SSRoboime

Generated by Doxygen 1.9.8

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Agent Namespace Reference

Classes

- class Agent

  Classe que representará os agentes de campo, possuindo métodos correspondentes.

5.2 AgentPenalty Namespace Reference

5.3 BaseAgent Namespace Reference

Classes

- class BaseAgent

  Classe que agrupará todas as funcionalidades comuns a qualquer agente.

5.4 Booting Namespace Reference

Classes

- class Booting

  Responsável por inicializar todas as necessidades de execução do time.

5.5 Printing Namespace Reference

Classes

- class Printing

  Responsável pela comunicação usuário - terminal.

5.6 run_player Namespace Reference

Variables

- boot = Booting()
- str if_debug_mode = '1'
- player = Agent(boot.options)

5.6.1 Variable Documentation

5.6.1.1 boot

```
run_player.boot = Booting()
```

Definition at line 4 of file run_player.py.

5.6.1.2 if_debug_mode

```
str run_player.if_debug_mode = '1'
```

Definition at line 5 of file run_player.py.

5.6.1.3 player

```
run_player.player = Agent(boot.options)
```

Definition at line 7 of file run_player.py.

5.7 ServerComm Namespace Reference

Classes

- class ServerComm

  Responsável pela comunicação com servidor.

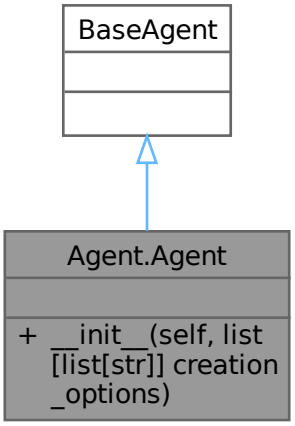
Chapter 6

Class Documentation

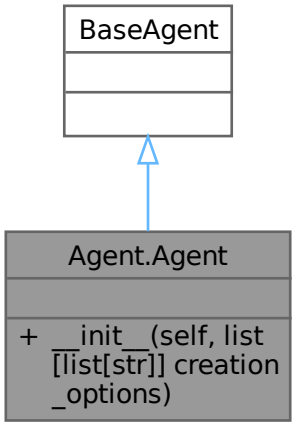
6.1 Agent.Agent Class Reference

Classe que representará os agentes de campo, possuindo métodos correspondentes.

Inheritance diagram for Agent.Agent:



Collaboration diagram for Agent.Agent:



Public Member Functions

- __init__ (self, list[list[str]] creation_options)

  Construtor da classe agente de campo, inicializando informações gerais.

6.1.1 Detailed Description

Classe que representará os agentes de campo, possuindo métodos correspondentes.

Definition at line 7 of file Agent.py.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
Agent.Agent.__init__ (
              self,
             list[list[str]] creation_options )
```

Construtor da classe agente de campo, inicializando informações gerais.

Parameters

creation_options	Lista de Parâmetros de Criação de Agente

Parâmetros presentes em `creation_options`:

-
-

Definition at line 11 of file Agent.py.

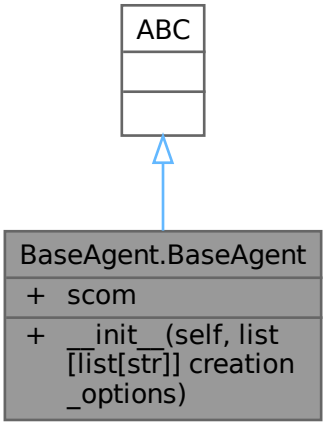
The documentation for this class was generated from the following file:

- src/agent/Agent.py

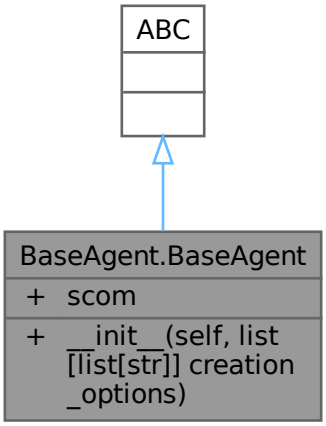
6.2 BaseAgent.BaseAgent Class Reference

Classe que agrupará todas as funcionalidades comuns a qualquer agente.

Inheritance diagram for BaseAgent.BaseAgent:



Collaboration diagram for BaseAgent.BaseAgent:



Public Member Functions

- __init__ (self, list[list[str]] creation_options)

  Construtor da classe base de agente, chamando todos os construtores de outras classes mínimas para cada agente.

Public Attributes

- scom

6.2.1 Detailed Description

Classe que agrupará todas as funcionalidades comuns a qualquer agente.

Definition at line 8 of file BaseAgent.py.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

```
BaseAgent.BaseAgent.__init__ (
              self,
             list[list[str]] creation_options )
```

Construtor da classe base de agente, chamando todos os construtores de outras classes mínimas para cada agente.

Parameters

creation_options	Lista de Parâmetros de Criação de Agente

Definition at line 13 of file BaseAgent.py.

6.2.3 Member Data Documentation

6.2.3.1 scom

`BaseAgent.BaseAgent.scom`

Definition at line 20 of file BaseAgent.py.

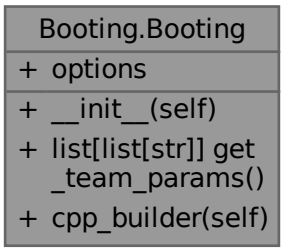
The documentation for this class was generated from the following file:

- src/agent/BaseAgent.py

6.3 Booting.Booting Class Reference

Responsável por inicializar todas as necessidades de execução do time.

Collaboration diagram for Booting.Booting:



Public Member Functions

- __init__ (self)

  Responsável por chamar as inicializações mínimas.

Static Public Member Functions

- list[list[str]] get_team_params ()

  Verifica existência de arquivo de parâmetros de time, caso não exista, usará o default.
- cpp_builder (self)

  Responsável por buildar os arquivos .cpp presentes na pasta cpp.

Public Attributes

- options

6.3.1 Detailed Description

Responsável por inicializar todas as necessidades de execução do time.

Assume as seguintes responsabilidades:

- Estabelece um arquivo de configurações default caso já não exista um.

Definition at line 9 of file Booting.py.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 __init__()

```
Booting.Booting.__init__ (
              self )
```

Responsável por chamar as inicializações mínimas.

Definition at line 17 of file Booting.py.

6.3.3 Member Function Documentation

6.3.3.1 cpp_builder()

```
Booting.Booting.cpp_builder (
              self )  [static]
```

Responsável por buildar os arquivos .cpp presentes na pasta cpp.

Returns

Funcionalidades C++ em condições de interoperabilidade.

Definition at line 74 of file Booting.py.

6.3.3.2 get_team_params()

```
list[list[str]] Booting.Booting.get_team_params ( )  [static]
```

Verifica existência de arquivo de parâmetros de time, caso não exista, usará o default.

Faremos em tupla para permitir uso mínimo de memória.

Returns

Definition at line 32 of file Booting.py.

6.3.4 Member Data Documentation

6.3.4.1 options

`Booting.Booting.options`

Definition at line 22 of file Booting.py.

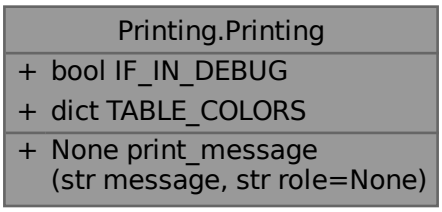
The documentation for this class was generated from the following file:

- src/term/Booting.py

6.4 Printing.Printing Class Reference

Responsável pela comunicação usuário - terminal.

Collaboration diagram for Printing.Printing:



Static Public Member Functions

- None print_message (str message, str role=None)

  Apresentará uma mensagem estilizada de forma específica.

Static Public Attributes

- bool IF_IN_DEBUG = False
- dict TABLE_COLORS

6.4.1 Detailed Description

Responsável pela comunicação usuário - terminal.

Definition at line 6 of file Printing.py.

6.4.2 Member Function Documentation

6.4.2.1 print_message()

```
None Printing.Printing.print_message (
            str message,
            str role = None )  [static]
```

Apresentará uma mensagem estilizada de forma específica.

Parameters

message	Mensagem a ser apresentada
role	String indicando qual o motivo da mensagem

Há uma quantidade específica de roles possíveis:

- info
- warning
- error Caso nenhuma dessas seja inserida, há a possibilidade de inserir o comando ASCII de uma vez.

Definition at line 18 of file Printing.py.

6.4.3 Member Data Documentation

6.4.3.1 IF_IN_DEBUG

```
bool Printing.Printing.IF_IN_DEBUG = False  [static]
```

Definition at line 10 of file Printing.py.

6.4.3.2 TABLE_COLORS

```
dict Printing.Printing.TABLE_COLORS  [static]
```

Initial value:

```
=  {
        "info": "\033[1;36m",
        "warning": "\033[1;33m",
        "error": "\033[1;31m"
    }
```

Definition at line 11 of file Printing.py.

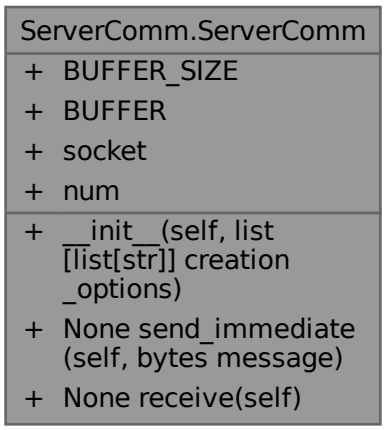
The documentation for this class was generated from the following file:

- src/term/Printing.py

6.5 ServerComm.ServerComm Class Reference

Responsável pela comunicação com servidor.

Collaboration diagram for ServerComm.ServerComm:



Public Member Functions

- __init__ (self, list[list[str]] creation_options)

  Construtor da classe, inicializando buffers e a conexão de cada agente com servidor.
- None send_immediate (self, bytes message)

  Envia uma mensagem instantânea ao servidor, verificando se a conexão continua ativa.
- None receive (self)

  Receberá informações diretamente do servidor.

Public Attributes

- BUFFER_SIZE
- BUFFER
- socket
- num

6.5.1 Detailed Description

Responsável pela comunicação com servidor.

Definition at line 10 of file ServerComm.py.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 __init__()

```
ServerComm.ServerComm.__init__ (
              self,
             list[list[str]] creation_options )
```

Construtor da classe, inicializando buffers e a conexão de cada agente com servidor.

Definition at line 15 of file ServerComm.py.

6.5.3 Member Function Documentation

6.5.3.1 receive()

```
None ServerComm.ServerComm.receive (
              self )
```

Receberá informações diretamente do servidor.

Definition at line 77 of file ServerComm.py.

6.5.3.2 send_immediate()

```
None ServerComm.ServerComm.send_immediate (
              self,
             bytes message )
```

Envia uma mensagem instantânea ao servidor, verificando se a conexão continua ativa.

Coloca-se na frente uma informação de tamanho da mensagem dentro de 4 bytes.

Definition at line 63 of file ServerComm.py.

6.5.4 Member Data Documentation

6.5.4.1 BUFFER

```
ServerComm.ServerComm.BUFFER
```

Definition at line 22 of file ServerComm.py.

6.5.4.2 BUFFER_SIZE

```
ServerComm.ServerComm.BUFFER_SIZE
```

Definition at line 21 of file ServerComm.py.

6.5.4.3 num

`ServerComm.ServerComm.num`

Definition at line 29 of file ServerComm.py.

6.5.4.4 socket

`ServerComm.ServerComm.socket`

Definition at line 23 of file ServerComm.py.

The documentation for this class was generated from the following file:

- src/communication/ServerComm.py

Chapter 7

File Documentation

7.1 src/agent/Agent.py File Reference

Implementação de Lógica de Agente de Campo.

Classes

- class Agent.Agent

  Classe que representará os agentes de campo, possuindo métodos correspondentes.

Namespaces

- namespace Agent

7.1.1 Detailed Description

Implementação de Lógica de Agente de Campo.

Definition in file Agent.py.

7.2 Agent.py

Go to the documentation of this file.

```
00001 """
00002 @file Agent.py
00003 @brief Implementação de Lógica de Agente de Campo
00004 """
00005 from agent.BaseAgent import BaseAgent
00006
00007 class Agent(BaseAgent):
00008     """
00009     @brief Classe que representará os agentes de campo, possuindo métodos correspondentes.
00010     """
00011     def __init__(self, creation_options: list[list[str]]):
00012         """
00013         @brief Construtor da classe agente de campo, inicializando informações gerais.
00014         @param creation_options Lista de Parâmetros de Criação de Agente
00015         @details
00016         Parâmetros presentes em `creation_options`:
00017             -
00018             -
00019         """
00020
00021         super().__init__(creation_options)
```

7.3 src/agent/AgentPenalty.py File Reference

Implementação de Lógica de Goleiro.

Namespaces

- namespace AgentPenalty

7.3.1 Detailed Description

Implementação de Lógica de Goleiro.

Definition in file AgentPenalty.py.

7.4 AgentPenalty.py

Go to the documentation of this file.

```
00001 """
00002 @file AgentPenalty.py
00003 @brief Implementação de Lógica de Goleiro
00004 """
```

7.5 src/agent/BaseAgent.py File Reference

Implementação da classe de jogador base, que deve ser comum a todos os agentes.

Classes

- class BaseAgent.BaseAgent

  Classe que agrupará todas as funcionalidades comuns a qualquer agente.

Namespaces

- namespace BaseAgent

7.5.1 Detailed Description

Implementação da classe de jogador base, que deve ser comum a todos os agentes.

Definition in file BaseAgent.py.

7.6 BaseAgent.py

Go to the documentation of this file.

```
00001 """
00002 @file BaseAgent.py
00003 @brief Implementação da classe de jogador base, que deve ser comum a todos os agentes.
00004 """
00005 from abc import ABC, abstractmethod # para conseguirmos criar classes abstratas em Python
00006 from communication.ServerComm import ServerComm
00007
00008 class BaseAgent(ABC):
00009     """
00010     @brief Classe que agrupará todas as funcionalidades comuns a qualquer agente.
00011     """
00012
00013     def __init__(self, creation_options: list[list[str]]):
00014         """
00015         @brief Construtor da classe base de agente, chamando todos os construtores de outras
00016         classes mínimas para cada agente.
00017         @param creation_options Lista de Parâmetros de Criação de Agente
00018         """
00019
00020         self.scom = ServerComm(creation_options)
00021         # Chamaremos os construtores mínimos conforme formos criando-os
00022
00023
```

7.7 src/communication/ServerComm.py File Reference

Implementação da Comunicação com Servidor.

Classes

- class ServerComm.ServerComm

  Responsável pela comunicação com servidor.

Namespaces

- namespace ServerComm

7.7.1 Detailed Description

Implementação da Comunicação com Servidor.

Definition in file ServerComm.py.

7.8 ServerComm.py

Go to the documentation of this file.

```
00001 """
00002 @file ServerComm.py
00003 @brief Implementação da Comunicação com Servidor
00004 """
00005 import socket
00006 from time import sleep
00007 from term.Printing import Printing
00008 from select import select
00009
00010 class ServerComm:
00011     """
00012     @brief Responsável pela comunicação com servidor.
00013     """
00014
00015     def __init__(self, creation_options: list[list[str]]):
00016         """
00017         @brief Construtor da classe, inicializando buffers e a conexão de cada agente com servidor.
00018         """
00019
00020         # Características da comunicação
00021         self.BUFFER_SIZE = 8192
00022         self.BUFFER = bytearray(self.BUFFER_SIZE)
00023         self.socket = socket.socket(
00024                                     socket.AF_INET,
00025                                     socket.SOCK_STREAM # TCP
00026                                     )
00027
00028         # Características alheias
00029         self.num = creation_options[4][1]
00030
00031
00032         # Fazemos a conexão com servidor
00033         Printing.print_message(f"Tentando conexão do jogador {self.num}", "info")
00034         while True:
00035             try:
00036
00037                 self.socket.connect(
00038                                     (
00039                                      creation_options[0][1],# Host
00040                                      int(creation_options[1][1]) # Porta de Agentes
00041                                     )
00042                                     )
00043                 break
00044             except ConnectionRefusedError:
00045                 sleep(1)
00046                 Printing.print_message(".")
00047
00048         Printing.print_message("Agente Conectado!", "info")
00049
00050         robot_type = 1
00051         # Fazemos o pedido de criação de robô
00052         self.send_immediate(
00053             b"(scene rsg/agent/nao/nao_hetero.rsg " + str(robot_type).encode() + b')'
00054         )
00055         self.send_immediate(
00056             b"(init (unum " + str(self.num).encode() + b") (teamname " + "Roboime".encode() + b"))"
00057         )
00058
00059         sleep(15)
00060         self.socket.close()
00061
00062
00063     def send_immediate(self, message: bytes) -> None:
00064         """
00065         @brief Envia uma mensagem instantânea ao servidor, verificando se a conexão continua ativa
00066         @details
00067         Coloca-se na frente uma informação de tamanho da mensagem dentro de 4 bytes.
00068         """
00069
00070         try:
00071             self.socket.send(
00072                              len(message).to_bytes(4, byteorder="big") + message
00073                              )
00074         except BrokenPipeError:
00075             Printing.print_message("Error: Socket foi fechado por rcssserver3d", "error")
00076
00077     def receive(self) -> None:
00078         """
00079         @brief Receberá informações diretamente do servidor.
00080         """
00081
00082         while True:
```

```
00083             try:
00084                 # Verificamos se há 4 bytes no cabeçalho e nos preparamos para ler.
00085                 if self.socket.recv_into(
00086                     self.BUFFER, nbytes=4
00087                 ) != 4:
00088                     raise ConnectionResetError
00089 
00090                 # Lemos o comprimento total da mensagem
00091                 msg_size = int.from_bytes(
00092                     self.BUFFER[:4], # Garantimos leitura de apenas 4 bytes
00093                     byteorder="big", # ordem de significativo
00094                     signed=False # se tem sinal
00095                 )
00096 
00097                 # Lemos o restante da mensagem
00098                 if(
00099                     self.socket.recv_into(
00100                         self.BUFFER,
00101                         nbytes=msg_size
00102                     )
00103                 ) != msg_size:
00104                     raise ConnectionResetError
00105 
00106             except ConnectionResetError:
00107                 Printing.print_message("\nError: socket foi fechado pelo rcssserver3d.")
00108 
00109             if len(
00110                 select( # Monitora sockets/arquivos para I/O
00111                     [self.socket], # Lista de sockets/arquivos para verificar leitura
00112                     [], # Lista vazia para escrita
00113                     [], # Lista vazia para exceções
00114                     0.0 # timeout zero (não bloqueante)
00115                 )[0] # Pegamos o primeiro socket para leitura
00116             ) == 0: # Logo, não há dados disponíveis para leitura
00117                 break
00118 
00119         # Vejamos o recebido do servidor
00120         print(self.BUFFER)
00121 
00122 
00123 
00124 
00125 
00126 
00127 
00128 
00129 
00130 
00131 
00132 
00133 
00134 
```

7.9 src/run_player.py File Reference

Namespaces

- namespace run_player

Variables

- run_player.boot = Booting()
- str run_player.if_debug_mode = '1'
- run_player.player = Agent(boot.options)

7.10 run_player.py

Go to the documentation of this file.

```
00001 from term.Booting import Booting
00002 from agent.Agent import Agent
00003 
00004 boot = Booting()
00005 if_debug_mode = boot.options[-1][1] == '1'
00006 
00007 player = Agent(boot.options)
```

7.11 src/term/Booting.py File Reference

Implementação do Booting do time.

Classes

- class Booting.Booting

  Responsável por inicializar todas as necessidades de execução do time.

Namespaces

- namespace Booting

7.11.1 Detailed Description

Implementação do Booting do time.

Definition in file Booting.py.

7.12 Booting.py

Go to the documentation of this file.

```
00001 """
00002 @file Booting.py
00003 @brief Implementação do Booting do time
00004 """
00005 import os
00006 import sys
00007 from term.Printing import Printing
00008
00009 class Booting:
00010     """
00011     @brief Responsável por inicializar todas as necessidades de execução do time
00012     @details
00013     Assume as seguintes responsabilidades:
00014         - Estabelece um arquivo de configurações default caso já não exista um.
00015     """
00016
00017     def __init__(self):
00018         """
00019         @brief Responsável por chamar as inicializações mínimas.
00020         """
00021
00022         self.options = Booting.get_team_params()
00023
00024
00025         if getattr(sys, 'frozen', False):
00026             # Então estamos executando o binário!
00027             # Devemos forçar que o debug seja 0.
00028             self.options[8][1] = '0'
00029             Printing.IF_IN_DEBUG = False
00030
00031     @staticmethod
00032     def get_team_params() -> list[list[str]]:
00033         """
00034         @brief Verifica existência de arquivo de parâmetros de time, caso não exista, usará o default.
00035         @details
00036         Faremos em tupla para permitir uso mínimo de memória.
00037         @return
00038         """
00039
00040         if os.path.exists("config_team_params.txt"):
00041             with open(
00042                     "config_team_params.txt",
00043                     "r"
```

```
00044             ) as file_team_params:
00045                 return [
00046                     string_tupla.split(",") for string_tupla in
      file_team_params.read().split("\n")[:-1]
00047                 ]
00048
00049         config_team_params = [
00050             ["IP Server",       "localhost"],
00051             ["Agent Port",      "3100"], # Onde nos conectaremos com rcssserver3d
00052             ["Monitor Port",    "3200"], # Onde nos conectaremos com Roboviz
00053             ["Team Name",       "RoboIME"],
00054             ["Uniform Number",  '1'],
00055             ["Robot Type",      '1'],
00056             ["Penalty Shootout",'0'],
00057             ["MagmaFatProxy",   '0'],
00058             ["Debug Mode",      '1']
00059         ]
00060
00061         # E criamos o arquivo
00062         with open(
00063             "config_team_params.txt",
00064             "w+"
00065         ) as file_team_params:
00066             for doc, value in config_team_params:
00067                 file_team_params.write(
00068                     f"{doc},{value}\n"
00069                 )
00070
00071         return config_team_params
00072
00073     @staticmethod
00074     def cpp_builder(self):
00075         """
00076         @brief Responsável por buildar os arquivos .cpp presentes na pasta cpp.
00077         @return Funcionalidades C++ em condições de interoperabilidade.
00078         """
00079         # Voltaremos para esta assim que tivermos desenvolvido pelo menos uma pasta cpp
00080         pass
00081
00082
```

7.13 src/term/Printing.py File Reference

Implementação de Interface no terminal.

Classes

- class Printing.Printing

  Responsável pela comunicação usuário - terminal.

Namespaces

- namespace Printing

7.13.1 Detailed Description

Implementação de Interface no terminal.

Definition in file Printing.py.

7.14 Printing.py

Go to the documentation of this file.

```
00001 """
00002 @file Printing.py
00003 @brief Implementação de Interface no terminal
00004 """
00005
00006 class Printing:
00007     """
00008     @brief Responsável pela comunicação usuário - terminal
00009     """
00010     IF_IN_DEBUG = False
00011     TABLE_COLORS = {
00012         "info": "\033[1;36m",
00013         "warning": "\033[1;33m",
00014         "error": "\033[1;31m"
00015     }
00016
00017     @staticmethod
00018     def print_message(message: str, role: str=None) -> None:
00019         """
00020         @brief Apresentará uma mensagem estilizada de forma específica
00021         @param message Mensagem a ser apresentada
00022         @param role String indicando qual o motivo da mensagem
00023         @details
00024         Há uma quantidade específica de roles possíveis:
00025             - info
00026             - warning
00027             - error
00028         Caso nenhuma dessas seja inserida, há a possibilidade de inserir
00029         o comando ASCII de uma vez.
00030         """
00031
00032         if not Printing.IF_IN_DEBUG:
00033             return
00034
00035         if role is None:
00036             print(message, end="", flush=True)
00037             return
00038
00039         if role in Printing.TABLE_COLORS:
00040             print(f"{Printing.TABLE_COLORS[role]}", end="", flush=True)
00041         else:
00042             if role.startswith("\\033["):
00043                 print(f"{role}", end="", flush=True)
00044             else:
00045                 Printing.print_message("Erro: `role` não específicada.", "error")
00046                 return
00047
00048         print(message, end="", flush=True)
00049         print("\033[0m", flush=True)
00050
00051
00052
00053
00054
00055
00056
00057
00058
00059
00060
00061
00062
```

Index