_4bucles/_13fibonacci.py

```python
def fibonacci(indice):
    if indice <= 1:
        return indice
    else:
        return fibonacci(indice - 1) + fibonacci(indice - 2)

# Ejemplo de uso:
print(fibonacci(7))  # Salida: 13

# Initialize an empty dictionary for caching Fibonacci numbers
fibonacci_cache = {}

def fibonacci_2(indice):
    # Check if the value is in the cache
    if indice in fibonacci_cache:
        return fibonacci_cache[indice]

    # Compute the Fibonacci number
    if indice <= 1:
        result = indice
    else:
        result = fibonacci_2(indice - 1) + fibonacci_2(indice - 2)

    # Store the result in the cache
    fibonacci_cache[indice] = result

    return result

# Call the function and print the result for indice 20
print(fibonacci_2(20))

from functools import lru_cache

@lru_cache(maxsize=20)
def fibonacci_cache_implicito(indice):
    if indice <= 1:
        return indice
    else:
        return fibonacci_cache_implicito(indice - 1) + fibonacci_cache_implicito(indice - 2)

def fibonacci_iter(indice):
    if indice <= 1:
        return indice
    secuencia = [0, 1]
    for i in range(2, indice + 1):
        secuencia.append(secuencia[-1] + secuencia[-2])
    return secuencia[-1]

# Testing both implementations
print(fibonacci_cache_implicito(20))  # Output: 6765
print(fibonacci_iter(20))             # Output: 6765
```