Ejercicio 1:

Haga una tabla en la que refleje los tiempos de ejecución (en milisegundos) del módulo PythonA1.py, para los valores de n expuestos (10000, 20000, 40000, 80000, 160000, 320000, 640000 y 1280000). Si para alguna n tardara más de 60 segundos, ponga FdT ("Fuera de Tiempo"), tanto en este apartado como en los apartados posteriores.

n	t (milisegundos)
10000	3640
20000	14236
40000	58375
80000	FdT
160000	FdT
320000	FdT
640000	FdT
1280000	FdT

Ejercicio 2:

Haga una tabla en la que refleje, al menos para dos ordenadores a los que tenga acceso, los tiempos de ejecución (en milisegundos) del módulo PythonA1.py para los valores de n expuestos (10000, 20000, …, 640000 y 1280000). Referencie claramente para cada ordenador qué CPU es la existente y cantidad de Memoria principal RAM.

	t (milisegundos)	
n	Ordenador 1	Ordenador 2
10000	3640	2264
20000	14236	9025
40000	58375	36508
80000	FdT	FdT
160000	FdT	FdT
320000	FdT	FdT
640000	FdT	FdT
1280000	FdT	FdT

	CPU	RAM
Ordenador 1	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz	12 GB
Ordenador 2	Intel(R) Core(TM) i5-6600 CPU @ 3.30GHz	8 GB

Ejercicio 3:

Programe una clase de nombre JavaA1.java que calcule el problema de los números primos expuesto y que utilice el mismo algoritmo A1 para ver si un número es primo que utilizaba PythonA1.py.

Posteriormente, hay que hacer una tabla en la que refleje los tiempos de ejecución (en milisegundos) SIN_OPTIMIZACIÓN de JavaA1.java, para los valores de n expuestos (10000, 20000, …, 640000 y 1280000).

Para finalizar, compare esos tiempos con los obtenidos en Python (en un apartado anterior) para ese mismo algoritmo A1.

	t (milisegundos)	
n	Java (SIN OPTIMIZACIÓN) [JavaA1.java]	Python [PythonA1.py]
10000	672	3640
20000	2501	14236
40000	18949	58375
80000	FdT	FdT
160000	FdT	FdT
320000	FdT	FdT
640000	FdT	FdT
1280000	FdT	FdT

Ejercicio 4:

Haga una tabla en la que refleje los tiempos de ejecución (en milisegundos) de los módulos PythonA1.py, PythonA2.py y PythonA3.py, para los valores de n expuestos (10000, 20000, …, 640000 y 1280000).

Implemente esos mismos algoritmos A2 y A3 en Java, en dos clases de nombres respectivamente JavaA2.java y JavaA3.java.

Realice una tabla en la que refleje los tiempos de ejecución (en milisegundos), ejecutando "SIN_OPTIMIZACIÓN", de las clases JavaA1.java, JavaA2.java y JavaA3.java, para los valores de n expuestos (10000, 20000, …, 640000 y 1280000).

Realice una tabla en la que refleje los tiempos de ejecución (en milisegundos), ejecutando "CON_OPTIMIZACIÓN", de las clases JavaA1.java, JavaA2.java y JavaA3.java, para los valores de n expuestos (10000, 20000, …, 640000 y 1280000).

Finalmente, razone las conclusiones finales obtenidas comparando los tiempos antes obtenidos: con Python, con Java "SIN_OPTIMIZACIÓN y con Java "CON_OPTIMIZACIÓN".

	t (milisegundos)		
n	**PythonA1.py**	**PythonA2.py**	**PythonA3.py**
10000	3640	470	309
20000	14236	1537	773
40000	58375	5872	2884
80000	FdT	21942	10936
160000	FdT	FdT	41413
320000	FdT	FdT	FdT
640000	FdT	FdT	FdT
1280000	FdT	FdT	FdT

SIN OPTIMIZACIÓN			
	t (milisegundos)		
n	**JavaA1.java**	**JavaA2.java**	**JavaA3.java**
10000	672	119	77
20000	2501	310	211
40000	18949	969	768
80000	FdT	3474	2501
160000	FdT	22376	15198
320000	FdT	FdT	FdT
640000	FdT	FdT	FdT
1280000	FdT	FdT	FdT

CON OPTIMIZACIÓN			
	t (milisegundos)		
n	**JavaA1.java**	**JavaA2.java**	**JavaA3.java**
10000	165	32	34
20000	614	74	40
40000	2388	282	166
80000	13127	893	436
160000	FdT	3042	1609
320000	FdT	19121	6220
640000	FdT	FdT	43176
1280000	FdT	FdT	FdT

<u>Conclusiones finales acerca de los tiempos obtenidos:</u>

Concluimos que Python es un lenguaje mucho más lento que Java por tener unos tiempos de ejecución notablemente mayores que los de Java sin optimización y con optimización.

Los tiempos de Java sin optimización son mayores que los tiempos de Java con optimización y son obtenidos desactivando la optimización automática del compilador JIT (Just In Time) de Java. Esto lo hacemos para obtener unas medidas de tiempo mucho más fieles a la complejidad del programa.

Los tiempos de Java con optimización son los más pequeños porque el compilador JIT realiza una serie de procesos para que la ejecución de los programas sea más rápida. Esta optimización no la contaba la versión sin optimización de Java, por eso es más rápida que la mencionada. Y al ser java un lenguaje más rápido que Python, concluimos que la versión de Java haciendo uso de la optimización del compilador JIT de Java es la más rápida.