Position	F(n)	f(n)
10th	0.000119 seconds	0.0000157 seconds
20th	0.01767 seconds	0.0000460 seconds
30th	1.36 seconds	0.00002169 seconds
40th	159.006 seconds	0.0000352 seconds

In this case, iterative is more consistent and faster than recursive. The recursive method takes so long because for each call to the recursive method (other than when a base case in involved), there are two separate recursive calls, rather than one. This takes up much more memory allocation and is more complex computationally than a simple for loop. The loop method only goes through each number in the Fibonacci sequence once, while the recursive method may go through multiple values of the sequence a second time.

```
# Garrett Gilliom and Matt Ryan

import time


def F(n):

if n == 0:

return 0

if n == 1 or n == 2:

return 1

else:

return F(n-1) + F(n-2)


def f(n):

added = 0

first = 0

second = 1

if n == 1:

return 1

for i in range(n - 1):

added = first + second
```

```
first = second

second = added

return added


tStartF = time.time()

print(F(40))

tFinishF = time.time()

FTime = tFinishF - tStartF

print("Recursion: " + str(FTime))

tStartf = time.time()

print(f(20))

tFinishf = time.time()

fTime = tFinishf - tStartf

print("Loop: " + str(fTime))
```