SecHack365

Unpep - Enhanced Golang Syntax

開発駆動コース 川合ゼミ Jantakorn Passawee

```
val, err := doSomething()
if err != nil {
    return err
}
```

ってめっちゃ書いてませんか？ 事情はわかります、必要だからですよね。Dockerのコア部分であるmoby/moby(Goの行数約**107万**行)でも **if err != nil {** でマッチさせると**17491**件マッチ(*1)します。そうです、**Goはシンプルな言語**ですが、それ故に**自明で煩雑なコードをかなり書かされます**。

*1: 2020/01/27 コミット2ebaeef943cc8c11d63a9175fd132792bcd7d376 にて

この問題を解決するためにいくつかの手段を考えました。**Go風の自作言語**、とか。ああ、ダメだ。新しい言語のセマンティクスを覚えるのはとてもコストがかかりますね。既存のGoユーザは使ってくれない・・・。**シンプルな言語**である特徴は守りたいです。であれば、**シンプルな拡張構文**を与えてあげるのはどうでしょうか。拡張構文であれば、**既存のセマンティクスを保ったまま、よりシンプルなコードの保守**に役立つはずです。

拡張構文の例

```
val := doSomething()?;
```

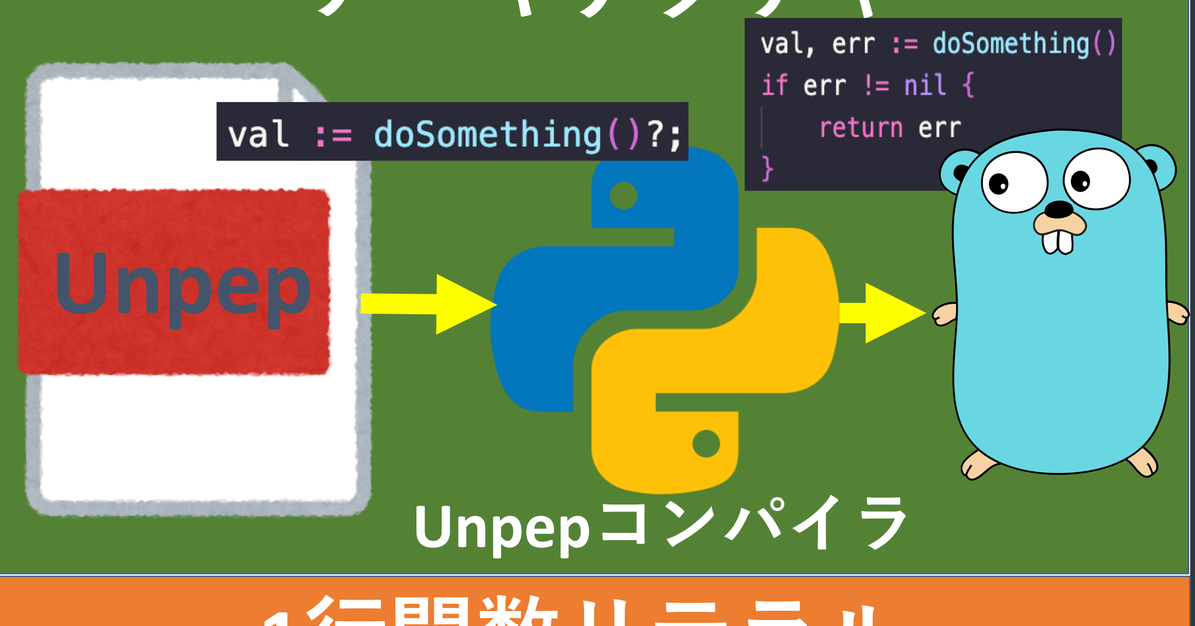
GoのShortVarDecl

```
ShortVarDecl =
    IdentifierList ":=" ExpressionList

ex.) a, b, err := f(1+2*3)[:], 0xff, nil
```

UnpepのShortVarDecl

```
ShortVarDecl =
  IdentifierList := ExpressionList [DeclAnnotation]

DeclAnnotation = "!" | "?" | "else" (ExpressionList | Block)
```

アーキテクチャ



1行関数リテラル

```
(x, y int) (int, int) -> x + y, x * y
```

パイプ演算子

```
n := 1 |> double |> triple
```

Unpepのこれから

- 意味解析の実装
- 既存Go言語文法の完全対応
- セミコロン自動挿入
- より新しい拡張構文の模索

NICT 国立研究開発法人 情報通信研究機構

Special Thanks

- **The Go Programming Language Specification**
- **github.com/lark-parser/lark**