今回やること

- プリミティブ型
- (もしかしたらやる) if for while

プリミティブ型(データ型、基本型)

プログラミングにおいて扱われるものが何を表しているものかを示す識別子

数字系

- Int(整数)
- Long(Intより大きい数を表すための整数)
- Float(浮動小数点数)
- Double(Floatより大きい数を表すための浮動小数点数)
- ...まだあるけど割愛

文字系

- Char(一文字)
- String(文字列)

特殊系

- Boolean(真偽)

なぜ型が必要なのか?

- 自動で型を判別するなら必要ないのでは？
  - いつかみんなで同じファイルに書いてあるプログラムを書くときに型がないと...
    - 「この変数は一体なにを意味しているものなんだ...わかんにゃい」
    - 「この変数は多分Int型だからこのまま足し算したらこの結果になるはず！」-> String型だったのでエラーが返ってきました
- ↑のような困りごとがないように型を必ずつけることが強要されているのが従来の古いプログラミング言語(C, Java)
- Python, Kotlinのような割と新しめのプログラミング言語だとは型をつけなくてもいい(型推論という)機能がついている
  - 基本型で変数が代入されているようなパッと見でわかるようなモノには型をつけるのがメンドくさいという歴史から生まれた産物

型が付いてないと少しだけ困ると感じるコード例

```
val a = 1  // Int
val b = "2"  // String

... // ここには数百行もの長い処理が書かれている...!

println(a + b)  // 3かな？
```

if式

```
val a = 1

if (a > 0) {  // もしもこの条件式に合っていればこの中身を実行する
  println("aは自然数です")
} else if (a < 0) {  // それかこれだったら
  println("aは負の数です")
} else {  // それでもないとき
  println("aは0です")
}
```

比較演算子

> 超過
< 未満
<= 以下
>= 以上
== 等価
!= 非等価
&& AND算(かつ)
|| OR算(または)

when式(条件つき)

if式のようにいちいちif〜else if〜elseと書かなくても良いのですっきりした文になってとても見やすい

```
val a = 1

when {
  a > 0 -> println("aは自然数です")
  a < 0 -> println("aは負の数です")
  else -> println("aは0です")
}
```

when式(特定の数)

```
val a = 1

when (a) {
  0 -> println("aは0です")
  1 -> println("aは1です")
  in 2..5 -> println("aは2から5の間のどれかです")
  else -> println("どの条件にも満たしていません")
}
```

次回

- for式
- while文