ATL Translator

```
module Translator
( SourceCode, TransCode, TargetCode
, Translator(Translator)
    , trans_blocks
    , trans_root_block
    , trans_convert_filepath
, Block(Block)
    , block_title
    , block_children
    , block_format
    , block_does_nest
, BlockChild(ChildCode, ChildBlock)
) where

import Utilities

type SourceCode = String
type TransCode  = String
type TargetCode = String

data Translator = Translator
    { trans_blocks           :: [Block]
    , trans_root_block       :: Block
    , trans_convert_filepath :: FilePath -> FilePath }

data Block = Block
    { block_title     :: String
    , block_children  :: [BlockChild]
    , block_format    :: [TargetCode] -> TargetCode
    , block_does_nest :: Bool }

data BlockChild
    = ChildCode  SourceCode
    | ChildBlock Block

instance Show Block where
    show block
        = "{ " ++ (block_title block) ++ " : "
        ++ (show $ reverse $ map show $ block_children block) ++ " }"

instance Show BlockChild where
    show child = case child of
        ChildCode x -> x
        ChildBlock x -> show x
```