50 CÓDIGOS PRONTOS FEITOS EM PYTHON



BY DIETRICH

Sumário

Introdução 4

1. Preparação do ambiente 5

2. Instalação das dependências 5

3.Instalação para cada ambiente 5

4.Enumeração dos 100 scripts feitos em Python 6

4.1 Obter arquivo em outro servidor usando Python 6

4.2 Renomear arquivos baseado em um padrão específico 7

4.3 Mover arquivos para diferentes diretórios baseado nas suas extensões 8

4.4 Excluir arquivos que são mais antigos do que uma data específica 9

4.5 Copiar arquivos de um local para outro 10

4.6 Comprimir arquivos usando formatos (zip) 11

4.7 Comprimir arquivos usando formatos (gz) 12

4.8 Percorrer uma árvore de diretórios usando Python 13

4.9 Extrair arquivos de arquivo zip 14

4.10 Criar um arquivo com um conteúdo específico. 15

4.11 Ler o conteúdo de um arquivo e armazená-lo em uma variável 16

4.12 Contar o número de linhas em um arquivo 17

4.13 Procurar uma string específica em um arquivo 18

4.14 Substituir uma string específica em um arquivo 19

4.15 Mesclar vários arquivos em um só 20

4.16 Dividir um arquivo grande em mais pequenos 21

4.17 Criptografar e descriptografar arquivos 22

4.18 Comparar o conteúdo de dois arquivos 23

4.19 Criar um arquivo com conteúdo aleatório 24

4.20 Renomear arquivos em massa 25

4.21 Deletar linha especifica de um arquivo de texto 26

4.22 Gerar um arquivo com um formato e estrutura específicos, como CSV 27

4.23 Gerar um arquivo com um formato e estrutura específicos, como JSON 28

4.24 Monitorar um diretório para novos arquivos e executar ações neles 29

4.25 Remover as linhas contendo palavra especifica 30

4.26 Procurar recursivamente por arquivo especifico em um diretório e seus subdiretórios 31

4.27 Remover caracteres não imprimíveis de um texto 32

4.28 Automatizando a organização de arquivos movendo-os para pastas apropriadas com base em seu tipo ou data de criação 33

4.29 Criptografar um arquivo fornecendo uma senha 34

4.30 Descriptografar um arquivo fornecendo uma senha 35

4.31 Verificando a integridade de arquivos usando checksums 36
4.32 Escrever intervalo de linhas de um arquivo de texto 37
4.33 Converter o primeiro caractere da palavra para letra maiúscula Após um ponto "." 38
4.34 Criando um script para baixar automaticamente arquivos de um site 39
4.35 Comparando o conteúdo de múltiplos arquivos e identificando diferenças 40
4.36 Alterar uma linha especifica de um arquivo de texto 41
4.37 Verificar se o tamanho de um arquivo é maior que um limite especificado 42
4.38 Procurar por arquivos com extensão específica em todos os subdiretórios de uma pasta 43
4.39 Renomear automaticamente arquivos com base na data de criação 44
4.40 Script que leia todos os arquivos em uma pasta e crie links simbólicos (ou atalhos) para cada arquivo na mesma pasta 45
4.41 Script para dividir automaticamente arquivos grandes em menores 46
4.42 Script para remover linhas duplicadas em arquivo de texto 47
4.43 Mover arquivos para pastas com base na data de criação 48
4.44 Dividir automaticamente arquivos em arquivos menores com base em um tamanho específico. 49
4.45 Remover caráter de controle presente no arquivo 50
4.46 Ler e escrever arquivo muito grande 51
4.47 Ler arquivo e remover linhas baseando em limite de caracter 52
4.48 Criando um script para remover automaticamente arquivos duplicados do mesmo tamanho 53
4.49 Concatenar dois arquivos com header igual e removendo duplicidade 54
4.50 Detectar o encoding de um arquivo 55
Conclusão 56

Introdução

Este e-book oferece uma coleção valiosa de 50 códigos completos e prontos feitos em Python para desenvolvedores. Com exemplos de código passo a passo e explicações detalhadas, você encontrará soluções para tarefas comuns e aumentará sua eficiência. É ideal para desenvolvedores iniciantes e avançados, bem como para aqueles que buscam aumentar sua produtividade no trabalho ou no dia a dia. Com os conhecimentos adquiridos neste livro, você será capaz de usar esses códigos prontos para resolver problemas e melhorar seu fluxo de trabalho.

Divirta-se! 

1.Preparação do ambiente

É recomendado instalar o Python e um editor. Eu sempre recomendo o Visual Studio Code da Microsoft. Para instalar o Python, acesse o site oficial https://www.python.org/downloads/ e baixe a versão mais recente. Depois disso, é recomendado criar um ambiente virtual para instalar as bibliotecas necessárias para o seu projeto. Isso pode ser feito usando o gerenciador de pacotes pip, que vem incluído com o Python. Para instalar o Visual Studio Code, acesse https://code.visualstudio.com/download e baixe a versão para o seu sistema operacional. Após a instalação, você pode usar o VSCode para escrever e executar seu código Python.

2.Instalação das dependências

Existem algumas etapas gerais que você pode seguir para configurar e instalar um pacote Python:

- Verifique se você tem a versão mais recente do Python instalada no seu sistema.
- Crie um ambiente virtual (opcional) usando o gerenciador de pacotes virtualenv ou conda, se você quiser manter as dependências do seu projeto isoladas.
- Ative o ambiente virtual, se você estiver usando um.
- Use o gerenciador de pacotes pip ou conda para instalar o pacote desejado. No caso do pip, o comando seria "**pip install nome_do_pacote**" e no caso do conda "conda install nome_do_pacote".
- Verifique se o pacote foi instalado corretamente, digitando "**pip freeze**" ou "conda list" para listar todos os pacotes instalados no ambiente virtual ativo.
- Inicie o seu projeto e importe o pacote para começar a usá-lo.
- Em alguns casos, você pode precisar instalar dependências adicionais antes de instalar o pacote desejado. Certifique-se de ler as instruções de instalação fornecidas pelo desenvolvedor do pacote e siga qualquer passo adicional específico.
- Lembre-se de que alguns pacotes podem ter problemas de compatibilidade com sua versão do Python ou sistema operacional, então sempre verifique as informações de compatibilidade antes de instalar qualquer pacote.

3.Instalação para cada ambiente

- **Windows** : pip install nome_do_pacote
- **Ambiente virtual usando anaconda**: conda install nome_do_pacote
- **Linux**: sudo pip install nome_do_pacote

4.Enumeração dos 100 scripts feitos em Python

4.1 Obter arquivo em outro servidor usando Python

Para obter um arquivo em outro servidor usando o módulo Python Paramiko, você pode usar o método SFTPClient.get() da classe SFTPClient. Primeiro, você precisa criar uma instância de Transport e conectá-la ao servidor usando as informações de autenticação. Em seguida, você pode criar uma instância de SFTPClient usando o método Transport.open_sftp() e chamar o método get() passando o caminho do arquivo no servidor e o local onde você deseja salvar o arquivo localmente.
Para instalar o pacote **paramiko** do Python:

- Instalação do pacote

pip install paramiko



```python
import paramiko

def get_file(hostname, username, password, remotepath, localpath):
    try:
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(hostname=hostname, username=username, password=password)
        sftp = ssh.open_sftp()
        sftp.get(remotepath, localpath)
    except Exception as e:
        print("Erro ao conectar no servidor %s" % hostname)
        print(e)
    finally:
        if sftp:
            sftp.close()
        if ssh:
            ssh.close()
# Exemplo de uso da função get_file():
get_file('192.168.1.10', 'usuario', 'senha', '/home/arquivo-remoto.txt', '/home/arquivo-local.txt')
```

4.2 Renomear arquivos baseado em um padrão específico

Esse código Python tem como objetivo mover arquivos de uma pasta para outra, separando os arquivos por extensão. O código importa o módulo os e o método move do módulo shutil. A função mover_arquivos recebe como parâmetros a origem e o destino dos arquivos. Em seguida, é feito um loop para percorrer todos os arquivos da pasta de origem. Se o arquivo terminar com .txt, ele é movido para a pasta destino/texto; se terminar com .jpg, é movido para a pasta destino/imagens; e se terminar com .pdf, é movido para a pasta destino/documentos. Por fim, é mostrado um exemplo de uso da função.



```python
import os

def renomear_arquivos(diretorio, padrao):
  for arquivo in os.listdir(diretorio):
    nome_atual = os.path.join(diretorio, arquivo)
    novo_nome = os.path.join(diretorio, padrao % arquivo[0:-4])
    os.rename(nome_atual, novo_nome)

# Exemplo: renomear todos os arquivos de um diretório para que tenham o formato "arquivo_<número>.txt"
renomear_arquivos("/caminho/para/diretorio", "arquivo_%s.txt")
```

4.3 Mover arquivos para diferentes diretórios baseado nas suas extensões

Este código é uma função que exclui arquivos de um diretório especificado que são mais antigos do que o número de dias especificado. Primeiro, obtém o tempo atual em segundos e converte o número de dias para segundos. Em seguida, itera sobre os arquivos no diretório especificado e obtém a data de modificação do arquivo em segundos. Se a data de modificação for menor que o limite estabelecido, o arquivo é excluído.



```python
import os
from shutil import move

def mover_arquivos(origem, destino):
    for arquivo in os.listdir(origem):
        if arquivo.endswith('.txt'):
            move(origem + arquivo, destino + 'texto/')
        elif arquivo.endswith('.jpg'):
            move(origem + arquivo, destino + 'imagens/')
        elif arquivo.endswith('.pdf'):
            move(origem + arquivo, destino + 'documentos/')
# Exemplo de uso:
mover_arquivos('/caminho/para/diretorio', '/caminho/para/diretorio/')
```

4.4 Excluir arquivos que são mais antigos do que uma data específica

Este código é uma função que exclui arquivos de um diretório especificado que são mais antigos do que o número de dias especificado. Primeiro, obtém o tempo atual em segundos e converte o número de dias para segundos. Em seguida, itera sobre os arquivos no diretório especificado e obtém a data de modificação do arquivo em segundos. Se a data de modificação for menor que o limite estabelecido, o arquivo é excluído.



```python
import os
import time

def delete_files_older_than(dir_name, days):
    # obtem o tempo atual em segundos
    now = time.time()
    # converte o numero de dias para segundos
    delta = days * 24 * 60 * 60
    # obtem o tempo limite em segundos
    delete_before = now - delta
    # itera sobre os arquivos do diretorio especificado
    for f in os.listdir(dir_name):
        # obtem o caminho completo do arquivo
        file_path = os.path.join(dir_name, f)
        # verifica se é um arquivo e não um diretorio
        if os.path.isfile(file_path):
            # obtem a data de modificação do arquivo em segundos
            mtime = os.stat(file_path).st_mtime
            # verifica se a data de modificação é menor que o limite estabelecido
            if mtime < delete_before:
                # exclui o arquivo caso esteja dentro do limite estabelecido
                os.remove(file_path)

#Exemplo
delete_files_older_than("/caminho/para/diretorio", 0)
```

4.5 Copiar arquivos de um local para outro

Esse código importa a biblioteca **shutil** e define uma função chamada copy_files que recebe dois parâmetros, src (origem) e dest (destino). A função tenta copiar o arquivo src para o destino e, caso haja algum erro, imprime uma mensagem de erro. O exemplo mostra como a função é chamada para copiar um arquivo de um diretório para outro.



```python
import shutil

def copy_files(src, dest):
    try:
        shutil.copy(src, dest)
    except IOError as e:
        print("Error na copia do arquivo. %s" % e)

# exemplo
copy_files("/caminho/para/diretorio/file1.txt", "/caminho/para/diretorio/destino/")
```