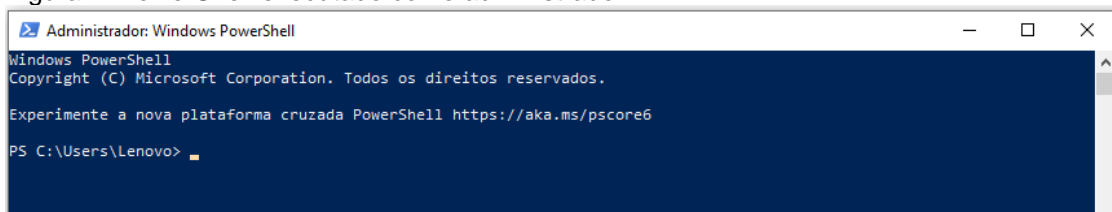


Miscellaneous

Rotinas de Criação, ativação e execução Scripts Python em ambiente virtual

- Checar permissão para rodar scripts
 - Abrir o powershell como administrador como Figura 1

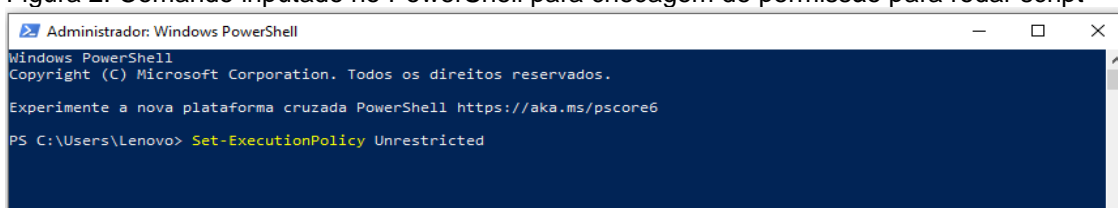
Figura 1: PowerShell executado como administrador



Fonte: Os autores

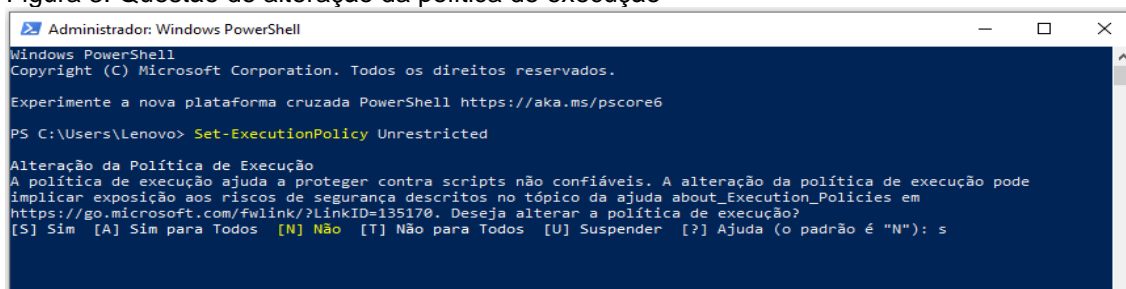
- Executar o comando `Set-ExecutionPolicy Unrestricted` e apertar ENTER como ilustrado na Figura 2
- Na pergunta ilustrada na Figura 3 digitar s e apertar o ENTER
- Deverá retornar sem erros como na Figura 4

Figura 2: Comando inputado no PowerShell para checagem de permissão para rodar script



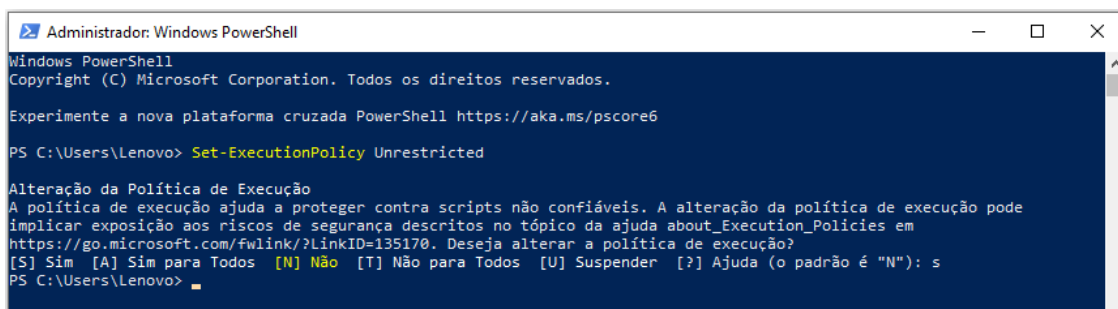
Fonte: os autores

Figura 3: Questão de alteração da política de execução



Fonte: Os autores

Figura 4: Confirmação da alteração da política de execução



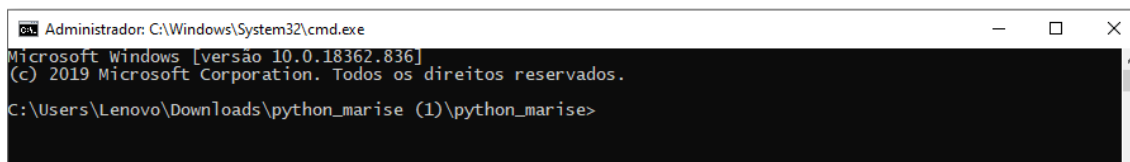
Fonte: Os autores

Miscellaneous

Rotinas de Criação, ativação e execução Scripts Python em ambiente virtual

- Ativação ou criação de ambiente virtual
 - Acesse o caminho da pasta que contém a API via terminal como ilustrado na Figura 5
 - Para ativar um ambiente virtual escreva o comando: **.\"nome_do_ambiente\"Scripts\activate.bat** como ilustrado na Figura 6

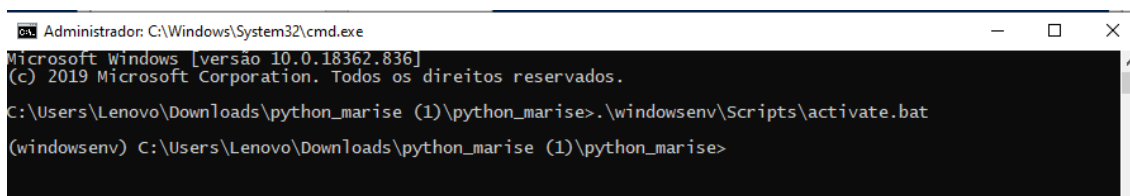
Figura 5: Acesso ao caminho da pasta no CMD



```
Administrador: C:\Windows\System32\cmd.exe
Microsoft Windows [versão 10.0.18362.836]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Lenovo\Downloads\python_marise (1)\python_marise>
```

Fonte: Os autores

Figura 6: Ativação do ambiente virtual nomeado "windowsenv"

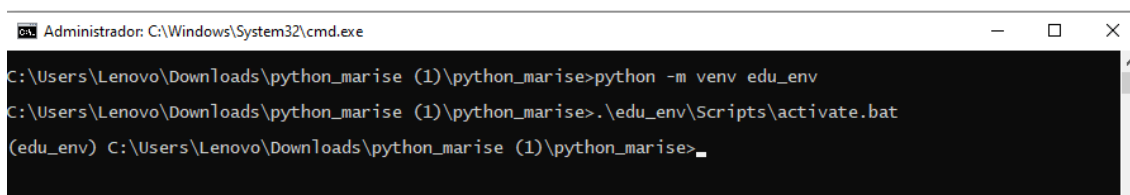


```
Administrador: C:\Windows\System32\cmd.exe
Microsoft Windows [versão 10.0.18362.836]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Lenovo\Downloads\python_marise (1)\python_marise>.\windowsenv\Scripts\activate.bat
(windowsenv) C:\Users\Lenovo\Downloads\python_marise (1)\python_marise>
```

Fonte: Os autores

- Para a criação de um novo ambiente virtual digite o comando **python -m venv "nome_do_ambiente"** e ative-o como ilustrado na Figura 7

Figura 7: Criação de um novo ambiente virtual e ativação

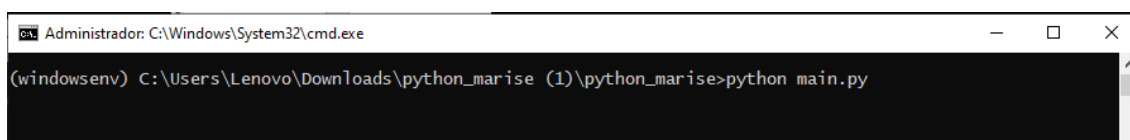


```
Administrador: C:\Windows\System32\cmd.exe
C:\Users\Lenovo\Downloads\python_marise (1)\python_marise>python -m venv edu_env
C:\Users\Lenovo\Downloads\python_marise (1)\python_marise>.\edu_env\Scripts\activate.bat
(edu_env) C:\Users\Lenovo\Downloads\python_marise (1)\python_marise>
```

Fonte: Os autores

- Ao acessar o ambiente virtual você pode inicializar a API com o comando **python main.py** como ilustrado na Figura 8

Figura 8: Inicialização da API



```
Administrador: C:\Windows\System32\cmd.exe
(windowsenv) C:\Users\Lenovo\Downloads\python_marise (1)\python_marise>python main.py
```

Fonte: Os autores

Miscellaneous

Rotinas de Criação, ativação e execução Scripts Python em ambiente virtual

- Para desativar o ambiente virtual utilize o comando:
.\"nome_do_ambiente"\Scripts\deactivate.bat como ilustrado na Figura 9

Figura 9: Desativação do ambiente virtual

```
Administrador: C:\Windows\System32\cmd.exe
(windowsenv) C:\Users\Lenovo\Downloads\python_marise (1)\python_marise>.\windowsenv\Scripts\deactivate.bat
C:\Users\Lenovo\Downloads\python_marise (1)\python_marise>
```

Fonte: Os autores

- Verificação das versões e instalação das bibliotecas necessárias
 - Verifique a versão do python instalado na sua máquina com o comando **python --version** ou **python -VV** ou **python -V** como ilustrado na Figura 10

Figura 10: Verificação da versão do python instalada

```
Administrador: Prompt de Comando
Microsoft Windows [versão 10.0.18362.836]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Lenovo>python --version
Python 3.8.5
C:\Users\Lenovo>
```

Fonte: Os autores

- Verificar as bibliotecas que estão instaladas no seu sistema com **pip list** como na Figura 11

Figura 11: verificação das bibliotecas instaladas no sistema

```
Administrador: Prompt de Comando
C:\Users\Lenovo>pip list
Package Version
-----
bernhard 0.2.6
bottle 0.12.18
cassandra-driver 3.24.0
certifi 2020.6.20
chardet 3.0.4
click 7.1.2
CouchDB 1.2
CPU-temperature-monitor 0.2
docker 4.3.1
elasticsearch 7.9.1
future 0.18.2
geom 0.2.1.post1
Glances 3.1.5
idna 2.10
ifaddr 0.1.7
influxdb 5.3.0
kafka-python 2.0.1
msgpack 0.6.1
mysql-connector 2.2.9
netifaces 0.10.9
paho-mqtt 1.5.0
pbkdf2 1.3
pika 1.1.0
pip 20.1.1
ply 3.11
potsdb 1.0.3
prometheus-client 0.8.0
protobuf 3.13.0
```

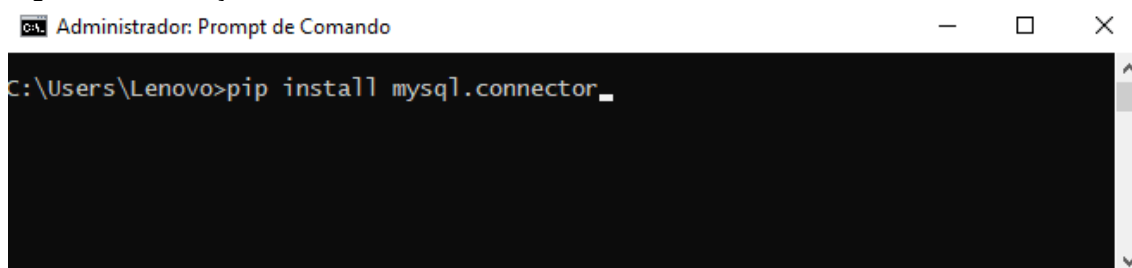
Fonte: Os autores

Miscellaneous

Rotinas de Criação, ativação e execução Scripts Python em ambiente virtual

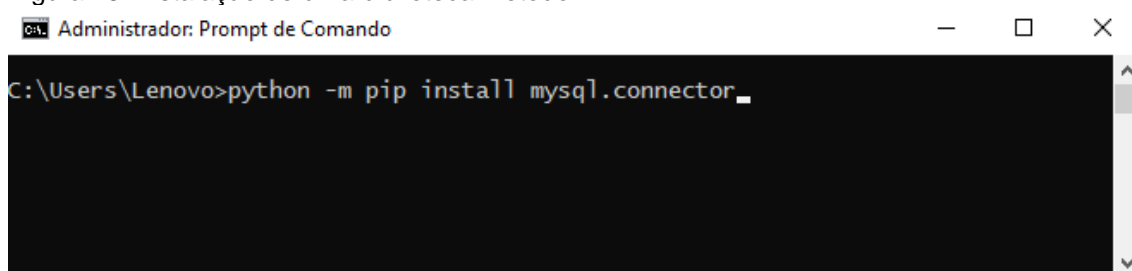
- Caso algum módulo não esteja instalado pode ser feito com o comando **pip install “nome_do_modulo”** ou **python -m pip install “nome_do_modulo”** (instalação forçada manual utilizando o python) como nas Figuras 12 e 13

Figura 12: Instalação de uma biblioteca método 1



Fonte: Os autores

Figura 13: Instalação de uma biblioteca método 2



Fonte: Os autores

- Pontos importantes relacionados ao banco de dados no script
 - Observe que a API utiliza de import de algumas bibliotecas como na Figura 14, caso não tenha instalado irá retornar erro, necessitando de instalação

Figura 14: Exemplo de bibliotecas utilizadas na API

```
from services.mysql import Mysql
from services.dataGenerator import getData
import time
```

Fonte: Os autores

- Atente à questão do usuário, esta API está escrita direto para o root Figura 15. Quais são as boas práticas? Devemos utilizar o root? Deixar aberto na API a senha para o nosso banco de dados?

Miscellaneous

Rotinas de Criação, ativação e execução Scripts Python em ambiente virtual

Figura 15: Parâmetros para conexão ao banco de dados

```
#Inserir user, password, host, database
mysql = Mysql('root', 'urubu100', 'localhost', 'projeto')
```

Fonte: Os autores

- Consistência dos parâmetros é importante, no comando **INSERT INTO `dataset-comp`(cpu, ram, ram_percent, disk) VALUES (%s, %s, %s, %s)** o nome utilizado no script corresponde ao nome da tabela que você criou no seu banco de dados? As colunas criadas no seu banco de dados possuem os mesmos nomes?

Figura 16

Figura 16: Parâmetros de inserção dos dados na tabela do banco de dados

```
24
25     def insert(self, data):
26         query = (
27             "INSERT INTO `dataset-comp`(cpu, ram, ram_percent, disk)"
28             "VALUES (%s, %s, %s, %s)"
29         )
30         values = data
31         try:
32             print('Inserindo Valores')
33             self.cursor.execute(query, values)
34             self.mysql.commit()
35         except Exception as err:
36             print(err)
37             self.mysql.rollback()
38             self.close()
```

Fonte: Os autores

- Erros
 - Ao tentar executar qualquer comando e o prompt retornar erro **PARE!** Analise o que você está lendo, guarde como recordação esse erro e ao pesquisá-lo para correção guarde na sua memória como conhecimento (em incidentes é nomeado de *knowledge base*) para engrandecimento da sua capacidade de resolução de problemas
- Utilização de bibliotecas prontas para sua aplicação
 - Ao utilizar soluções prontas disponibilizadas na internet se certifique de que são compatíveis com seu sistema e mais importante, se certifique do que você está recebendo para garantir acuracidade do seu programa. A Figura 17 mostra o código aberto da **psutil.sensors_temperatures()** que apenas roda em Linux. Você consegue observar algo curioso nesse código?

Miscellaneous

Rotinas de Criação, ativação e execução

Scripts Python em ambiente virtual

Figura 17: Código da psutil.sensors_temperatures()

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Copyright (c) 2009, Giampaolo Rodola'. All rights reserved.
5  # Use of this source code is governed by a BSD-style license that can be
6  # found in the LICENSE file.
7
8  """
9  A clone of 'sensors' utility on Linux printing hardware temperatures.
10
11  $ python scripts/sensors.py
12  asus
13      asus                47.0 °C (high = None °C, critical = None °C)
14
15  acpitz
16      acpitz              47.0 °C (high = 103.0 °C, critical = 103.0 °C)
17
18  coretemp
19      Physical id 0       54.0 °C (high = 100.0 °C, critical = 100.0 °C)
20      Core 0              47.0 °C (high = 100.0 °C, critical = 100.0 °C)
21      Core 1              48.0 °C (high = 100.0 °C, critical = 100.0 °C)
22      Core 2              47.0 °C (high = 100.0 °C, critical = 100.0 °C)
23      Core 3              54.0 °C (high = 100.0 °C, critical = 100.0 °C)
24  """
25
26  from __future__ import print_function
27  import sys
28
29  import psutil
30
31
32  def main():
33      if not hasattr(psutil, "sensors_temperatures"):
34          sys.exit("platform not supported")
35      temps = psutil.sensors_temperatures()
36      if not temps:
37          sys.exit("can't read any temperature")
38      for name, entries in temps.items():
39          print(name)
40          for entry in entries:
41              print("    %-20s %s °C (high = %s °C, critical = %s °C)" % (
42                  entry.label or name, entry.current, entry.high,
43                  entry.critical))
44          print()
45
46
47  if __name__ == '__main__':
48      main()
```

Fonte: <https://github.com/giampaolo/psutil/blob/master/scripts/temperatures.py>

Miscellaneous

Rotinas de Criação, ativação e execução Scripts Python em ambiente virtual

Ambiente virtual em Python

Conforme descrito em <https://docs.python.org/pt-br/3/tutorial/venv.html>

“Aplicações em Python normalmente usam pacotes e módulos que não vêm como parte da instalação padrão. Aplicações às vezes necessitam uma versão específica de uma biblioteca, porque ela requer que algum problema em particular tenha sido corrigido ou foi escrito utilizando-se de uma versão obsoleta da interface da biblioteca.

Isso significa que talvez não seja possível que uma instalação Python preencha os requisitos de qualquer aplicação. Se uma aplicação A necessita a versão 1.0 de um módulo específico, mas a aplicação B necessita a versão 2.0, os requisitos entrarão em conflito, e instalar qualquer uma das duas versões 1.0 ou 2.0 fará com que uma das aplicações não consiga ser executada.

A solução para este problema é criar um ambiente virtual, uma árvore de diretórios que contém uma instalação Python para uma versão particular do Python, além de uma série de pacotes adicionais.

Diferentes aplicações podem então usar diferentes ambientes virtuais. Para resolver o exemplo anterior de requisitos conflitantes, a aplicação A deve ter seu próprio ambiente virtual com a versão 1.0 instalada enquanto a aplicação B vai possuir outro ambiente virtual com a versão 2.0. Se a aplicação B precisar fazer uma atualização para a versão 3.0, isso não afetará o ambiente da aplicação A.”

Criando o ambiente virtual:

```
python3 -m venv tutorial-env
```

python3	é a versão
-m	é a especificação do módulo
venv	é o módulo do ambiente virtual
tutorial-env	é o nome do ambiente virtual em nosso projeto é windowsenv

Isto irá criar um diretório, incluindo o os diretórios do interpretador python, suas bibliotecas e alguns pacotes. Neste ambiente poderão ser instalados os pacotes necessários à sua aplicação isolada.

É criado um diretório de localização comum para o ambiente virtual, é o `.venv` e ficará oculto em seu ambiente. Também previne conflitos com o `.env`, que são arquivos de definição de variáveis de ambiente que algumas ferramentas utilizam.

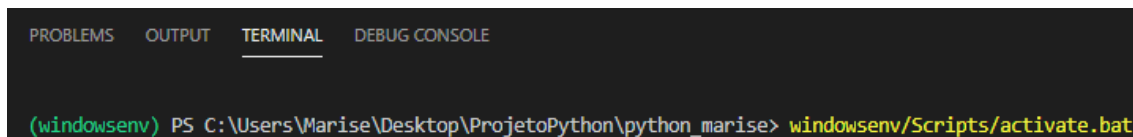
Miscellaneous

Rotinas de Criação, ativação e execução Scripts Python em ambiente virtual

Ativando o ambiente virtual:

Em Windows no cmd:

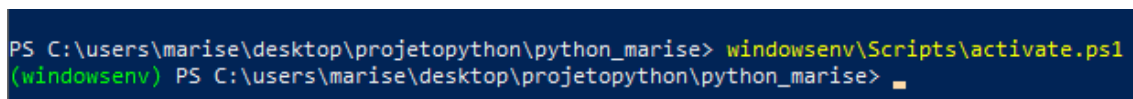
```
tutorial-env\Scripts\activate.bat
```



The screenshot shows a Windows command prompt window with the title bar 'PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE'. The terminal text is: (windowsenv) PS C:\Users\Marise\Desktop\ProjetoPython\python_marise> windowsenv/Scripts/activate.bat

Em Windows no powershell:

```
tutorial-env\Scripts\activate.ps1
```



The screenshot shows a Windows PowerShell window with the title bar 'PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE'. The terminal text is: PS C:\users\marise\desktop\projetopython\python_marise> windowsenv\Scripts\activate.ps1 (windowsenv) PS C:\users\marise\desktop\projetopython\python_marise> █

Em Linux, terminal bash:

```
source tutorial-env/bin/activate
```

ou

```
source xxxxxenv/Scripts/activate
```

Agora é só executar o script:

```
python .\main.py
```