Hecho por:
Juan Manuel Castro De La Hoz | 202312763
Sebastián Torres González | 202223811

P0-Lenguajes y Máquinas

Primero se define todas las reglas que se van a utilizar en regex:

```
##Numero
numero = r"^-?\d+(\.\d+)?$"

##bloque
bloque = r"\n*{.*;*}\n*"

##var
var_def_regex = rf"NEW VAR (\w+)\s*=\s*{numero}]"

##Macro
macro_def_regex = rf"NEW MACRO\s*(\w+)\s*\((\w*\s*)\)*{bloque}*"

##EXEC
exec_block_regex = rf"EXEC\s*{bloque}"

##Valores
lista_valores = ["size", "myX", "myY", "myChips", "myBalloons", "balloonsHere", "chipsHere", "roomForChips"]
Valores_Patron = "|".join(re.escape(palabra) for palabra in lista_valores)
valores_regex = rf"{Valores_Patron}"

##Comandos
    ##direcciones
direcciones_simples = r"\s*(forward|left|right|back|backwards)\s*"
direcciones_complejas = r"\s*(north|south|east|west)\s*"
    ##Todos los comandos
turnToMy = rf"turnToMy\s*\(\s*{direcciones_simples}\s*\)"
turnToThe = rf"turnToMy\s*\(\s*{direcciones_complejas}\s*\)"
comandos_para_w_j_d_p_g_lG_pop = r"(walk|jump|drop|pick|grab|letGo|pop)\s*"
w_j_d_p_g_lG_pop = rf"{comandos_para_w_j_d_p_g_lG_pop}\(\s*{numero}\s*\)"
moves = rf"moves\s*\(\s*({direcciones_simples},)*\s*\)"
nop = r""
todos_comandos_msafeExe = rf"\s*({turnToMy}|{turnToThe}|{w_j_d_p_g_lG_pop}|{moves}|{nop})\s*"
safeExe = rf"safeExe\s*\({todos_comandos_msafeExe}\)"
##para añadir valores añadir a la lista
lista_comandos = [r"{turnToMy}", r"{turnToThe}", r"{w_j_d_p_g_lG_pop}", r"{moves}", r"{safeExe}", r"{nop}"]
comandos_Patron = "|".join(re.escape(palabra) for palabra in lista_comandos)
command_regex = rf"({comandos_Patron})\s*\(\s*(.*)\s*\)"

##Condicionales y condicion
condicion = r"(isBlocked\?|isFacing\?|zero\?)\s*\(\s*(.*)\s*\)"
condicional_then = rf"\n+then\s*{bloque}"
condicional_else = rf"(\n+else\s*{bloque})?"
condicional_then_else = rf"({condicional_then}{condicional_else})*"
condicnoal_not = r"\s+(not?)"
condicional = rf"if{condicnoal_not}\s*\(({condicion})\s*\){condicional_then_else}"

##Loop
loop = rf"do\s*{condicion}\s*{bloque})\s*"

##repetirnumveces
Repeticion = rf"rep\s*{numero}\s*{bloque}"
```

En el comentario **“Numero”** se define la estructura de un número.

```
##Numero
numero = r"^-?\d+(\.\d+)?$"
```

En el comentario **“bloque”** se define la estructura de un bloque.

```
##bloque
bloque = r"\n*{.*;*}\n*"
```

En el comentario **“var”** se define la estructura de una variable a la cual se le asigna un nombre y la variable **“numero”**.

```
##var
var_def_regex = rf"NEW VAR (\w+)\s*=\s*{numero}]"
```

En el comentario **“Macro”** se le asigna un nombre a dicha macro y se le pide que se le añada algo o nada dentro de un paréntesis, en el documento no decía que la macro llevaba un **“bloque”** pero en los ejemplos sí la lleva debido a esto se añadió la posibilidad de tener un “**bloque”** al final de la macro.

```
##Macro
macro_def_regex = rf"NEW MACRO\s*(\w+)\s*\((\w*\s*)\)*{bloque}*"
```

En el comentario **“EXEC”** se le añade un bloque de código donde presuntamente se ejecuta todo.

```
##EXEC
exec_block_regex = rf"EXEC\s*{bloque}"
```

En el comentario **“Valores”** se hace una lista de valores la cuál se añade a la regla al final del proceso mediante un for loop que recorre la lista y la delimita con |, haciendo la regla (x|y|z|...)*. Esto se hace con el propósito de poder añadirle un valor nuevo a la lista cuando sea necesario.

```
##Valores
lista_valores = ["size", "myX", "myY", "myChips", "myBalloons", "balloonsHere", "chipsHere", "roomForChips"]
Valores_Patron = "|".join(re.escape(palabra) for palabra in lista_valores)
valores_regex = rf"({Valores_Patron})"
```

En el comentario **“Comandos”** hay múltiples sub comentarios.

```
##Comandos
    ##direcciones
direcciones_simples = r"\s*(forward|left|right|back|backwards)\s*"
direcciones_complejas = r"\s*(north|south|east|west)\s*"
    ##Todos los comandos
turnToMy = rf"turnToMy\s*\(\s*{direcciones_simples}\s*\)"
turnToThe = rf"turnToMy\s*\(\s*{direcciones_complejas}\s*\)"
comandos_para_w_j_d_p_g_lG_pop = r"(walk|jump|drop|pick|grab|letGo|pop)\s*"
w_j_d_p_g_lG_pop = rf"{comandos_para_w_j_d_p_g_lG_pop}\(\s*{numero}\s*\\)"
moves = rf"moves\s*\(\s*({direcciones_simples},)*\s*\)"
nop = r""
todos_comandos_msafeExe = rf"\s*({turnToMy}|{turnToThe}|{w_j_d_p_g_lG_pop}|{moves}|{nop})\s*"
safeExe = rf"safeExe\s*\({todos_comandos_msafeExe}\)"
##para añadir valores añadir a la lista
lista_comandos = [r"{turnToMy}", r"{turnToThe}", r"{w_j_d_p_g_lG_pop}", r"{moves}", r"{safeExe}", r"{nop}"]
comandos_Patron = "|".join(re.escape(palabra) for palabra in lista_comandos)
command_regex = rf"({comandos_Patron})\s*\(\s*(.*)\s*\)"
```

En el comentario **“direcciones”** se facilita el acceso a las diferentes direcciones que se puedan llegar a necesitar.

```
    ##direcciones
direcciones_simples = r"\s*(forward|left|right|back|backwards)\s*"
direcciones_complejas = r"\s*(north|south|east|west)\s*"
```

En el comentario **“Todos los comandos”** como indica el nombre se encuentran todos los tipos de comandos y lo que reciben dichos comandos, en un caso hubieron varios comandos los cuales se facilitaba meterlos en una variable ya que su comportamiento es prácticamente el mismo.

```
    ##Todos los comandos
turnToMy = rf"turnToMy\s*\(\s*{direcciones_simples}\s*\)"
turnToThe = rf"turnToMy\s*\(\s*{direcciones_complejas}\s*\)"
comandos_para_w_j_d_p_g_lG_pop = r"(walk|jump|drop|pick|grab|letGo|pop)\s*"
w_j_d_p_g_lG_pop = rf"{comandos_para_w_j_d_p_g_lG_pop}\(\s*{numero}\s*\\)"
moves = rf"moves\s*\(\s*({direcciones_simples},)*\s*\)"
nop = r""
todos_comandos_msafeExe = rf"\s*({turnToMy}|{turnToThe}|{w_j_d_p_g_lG_pop}|{moves}|{nop})\s*"
safeExe = rf"safeExe\s*\({todos_comandos_msafeExe}\)"
```

En el comentario **“para añadir valores añadir a la lista”** Se encuentra un código similar a el que se encuentra en **“Valores”** la diferencia se da debido a que los valores dentro de la lista no son str sino unas variables complejas que contienen código regex.

```
##para añadir valores añadir a la lista
lista_comandos = [r"{turnToMy}", r"{turnToThe}", r"{w_j_d_p_g_lG_pop}", r"{moves}", r"{safeExe}", r"{nop}"]
comandos_Patron = "|".join(re.escape(palabra) for palabra in lista_comandos)
command_regex = rf"({comandos_Patron})\s*\(\s*(.*)\s*\)"
```

En el comentario **“Condicionales y condicion”** se encuentran todas las posibles condiciones y condicionales, a las condiciones se le dan los posibles valores que pueden tomar y las condicionales se estructuran con bloques y con las mismas condiciones. En este se usan **“direcciones_simples”, direcciones_complejas”, “numero”, “bloque”** y las que vienen dentro del comentario.

```
##Condicionales y condicion
isBlocked = rf"isBlocked\?\s*\(\s*{direcciones_simples}\s*\)"
isFacing = rf"isFacing\?\s*\(\s*{direcciones_complejas}\s*\)"
zero = rf"zero\?)\s*\(\s*{numero}\s*\)"
condicion = rf"({isBlocked}|{isFacing}|{zero})"
condicional_then = rf"\n+then\s*{bloque}"
condicional_else = rf"(\n+else\s*{bloque})?"
condicional_then_else = rf"({condicional_then}{condicional_else})*"
condicinoal_not = r"\s+(not?)"
condicional = rf"if{condicinoal_not}\s*\(({condicion})\s*\){condicional_then_else}"
```

En el comentario **“Loop”** se crea la lectura se loop y su estructura, también se usa **“Condición”** y **“bloque”**

```
##Loop
loop = rf"do\s*{condicion}\s*{bloque})\s*"
```

En el comentario **“repetirnumveces”** se hace la estructura necesaria para que un bloque se repita x veces, para esto se utiliza **“numero”** y **“bloque”**.

```
##repetirnumveces
Repeticion = rf"rep\s*{numero}\s*{bloque}"
```

El resto del código se utiliza para poder leer un comando o un archivo txt y chequear syntaxis teniendo en cuenta todas las normas/lógica escrita en regex.