Dictionary Hash Check

Background

Hash functions are used to uniquely represent data in a fixed-size digest. In Python, the hashlib library can compute cryptographic hashes such as SHA-256. This problem challenges you to compute the SHA-256 hash of a dictionary (after converting it to a string) and validate the correctness of the hashing process.

Task

You are given a Python dictionary that represents a user profile. Your tasks are:
- Convert the dictionary to a string and compute its SHA-256 hash.
- Validate that two different methods of hashing the same dictionary yield the same result.
- Demonstrate that modifying the dictionary yields a different hash value.
- Print two result lines based on whether the hashes match or not.

Input

There is no input. The dictionary is fixed as follows:

```
{
    "name": "Alice",
    "age": 30,
    "is_admin": True,
    "skills": ["Python", "Network Programming", "Digital Forensics"]
}
```

Example output

```
The hashes do not match. d072e72e09134210744569882ec936478b2c2e71e65fdf73cd50385b013db0d7 f4afe43c729b7706d27c59c37fa8d08fdb2e04d0ab2b43d1bd7d1ca6fae88e83
The hashes match. d072e72e09134210744569882ec936478b2c2e71e65fdf73cd50385b013db0d7
```

Constraints

- Use str() to convert the dictionary to a string before hashing.
- Use hashlib.sha256().hexdigest() to compute SHA-256.
- String conversion of a dictionary may vary if unordered—this is acceptable for this task since the dictionary structure is fixed.

Implementation guide

Your program should implement:

```
def hash_dict(data):
    # Return SHA-256 hash digest (hex) of dictionary string
    pass
```

And in your main block:
- Compute two hashes of the same dictionary using two methods and compare them.
- Modify the dictionary (e.g. change age to 31) and hash again.
- Print the two lines indicating whether the hashes match or not.