Function Caching in python

Function caching is a technique for improving the performance of a program by string the results of a function call so that you can reuse the results instead of recomputing them every time the function is called. This can be particularly useful when a function is computationally expensive, or when the inputs to the function are unlikely to change frequently.

In Python, function caching can be achieved using the methods lru_caching decorator. The functools lru_cache decorator is used to cache the results of a function so that you can reuse the results instead of recomputing them every time the function is called.

Here's an example

```
from functools import lru_cache
import time

@lru_cache(maxsize=None)

def fx(n):
    time.sleep(5)
    return n*5

print(fx(20))
print("Done for 20")

print(fx(2))
print("Done for 2")

print(fx(5))
print("Done for 5")

# it used the term memoization.
# do not fetch the values from cache again
print(fx(20))
print("Done for 20")

print(fx(2))
print("Done for 2")

print(fx(5))
```

```
print("Done for 5")
```

Source Code

```
from functools import lru_cache
import time

@lru_cache(maxsize=None)

def fx(n):
    time.sleep(5)
    return n*5

print(fx(20))
print("Done for 20")

print(fx(2))
print("Done for 2")

print(fx(5))
print("Done for 5")

# it used the term memoization.
# do not fetch the values from cache again
print(fx(20))
print("Done for 20")

print(fx(2))
print("Done for 2")

print(fx(5))
print("Done for 5")
```

Thank You