Initiation_Python_Jupyter_TP1

October 15, 2021

```
In [1]: 5 + 3
```

```
Out[1]: 8
```

```
In [3]: 5 - 3
        5 * 3
        5 / 3
        5 // 3
        5 % 3
        5 ^ 3
```

```
Out[3]: 125
```

```
In [4]: print(5 - 3)
        print(5 * 3)
        print(5 / 3)
        print(5 // 3)
        print(5 % 3)
        print(5^3)
```

```
2
15
5/3
1
2
125
```

Le symbole // donne le quotient (partie entière de la division), le symbole % donne le modulo.

```
In [7]: print(5/3)
        print(5.0/3)
        print(5./3)
        print(RR(5/3))
        print(RR(5)/3)
        print(QQ(5.0)/3)
```

```
5/3
1.66666666666667
1.66666666666667
1.66666666666667
1.66666666666667
5/3
```

```
In [14]: print(type(5))
         print(type(5.0))
         print(type(RR(5)))
         print(type(5/3))
         print(type(int(5)))
         print(type(True))
         print(type("a"))
         print(type(print))
```

```
<class 'sage.rings.integer.Integer'>
<class 'sage.rings.real_mpfr.RealLiteral'>
<class 'sage.rings.real_mpfr.RealNumber'>
<class 'sage.rings.rational.Rational'>
<class 'int'>
<class 'bool'>
<class 'str'>
<class 'builtin_function_or_method'>
```

```
In [31]: print(1 == 2)
         print(5 + 3 == 8)
         print()
         print(5 * 3 > 10)
         print(5 * 3 > 15)
         print(5 * 3 >= 15)
         print()
         print("1" == 1)
         print(5 == 5.0)
         print()
         print(3 == 2 + 1 and 7 == 8 - 1)
         print(1 == 2 or 7 == 8 - 1)
         print(not 1 == 2)
         print()
         print(3 in NN)
         print(1.5 in ZZ)
         print(sqrt(2) in QQ)
```

```
False
True

True
```

```
False
True

False
True

True
True
True

True
False
False
```

```
In [34]: if 1 + 1 == 2:
             print("Tout va bien.")
         if 1 > 2:
             print("Huston, nous avons un problème !")
```

```
Tout va bien.
```

```
In [35]: C1 = {1,2,3,4,5,0}
         C2 = {2,3,4,5,6,7}

         Ind1, Ind2, Shared = set(), set(), set()
         for x in C1:
             if not x in C2:
                 Ind1.add(x)
                 if len(Ind1) >= 2:
                     break
             else:
                 Shared.add(x)
         Ind2 = C2.difference(Shared)

         print(Ind1, Ind2, Shared)
```

```
{0, 1} {2, 3, 4, 5, 6, 7} set()
```

```
In [60]: x = random()
         print(x)
         if x > 0.5:
             print(+1)
         else:
             print(-1)
```

```
0.8214328979112542
1
```

Ce programme simule une loi de Radmarcher : il tire une variable aléatoire qui vaut +1 avec probabilité 1/2 et -1 avec probabilité 1/2.

```
In [1]: x, y = random(), random()
        if x < y:
            print(y)
        else:
            print(x)
```

```
0.9347352069462768
```

```
In [15]: x, y, z = random(), random(), random()
         if x < y:
             if y < z:
                 print(z)
             else:
                 print(y)
         else:
             if x < z:
                 print(z)
             else:
                 print(x)
```

```
0.5391746080268893
```

```
In [68]: for x in {1,2,3,4,5}:
             print(x^2)
             print(x % 2 == 0)
         print()
         print(x)
```

```
1
False
4
True
9
False
16
True
25
False

5
```

```
In [69]: for de1 in {1,2,3,4,5,6}:
             for de2 in {1,2,3,4,5,6}:
                 print(abs(de1-de2))
```

```
0
1
2
3
4
5
1
0
1
2
3
4
2
1
0
1
2
3
3
2
1
0
1
2
4
3
2
1
0
1
5
4
3
2
1
0
```

Il y a 36 couples de dés possibles. On constate ci-dessus que 6 d'en eux ont une différence de 3 : la probabilité d'obtenir 3 vaut donc $6/36 = 1/6$.

```
In [70]: DE = [abs(de1-de2) for de1 in {1,2,3,4,5,6} for de2 in {1,2,3,4,5,6}]
         {k : len([1 for i in range(len(DE)) if DE[i] == k]) for k in {0,1,2,3,4,5}}

Out[70]: {0: 6, 1: 10, 2: 8, 3: 6, 4: 4, 5: 2}

In [67]: for x in NN:
             print(x,x%2==0)
```

```
0 True
1 False
2 True
3 False
4 True
5 False
6 True
7 False
8 True
9 False
10 True
11 False
12 True
13 False
14 True
15 False
16 True
17 False
18 True
19 False
20 True
21 False
22 True
23 False
24 True
25 False
26 True
27 False
28 True
29 False
30 True
31 False
32 True
33 False
34 True
35 False
36 True
37 False
38 True
39 False
40 True
41 False
42 True
43 False
44 True
45 False
46 True
47 False
```

```
48 True
49 False
50 True
51 False
52 True
53 False
54 True
55 False
56 True
57 False
58 True
59 False
60 True
61 False
62 True
63 False
64 True
65 False
66 True
67 False
68 True
69 False
70 True
71 False
72 True
73 False
74 True
75 False
76 True
77 False
78 True
79 False
80 True
81 False
82 True
83 False
84 True
85 False
86 True
87 False
88 True
89 False
90 True
91 False
92 True
93 False
94 True
95 False
```

```
96 True
97 False
98 True
99 False
100 True
101 False
102 True
103 False
104 True
105 False
106 True
107 False
108 True
109 False
110 True
111 False
112 True
113 False
114 True
115 False
116 True
117 False
118 True
119 False
120 True
121 False
122 True
123 False
124 True
125 False
126 True
127 False
128 True
129 False
130 True
131 False
132 True
133 False
134 True
135 False
136 True
137 False
138 True
139 False
140 True
141 False
142 True
143 False
```

144 True
145 False
146 True
147 False
148 True
149 False
150 True
151 False
152 True
153 False
154 True
155 False
156 True
157 False
158 True
159 False
160 True
161 False
162 True
163 False
164 True
165 False
166 True
167 False
168 True
169 False
170 True
171 False
172 True
173 False
174 True
175 False
176 True
177 False
178 True
179 False
180 True
181 False
182 True
183 False
184 True
185 False
186 True
187 False
188 True
189 False
190 True
191 False

192 True
193 False
194 True
195 False
196 True
197 False
198 True
199 False
200 True
201 False
202 True
203 False
204 True
205 False
206 True
207 False
208 True
209 False
210 True
211 False
212 True
213 False
214 True
215 False
216 True
217 False
218 True
219 False
220 True
221 False
222 True
223 False
224 True
225 False
226 True
227 False
228 True
229 False
230 True
231 False
232 True
233 False
234 True
235 False
236 True
237 False
238 True
239 False

240 True
241 False
242 True
243 False
244 True
245 False
246 True
247 False
248 True
249 False
250 True
251 False
252 True
253 False
254 True
255 False
256 True
257 False
258 True
259 False
260 True
261 False
262 True
263 False
264 True
265 False
266 True
267 False
268 True
269 False
270 True
271 False
272 True
273 False
274 True
275 False
276 True
277 False
278 True
279 False
280 True
281 False
282 True
283 False
284 True
285 False
286 True
287 False

288 True
289 False
290 True
291 False
292 True
293 False
294 True
295 False
296 True
297 False
298 True
299 False
300 True
301 False
302 True
303 False
304 True
305 False
306 True
307 False
308 True
309 False
310 True
311 False
312 True
313 False
314 True
315 False
316 True
317 False
318 True
319 False
320 True
321 False
322 True
323 False
324 True
325 False
326 True
327 False
328 True
329 False
330 True
331 False
332 True
333 False
334 True
335 False

336 True
337 False
338 True
339 False
340 True
341 False
342 True
343 False
344 True
345 False
346 True
347 False
348 True
349 False
350 True
351 False
352 True
353 False
354 True
355 False
356 True
357 False
358 True
359 False
360 True
361 False
362 True
363 False
364 True
365 False
366 True
367 False
368 True
369 False
370 True
371 False
372 True
373 False
374 True
375 False
376 True
377 False
378 True
379 False
380 True
381 False
382 True
383 False

384 True
385 False
386 True
387 False
388 True
389 False
390 True
391 False
392 True
393 False
394 True
395 False
396 True
397 False
398 True
399 False
400 True
401 False
402 True
403 False
404 True
405 False
406 True
407 False
408 True
409 False
410 True
411 False
412 True
413 False
414 True
415 False
416 True
417 False
418 True
419 False
420 True
421 False
422 True
423 False
424 True
425 False
426 True
427 False
428 True
429 False
430 True
431 False

```
432 True
433 False
434 True
435 False
436 True
437 False
438 True
439 False
440 True
441 False
442 True
443 False
444 True
445 False
446 True
447 False
448 True
449 False
450 True
451 False
452 True
453 False
454 True
455 False
456 True
457 False
458 True
459 False
460 True
461 False
462 True
463 False
464 True
465 False
466 True
467 False
468 True
469 False
470 True
471 False
472 True
473 False
474 True
475 False
476 True
477 False
478 True
479 False
```

480 True
481 False
482 True
483 False
484 True
485 False
486 True
487 False
488 True
489 False
490 True
491 False
492 True
493 False
494 True
495 False
496 True
497 False
498 True
499 False
500 True
501 False
502 True
503 False
504 True
505 False
506 True
507 False
508 True
509 False
510 True
511 False
512 True
513 False
514 True
515 False
516 True
517 False
518 True
519 False
520 True
521 False
522 True
523 False
524 True
525 False
526 True
527 False

528 True
529 False
530 True
531 False
532 True
533 False
534 True
535 False
536 True
537 False
538 True
539 False
540 True
541 False
542 True
543 False
544 True
545 False
546 True
547 False
548 True
549 False
550 True
551 False
552 True
553 False
554 True
555 False
556 True
557 False
558 True
559 False
560 True
561 False
562 True
563 False
564 True
565 False
566 True
567 False
568 True
569 False
570 True
571 False
572 True
573 False
574 True
575 False

576 True
577 False
578 True
579 False
580 True
581 False
582 True
583 False
584 True
585 False
586 True
587 False
588 True
589 False
590 True
591 False
592 True
593 False
594 True
595 False
596 True
597 False
598 True
599 False
600 True
601 False
602 True
603 False
604 True
605 False
606 True
607 False
608 True
609 False
610 True
611 False
612 True
613 False
614 True
615 False
616 True
617 False
618 True
619 False
620 True
621 False
622 True
623 False

624 True
625 False
626 True
627 False
628 True
629 False
630 True
631 False
632 True
633 False
634 True
635 False
636 True
637 False
638 True
639 False
640 True
641 False
642 True
643 False
644 True
645 False
646 True
647 False
648 True
649 False
650 True
651 False
652 True
653 False
654 True
655 False
656 True
657 False
658 True
659 False
660 True
661 False
662 True
663 False
664 True
665 False
666 True
667 False
668 True
669 False
670 True
671 False

672 True
673 False
674 True
675 False
676 True
677 False
678 True
679 False
680 True
681 False
682 True
683 False
684 True
685 False
686 True
687 False
688 True
689 False
690 True
691 False
692 True
693 False
694 True
695 False
696 True
697 False
698 True
699 False
700 True
701 False
702 True
703 False
704 True
705 False
706 True
707 False
708 True
709 False
710 True
711 False
712 True
713 False
714 True
715 False
716 True
717 False
718 True
719 False

720 True
721 False
722 True
723 False
724 True
725 False
726 True
727 False
728 True
729 False
730 True
731 False
732 True
733 False
734 True
735 False
736 True
737 False
738 True
739 False
740 True
741 False
742 True
743 False
744 True
745 False
746 True
747 False
748 True
749 False
750 True
751 False
752 True
753 False
754 True
755 False
756 True
757 False
758 True
759 False
760 True
761 False
762 True
763 False
764 True
765 False
766 True
767 False

768 True
769 False
770 True
771 False
772 True
773 False
774 True
775 False
776 True
777 False
778 True
779 False
780 True
781 False
782 True
783 False
784 True
785 False
786 True
787 False
788 True
789 False
790 True
791 False
792 True
793 False
794 True
795 False
796 True
797 False
798 True
799 False
800 True
801 False
802 True
803 False
804 True
805 False
806 True
807 False
808 True
809 False
810 True
811 False
812 True
813 False
814 True
815 False

816 True
817 False
818 True
819 False
820 True
821 False
822 True
823 False
824 True
825 False
826 True
827 False
828 True
829 False
830 True
831 False
832 True
833 False
834 True
835 False
836 True
837 False
838 True
839 False
840 True
841 False
842 True
843 False
844 True
845 False
846 True
847 False
848 True
849 False
850 True
851 False
852 True
853 False
854 True
855 False
856 True
857 False
858 True
859 False
860 True
861 False
862 True
863 False

```
864 True
865 False
866 True
867 False
868 True
869 False
870 True
871 False
872 True
873 False
874 True
875 False
876 True
877 False
878 True
879 False
880 True
881 False
882 True
883 False
884 True
885 False
886 True
887 False
888 True
889 False
890 True
891 False
892 True
893 False
894 True
895 False
896 True
897 False
898 True
899 False
900 True
901 False
902 True
903 False
904 True
905 False
906 True
907 False
908 True
909 False
910 True
911 False
```

912 True
913 False
914 True
915 False
916 True
917 False
918 True
919 False
920 True
921 False
922 True
923 False
924 True
925 False
926 True
927 False
928 True
929 False
930 True
931 False
932 True
933 False
934 True
935 False
936 True
937 False
938 True
939 False
940 True
941 False
942 True
943 False
944 True
945 False
946 True
947 False
948 True
949 False
950 True
951 False
952 True
953 False
954 True
955 False
956 True
957 False
958 True
959 False

960 True
961 False
962 True
963 False
964 True
965 False
966 True
967 False
968 True
969 False
970 True
971 False
972 True
973 False
974 True
975 False
976 True
977 False
978 True
979 False
980 True
981 False
982 True
983 False
984 True
985 False
986 True
987 False
988 True
989 False
990 True
991 False
992 True
993 False
994 True
995 False
996 True
997 False
998 True
999 False
1000 True
1001 False
1002 True
1003 False
1004 True
1005 False
1006 True
1007 False

1008 True
1009 False
1010 True
1011 False
1012 True
1013 False
1014 True
1015 False
1016 True
1017 False
1018 True
1019 False
1020 True
1021 False
1022 True
1023 False
1024 True
1025 False
1026 True
1027 False
1028 True
1029 False
1030 True
1031 False
1032 True
1033 False
1034 True
1035 False
1036 True
1037 False
1038 True
1039 False
1040 True
1041 False
1042 True
1043 False
1044 True
1045 False
1046 True
1047 False
1048 True
1049 False
1050 True
1051 False
1052 True
1053 False
1054 True
1055 False

1056 True
1057 False
1058 True
1059 False
1060 True
1061 False
1062 True
1063 False
1064 True
1065 False
1066 True
1067 False
1068 True
1069 False
1070 True
1071 False
1072 True
1073 False
1074 True
1075 False
1076 True
1077 False
1078 True
1079 False
1080 True
1081 False
1082 True
1083 False
1084 True
1085 False
1086 True
1087 False
1088 True
1089 False
1090 True
1091 False
1092 True
1093 False
1094 True
1095 False
1096 True
1097 False
1098 True
1099 False
1100 True
1101 False
1102 True
1103 False

1104 True
1105 False
1106 True
1107 False
1108 True
1109 False
1110 True
1111 False
1112 True
1113 False
1114 True
1115 False
1116 True
1117 False
1118 True
1119 False
1120 True
1121 False
1122 True
1123 False
1124 True
1125 False
1126 True
1127 False
1128 True
1129 False
1130 True
1131 False
1132 True
1133 False
1134 True
1135 False
1136 True
1137 False
1138 True
1139 False
1140 True
1141 False
1142 True
1143 False
1144 True
1145 False
1146 True
1147 False
1148 True
1149 False
1150 True
1151 False

1152 True
1153 False
1154 True
1155 False
1156 True
1157 False
1158 True
1159 False
1160 True
1161 False
1162 True
1163 False
1164 True
1165 False
1166 True
1167 False
1168 True
1169 False
1170 True
1171 False
1172 True
1173 False
1174 True
1175 False
1176 True
1177 False
1178 True
1179 False
1180 True
1181 False
1182 True
1183 False
1184 True
1185 False
1186 True
1187 False
1188 True
1189 False
1190 True
1191 False
1192 True
1193 False
1194 True
1195 False
1196 True
1197 False
1198 True
1199 False

```
1200 True
1201 False
1202 True
1203 False
1204 True
1205 False
1206 True
```

```
---------------------------------------------------------------------------

KeyboardInterrupt                         Traceback (most recent call last)

<ipython-input-67-db2cf98c32ce> in <module>()
  1 for x in NN:
----> 2     print(x,x%Integer(2)==Integer(0))


/opt/sagemath-9.1/local/lib/python3.7/site-packages/ipykernel/iostream.py in write(sel
364                 parent=self.parent_header, ident=self.topic)
365
--> 366     def write(self, string):
367         if self.pub_thread is None:
368             raise ValueError('I/O operation on closed file')


src/cysignals/signals.pyx in cysignals.signals.python_check_interrupt()


KeyboardInterrupt:
```

In [71]:
```
x = 3
print(x)
x = 5
print(x)
y = 4
print(x,y)
y = 2*x + y
print(x,y)
z = x==1
print(x,y,z)
x = 1
print(x,y,z)
```

```
3
5
5 4
```

```
5 14
5 14 False
1 14 False
```

```
In [73]: i = 0
         while i <= 5:
             print(i^2)
             print(i % 2 == 0)
             i = i+1
         print()
         print(i)
```

```
0
True
1
False
4
True
9
False
16
True
25
False

6
```

```
In [16]: n = 14
         print(n)
         while not n == 1:
             if n%2 == 0:
                 n //= 2
             else:
                 n = 3*n + 1
             print(n)
```

```
14
7
22
11
34
17
52
26
13
40
20
```

```
10
5
16
8
4
2
1
```

```
In [17]: n = 14
         c = 1
         while not n == 1:
             if n%2 == 0:
                 n //= 2
             else:
                 n = 3*n + 1
             c += 1
         print(c)
```

```
18
```

```
In [19]: for N in range(1,21):
             c = 1
             n = N
             while not n == 1:
                 if n%2 == 0:
                     n //= 2
                 else:
                     n = 3*n + 1
                 c += 1
             print(N,c)
```

```
1 1
2 2
3 8
4 3
5 6
6 9
7 17
8 4
9 20
10 7
11 15
12 10
13 10
14 18
15 18
16 5
```

```
17 13
18 21
19 21
20 8
```

```
In [2]: n = 1230
        for d in range(1,n+1):
            if n%d == 0:
                print(d)
```

```
1
2
3
5
6
10
15
30
41
82
123
205
246
410
615
1230
```

```
In [3]: n = 1230
        c = 0
        for d in range(1,n+1):
            if n%d == 0:
                c += 1
        print(c)
```

```
16
```

```
In [28]: def nombre_de_diviseurs(n):
             c = 0
             for d in range(1,n+1):
                 if n%d == 0:
                     c += 1
             return c

         def est_premier(p):
             return nombre_de_diviseurs(p) == 2
```

```
        def plus_grand_divseur_premier(n):
            for d in range(n,1,-1):
                if n%d == 0:
                    if est_premier(d):
                        return d
```

```
In [29]: for n in range(2,101):
             print("Le plus grand diviseur premier de",n,"est",plus_grand_divseur_premier(n))
```

```
Le plus grand diviseur premier de 2 est 2
Le plus grand diviseur premier de 3 est 3
Le plus grand diviseur premier de 4 est 2
Le plus grand diviseur premier de 5 est 5
Le plus grand diviseur premier de 6 est 3
Le plus grand diviseur premier de 7 est 7
Le plus grand diviseur premier de 8 est 2
Le plus grand diviseur premier de 9 est 3
Le plus grand diviseur premier de 10 est 5
Le plus grand diviseur premier de 11 est 11
Le plus grand diviseur premier de 12 est 3
Le plus grand diviseur premier de 13 est 13
Le plus grand diviseur premier de 14 est 7
Le plus grand diviseur premier de 15 est 5
Le plus grand diviseur premier de 16 est 2
Le plus grand diviseur premier de 17 est 17
Le plus grand diviseur premier de 18 est 3
Le plus grand diviseur premier de 19 est 19
Le plus grand diviseur premier de 20 est 5
Le plus grand diviseur premier de 21 est 7
Le plus grand diviseur premier de 22 est 11
Le plus grand diviseur premier de 23 est 23
Le plus grand diviseur premier de 24 est 3
Le plus grand diviseur premier de 25 est 5
Le plus grand diviseur premier de 26 est 13
Le plus grand diviseur premier de 27 est 3
Le plus grand diviseur premier de 28 est 7
Le plus grand diviseur premier de 29 est 29
Le plus grand diviseur premier de 30 est 5
Le plus grand diviseur premier de 31 est 31
Le plus grand diviseur premier de 32 est 2
Le plus grand diviseur premier de 33 est 11
Le plus grand diviseur premier de 34 est 17
Le plus grand diviseur premier de 35 est 7
Le plus grand diviseur premier de 36 est 3
Le plus grand diviseur premier de 37 est 37
Le plus grand diviseur premier de 38 est 19
Le plus grand diviseur premier de 39 est 13
Le plus grand diviseur premier de 40 est 5
```

```
Le plus grand diviseur premier de 41 est 41
Le plus grand diviseur premier de 42 est 7
Le plus grand diviseur premier de 43 est 43
Le plus grand diviseur premier de 44 est 11
Le plus grand diviseur premier de 45 est 5
Le plus grand diviseur premier de 46 est 23
Le plus grand diviseur premier de 47 est 47
Le plus grand diviseur premier de 48 est 3
Le plus grand diviseur premier de 49 est 7
Le plus grand diviseur premier de 50 est 5
Le plus grand diviseur premier de 51 est 17
Le plus grand diviseur premier de 52 est 13
Le plus grand diviseur premier de 53 est 53
Le plus grand diviseur premier de 54 est 3
Le plus grand diviseur premier de 55 est 11
Le plus grand diviseur premier de 56 est 7
Le plus grand diviseur premier de 57 est 19
Le plus grand diviseur premier de 58 est 29
Le plus grand diviseur premier de 59 est 59
Le plus grand diviseur premier de 60 est 5
Le plus grand diviseur premier de 61 est 61
Le plus grand diviseur premier de 62 est 31
Le plus grand diviseur premier de 63 est 7
Le plus grand diviseur premier de 64 est 2
Le plus grand diviseur premier de 65 est 13
Le plus grand diviseur premier de 66 est 11
Le plus grand diviseur premier de 67 est 67
Le plus grand diviseur premier de 68 est 17
Le plus grand diviseur premier de 69 est 23
Le plus grand diviseur premier de 70 est 7
Le plus grand diviseur premier de 71 est 71
Le plus grand diviseur premier de 72 est 3
Le plus grand diviseur premier de 73 est 73
Le plus grand diviseur premier de 74 est 37
Le plus grand diviseur premier de 75 est 5
Le plus grand diviseur premier de 76 est 19
Le plus grand diviseur premier de 77 est 11
Le plus grand diviseur premier de 78 est 13
Le plus grand diviseur premier de 79 est 79
Le plus grand diviseur premier de 80 est 5
Le plus grand diviseur premier de 81 est 3
Le plus grand diviseur premier de 82 est 41
Le plus grand diviseur premier de 83 est 83
Le plus grand diviseur premier de 84 est 7
Le plus grand diviseur premier de 85 est 17
Le plus grand diviseur premier de 86 est 43
Le plus grand diviseur premier de 87 est 29
Le plus grand diviseur premier de 88 est 11
```

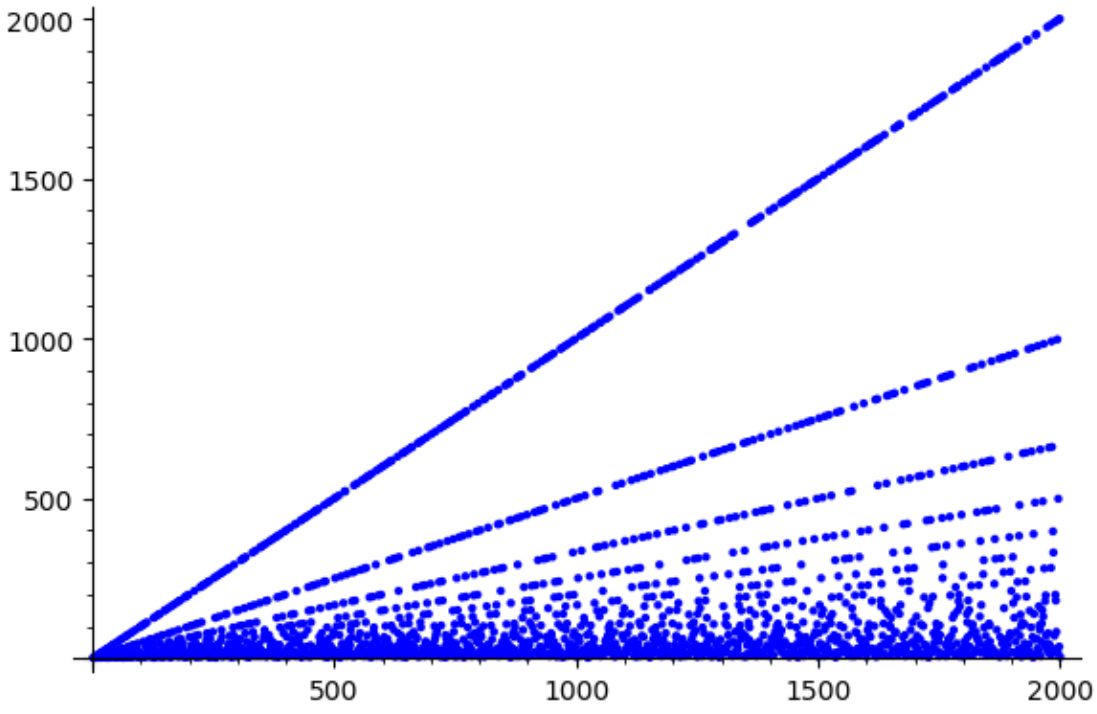
```
Le plus grand diviseur premier de 89 est 89
Le plus grand diviseur premier de 90 est 5
Le plus grand diviseur premier de 91 est 13
Le plus grand diviseur premier de 92 est 23
Le plus grand diviseur premier de 93 est 31
Le plus grand diviseur premier de 94 est 47
Le plus grand diviseur premier de 95 est 19
Le plus grand diviseur premier de 96 est 3
Le plus grand diviseur premier de 97 est 97
Le plus grand diviseur premier de 98 est 7
Le plus grand diviseur premier de 99 est 11
Le plus grand diviseur premier de 100 est 5
```

Par exemple, on peut afficher le plus grand diviseur premier en fonction de n pour n de 2 à 2000 (ça prend quelques secondes car notre programme est très mal optimisé).

Vous apprendrez à la prochaine séance à tracer une telle courbe.

```
In [42]: L = [(RR(n),RR(plus_grand_divseur_premier(n))) for n in range(2,2001)]
         list_plot(L)
```

Out[42]:




```
In [113]: help(range)
```

```
Help on class range in module builtins:

class range(object)
 |  range(stop) -> range object
 |  range(start, stop[, step]) -> range object
 |
 |  Return an object that produces a sequence of integers from start (inclusive)
 |  to stop (exclusive) by step.  range(i, j) produces i, i+1, i+2, ..., j-1.
 |  start defaults to 0, and stop is omitted!  range(4) produces 0, 1, 2, 3.
 |  These are exactly the valid indices for a list of 4 elements.
 |  When step is given, it specifies the increment (or decrement).
 |
 |  Methods defined here:
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, key, /)
 |      Return self[key].
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
```

```
|      Return self<value.
|
|  __ne__(self, value, /)
|      Return self!=value.
|
|  __reduce__(...)
|      Helper for pickle.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(...)
|      Return a reverse iterator.
|
|  count(...)
|      rangeobject.count(value) -> integer -- return number of occurrences of value
|
|  index(...)
|      rangeobject.index(value, [start, [stop]]) -> integer -- return index of value.
|      Raise ValueError if the value is not present.
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
|  ----------------------------------------------------------------------
|  Data descriptors defined here:
|
|  start
|
|  step
|
|  stop
```