```
!pip install pefile
```

```
Collecting pefile
  Downloading https://files.pythonhosted.org/packages/36/58/acf7f35859d541985f0a6ea3c34b
     |████████████████████████████████| 71kB 3.6MB/s
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages (from pe
Building wheels for collected packages: pefile
  Building wheel for pefile (setup.py) ... done
  Created wheel for pefile: filename=pefile-2019.4.18-cp36-none-any.whl size=60824 sha25
  Stored in directory: /root/.cache/pip/wheels/1c/a1/95/4f33011a0c013c872fe6f0f364dc463a
Successfully built pefile
Installing collected packages: pefile
Successfully installed pefile-2019.4.18
```

```
from google.colab import files
uploaded =files.upload()
```

Escolher arquivos 8 arquivos

- **cleanmgr.exe**(application/x-msdownload) - 299008 bytes, last modified: 07/12/2019 - 100% done
- **cmd.exe**(application/x-msdownload) - 289792 bytes, last modified: 15/01/2021 - 100% done
- **Defrag.exe**(application/x-msdownload) - 210432 bytes, last modified: 15/01/2021 - 100% done
- **ftp.exe**(application/x-msdownload) - 58880 bytes, last modified: 07/12/2019 - 100% done
- **LICLUA.EXE**(application/x-msdownload) - 485776 bytes, last modified: 01/02/2002 - 100% done
- **mediainfo.exe**(application/x-msdownload) - 1048576 bytes, last modified: 21/07/2009 - 100% done
- **notepad.exe**(application/x-msdownload) - 202240 bytes, last modified: 15/01/2021 - 100% done
- **OSE.EXE**(application/x-msdownload) - 214832 bytes, last modified: 01/02/2002 - 100% done

```
Saving cleanmgr.exe to cleanmgr.exe
Saving cmd.exe to cmd.exe
Saving Defrag.exe to Defrag.exe
Saving ftp.exe to ftp.exe
Saving LICLUA.EXE to LICLUA.EXE
Saving mediainfo.exe to mediainfo.exe
Saving notepad.exe to notepad.exe
Saving OSE.EXE to OSE.EXE
```

```
#1) Escreva um script em Python (ou .ipynb) que receba como entrada um arquivo ou
#diretório e enumere a seções executáveis do(s) binário(s), imprimindo na saída padrão
#um dicionário de listas, onde a chave é o nome do binário e o valor é uma lista de seções ex
```

```
import pefile

#! localizacao do arquivo
file_location = 'ftp.exe'
# abre arquivo
pe = pefile.PE(file_location)
dict = {}
listasecoes = []

chave = file_location
for section in pe.sections:
```

```
    if section.IMAGE_SCN_CNT_CODE == True or section.IMAGE_SCN_MEM_EXECUTE ==  True:
       #rotulo = 'nome do Arquivo Binario'
       listasecoes.append (section.Name.decode('utf-8'))

dict[chave] = listasecoes
print(dict)
```

```
{'ftp.exe': ['.text\x00\x00\x00']}
```

```
#2)Escreva outro script em Python (ou .ipynb) que receba como entrada dois arquivos .exe e os
#imprimindo na saída padrão quais seções são comuns a ambos os binários, quais somente estão
#no binário 1 e quais somente estão presentes no binário 2.
```

```
import pefile
#! localizacao do arquivo
file_location = 'mediainfo.exe'
# abre binário1
binario1 = pefile.PE(file_location)

file_location = 'notepad.exe'
# abre binário2
binario2 = pefile.PE(file_location)

#cria listas para armazenar as seções
secoes1 = []
secoes2 = []

#acrescenta na lista as seções do binário 1
for section in binario1.sections:
  secoes1.append(section.Name.decode('utf-8'))

#acrescenta na lista as seções do binário 2
for section in binario2.sections:
  secoes2.append(section.Name.decode('utf-8'))


#verifica seções comuns aos 2 binários
comum = [x for x in secoes2 if x in secoes1]

#verifica seções apenas do binário 1
so_secao1 = list(set(secoes1) - set(secoes2))

#verifica seções apenas do binário 2
so_secao2 = list(set(secoes2) - set(secoes1))

#imprime resultados
print("Seções comuns nos dois binarios %s" % comum)
```

```
print("Seções apenas do binario 1 %s" % so_secao1)
print("Seções apenas do binario 2 %s" % so_secao2)
```

```
Seções comuns nos dois binarios ['.rdata\x00\x00', '.rsrc\x00\x00\x00', '.reloc\x00\x00
Seções apenas do binario 1 ['.idata\x00\x00', 'BSS\x00\x00\x00\x00\x00', '.tls\x00\x00\x
Seções apenas do binario 2 ['.data\x00\x00\x00', '.didat\x00\x00', '.pdata\x00\x00', '.t
```