Documento de especificação do instalador de projetos github Tevec

Sumário

[Introdução 1](#_Toc427831105)

Objetivo [1](#_Toc427831105)

Funcionalidades [1](#_Toc427831105)

Estrutura [1](#_Toc427831106)

1[. Classes:](#_Toc427831108) 1

2[. Métodos:](#_Toc427831108) 1

[Regras](#_Toc427831109)6

[Acesso](#_Toc427831110)6

# Introdução

Este documento tem como objetivo detalhar a estrutura de código e utilização do programa **instalador de projetos github**.

O **instalador de projetos github** é utilizado para clonar um projeto alocado no github e instalar o mesmo dentro do ambiente python. Onde este pode ser chamado como um modulo em qualquer programa python.

A tarefa de trazer o projeto para dentro do ambiente python é muito interessante para por exemplo: realizar testes facilmente fora do ambiente de desenvolvimento.

É possivel tanto instalar quanto desinstalar um projeto do github no ambiente python, apenas setando o nome do projeto alocado no github. O programa também suporta instalar a versão de um release especifico de um projeto.

# Objetivo

O objetivo do **instalador de projetos github** é disponibilizar um projeto alocado no github dentro do ambiente python.

# Funcionalidades

1 – Instalação de projetos no ambiente python: O programa possui a feature de instalar um projeto que está alocado no servidor do github dentro do ambiente python, para isto basta apenas setar o nome do projeto no github.

2 – Desinstalação de projetos do ambiente python: O programa possui a feature de desinstalar um projeto que foi instalado anteriormente dentro do ambiente python, para isto basta apenas setar o nome do projeto no github.

3 – Atualização do python: O programa também verifica se a versão do python e igual a versão do python utilizada pelo strike. Caso a versão seja diferente o programa tem a funcionalidade de atualizar a versão do python, deixando igual a versão utilizada pelo strike.

4 – Validações: O programa possuí também uma serie de validações para verificar: Se o ambiente python está ativo (o ambiente python precisa estar ativo para a instalação do projeto dentro do mesmo), versão do python, quantidade de parametros inseridos na execução do programa.

5 – Listar os ambientes python: Com apenas um simples comando o programa consegue listar todos os ambiente python locais. Esta feature é interessante para consultar/lembrar os ambiente python.

6 – Instalar a versão de um release especifico: O programa também pode instalar uma versão especifica de um projeto alocado no github dado o nome da tag (versão) no github.

7 – Ajuda: O programa possui a funcionalidade de exibir uma ajuda com um pequeno manual de utilização do programa.

# Estrutura

Para a instalação de projetos do github serão utilizados um conjunto de métodos.

## Classes:

* 1. Instalador:

Classe reponsável pelo comportamento de todo o programa.

## Métodos:

* 1. \_\_init\_\_():

Esse é um método construtor, nele é executada toda a lógica do programa.

O método não possui parâmetros.

|  |  |
| --- | --- |
| **Detalhe** | Neste momento o metodo verifica se há algum parametro com o valor '--atualizar-python', caso haja é chamado o metodo 'atualizarVersaoPython()' para atualizar a versão do python. |
| **Código** | # Atualizar versao do python  for indice, valor\_parametro in enumerate(lista\_parametros):  if valor\_parametro == '--atualizar-python':  concluido = Instalador.atualizarVersaoPython(self, lista\_parametros[indice + 1])  exit() |
| **Detalhe** | Verifica se há algum parametro com o valor '-h', se houver o programa chama o método 'exibirAjuda()' que imprime um texto para ajuda de utilização do programa. |
| **Código** | # Exibir ajuda  for indice, valor\_parametro in enumerate(lista\_parametros):  if valor\_parametro == '-h':  print(Instalador.exibirAjuda(self))  exit() |
| **Detalhe** | Verifica se há algum parametro com o valor '-l', se houver o programa chama o método 'listarAmbientes()' que imprime uma lista de todos os ambientes python locais. |
| **Código** | # Exibir lista de ambientes  for indice, valor\_parametro in enumerate(lista\_parametros):  if valor\_parametro == '-l':  concluido = Instalador.listarAmbientes(self)  exit() |
| **Detalhe** | Verifica se há algum parametro com o valor '-v', se houver o programa atribui o próximo valor da lista de parametros para uma variavel global chamada 'versao\_release'. |
| **Código** | # Setando a versão do release  for indice, valor\_parametro in enumerate(lista\_parametros):  if valor\_parametro == '-v':  # Verifica se o indice existe na lista  if (indice + 1) in lista\_parametros:  self.versao\_release = (lista\_parametros[indice + 1]) |
| **Detalhe** | Verifica se há algum parametro com o valor '-i' ou 'instalar', se houver o programa verifica se existe um próximo valor na lista de parametros (este próximo valor deve ser o nome do projeto no github), caso haja, o próximo valor é atribuido a uma veriavel chamada 'nome\_cliente'. Em seguida é chamado o método 'instalarPacote()' que tem a função de instalar o projeto do github no ambiente python. |
| **Código** | # Instalando o projeto do cliente e suas dependencias no ambiente python  for indice, valor\_parametro in enumerate(lista\_parametros):  if valor\_parametro == '-i' or valor\_parametro == 'instalar':  if len(lista\_parametros) > 2 and len(lista\_parametros) < 6:  self.nome\_cliente = lista\_parametros[indice + 1]  # Instalando o projeto cliente e suas dependencias  concluido = Instalador.instalarPacote(self)  else:  print(self.lista\_mensagens[8] + self.lista\_mensagens[9])  exit() |
| **Detalhe** | Verifica se há algum parametro com o valor '-r' ou 'remover', se houver o programa verifica se existe um próximo valor na lista de parametros (este próximo valor deve ser o nome do projeto no github), caso haja, o próximo valor é atribuido a uma veriavel chamada 'nome\_cliente'. Em seguida é chamado o método 'desinstalarPacote()' que tem a função de desinstalar o projeto do github do ambiente python. |
| **Código** | # Desinstalando o projeto do cliente e suas dependencias do ambiente python  for indice, valor\_parametro in enumerate(lista\_parametros):  if valor\_parametro == '-r' or valor\_parametro == 'remover':  if (indice + 1) in lista\_parametros:  self.nome\_cliente = lista\_parametros[indice + 1]  # Desinstalando o projeto cliente e suas dependencias  concluido = Instalador.desinstalarPacote(self)  else:  print(self.lista\_mensagens[8] + self.lista\_mensagens[9])  exit() |

* 1. instalarPacote()

Esse método tem como objetivo instalar um projeto do github no ambiente python dado o nome do projeto.

O método não possui parâmetros.

|  |  |
| --- | --- |
| **Detalhe** | O programa faz uma pergunta ao usuário, perguntando se ele deseja realmente instalar o projeto, se o usuario digitar 'y' e der enter o programa entende que ele deseja instalar o projeto no ambiente python. |
| **Código** | if input(self.lista\_mensagens[2]) == 'y': |
| **Detalhe** | Neste momento o programa pega o caminho do ambiente python atravez do atributo 'prefix' da biblioteca 'sys'. Após é concatenado com o caminho até a paste '/site-packages/' onde são instalador os pacotes no ambiente python. |
| **Código** | # Pegando o caminho do ambiente python  print(self.lista\_mensagens[5])  caminho\_ambiente = str(sys.prefix) + '/lib/python' + self.versao\_python + '/site-packages/' |
| **Detalhe** | O programa verifica se a variavel global 'versao\_release' está preenchida. Se sim ela é concatenada com o valor atribuido a variavel 'comando\_pip', caso contrario ela não é concatenada.  A variavel 'comando\_pip' recebe o comando utilizado pelo 'pip (instalador de pacotes python)' para realizar a instalação do projeto no ambiente python. |
| **Código** | # Gerando o comando para o 'pip install'  print(self.lista\_mensagens[11])  if self.versao\_release == None:  comando\_pip = 'pip install -e git+http://github.com/TEVEC/' + str(self.nome\_cliente) + '#egg=' + str(self.nome\_cliente)  else:  comando\_pip = 'pip install -e git+https://github.com/TEVEC/' + str(self.nome\_cliente) + '@' + str(self.versao\_release) + '#egg=' + str(self.nome\_cliente) |
| **Detalhe** | O programa abra uma sessão no terminal (ou cmd, no caso do windows) e vai até o diretório armazenado na variavel 'caminho\_ambiente', após acessar o diretório executa o comando armazenado na variavel 'comando\_pip'. Se um dos comandos falharem é exibida uma mensagem de erro e o programa é encerrado. |
| **Código** | # Utilizando os comandos  if os.system('cd ' + caminho\_ambiente + ' && ' + comando\_pip):  exit() |

* 1. desinstalarPacote()

Esse método tem como objetivo desintalar projetos do ambiente python;

O método não possui parâmetros.

|  |  |
| --- | --- |
| **Detalhe** | O programa faz uma pergunta ao usuário, perguntando se ele deseja realmente instalar o projeto, se o usuario digitar 'y' e der enter o programa entende que ele deseja instalar o projeto no ambiente python. |
| **Código** | if input(self.lista\_mensagens[3]) == 'y': |
| **Detalhe** | O programa pega o caminho do ambiente python atravez do atributo 'prefix' da biblioteca 'sys'. Após é concatenado com o caminho até a paste '/site-packages/' onde são instalador os pacotes no ambiente python. |
| **Código** | # Pegando o caminho do ambiente python  print(self.lista\_mensagens[5])  caminho\_ambiente = str(sys.prefix) + '/lib/python' + self.versao\_python + '/site-packages/' |
| **Detalhe** | Passando o caminho armazenado na variavel 'caminho\_ambiente' como parâmetro do metodo 'remove()' da biblioteca 'os', onde este realiza a exclusão do diretório do projeto. Desinstalando o projeto do ambiente python. |
| **Código** | # Pegando o caminho do ambiente python  print(self.lista\_mensagens[12])  os.remove(caminho\_ambiente + 'src/' + str(self.nome\_programa)) |

* 1. listarAmbientes()

Esse método tem como objetivo listar todos os ambintes python existentes localmente.

O método não possui parâmetros.

|  |  |
| --- | --- |
| **Detalhe** | O programa abre uma sessão do terminal (ou cmd, no caso do windows) e executa o comando 'conda env list', utilizado para listar os ambientes python. |
| **Código** | # Listando ambientes python  if os.system('conda env list'):  exit() |

* 1. atualizarVersaoPython():

Esse método tem como objetivo atualizar o versão do python quando o usuário solicitar.

O método receberá como parâmetro:

1. versao – Utilizado para setar a versão desejada do python.

|  |  |
| --- | --- |
| **Detalhe** | O programa abre uma sessão do terminal (ou cmd, no caso do windows) e executa o comando 'conda install python=' + str(versao))', onde é setado o parâmetro 'versao' para determinar a versão do python desejada. O python é instalado ou atualizado com este comando. |
| **Código** | # Atualizando versão do python  if os.system('conda install python=' + str(versao)):  exit() |

* 1. validacoes()

Esse método tem como objetivo agrupar e executar validações para o programa.

O método receberá como parâmetro:

1. lista\_parametros – Lista de parâmetros recebidos no console.

|  |  |
| --- | --- |
| **Detalhe** | Valida se a variavel 'versao\_python' que contém a versão do python instalada é diferente da varivavel 'versao\_python\_strike' que guarda a versão do python utilizada pelo strike. Se for diferente é exibida uma mensagem de erro e o programa é encerrado. |
| **Código** | # Validando versão do python  if self.versao\_python != self.versao\_python\_strike:  print(self.lista\_mensagens[4])  exit() |
| **Detalhe** | Valida se o tamanho da lista 'lista\_parametros' (que contém uma lista dos parâmetros setados no console) é menor do que 2. Se verdadeiro é exibida uma mensagem de erro e o programa é encerrado. |
| **Código** | # Validando se falta argumentos  if len(lista\_parametros) < 2:  print(self.lista\_mensagens[7] + self.lista\_mensagens[9])  exit() |
| **Detalhe** | Valida se o tamanho da lista 'lista\_parametros' (que contém uma lista dos parâmetros setados no console) é maior do que 5. Se verdadeiro é exibida uma mensagem de erro e o programa é encerrado. |
| **Código** | # Validando se há excesso de argumentos  if len(lista\_parametros) > 5:  print(self.lista\_mensagens[10] + self.lista\_mensagens[9])  exit() |

* 1. exibirAjuda()

Esse método tem como objetivo de exibir uma ajuda ao usuário, quando o mesmo solicitar.

O método não possui parâmetros.

|  |  |
| --- | --- |
| **Detalhe** | Retorna uma string contendo o texto de ajuda. |
| **Código** | return texto\_ajuda |

* 1. carregarListaMensagens()

Esse método tem como objetivo carregar uma lista com as mensagens utilizadas pelo programa.

O método não possui parâmetros.

|  |  |
| --- | --- |
| **Detalhe** | Carrega uma lista com as mensagens utilizadas pelo programa. |
| **Código** | self.lista\_mensagens = ['\nInstalação concluída.', ... |

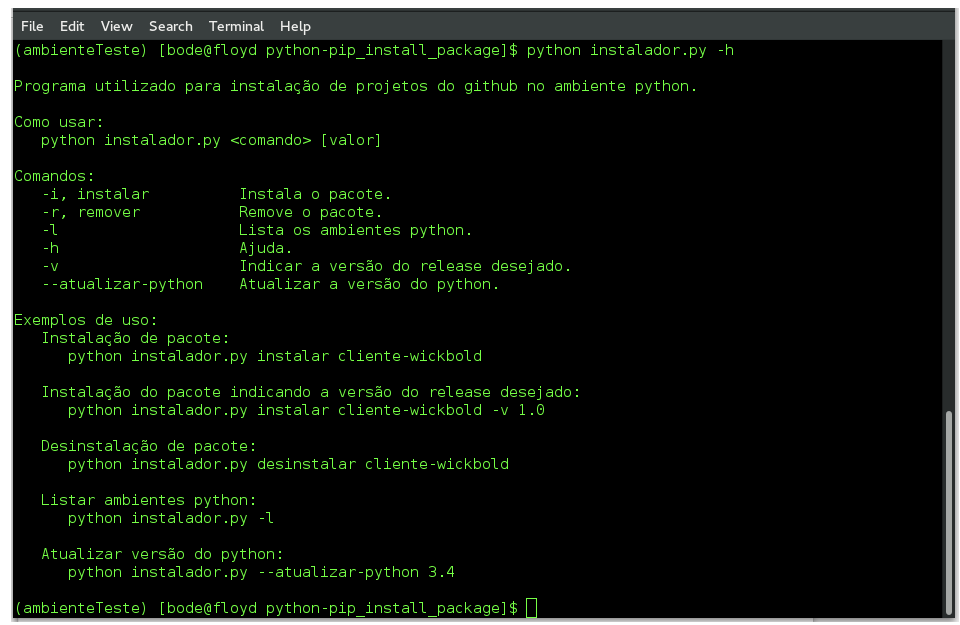
# Regras

1 - O ambiente python deve estar ativo antes de instalar ou desinstalar um projeto utilizando o programa.

2 - Deverá ser passado no minimo um e no máximo quatro parâmetros quando executar o programa.

# Utilização

Deverá ter em maquina o programa **instalador.py** e rodar o mesmo com o comando 'python **instalador.py** <comando> [valor]'

Imagem 1 – Exemplo de utilização do programa