# ATL Parse

```
module Parse (parse) where
```

This module supplies a single function, `parse`, which traslates the text of an input file into an array of `ParseItem` which are easy to transpile using a transpilation blueprint (provided during compilation).

```
strings_to_separate :: [String]
strings_to_separate =
    [" "]

strings_to_ignore :: [String]
strings_to_ignore =
    ["\n"]

begins :: String -> String -> Bool
begins a b = case a, b of
    ""  , _    -> True
    _   , ""   -> False
    x:xs, y:ys -> if x == y
        then xs `begins` ys
        else False

begins_any_of :: String -> [String] -> Bool
begins_any_of a bs = any $ map (\b -> a `begins` b) bs
```

Parsing breaks the input file into a list of `Token`s

```
type Token = String



parse :: String -> [Token]
parse lines
    = ignore
    $ separate lines ""

ignore :: [Token] -> [Token]
ignore [] = []
ignore (t:ts) = if t `elem` strings_to_ignore
```

```
    then ignore ts
    else t : ignore ts

separate :: String -> Token -> [Token]
separate [] _ = []
separate (c:cs) working_token =
    let new_token = working_token ++ [c]
    in if new_token `elem` strings_to_separate
        then new_token : separate cs ""
        else separate cs new_token
```