# Manners

Uma linguagem textual e educada em inglês

**Gabriel Moreira**

# Descrição

A ideia da linguagem era fazer uma linguagem para ensinar as pessoas a serem devidamente educadas com o computador, deste modo é possível aprender boas maneiras e programação ao mesmo tempo. Além disso a linguagem por ser muito textual, pode ser boa para iniciantes que precisam deixar seu código mais claro possível

# Como usar a linguagem

# Variáveis

```
please store flag as boolean
please change the value of flag to false
```

Equivalente em Python:

flag = False

# Loop

```
could you please do {
        print flag
        please change the value of flag to flag+1
    } while the following condition is true (flag < a)?
```

Equivalente em Python:

```
while flag < a:
        print(flag)
        flag +=1
```

# Condicionais

```
please do this {
        please change the value of flag to true
    } if the following condition is true (n = 1)
    else{}
```

Equivalente em Python:

```
if n ==1:
        flag = True
```

# Funções que não retornam

```
computer, the void called printplusone will receive (a as
integer), whenever I call it, please do {
    print a+1
}
```

Equivalente em Python:

```
def printplusone(a):
    print(a+1)
```

# Funções que retornam

```
computer, the function called maior will plusone (a as
integer, b as integer) and return integer, whenever I call
it, please do {
    plusone = a + 1
}
```

Equivalente em Python:

```
def plusone(a):
        return a+1
```

# EBNF

```
Program = "computer", ",", "the",  SubDec|FuncDec;
SubDec = "void", "called", "identifier", "will", "receive", "(", { | ("identifier", as, Type)}, ")", ",", "whenever", "I",
"call", "it", "please", "do", "{", { | ( Statement, "\n")}, "}";
FuncDec = "function", "called", "identifier", "will", "receive", "(", { | ("identifier", as, Type)}, ")", "and", "return",
Type, ",", "whenever", "I", "call", "it", "please", "do", "{", { | ( Statement, "\n")}, "}";
RelExpression = Expression, {"=" | ">" | "<"}, Expression;
Expression = Term, {("+" | "-" | "or"),Term | ;
Term = Factor, {("*" | "/" | "and"), Factor} | ;
Factor = "number" | {"boolean" | ("identifier",{| {"("{(| RelExpression, {| ","})}}}) | {("+" | "-" | "not"), Factor} | "(",
RelExpression, ")" | "input";
Statement = |("please", "change", "the", "value", "of", "identifier", "to", RelExpression ) | ("please", "print", RelExpression)
| ("please", "store", "identifier", "as",  Type) | "please", "do", "this", "{", {| (Statement, "\n"), "}", "if", "the",
"following", "condition", "is", "true" "(", RelExpression, ")", {| ("else", "{", {| (Statement, "\n")}}, "}") | ("please",
"call", "identifier", "(", {| {RelExpression, {| ","}}) | ("could", "you", "please", "do", "this", "{", |(Statement, "\n"),"}",
"while", "the", "following", "condition", "is", "true", "{",  RelExpression, "}", "?"));
Type = "integer"|"boolean"
```

# Observações

Apesar dos exemplos estarem ligados ao Python, a linguagem deve possuir um void chamado main, como feito nos programas escritos em C.

```
computer, the void called main will receive (),
whenever I call it, please do {
    please print fibonacci(9)
    please call countdown(8)
}
```

# Exemplo

```
computer, the function called factorial will receive (n as integer) and return
integer, whenever I call it, please do {
   please do this {
       please change the value of factorial to n
   } if the following condition is true (n = 1)
   else{
       please change the value of factorial to n*factorial(n-1)
   }
}

computer, the void called main will receive (), whenever I call it, please do {
   please print factorial(3)
}
```

output: 6