Rustに触れる３

ここでは変数の型についてより詳しく解説します。

配列

```
fn main()
{
    let int_hairetu = [1, 3, 5];

    println!("{},{},{}", int_hairetu[0],int_hairetu[1],int_hairetu[2]);
}
```

これは各値を指定する場合の宣言で、すべての値を０にして宣言する場合には以下のようにします。

```
fn main()
{
    let buffer = [0_u8; 256];
}
```

０とは言いましたが０は０でも色々な型の０があるので***u8***（正の整数８bit）の０として要素数256の配列を宣言してみました。

構造体

構造体とは複数の変数を格納しひとまとめにしたもので、Rustではさらにその構造体に専用の関数を定義することが可能です。

まずは基本的な構造体について見ていきましょう。

```
fn main()
{
    let info = Person {age : 18, favorite_num : 3.14};

    println!("Age:{},Favorite Number:{}", info.age, info.favorite_num);
}

struct Person
{
    age : i32,
    favorite_num : f32
}
```

インプリメント

ここでRustで私が最も好きな部分に触れていきます。C言語ではぎりClassに相当するのかなと勝手に思っているものです。

まずは実際のコードを見ていきましょう。

```
fn main()
{
    let new_person = Person::new(18, 3,14);

    new_person.info();

    // 「18,3.14」と出力される
}

struct Person
{
    age : i32,
    favorite_num : f32
}

impl Person
{
    pub fn new(age_ : i32, favorite_number_ : f32)->Person
    {
        Person {age : age_, favorite_num : favorite_number_};
    }

    fn info(&self)
    {
        println!("{},{}", self.age, self.favorite_num);
    }
}
```

ここは重要なところなのでじっくり解説します。まず**インプリメント**とは構造体に対して「振る舞い」として関数を実装するものです。上記コードでは２つの関数をインプリメントしてみました。それぞれ順を追って説明します。

まずはじめに**new**関数です。こちらはintのageとfloatのfavorite_numberを引数とした関数で返り値としてPersonという型の変数を返しています。