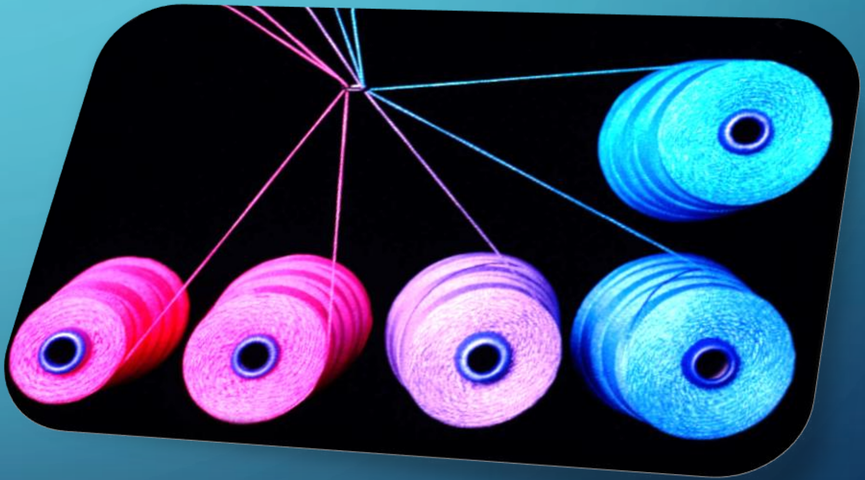
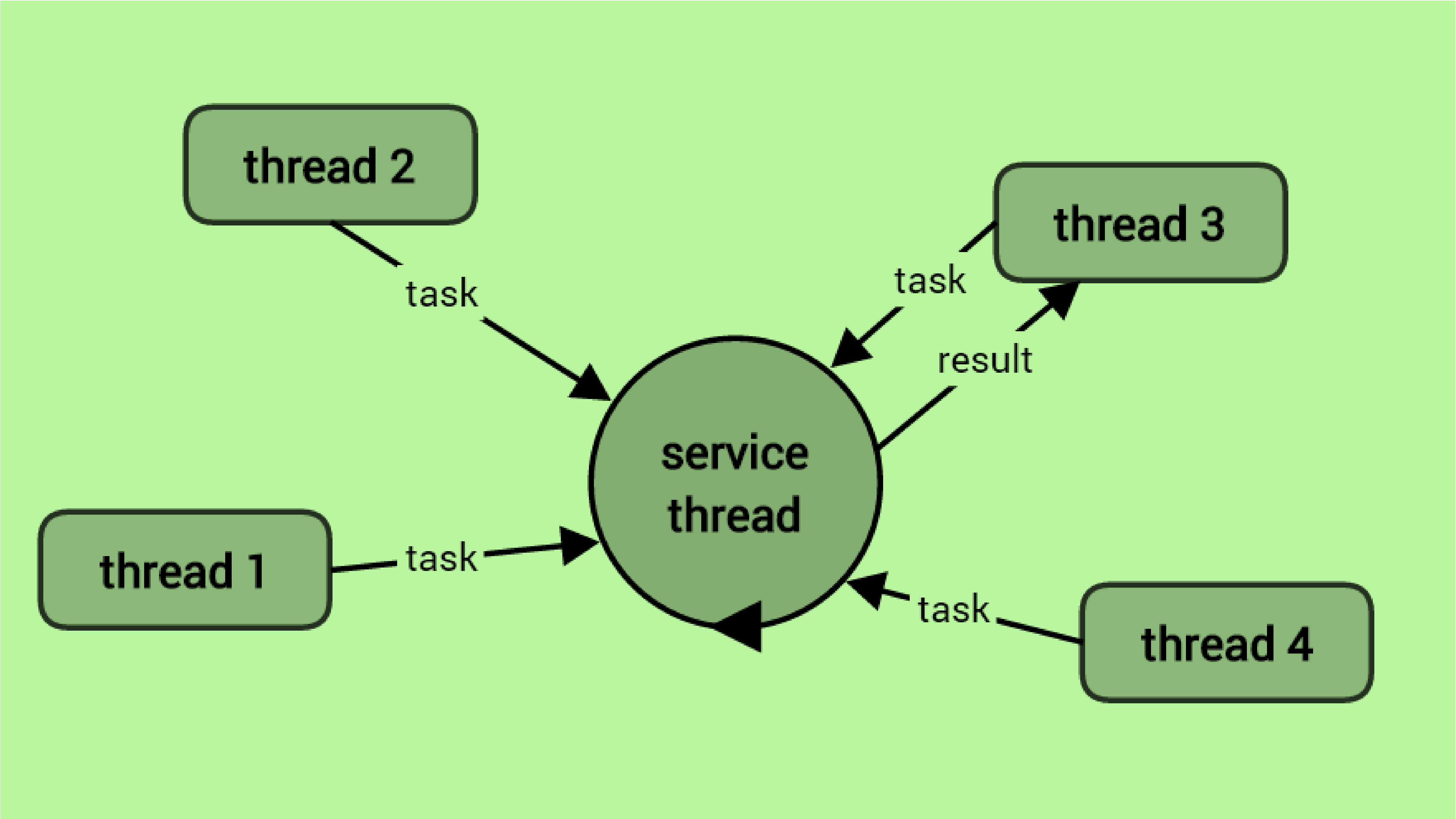
THREADING

ASYNCHRONOUS PROGRAMMING



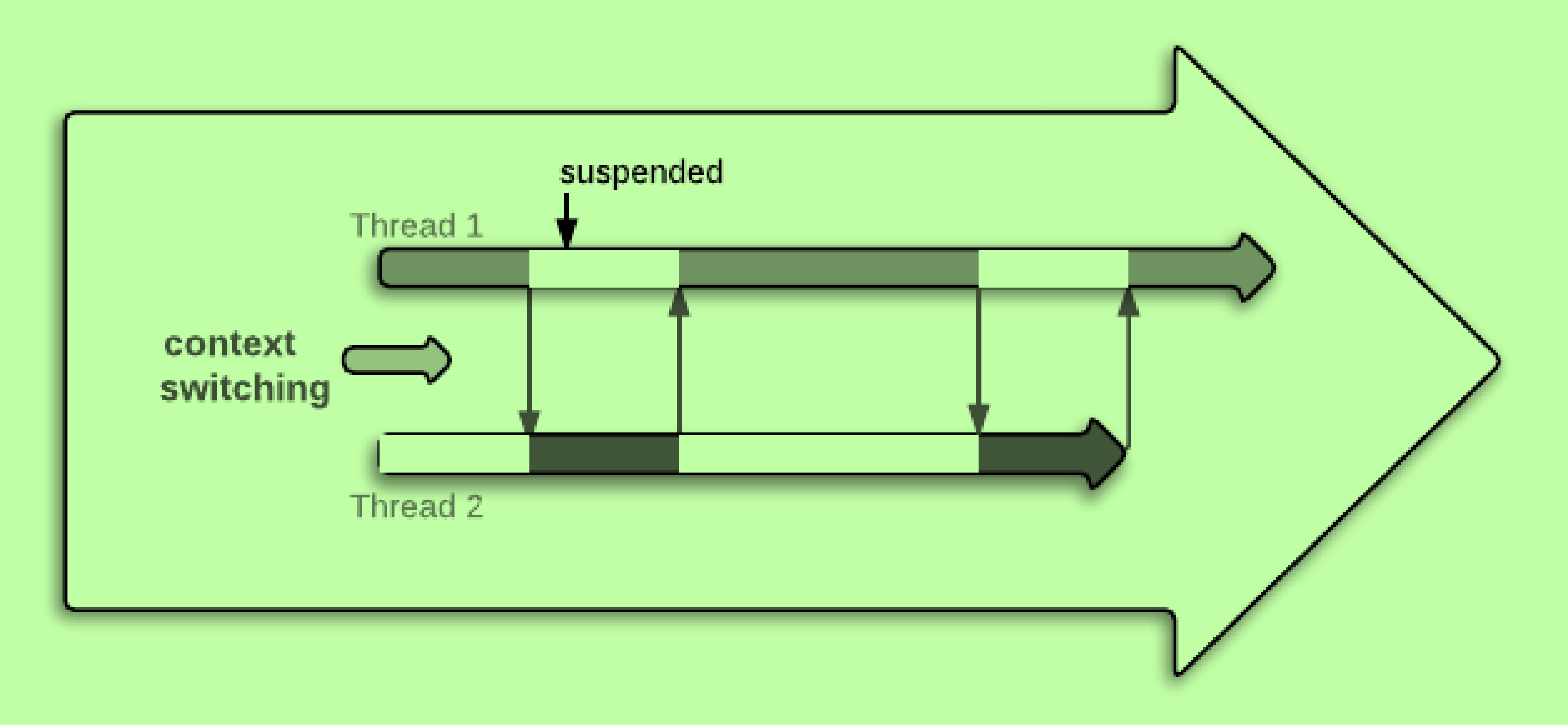

thread 2
task
thread 3
task
result
service thread
thread 1
task
task
thread 4

Introduction

- A **thread** of execution is a sequence of programmed instructions that can be managed independently by a scheduler.
- A **scheduler** is part of the operating system.
- A **thread** an element of a process.
- A **process** is the instance of a computer program that is being executed.
- One process can include multiple threads, which can be executing simultaneously also called **Multithreading.**

Context Switching

- In Python 3 implementations the different threads do not actually execute at the same time: they simply appear to.
- Threading is achieved by using frequent switching between threads. This is termed as **context switching**.
- In context switching, the state of a thread is saved and state of another thread is loaded whenever any interrupt (due to I/O or manually set) takes place.
- Context switching takes place so frequently that all the threads appear to be running parallelly (this is termed as **multitasking**).


suspended
Thread 1
context switching
Thread 2

Techniques

- Python’s threading module provides a Thread class to create and manage threads.
- Extend this class to create a Thread.
- Directly create Thread class object and pass member function of other class.

Example

```
from ThreadingClass import MyThread


thread_A = MyThread('Task_A', 1)
thread_B = MyThread('Task_B’, 2)

thread_A.start()
thread_B.start()

thread_A.join()
thread_B.join()


print('Finished.')
```

Serial Threads

```
th = FileLoaderThread('users.csv','ABC')
th.start()
for i in range(5):
        print('Hi from Main Function')
        time.sleep(1)
th.join()
```

Inheritance

```
class FileLoaderThread(Thread):
    def __init__(self, fileName, encryptionType):
        Thread.__init__(self)
        self.fileName = fileName
        self.encryptionType = encryptionType
    def run(self):
        for i in range(5):
            print('Loading ... ‘)
            time.sleep(1)
        print('Finished loading contents from file : ', self.fileName)
```

Start

- **start()** will start a new thread, which will execute the function threadFunc() in parallel to main thread.

- After calling start() function on thread object, control will return to Main thread.

Run

- The run() method represents the thread’s activity, and can be overridden this method in a subclass.

- The standard run() method invokes the callable object passed to the object’s constructor as the target argument.

Join

- If the join() function is called in the main thread, then main() function will wait for the threads to finish.

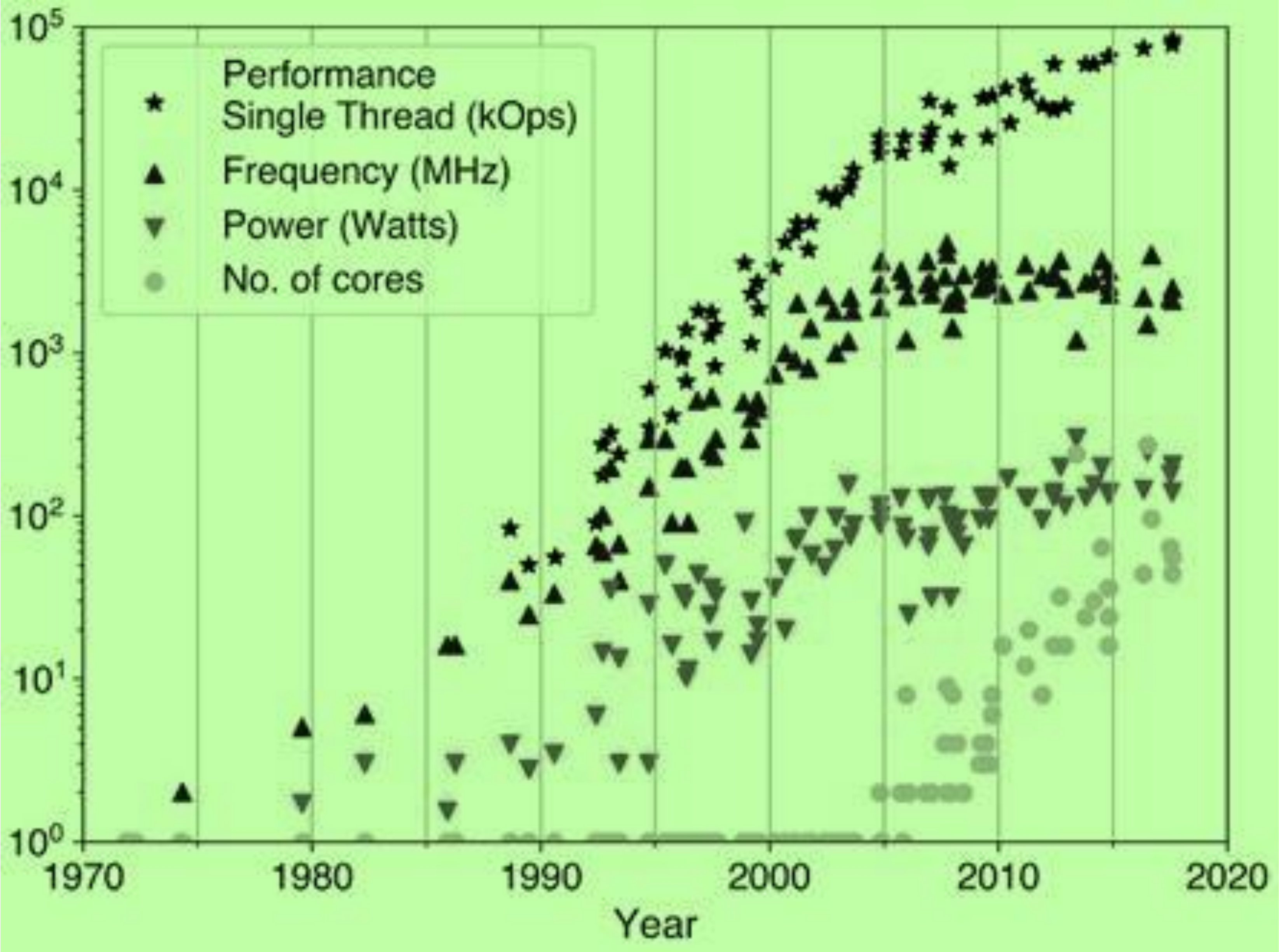
- If main() function finishes it’s work first, it can exit without for other threads to finish.

Locks

- Locks are the most fundamental synchronization mechanism provided by the **threading** module.

- At any time, a lock can be held by a single thread, or by no thread at all.

- If a thread attempts to hold a lock that’s already held by some other thread, execution of the unlocked thread is halted until the lock is released.

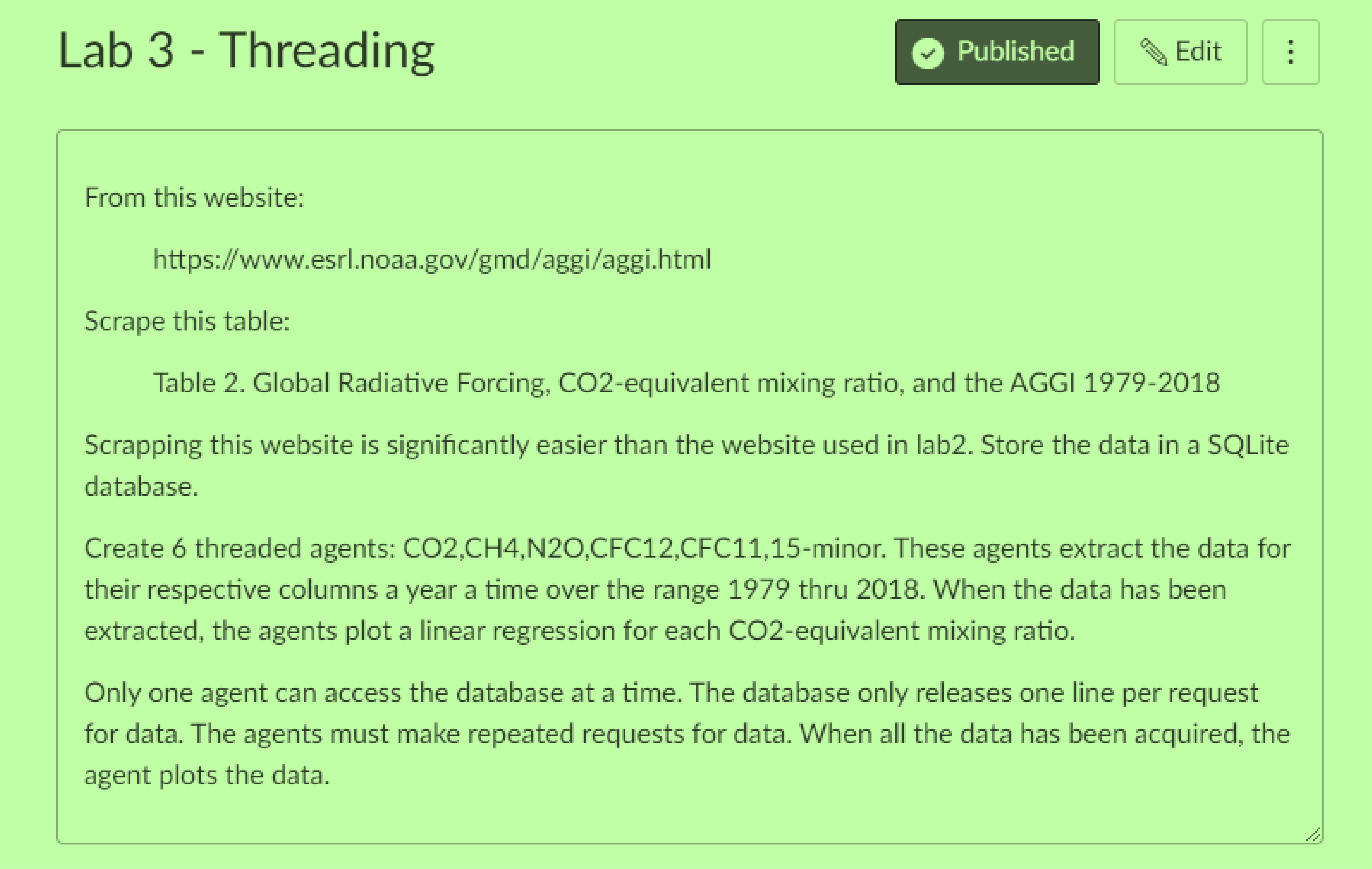


Performance Single Thread (kOps)
Frequency (MHz)
Power (Watts)
No. of cores
10^5
10^4
10^3
10^2
10^1
10^0
1970
1980
1990
2000
2010
2020
Year

Python Threading Tutorial

https://www.geeksforgeeks.org/multithreading-python-set-1/

- A beginners tutorial to the basics of thread programming



Lab 3 - Threading

From this website:

https://www.esrl.noaa.gov/gmd/aggi/aggi.html

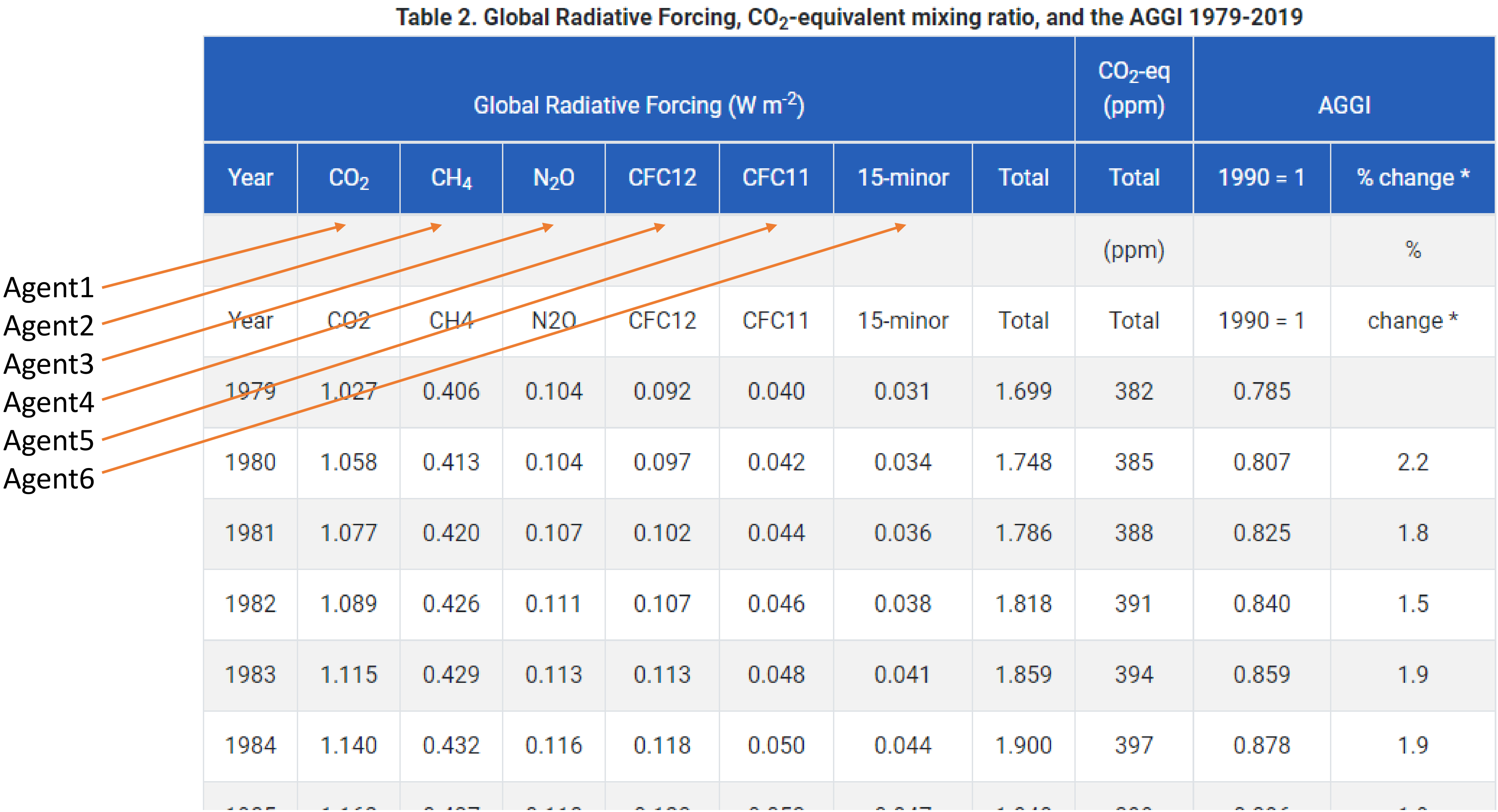
Scrape this table:

Table 2. Global Radiative Forcing, CO2-equivalent mixing ratio, and the AGGI 1979-2018

Scrapping this website is significantly easier than the website used in lab2. Store the data in a SQLite database.

Create 6 threaded agents: CO2,CH4,N2O,CFC12,CFC11,15-minor. These agents extract the data for their respective columns a year a time over the range 1979 thru 2018. When the data has been extracted, the agents plot a linear regression for each CO2-equivalent mixing ratio.

Only one agent can access the database at a time. The database only releases one line per request for data. The agents must make repeated requests for data. When all the data has been acquired, the agent plots the data.



Agent1

Agent2

Agent3

Agent4

Agent5

Agent6

Table 2. Global Radiative Forcing, CO_2-equivalent mixing ratio, and the AGGI 1979-2019

Global Radiative Forcing (W m^{-2})								CO_2-eq (ppm)	AGGI	
Year	CO_2	CH_4	N_2O	CFC12	CFC11	15-minor	Total	Total	1990 = 1	% change *
								(ppm)		%
Year	CO2	CH4	N2O	CFC12	CFC11	15-minor	Total	Total	1990 = 1	change *
1979	1.027	0.406	0.104	0.092	0.040	0.031	1.699	382	0.785	
1980	1.058	0.413	0.104	0.097	0.042	0.034	1.748	385	0.807	2.2
1981	1.077	0.420	0.107	0.102	0.044	0.036	1.786	388	0.825	1.8
1982	1.089	0.426	0.111	0.107	0.046	0.038	1.818	391	0.840	1.5
1983	1.115	0.429	0.113	0.113	0.048	0.041	1.859	394	0.859	1.9
1984	1.140	0.432	0.116	0.118	0.050	0.044	1.900	397	0.878	1.9