Zlib Compression Roundtrip

Background

In many applications, data needs to be both serialized and compressed before sto
rage or transmission. Python supports serialization using pickle and compression
 via zlib. This problem challenges you to verify the correctness of both process
es on a given dictionary.

Task

You are given a dictionary containing user information. Your task is to:
- Serialize and compress the dictionary into a byte string using pickle and zlib
.
- Decompress and deserialize the result back into the original dictionary.
- Compare both the compressed byte string and the reconstructed dictionary to ve
rify correctness.
- Print two specific output messages depending on whether the results match expe
ctations.

Input

There is no input. The dictionary is hardcoded and defined as:

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

Example output

```
The compressed byte strings match. b'x\x9ck`\x99\x9a\xc1\x00\x01\xb5S4zX\xf2\x12
sS\xa7\xf4\xb0:\xe6d&\x03i\xe6\xc4\xf4\xd4)\xder=\x1c\x99\xc5\xf1\x89)\xb9\x99yS
:z\xd8\x8a\xb33sr\x8a\xa7\xc4\x02\x95\xb3\x05T\x96d\xe4\xe7M\xe9\x11\xf6K-)\xcf/
\xcaV\x08(\xcaO/J\xcc\x05*M\x9f\xd2#\xe8\x92\x99\x9eY\x92\x98\xa3\xe0\x96_\x94\x
9aW\x9c\x99\\<%\xb5T\x0f\x00\xf4\xc6*,'
The dictionaries match. {'name': 'Alice', 'age': 30, 'is_admin': True, 'skills':
 ['Python', 'Network Programming', 'Digital Forensics']} {'name': 'Alice', 'age'
: 30, 'is_admin': True, 'skills': ['Python', 'Network Programming', 'Digital For
ensics']}
```

Constraints

- Use pickle.dumps() for serialization and zlib.compress() for compression.
- Use zlib.decompress() followed by pickle.loads() for decompression.
- Byte comparison must be done using strict equality (==).
- Dictionary comparison must ensure all keys and values match exactly.

Implementation guide

Your program should define the following functions:

```
def compress_dict(data):
    # Compress using pickle + zlib
```

```
    pass

def decompress_dict(compressed_data):
    # Decompress and deserialize
    pass
```

Then write a main block that:
- Compresses the data using compress_dict().
- Decompresses the result using decompress_dict().
- Prints the two required output lines.