一个简单的基于paramiko的类

最近想要在jupyterlab中构建主机和服务器的通信，对paramiko做了一个简单的封装

一些事项

- 采用公钥—私钥—私钥密码的方式进行连接
- 简单定义了 `__enter__` 和 `__exit__` ，通过上下文管理器进行连接通道管理
- upload和download方法检测文件或是文件夹进行上传和下载

一些学习

- `os.walk` 递归遍历一个路径，返回一个生成器，解包后分别为源路径本身，路径下文件夹，路径下文件地址

```
for direc, folder, files in os.walk(directory):
    for file in files:
        all_files.append(os.path.join(direc, file))
```

- 列表的 `extend` 和 `append` , `extend` 接收一个可迭代对象，把这个可迭代对象解包了分别添加到列表的尾部，而 `append` 接收一个任意对象，把这个对象直接添加到列表的尾部. 这两个命令都是就地(in-place)操作，返回值均为None.

- Shell命令中

```
mkdir -p(--parents) path：递归生成一个路径，子目录的父目录不存在时，会直接生成
mkdir path：只能生成子目录，父目录不存在时会报错
```

`os.makedirs` 和 `os.mkdir` 分别对应上面这两个命令

- `str.replace(old,new)` 将字符串中的旧字符替换成新字符

- 在本机上，可以用 `os.path.isdir` 或 `os.path.isfile` 判断一个路径是文件还是文件夹，远程连接服务器时，可以用 `SFTPAttributes` 对象的st_mode码来进行判断

```
stat.S_ISREG(self.sftp.lstat(remotepath).st_mode)
```

Probe类

```
import os
import stat

import paramiko

class Probe:
    def __init__(self):
        self.pkey = paramiko.RSAKey.from_private_key_file(
            "your_rsa_private_key",
            password="private_key_password",
        )
        self.ip = "server ip"
        self.username = "server name"
        self.trans = paramiko.Transport((self.ip, 22))
        self.trans.connect(username=self.username, pkey=self.pkey)
        self.sftp = paramiko.SFTPClient.from_transport(self.trans)
        self.ssh = paramiko.SSHClient()
        self.ssh._transport = self.trans

    def __enter__(self):
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        self.trans.close()

    def __get_all_files(self, directory, remote=True):  # revise from CSDN:littleRpl
        all_files = []
        if remote == True:
            files = self.sftp.listdir_attr(directory)
            for file in files:
                filename = directory + "/" + file.filename

                if stat.S_ISDIR(file.st_mode):
                    all_files.extend(self.__get_all_files(filename))
                else:
                    all_files.append(filename)

            return all_files
        else:
            for direc, folder, files in os.walk(directory):
                for file in files:
                    all_files.append(os.path.join(direc, file))
            return all_files

    def upload(self, localpath, remotepath):
```

```
        if os.path.isfile(localpath):
            self.sftp.put(localpath, remotepath)
        else:
            all_files = self.__get_all_files(localpath, remote=False)
            for file in all_files:

                remote_filename = file.replace(localpath, remotepath).replace("\\", "/")
                print(remote_filename)
                remote_path = os.path.dirname(remote_filename)

                try:
                    sftp.stat(remote_path)
                except:
                    self.run_shell("mkdir -p {}".format(remote_path))

                self.sftp.put(file, remote_filename)

    def download(self, localpath, remotepath):
        if stat.S_ISREG(self.sftp.lstat(remotepath).st_mode):
            self.sftp.get(remotepath, localpath)
        else:
            all_files = self.__get_all_files(remotepath, remote=True)
            for file in all_files:

                local_filename = file.replace(remotepath, localpath)
                local_path = os.path.dirname(local_filename)
                if not os.path.exists(local_path):
                    os.makedirs(local_path)
                self.sftp.get(file, local_filename)

    def run_shell(self, command):
        command = "bash -lc '{}'".format(command)
        stdin, stdout, stderr = self.ssh.exec_command(command, get_pty=True)
        out = stdout.read().decode()
        error = stderr.read().decode()
        return out
```