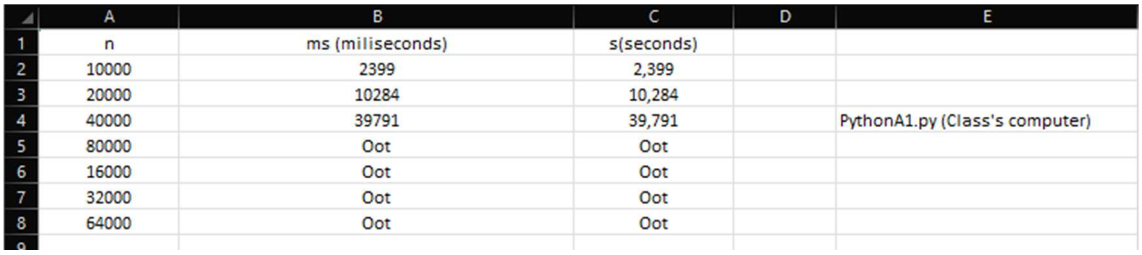
Algorithmics	Student information	Date	Number of session
	UO: 294866	31/01/24	0
	Surname: Menéndez		
	Name: Javier		

Escuela de Ingeniería Informática
Universidad de Oviedo

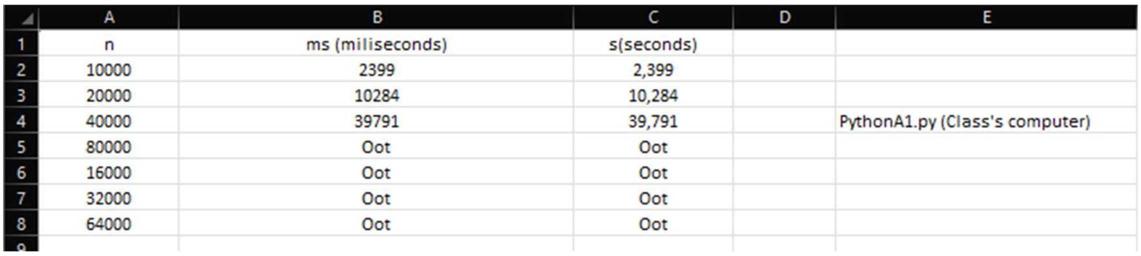
Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. [Problem size]



	A	B	C	D	E
1	n	ms (miliseconds)	s(seconds)		
2	10000	2399	2,399		
3	20000	10284	10,284		
4	40000	39791	39,791		PythonA1.py (Class's computer)
5	80000	Oot	Oot		
6	16000	Oot	Oot		
7	32000	Oot	Oot		
8	64000	Oot	Oot		

Activity 2. [Computer power]

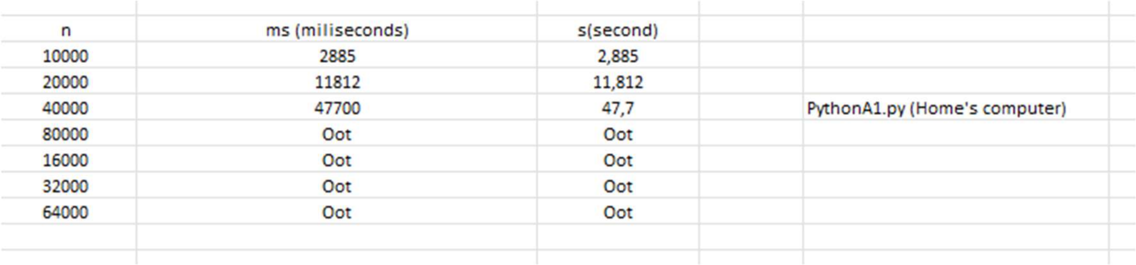


	A	B	C	D	E
1	n	ms (miliseconds)	s(seconds)		
2	10000	2399	2,399		
3	20000	10284	10,284		
4	40000	39791	39,791		PythonA1.py (Class's computer)
5	80000	Oot	Oot		
6	16000	Oot	Oot		
7	32000	Oot	Oot		
8	64000	Oot	Oot		

CPU: Intel® Core ™ i7-4790 CPU @ 3.60GHz

RAM Memory: 8,0 GB DDR3

	Student information	Date	Number of session
Algorithmics	UO: 294866	31/01/24	0
	Surname: Menéndez		
	Name: Javier		



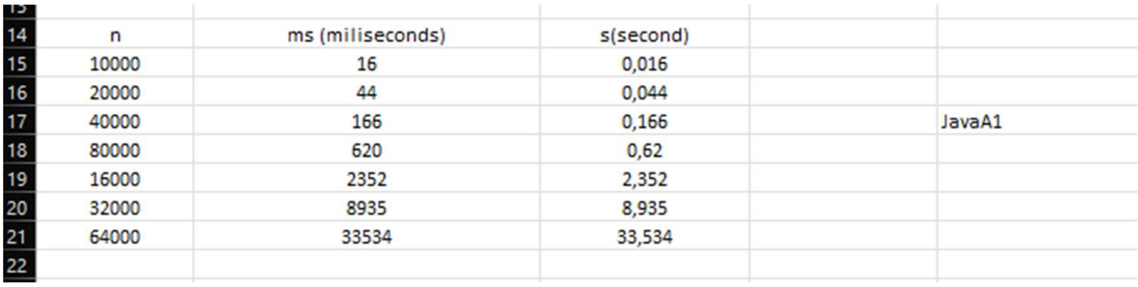
n	ms (miliseconds)	s(second)		
10000	2885	2,885		
20000	11812	11,812		
40000	47700	47,7		PythonA1.py (Home's computer)
80000	Oot	Oot		
16000	Oot	Oot		
32000	Oot	Oot		
64000	Oot	Oot		

CPU: Intel® Core™ i5-9400 CPU @ 2.90GHz

RAM: 16,0 GB 2666mHz

Activity 3. [Implementation environment]

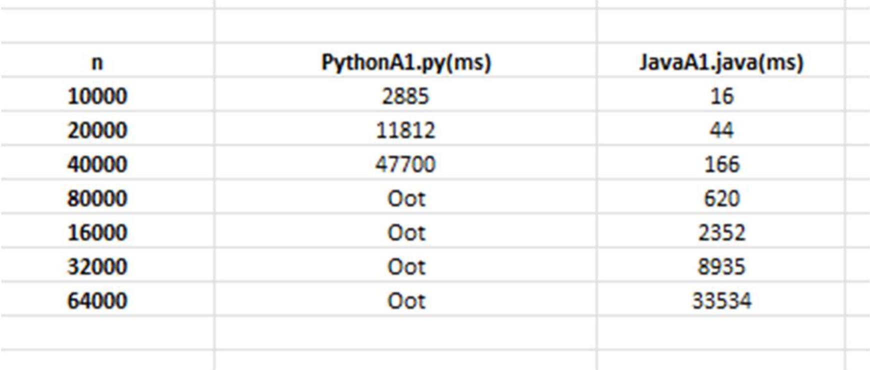
Now I created a class called JavaA1.java which uses the same A1 algorithm to find out if a number is a prime number, the same as PythonA1.py.



14	n	ms (miliseconds)	s(second)		
15	10000	16	0,016		
16	20000	44	0,044		
17	40000	166	0,166		JavaA1
18	80000	620	0,62		
19	16000	2352	2,352		
20	32000	8935	8,935		
21	64000	33534	33,534		
22					

I used the same “n” values as in PythonA1.py and as it can be seeing the time for doing the algorithm in Java it is much less. Here is a table comparing the two different implementation environments.

	Student information	Date	Number of session
Algorithmics	UO: 294866	31/01/24	0
	Surname: Menéndez		
	Name: Javier		

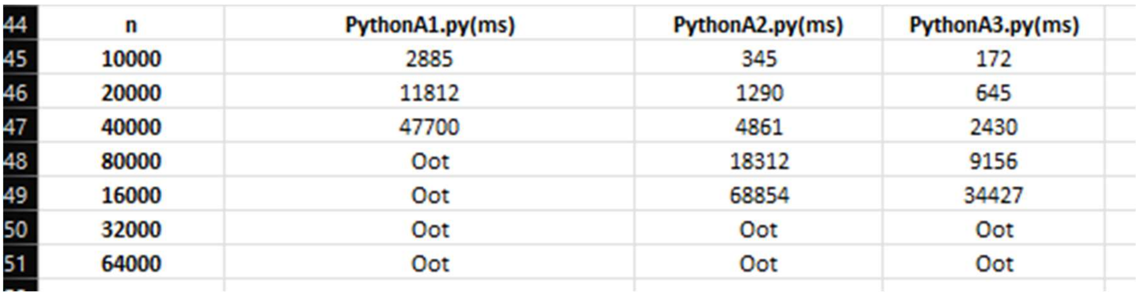


n	PythonA1.py(ms)	JavaA1.java(ms)
10000	2885	16
20000	11812	44
40000	47700	166
80000	Oot	620
16000	Oot	2352
32000	Oot	8935
64000	Oot	33534

Both measurements were done with the CPU: Intel® Core™ i5-9400 CPU @ 2.90GHz and RAM: 16,0 GB 2666mHz.

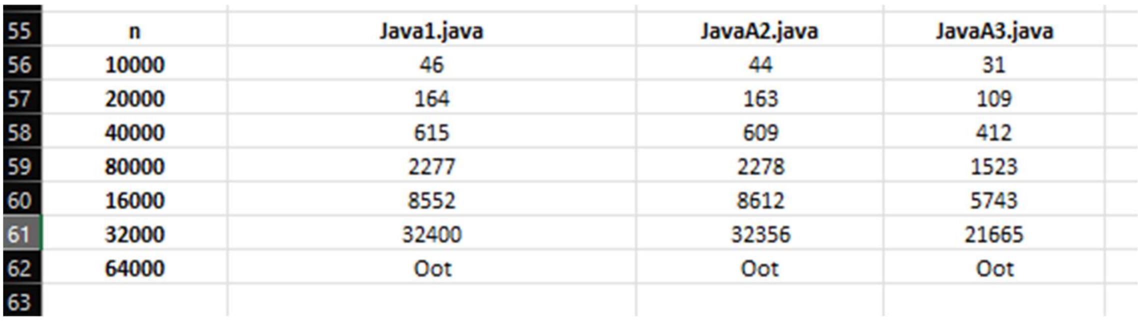
Activity 4. [Algorithm that is used]

Table reflecting the execution times of the modules PythonA1, PythonA2, PythonA3.



44	n	PythonA1.py(ms)	PythonA2.py(ms)	PythonA3.py(ms)
45	10000	2885	345	172
46	20000	11812	1290	645
47	40000	47700	4861	2430
48	80000	Oot	18312	9156
49	16000	Oot	68854	34427
50	32000	Oot	Oot	Oot
51	64000	Oot	Oot	Oot

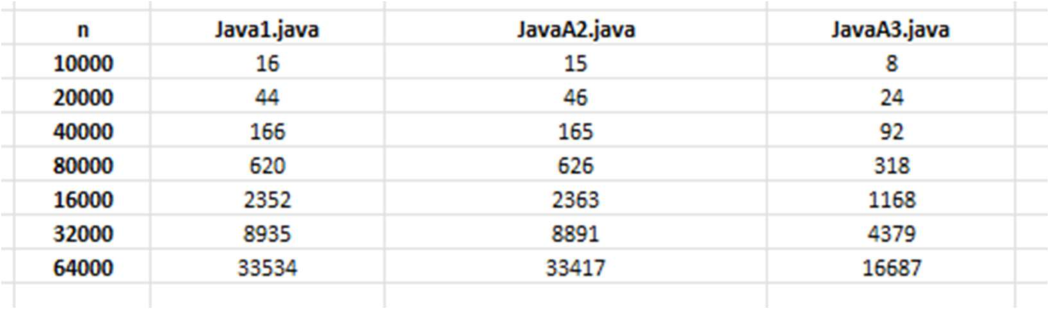
Times of the Java files without optimization (done by CPU: Intel® Core™ i5-9400 CPU @ 2.90GHz)



55	n	Java1.java	JavaA2.java	JavaA3.java
56	10000	46	44	31
57	20000	164	163	109
58	40000	615	609	412
59	80000	2277	2278	1523
60	16000	8552	8612	5743
61	32000	32400	32356	21665
62	64000	Oot	Oot	Oot
63				

Algorithmics	Student information	Date	Number of session
	UO: 294866	31/01/24	0
	Surname: Menéndez		
	Name: Javier		

Times of the Java files with the java optimization (done by CPU: Intel® Core™ i5-9400 CPU @ 2.90GHz)



n	Java1.java	JavaA2.java	JavaA3.java
10000	16	15	8
20000	44	46	24
40000	166	165	92
80000	620	626	318
16000	2352	2363	1168
32000	8935	8891	4379
64000	33534	33417	16687

The conclusions we can take thanks to this project is, first that with the different measurements is clear that Java is faster than Python, then the different algorithms in Java and in Python while increasing are better, for example JavaA3 is faster than JavaA2 and this one is faster than JavaA1. The same happens with Python, the algorithms were improving. Then is clearly saw that the java optimization helps a lot to reduce the execution time.