

nphullmen_12xa.py

```
1  #!/usr/bin/env python3.6
2  # -*- coding: utf-8 -*-
3  # Série première. En Général Python
4  # Objet : Outil pratique
5  # MusicAtoumic nphullmen_12xa Ce samedi 20 février 2018
6
7  """Pyhton Compulsion Commune
8  Programme évoluant sur une table de travail
9  L'histoire des premiers dividendes aux nombres communs
10  Produisant des multiples à structures uniques
11  Associant les unités en une ligne composée:
12     Structure commune ' Nombre Premier = Premier multiple
13     (Un nombre peut composer plusieurs premiers communs)'
14  La première finalité est alors, liée à une nouvelle fin
15     Un nombre peut avoir plusieurs structures communes
16     Les structures se chevauchent ou se suivent
17  Ce qui détermine la fin, c'est l'(ip.sup & dv.inf)
18     (dv.inf): L'unité terminale proche du carré
19     Soit, elle est un nombre premier
20     Soit, elle multiplie les premiers
21     Soit, elle voit (Batterie à essais:)
22  """
23
24  import time
25
26  time_ = time.time()
27
28
29  def wifi():
30      print('nphullmen_12xa.py En: \n', time.time() - time_)
31
32
33  def zero(z):
34      if z == 1:
35          z = 0
36          print('Premiers (', z, ')ex\n', '[0]')
37          print('{} * {} Types {}&{}'.format(z, z, z % 6, z % 6))
38          wifi()
39      else:
40          wifi()
41
42
43
44  """ Batterie à essais:
45  Ces nombres sur un alignement de mesures:
46  Ligne(#): Entier Dv.inf Break(Type) Temps(s) Multi(|*|)
47  Dv.inf premier est signé (°)
48  Cette signature (²) indique sa divisibilité
49  |°*°|: Les nombres premiers côtés (ip*dv)"""
50  # 93256514687897874652810414 302367527315767°
51  # 913746825159575396321478 865157831052031219°
52  # 987898765456543212321 34584163467²
53  # En Cours ..T loop sqr2/3: 987898765456543212321
54  # 1234567898765432112345678987654321 35922330097087379²
55  # En Cours ..T loop sqr3/6: 1234567898765432112345678987654321
56  # 999888777666555666777888999 107407407407411²
57  # 222557130146747222557130146747 476012969567993²
58  # En Cours ..T loop sqr: 222557130146747222557130146747
```

```

59 # 111222333222111000111222333222111 112109887890112112
60 # En Cours ..T loop sqr: 111222333222111000111222333222111
61 # 5555555551555555555 439445475352
62 # 999888777666555444333222111 3720779432467990087981°
63 # En Cours ..T loop sqr3/3: 999888777666555444333222111
64 # 999888777666555000111222333444555 1304562926286024619985807821°
65 # En Cours ..T loop sqr: 999888777666555000111222333444555
66 # 22255713012225571301 10000000012
67 # 987898765456543212317 2536415380787°
68 # 6532987845121422 1088831307520237°
69 """
70 Chaque nombre a sa finalité
71 Break ? ; En cours
72 Break om1>q1 ; Stop des lecteurs (om1&q1)
73 """
74 # 9325651652810414 8426829197°
75 # 9137468251596321478 1696522141031623°
76 # 98789875654321232 6174367228395077°
77 # 123456788987654320 3556593982
78 # 888777666666777888 9456122782
79 # 22255301467146747 76478699199817°
80 # 9998887776665554 1056842542
81 # 9998887776665 112551292
82 # 11122233110001321 4331042123°
83 # En Cours ..T loop sqr: 11122233110001321
84 # En Cours 5555555551555555 1278297169° break p1>borne (0s)
85 #
86 # :nombre: objet unique
87 nombre = 456789215975346
88 print('Nombre =', nombre, '; typ =', nombre % 6)
89 # :breakComm: printage communs
90 breakComm = 0
91 # :zero: instruction zéro
92 if nombre == 0:
93     zero(breakComm)
94 elif nombre < 0:
95     nombre = abs(nombre)
96 # :carre: service axial
97 carre = int(nombre ** .5)
98 print('carre =', carre)
99 cartyp6 = [carre]
100 borne = [0]
101
102 hautniveau = [] # Liste premiers
103 horscourse = [] # Liste communs ip&dv
104 hautmulti_ = [1] # Mult.hautniveau
105 hautcumul_ = [] # Cum.Mult.hautniveau
106 horspluri_ = [0] # Plus.horscourse
107 horsmulti_ = [0] # Mult.horscourse
108 horscumul_ = [] # Cum.Mult.horscourse
109 cl2tableau = [] # Liste communs dv
110 dvinftable = [] # Liste tranches dv
111 ipsuptable = [] # Liste tranches ip
112 ipsupcumul = [1] # Mult.ipsuptable
113 rapcartime = [0] # Indice niveau dv
114
115 # Lecteurs initiaux
116 p1 = [7] # Intro +6 type 1
117 p5 = [11]
118
119 ed1 = [0] # Mi inf -6 type 1

```

```

120 ed5 = [0]
121 em1 = [0] # Mi inf +6 type 1
122 em5 = [0]
123
124 d1 = [0] # Mi carre -6 type 1
125 d5 = [0]
126 m1 = [0] # Mi carre +6 type 1
127 m5 = [0]
128
129 od1 = [0] # Mi sup -6 type 1
130 od5 = [0]
131 om1 = [0] # Mi sup +6 type 1
132 om5 = [0]
133
134 q1 = [0] # Carre -6 type 1
135 q5 = [0]
136
137 p0 = [0] # Portion vide
138
139
140 def formel15(rng_, r6_, mng_, m6_):
141     """Positionnement des Lecteurs (bas/hauts)
142         P1: Point (+6) Type (1)
143         Position inférieure :ip: Inf.carre
144         Q1: Point (-6) Type (5)
145         Position supérieure :dv: Sup.carre"""
146
147     # Max = rng_ | Axe = mng_
148     dom_ = mng_ // 2 # Coaxe = dom_
149     edm_ = mng_ - dom_ # Report inf = edm_
150     odm_ = mng_ + dom_ # Report sup = odm_
151     e6_ = edm_ % 6 # Type inf e6_
152     o6_ = odm_ % 6 # Type sup o6_
153
154     if r6_ == 1:
155         q1[0] = rng_
156     else:
157         dfo_ = r6_ - 1
158         q1[0] = rng_ - dfo_
159
160     if m6_ == 1:
161         m1[0] = mng_
162         d1[0] = mng_
163     else:
164         dfo_ = m6_ - 1
165         m1[0] = mng_ - dfo_
166         d1[0] = mng_ - dfo_
167     if e6_ == 1:
168         ed1[0] = edm_
169         em1[0] = edm_
170     else:
171         dfo_ = e6_ - 1
172         ed1[0] = edm_ - dfo_
173         em1[0] = edm_ - dfo_
174     if o6_ == 1:
175         od1[0] = odm_
176         om1[0] = odm_
177     else:
178         dfo_ = o6_ - 1
179         od1[0] = odm_ - dfo_
180         om1[0] = odm_ - dfo_

```

```

181
182     # Position P5 (début)
183     p5[0] = p1[0] - 2
184     # Position E5 (bas)
185     ed5[0] = ed1[0] - 2
186     em5[0] = em1[0] - 2
187     # Position M5 (milieu)
188     m5[0] = m1[0] - 2
189     d5[0] = d1[0] - 2
190     # Position O5 (haut)
191     od5[0] = od1[0] - 2
192     om5[0] = om1[0] - 2
193     # Position Q5 (fin)
194     q5[0] = q1[0] - 2
195     print(' f15=0; p1=%s; ed1=%s; q1=%s' % (p1[0], ed1[0], q1[0]))
196     print(' f15=0; d1=%s; od1=%s' % (d1[0], od1[0]))
197
198
199     def compare(c):
200         """Réception et traitement (c)"""
201         if c < 1:
202             return
203         hautmulti_[0] = 1
204         didi = [c]
205         print('intro DIDI', didi)
206         con = 1
207         if c not in horscourse:
208             if carre >= c > 1:
209                 horscourse.append(c)
210                 horscourse.sort()
211                 sqc = int(c ** .501)
212                 # Démultiplications
213                 for dd in range(sqc, max(hautniveau), -1):
214                     con += 1
215                     if not c % dd or not c % con:
216                         # Identifier (dd&con)
217                         if not c % dd:
218                             dddv = c // dd
219                             print('**dd', dd, dddv)
220                         else:
221                             dddv = c // con
222                             dd = con
223                             print('**con', con, dddv)
224                         # :dd: Explore
225                         if dd not in horscourse \
226                             and dd % 6 in (1, 5) \
227                             and dd not in didi:
228                             print('**_____C app(dd)', dd)
229                             didi.append(dd)
230                             didi.append(dddv)
231                             print('DIDI dd', didi)
232                             print('DD DDDV', dd, dddv)
233                             break
234                         # :ddv: Explore
235                         if dddv >= carre:
236                             sqdd = int(dddv ** .4)
237                             noc = 1
238                             for fd in range(sqdd, max(hautniveau), -1):
239                                 noc += 1
240                                 # Condition de fin (noc&fd)
241                                 if noc > fd:

```

```

242         print('boucle: dddv>carre:', noc)
243         break
244
245     elif not c % fd or not c % noc:
246         # Identifieur (fd&noc)
247         if not c % fd:
248             dddv = c // fd
249             print('**fd', fd, dddv, 'sqdd', sqdd)
250         else:
251             dddv = c // noc
252             fd = noc
253             print('**noc', noc, dddv)
254         # :fd: Explore
255         if fd not in horscourse \
256             and fd % 6 in (1, 5) \
257             and fd not in didi:
258             print('**_____C app(fd)', fd, 'sqdd',
sqdd)
259             didi.append(fd)
260             print('DIDI fd', didi)
261             break
262     if dddv not in horscourse \
263         and dddv % 6 in (1, 5) \
264         and dddv <= carre \
265         and dddv not in didi:
266         print('**_____C app(dddv)', dddv)
267         didi.append(dddv)
268     sqd = int(dddv ** .501)
269     if dddv > carre:
270         sqd = int(sqd ** .6)
271         # print('C sqd', sqd)
272     for vv in range(2, sqd):
273         if not dddv % vv:
274             vvd = dddv // vv
275             if vv not in horscourse \
276                 and vv % 6 in (1, 5) \
277                 and vv not in didi:
278                 print('**_____C app(vv)', vv)
279                 didi.append(vv)
280                 break
281             if vvd not in horscourse \
282                 and vvd % 6 in (1, 5) \
283                 and vvd not in didi:
284                 sqv = int(vvd ** .5)
285                 ffd = []
286                 for q in range(2, sqv):
287                     if not vvd % q and vvd not in didi:
288                         veq = vvd // q
289                         ffd.append(q)
290                         ffd.append(veq)
291                         print('** veq =', veq, q)
292                         break
293                 if ffd:
294                     print('** ffd =', ffd)
295                     for f in ffd:
296                         vvd = f
297                         if vvd not in dvinftable and vvd
not in didi:
298                             print('**_____C app(vvd1)',
vvd)
299                             didi.append(vvd)

```

```

300                                     break
301     else:
302         if vvd not in dvinftable:
303             dvinftable.append(vvd)
304             print('**          C app(vvd0)', vvd)
305             didi.append(vvd)
306             break
307     didi.sort()
308     print('fin DIDI', didi)
309     # Priorité :didi:
310     break
311
312 for di in sorted(didi):
313     c = di
314     for haut in hautniveau[1:]:
315         if not c % haut:
316             break
317     else:
318         print('****DIDIDI', c)
319         if c <= carre:
320             hautniveau.append(c)
321         elif c not in dvinftable:
322             print('****DIDIDI dvinftable', c)
323             dvinftable.append(c)
324     # Production haut niveau
325     for hha in hautniveau:
326         hautmulti_[0] *= hha
327         if hautmulti_[0] not in hautcumul_:
328             hautcumul_.append(hautmulti_[0])
329
330 if hautmulti_[0] != 0:
331     print(' Oniveau =', sorted(hautniveau))
332     print(' Omulti_ =', hautmulti_[0])
333     for c1 in sorted(hautniveau):
334         for c2 in horscourse:
335             c12 = c1 * c2
336             # Séparation :carre:
337             if not nombre % c12 \
338                 and c12 not in horscourse:
339                 if c12 <= carre:
340                     horscourse.append(c12)
341                     horscourse.sort()
342                 elif c12 not in c12tableau:
343                     c12tableau.append(c12)
344                     c12tableau.sort()
345     print('C Orscourse =', max(horscourse))
346     if c12tableau:
347         for ctab in c12tableau:
348             iptab = nombre // ctab
349             # :ctab: Occurence :c12:
350             if iptab not in horscourse:
351                 for hh in hautniveau[1:]:
352                     if not iptab % hh:
353                         break
354             else:
355                 print('C c12 compare: iptab =', iptab)
356                 compare(iptab)
357
358 if hautmulti_[0] > 0:
359     horspluri_[0] = 0
360     horsmulti_[0] = 1

```

```

361         # Production hors course
362         for hho in horscourse:
363             if horspluri_[0] <= nombre:
364                 horspluri_[0] += hho
365                 horsmulti_[0] *= hho
366             horscumul_.append(max(horscourse))
367             cartyp6[0] = int((nombre // max(horscourse)) ** .5)
368             brn = int(cartyp6[0] ** .9478)
369             borne[0] = brn
370
371     # Terme commun
372     if max(horscourse) <= carre:
373         if hautmulti_[0]:
374             hdv = hautmulti_[0] * hautniveau[1]
375             # :hdv: Son.carre(dv) par Min.hautniveau
376             if hdv >= carre and not nombre % hdv:
377                 hip = nombre // hdv
378                 # :hip: Son(ip) du Son.carre(dv)
379                 if hip not in horscourse:
380                     print('Cm compare: hip =', hip)
381                     compare(hip)
382
383     if hautmulti_[0]:
384         # Termes racine
385         if hautmulti_[0] == nombre:
386             hautmulti_[0] = 0
387             print('E BREAK multi==', nombre)
388         elif hautmulti_[0] * hautniveau[1] >= carre \
389              and hautmulti_[0] == max(hautniveau):
390             hautmulti_[0] = 0
391             print('E BREAK multi*min(c)', hautmulti_)
392         elif hautmulti_[0] * hautniveau[1] >= nombre:
393             hautmulti_[0] = 0
394             print('E BREAK multi*min(n)', hautmulti_)
395         elif (nombre // hautmulti_[0]) <= max(hautniveau):
396             hautmulti_[0] = 0
397             print('E BREAK n/multi <= max(haut)', hautmulti_)
398         elif max(horscourse) == nombre:
399             hautmulti_[0] = 0
400             print('E BREAK hors(nombre)', nombre)
401         elif max(horscourse) ** 2 > nombre:
402             hautmulti_[0] = 0
403             print('E BREAK hors**2', nombre)
404         elif hautmulti_[0]:
405             # Différentiel comparatif
406             dvinf = nombre // max(horscourse)
407             ipsup = dvinf // max(horscourse)
408             sqinf = int(dvinf ** .501)
409             if max(horscourse) not in ipsuptable:
410                 ipsuptable.append(max(horscourse))
411                 ipsupcumul[0] *= max(horscourse)
412                 print('E ipsuptable =', ipsuptable)
413                 if ipsupcumul[0] > carre:
414                     print('E ipsupcumul =',
415                           nombre // ipsupcumul[0])
416                 print('E dvinf//max(hors) ipsup =', ipsup)
417                 print('*****')
418                 print('*****')
419             if dvinf not in dvinftable:
420                 dvinftable.append(dvinf)
421             if ipsuptable:

```

```

422 rapcartime[0] = 0
423 rapips = max(ipsuptable)
424 rapdvs = min(dvinftable)
425 sqrdvs = int(rapdvs ** .5)
426 print('.T sqrdvs =', sqrdvs)
427 rapnbr = nombre // (rapdvs // rapips)
428 print('.T rapnbr =', rapnbr)
429 if rapnbr > max(ipsuptable):
430     print('..T rapips =', rapips)
431     print('..T rapdvs =', rapdvs)
432     raprap = rapdvs // rapips
433     if raprap >= carre:
434         print('..T rapsup =', raprap)
435         rapcar = raprap // carre
436     else:
437         print('..T rapinf =', raprap)
438         rapcar = carre // raprap
439     print('..T rapcar =', rapcar)
440     if rapcar > sqrdvs:
441         print('.._T loop sqrdvs', sqrdvs)
442         rapcartime[0] = sqrdvs
443     elif max(hautniveau) > rapcar > 1:
444         if rapnbr > sqrdvs:
445             rapcartime[0] =
446                 print('.._T loop rapnbr',
447                       rapcartime[0])
448     elif rapcar < 2:
449         sqrap = int(raprap ** .5)
450         if sqrap > max(hautniveau):
451             rapcartime[0] = sqrap
452             print('.. .T loop sqrap',
453                   sqrap)
454         else:
455             rapcartime[0] = raprap
456             print('.. .T loop raprap')
457     print('.._T rapcartime',
458           rapcartime[0])
459     else:
460         if raprap > carre:
461             if rapcar < int(sqrdvs ** .5):
462                 rapcartime[0] = int(sqrdvs
463 ** .6)
464                 print('..T loop sqr',
465                       rapcartime[0])
466             else:
467                 rapcartime[0] = int(sqrdvs
468 ** .948)
469                 print('..T loop 94',
470                       rapcartime[0])
471             else:
472                 rapcartime[0] = int(sqrdvs **
473 .987)
474                 print('..T else r<c', )
475     print('.._T else rapcartime[0]',
476           rapcartime[0])
477     else:
478         rapcartime[0] = rapnbr
479         print('...T for/else rapcartime',
480               rapcartime[0])

```



```

534                                     'E (compare): hodv =',
hodv)
535                                     compare(hodv)
536                                     break
537                                     elif hoip > sqinf:
538                                     break
539
540
541 """Bas niveau des premiers:
542 Création du haut niveau des nombres premiers
543 Lié au bas niveau des nombres premiers
544 Composition des hors course"""
545 basniveau = [1, 2, 3, 5]
546 for i in range(1, 7):
547     if not nombre % i and i <= carre:
548         def rang(ti, i_):
549             if ti == 1: # Nombre Premier(basniveau)
550                 hautniveau.append(i_)
551                 horscourse.append(i_)
552             else: # Nombre Commun(basniveau)
553                 horscourse.append(i_)
554
555
556         if i in basniveau:
557             rang(1, i)
558         else:
559             rang(2, i)
560 if horscourse:
561     # Traitement basique
562     for h in horscourse:
563         for o in horscourse:
564             oh = h * o
565             if not nombre % oh and oh not in horscourse:
566                 if oh > carre and not nombre % (nombre // oh):
567                     if (nombre // oh) not in horscourse:
568                         print('Basic compare: n>carre =', oh)
569                         compare(nombre // oh)
570                         break
571                 elif oh <= carre:
572                     print('Basic compare: n<carre =', oh)
573                     compare(oh)
574                     break
575             dvhors = nombre // max(horscourse)
576             borne[0] = int(dvhors ** .5)
577             print('Basic borne[0]', borne[0])
578
579 """Borne & cartyp6
580 Borne : Limite en pleine forêt
581 Cartyp6 : Limite en plein ciel"""
582 if not borne:
583     borne = cartyp6
584 print('cartyp6=', cartyp6[0], ';borne=', borne[0])
585 print('horscourse', horscourse)
586 if nombre != 0:
587     mib0 = borne[0] // 2
588     formel5(borne[0], borne[0] % 6, mib0, mib0 % 6)
589
590 """Hauts niveaux premiers: cartyp6
591 Itérer dans l'alignement des nombres premiers
592 Condition évolutive des lecteurs"""
593 while pl[0] <= cartyp6[0]:

```

```

594
595     """if p1[0] > int(borne[0]):
596         print('    W break p1>borne', p1[0], int(borne[0]))
597         break"""
598 if hautmulti_[0] == 0:
599     print('    W break hautmulti_', hautmulti_)
600     break
601 elif om1[0] > q1[0]:
602     print('    W break om1>q1', om1[0], q1[0])
603     break
604
605 # Lecteurs en lecture
606 # Lecteurs P: Début (P)
607 if od1[0] == 0: # Révision nombre
608     print('    W OD1', p1[0], int(borne[0] ** .916))
609     print('    W n%OD1', nombre % p1[0])
610 if not nombre % p1[0] and p1[0] not in hautniveau \
611     and p1[0] not in horscourse:
612     print('    W ___P1', p1[0])
613     compare(p1[0])
614 if not nombre % p5[0] and p5[0] not in hautniveau \
615     and p5[0] not in horscourse:
616     print('    W ___P5', p5[0])
617     compare(p5[0])
618 # Lecteurs
619 if not nombre % q1[0] and q1[0] not in hautniveau \
620     and q1[0] not in horscourse:
621     print('    W ___Q1', q1[0])
622     compare(q1[0])
623 if not nombre % q5[0] and q5[0] not in hautniveau \
624     and q5[0] not in horscourse:
625     print('    W ___Q5', q5[0])
626     compare(q5[0])
627 # Lecteurs M: Entre carre (Q) et début (P)
628 if not nombre % m1[0] and m1[0] not in hautniveau \
629     and m1[0] not in horscourse:
630     print('    W ___M1', m1[0])
631     compare(m1[0])
632 if not nombre % m5[0] and m5[0] not in hautniveau \
633     and m5[0] not in horscourse:
634     print('    W ___M5', m5[0])
635     compare(m5[0])
636 # Lecteurs
637 if not nombre % d1[0] and d1[0] not in hautniveau \
638     and d1[0] not in horscourse:
639     print('    W ___D1', d1[0])
640     compare(d1[0])
641 if not nombre % d5[0] and d5[0] not in hautniveau \
642     and d5[0] not in horscourse:
643     print('    W ___D5', d5[0])
644     compare(d5[0])
645 # Lecteurs E: Entre début (P) et mi course (M)
646 if not nombre % ed1[0] and ed1[0] not in hautniveau \
647     and ed1[0] not in horscourse:
648     print('    W ___ED1', ed1[0])
649     compare(ed1[0])
650 if not nombre % ed5[0] and ed5[0] not in hautniveau \
651     and ed5[0] not in horscourse:
652     print('    W ___ED5', ed5[0])
653     compare(ed5[0])
654 if not nombre % em1[0] and em1[0] not in hautniveau \

```

```

655         and em1[0] not in horscourse:
656     print('  W___EM1', em1[0])
657     compare(em1[0])
658     if not nombre % em5[0] and em5[0] not in hautniveau \
659         and em5[0] not in horscourse:
660     print('  W___EM5', em5[0])
661     compare(em5[0])
662     # Lecteurs 0: Entre carre (Q) et mi course (M)
663     if not nombre % od1[0] and od1[0] not in hautniveau \
664         and od1[0] not in horscourse:
665     print('  W___OD1', od1[0])
666     compare(od1[0])
667     if not nombre % od5[0] and od5[0] not in hautniveau \
668         and od5[0] not in horscourse:
669     print('  W___OD5', od5[0])
670     compare(od5[0])
671     if not nombre % om1[0] and om1[0] not in hautniveau \
672         and om1[0] not in horscourse:
673     print('  W___OM1', om1[0])
674     compare(om1[0])
675     if not nombre % om5[0] and om5[0] not in hautniveau \
676         and om5[0] not in horscourse:
677     print('  W___OM5', om5[0])
678     compare(om5[0])
679
680     p1[0] += 6
681     p5[0] += 6
682     q1[0] -= 6
683     q5[0] -= 6
684     m1[0] += 6
685     m5[0] += 6
686     d1[0] -= 6
687     d5[0] -= 6
688     # 3ème niveau de profondeur
689     ed1[0] -= 6
690     ed5[0] -= 6
691     em1[0] += 6
692     em5[0] += 6
693     od1[0] -= 6
694     od5[0] -= 6
695     om1[0] += 6
696     om5[0] += 6
697
698     if nombre != 0:
699         """Désigne la communauté:
700         Les nombres premiers associés :hautniveau:
701         Produisent la communauté des multiples communs"""
702
703         hi = [u for u in horscourse if u <= carre]
704         hi = len(hi)
705         hautniveau.sort()
706         """print('car %s car6 %s bor %s \n D1 %s OM1 %s Q1=%s'
707             % (carre, cartyp6, borne, d1[0], om1[0], q1[0]))"""
708         print('Premiers (', hi, ')ex\n', hautniveau, carre)
709         for i in horscourse[:hi]:
710             if breakComm == 0:
711                 iii = -1
712                 while horscourse[iii] >= carre:
713                     iii -= 1
714                 pass
715             else:

```

```
716         ha = horscourse[iii]
717     hu = horscourse[0]
718     print("Duos (inf)*(sup)|(sup)*(inf) \n {} * {} "
719           "Types {}&{}".format(hu, nombre // hu, hu % 6,
(nombre // hu) % 6))
720     print(
721         ' {} * {} Types {}&{}'.format(
722             ha, nombre // ha, ha % 6, (nombre // ha) % 6))
723     break
724 else:
725     print(
726         '{} * {} Types {}&{}'.format(
727             i, nombre // i, i % 6, (nombre // i) % 6))
728     wifi()
729
730 #
731
```