SCHwiftc

# Introduction

Dans le cadre du cours de compilateur, nous avons décidé de définir un langage de programmation dénommé ***schwiftC***, inspiré de la série ***Rick and Morty***.

# But

Créer un compilateur ***schwiftC*** en python à l’aide des modules ***PLY*** et ***AST***. Le compilateur lit le ***schwiftCode*** et génère du code C.

Le code devra être écrit en deux parties, une décrivant les méthodes utilisées par le programme et l’autre le code principal qui sera exécuté, l’équivalent d’un *main()*.

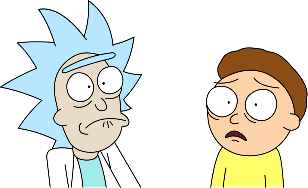
*« Come on, get schwifty »*

# Description du langage

## Structure

Le code se sépare en deux parties bien définies, les méthodes utilisateurs et le code principal.

Elles seront séparées par 42 ‘=‘ contigus.



Méthodes utilisateurs

==========================================

=======

=======

Code principal

### Méthodes utilisateur

On commence la déclaration d’une méthode à l’aide du mot-clé « meeseeks», suivi des paramètres. Le corps débute par un PIF et se termine par un PAF (oui.) qui doit être suivi du type de retour de la méthode.

meeseeks <name> <type1> <attr1>, <type2> <attr2> ...

PIF

<code>

didit <return\_value>~

PAF <return\_type>~

## Mots-clés

### Opérateurs de base

|  |  |
| --- | --- |
| *SchwiftC* | *C* |
| \* | + |
| / | - |
| - | / |
| + | \* |
| got | = |

### Opérateurs de comparaison

|  |  |
| --- | --- |
| *SchwiftC* | *C* |
| tiniest | < |
| tinier | <= |
| fattest | > |
| fatter | >= |
| is | == |
| isnot | != |

### Types simples

|  |  |
| --- | --- |
| *SchwiftC* | *C* |
| hey | int |
| thong | string |
| isit | boolean |
| fake | float |
| schmeckle | char |
| mpfh | void |

### Autres mots-clés

|  |  |
| --- | --- |
| *SchwiftC* | *C* |
| jeez | if |
| WUBBALUBBADUBDUBS | for |
| whale | while |
| shutupmorty | break |
| cando | do |
| schwift | switch |
| heyrick | case |
| didit | return |
| SHOWMEWHATYOUGOT | print |

### Délimiteurs

|  |  |
| --- | --- |
| *SchwiftC* | *C* |
| PIF | { |
| PAF | } |
| ~ | ; |

## Utilisation