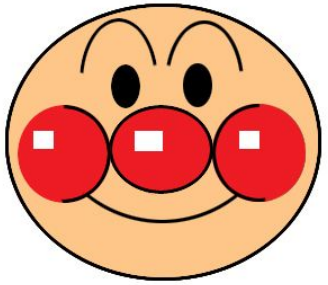
関数

特定の処理を行う受付窓口

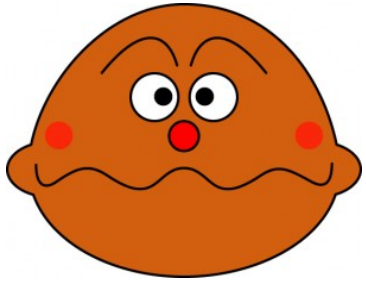
内部の細かい処理内容を隠す（隠蔽）



プログラム（パンの作成関数）

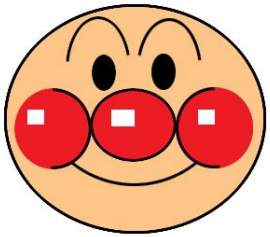


あんパン



カレーパン

あんパンの作り方



こねる

材料を用意
粉などをまぜる
こねる

発酵

生地を整形
放置

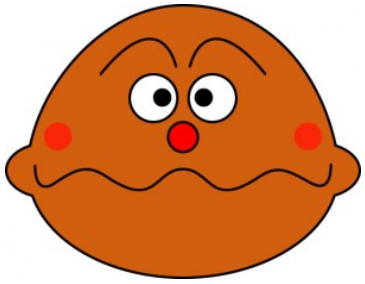
あんパンのたね作り

生地をとる
あんを詰める
丸める

焼く

オーブンを温める
たねを入れる
しばらく放置
取り出す

カレーパンの作り方



こねる

材料を用意
粉などをまぜる
こねる

発酵

生地を整形
放置

カレーパンのたね作り

生地をとる
カレーを詰める
丸める

焼く

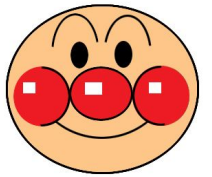
油を入れる
たねをいれれる
しばらく放置
取り出す

関数を使わないと

処理の区切りがわかりづらく、重複が多い

こねる
材料を用意
粉などをまぜる
こねる
発酵
生地を整形
放置
あんパンのたね作り
生地をとる
あんを詰める
丸める
焼く
オーブンを温める
たねを入れる
しばらく放置
取り出す

こねる
材料を用意
粉などをまぜる
こねる
発酵
生地を整形
放置
カレーパンのたね作り
生地をとる
カレーを詰める
丸める
焼く
オーブンを温める
たねを入れる
しばらく放置
取り出す

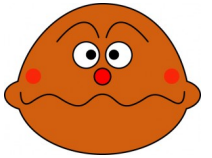


こねる

発酵

あんパンの
たね作り

焼く



カレーパンの
たね作り

揚げる

関数による
プログラムの整理

関数：makeあんパン
　こねる
　発酵
　あんパンのたね作り
　焼く

関数：makeカレーパン
　こねる
　発酵
　カレーパンのたね作り
　揚げる

関数：こねる
　材料を用意
　粉を混ぜる
　こねる
関数：発酵
　生地を整形
　放置
関数：あんパンのたね作り
　生地を適量とる
　あんを入れる
　整形
関数：焼く
　オーブンを温める
　たねを入れる
　一定時間待つ
　取り出す
関数：カレーパンのたね作り
　生地を適量とる
　カレーを入れる
　整形
　　以下略

関数宣言

同じ処理はひとかたまりに–＞関数を利用

関数名として使えるもの：変数と同じ

```
void 関数名(){
        処理内容を書く
}
```

関数名(); // <-処理内容を実行できる

組み込み関数 → Processing言語側で用意している関数

組み込み関数の大まかの分類

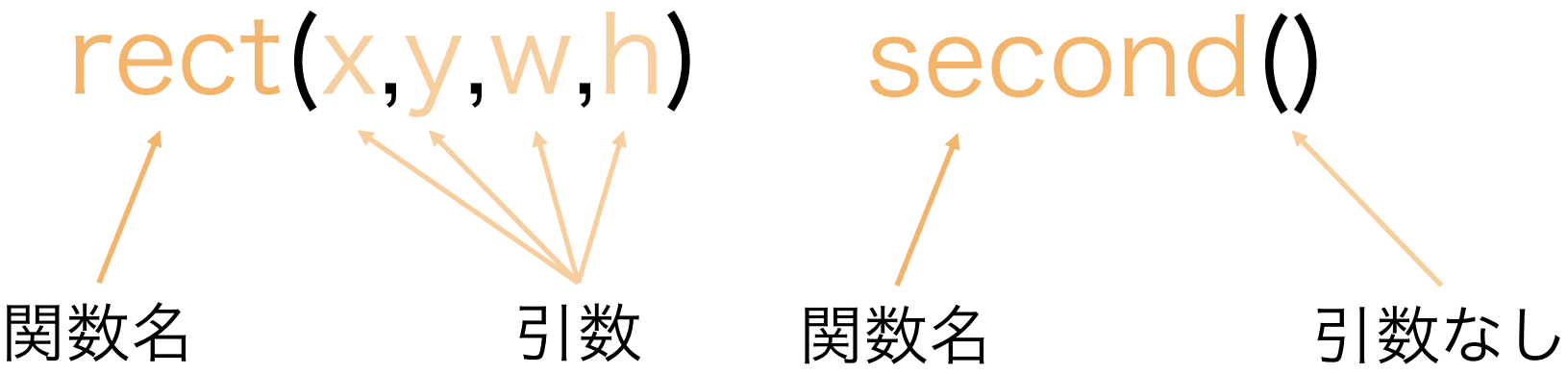
- 図形描画の関数
- ファイル関連の関数
- 時刻に関する関数
- 文字列関連の関数
- 数学関連の関数
- その他の関数

詳しくは、リファレンスを参照のこと

関数の分類

	引数がない	引数がある
戻り値がある	secondなど	mapなど
戻り値がない void型	noFillなど	rectなど

手続き



関数の分類

戻り値がない

何らかの処理を
ひとまとめにしたもの

手続き、Procedure

戻り値がある

何らかの処理の結果、
何かの答え（戻り値）が
求まるもの

戻り値：リターンバリュー,return value

関数の宣言
（引数無し、戻り値なし）

```
void 関数名(){
    関数処理の内容を書きます。
    変数なども使うことができます。
}
```

setup関数やdraw関数は、このタイプの
関数の例となっている

関数の宣言
（引数無し、戻り値なし）

戻り値はvoid型の値

引数がないので空欄

関数名は変数名と同じように決めることができる

```
void drawPackMan(){
  pushMatrix();
  float angle=PI/6;
  arc(0, 0, 40, 40, angle, 2*PI-angle);
  rotate(angle);
  line(0, 0, 20, 0);
  rotate(-2*angle);
  line(0, 0, 20, 0);
  popMatrix();
}
```

まとめたい処理内容を書く

関数の宣言
（引数有り、戻り値なし）

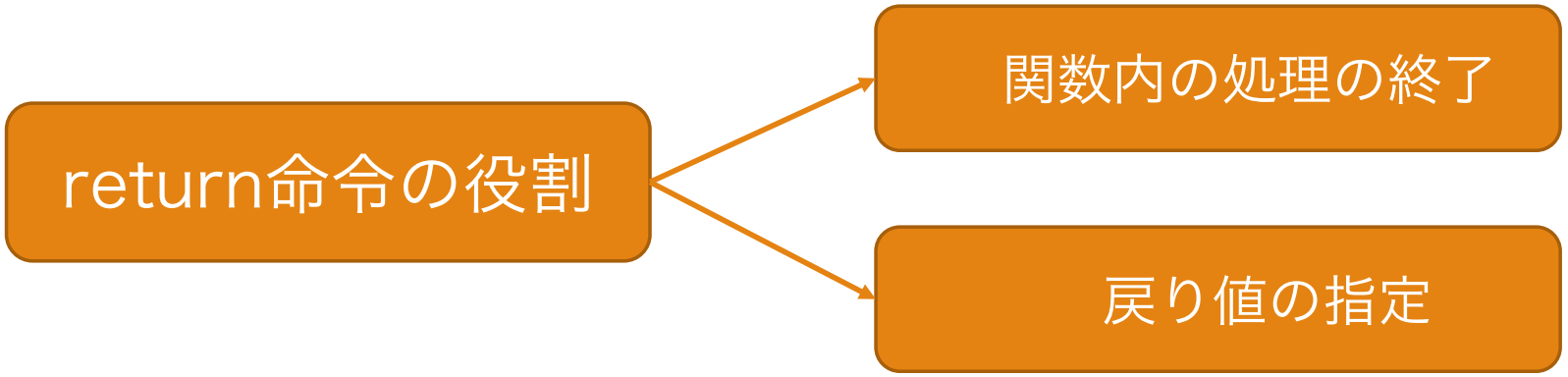
```
void 関数名(データ型名1　引数名1,
            データ型名2　引数名2…){
    関数処理の内容を書きます。
    変数なども使うことができます。
}
```

仮引数

引数名の有効範囲は関数の中だけ
rectなどのこのパターン

関数の宣言
（引数無し、戻り値あり）

戻り値のデータ型 関数名(){
　　関数処理の内容を書きます。
　　どこかに、return命令が必要です。
　　変数なども使うことができます。
}



関数の宣言
（引数無し、戻り値あり）

戻り値はString型の値　　引数がないので空欄

```
String today(){
  String msg = year() +”/”+month() + “/”+day();
  return msg;
}
```

文字列
処理の基礎

文字列+文字列–＞文字列の連結
“kait”+”DeNA” -> “kaitDeNA”
数字+文字列、文字列+数字->文字列の連結

関数の宣言
（引数無し、戻り値あり）

```
PFont font;
String today(){
  String msg = year() + "/" + month() + "/" + day();
  return msg;
}
void setup(){
  size(400,200);
  font = loadFont("SansSerif-48.vlw");
  textFont(font,48);
}
void draw(){
  background(255);
  fill(0);
  String str = today();
  text(str,100,height/2);
}
```

②ここに来るともとに戻る
戻り値は変数strに保存される

①ここに来ると関数todayに
処理が移動する

関数の宣言
（引数あり、戻り値あり）

仮引数（かりひきすう）

戻り値のデータ型 関数名(データ型名1　引数名1,
データ型名2　引数名2…){

関数処理の内容を書きます。
どこかに、return命令が必要です。
変数なども使うことができます。
引数は自由に使うことができる。

}

引数：関数の処理に利用するデータ（値）を与える

dist(x0,y0,x1,y1)
2点（x0,y0）と(x1,y1)の距離を求める組み込み関数
abs(x)
xの絶対値を求める組み込み関数

絶対値

```
float abs(float x){
  if(x > 0){
    return x;
  }else{
    return –x;
  }
}
```

仮引数の有効範囲

仮引数の有効範囲は定義した関数の中だけ

関数が異なれば同じ名前の仮引数を使っても良い

```
戻り値のデータ型 関数名(データ型名1　引数名1,
                        データ型名2　引数名2…){
    関数処理の内容を書きます。
    どこかに、return命令が必要です。
    変数なども使うことができます。
    引数は自由に使うことができる。
}
```

スケッチ（プログラム）の提出に関して

キャリアポートフォリオ上で解答して下さい。
「スケッチをアーカイブ」で保存されるファイルをアップロードして下さい。

今日やった内容

- 関数の定義
- returnの役割
- 仮引数の有効範囲

授業時に配布した資料

http://www.sato-lab.jp/imfu/index.html

においてあります。