Processamento de Linguagem - LOGO

João Azevedo Nº18845

João Rodrigues Nº19431

Carlos Santos Nº19432

Instituto Politécnico Cávado do Ave

Data: 19/01/2021

# Resumo

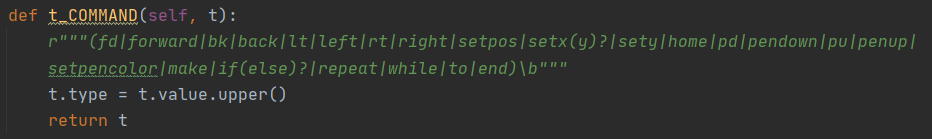
O Logo é uma linguagem de programação educativa. Pretende-se implementar um interpretador um interpretador básico, capaz de simular o maior número possível de comandos da linguagem original. Nesta linguagem o programador controla uma tartaruga, que se vai mexendo através do espaço, e desenhando linhas por onde passa.

O programa será um ficheiro de texto com uma ou mais linhas, e deve implementar, pelo menos, os seguintes comandos:

* O comando **fd** ou **foward** – move a tartaruga n pixies em frente: **fd** 10;
* O comando **bk** ou **back** – move a tartaruga, para trás: **back** 20;
* O comando **lt** ou **left** roda a tartaruga, para a esquerda, n graus: **left** 90;
* O comando **rt** ou **right** roda a tartaruga, para a direita, n graus: **right** 180;
* O comando **setpos**, **setxy**, **setx**, **sety** permite definir uma posição para a qual a tartaruga se deve movimentar: **setpos** [100 100], ou **setxy** 100 100 ou um eixo apenas, mantendo o outro eixo: **setx** -100;
* O comando gome move a tartaruga para o ponto inicial (0,0), e roda-a para a orientação original: **home**;
* Os comandos **pendown** e **pwnup**, respetivamente abreviados por **pd** e **pu**, permitem alternar entre o modo de desenho e o modo de movimentação livre;
* O comando **setpencolor** permite alterar a cor das linhas para os comandos que se seguem: **setpencolor** [99 0 0];
* O comando **make** permite definir o valor de uma variável: **make“varname**20;
* O comando **if** e **ifelse** permitem definir estruturas condicionais;
* O comando **repeat** repete um conjunto de comandos;
* O comando **while** permite definir ciclos;
* O comando **to** permite criar funções.

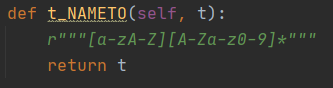
# Expressões Regulares

Esta foi a expressão regular usada de forma a ler apenas os comandos corretos:



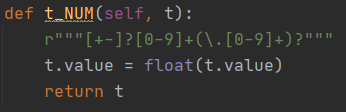
Nesta expressão são lidos apenas os comandos pretendidos e suas abreviaturas, como o caso do **foward** ou **fd**.

Esta foi a expressão regular usada de forma a ler apenas os nomes das funções:



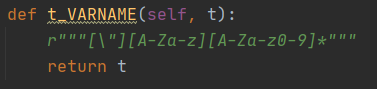
Nesta expressão são lidos apenas os nomes das funções.

Esta foi a expressão regular usada de forma a ler apenas os números:



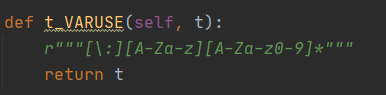
Nesta expressão regular é lido qualquer número positivo ou negativo e com ou sem vírgulas, isto é -3 ou -3,14.

Esta foi a expressão regular usada de forma a ler apenas os nomes e os respetivos valores das varáveis:



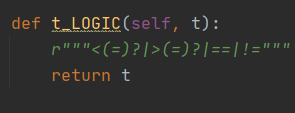
Nesta expressão regular são lidas as variáveis que contenham o sinal **“**antes do seu nome e o respetivo valor associado, como por exemplo “numero10.

Esta foi a expressão regular usada de forma a ler apenas os nomes das varáveis que já possuem valores associados:



Nesta expressão regular são lidas as variáveis que já possuam valores ou seja variáveis com “**:”** antes do nome da variável, um exemplo prático seria se já existir uma variável (“numero 10) e no código possuir “:numero” o valor associado a essa variável (10) irá ser usado.

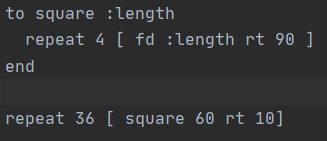
Esta foi a expressão regular usada de forma a ler operadores lógicos:



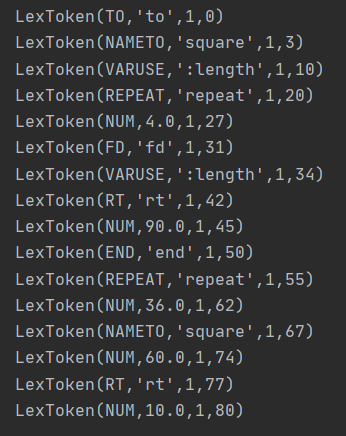
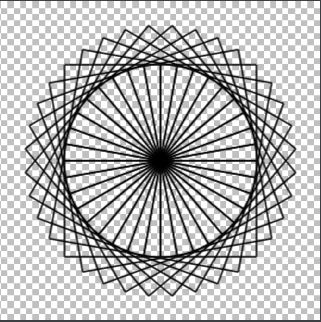
Nesta expressão regular são lidos operadores lógicos que possam ser usados como comparadores (==, >, <, <=, >=, !=).

# Testes

O input posto foi o seguinte:

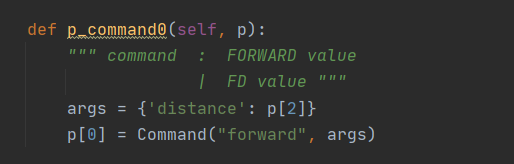


Com base neste código o output foi o seguinte:

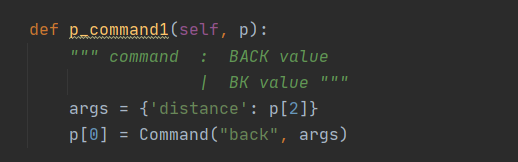


# Gramática

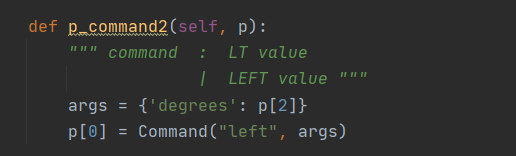
No Parser.py tem um conjunto de analisadores sintáticos de cada comando.



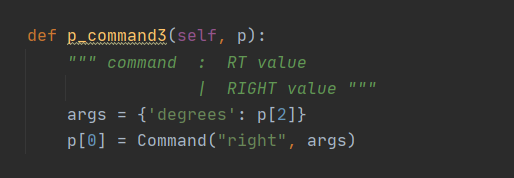
Na funcão p\_command0, o comando forward ou fd é precedido de um valor, esse valor corresponde à distância que irá andar para a frente.



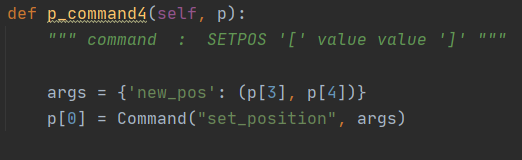
Na função p\_command1, comando back ou bk é precedido de um valor, esse valor corresponde à distância que irá andar para trás.

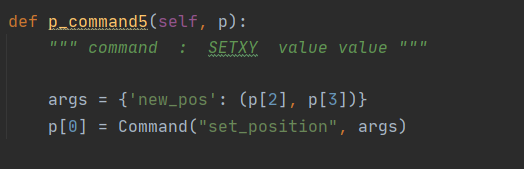


Na função p\_command2, comando left ou lf é precedido de um valor, esse valor corresponde aos graus que pretende virar no sentido anti-horário.

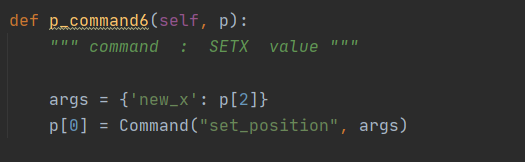


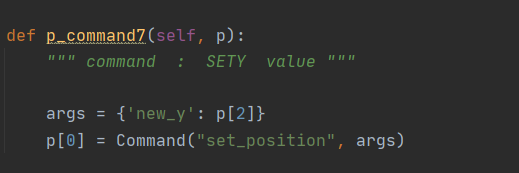
Na função p\_command3, o comando right ou rt é precedido de um valor, esse valor corresponde aos graus que pretende virar no sentido horário.



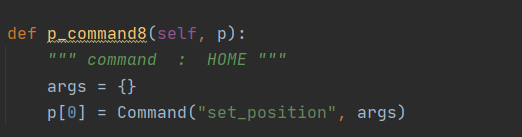


Na função p\_command4 e p\_command5, o comando setpos e setxy são precedidos de dois valores entre parênteses ([10 10]) e sem eles (10 10) respetivamente.

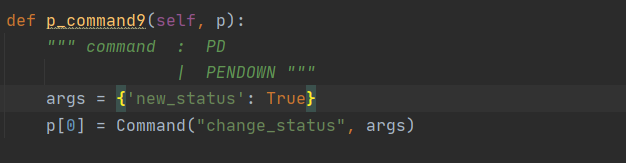


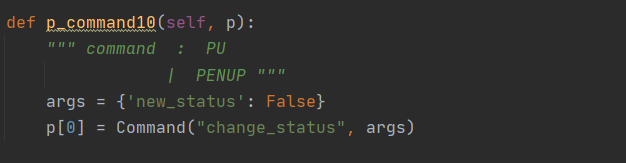


Na função p\_command6 e p\_command7, os comandos setx ou sety são precedidos de um valor, esse valor corresponde a posição x que se pretende alterar no caso do comando setx e posição y no caso do comando sety.



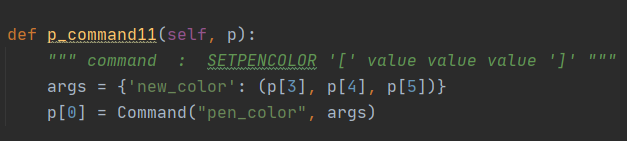
Na função p\_command8, o comando home tem como função voltar a posição inicial do desenho.



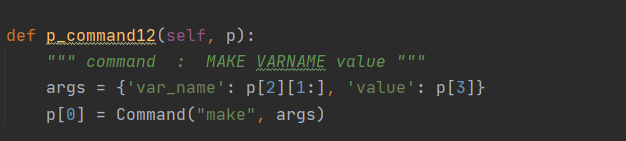


Na função p\_command9, o comando pendown ou pd tem a função de desenhar as instruções seguintes de forma a que se consiga ver o desenho feito.

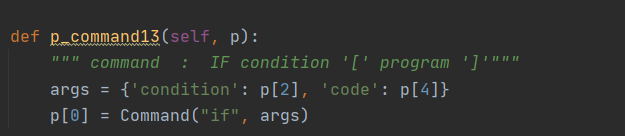
No caso da p\_command10, o comando penup ou pu tem a função de desenhar as instruções seguintes de forma a que não se consiga ver o desenho feito.



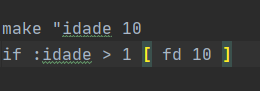
Na função p\_command11, o comando setpencolor é precedido de três valores entre parênteses [100 0 100], esses valores correspondem a uma cor que se pretenda ver no desenho.

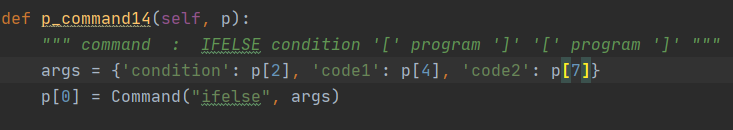


Na função p\_command12, o comando make é precedido de uma variável (varname) e de um valor, esse valor corresponde ao valor da variável que poderá ser usada em outras estruturas com as estruturas condicionais, entre outras.

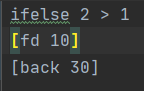


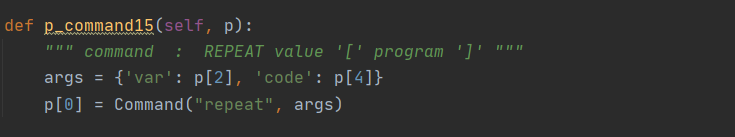
Na função p\_command13, o comando if é precedido de uma condição, e entre parênteses o código que se pretenda ser executado.





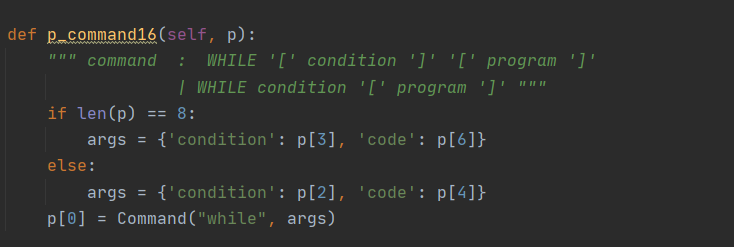
Na função p\_command14, o comando ifelse é precedido de uma condição, e entre os primeiros parênteses o código que se pretenda ser executado no caso de essa condição seja verdadeira e os segundos parênteses o código a executar caso a condição seja falsa.



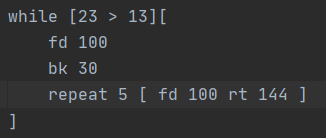


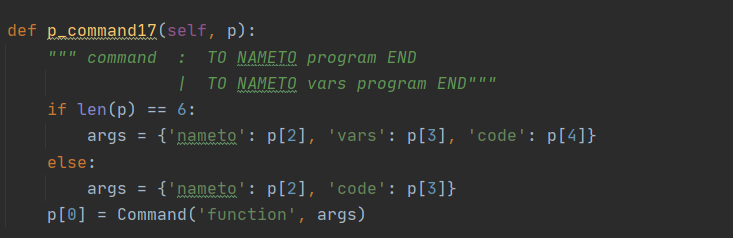
Na função p\_command15, o comando **repeat** é precedido de um valor, que representa o número de repetições que devem ser feitas ao código que esta entre parênteses.





Na função p\_command16, o comando while tem duas variantes, no caso da primeira a condição deve estar entre parênteses assim como o código a executar. Na segunda variante a condição não necessita de estar entre parênteses, no entanto o código a executar sim. Em ambas as situações o código só executa caso a condição seja verdadeira.





Na função p\_command17, o comando to tem duas variantes, no caso da primeira possui o nome da função (nameto), o conjunto de código a ser executado (program) e o terminador de uma função (end). Na segunda possui o nome da função (nameto), os argumentos (vars) que a função necessita para ser executada, o conjunto de código a ser executado (program) e o terminador de uma função (end).

