

😎 Ciclos o bucles

for

Un ciclo **for** es un ciclo que repite el bloque de instrucciones un número predeterminado de veces. El bloque de instrucciones que se repite se suele llamar cuerpo del ciclo y cada repetición se suele llamar iteración. La sintaxis de un ciclo for es la siguiente:

for variable in elemento_iterable (lista, cadena, range, etc.):

cuerpo del ciclo

Ejemplos de ciclos for

1. En Python no es necesario definir la variable de control antes del ciclo, aunque se puede utilizar como variable de control una variable ya definida en el programa. El cuerpo del ciclo se ejecuta tantas veces como elementos tenga el elemento que se está recorriendo por ejemplo: elementos de una lista o de un range(), caracteres de una cadena, etc.). Por ejemplo:

```
In [1]: ▶ print("Comienzo")
for i in [0, 1, 2]:
    print("Hola ", end="")
print()
print("Final")
```

Comienzo
Hola Hola Hola
Final

```
1 print("Comienzo")
2 for i in [0, 1, 2]:
3     print("Hola ", end="")
4 print()
⇒ 5 print("Final")
```

Print output (drag lower right corner to resize)

Comienzo
Hola Hola Hola
Final

Frames

Objects

Global frame
i 2

2. Si la lista está vacía, el ciclo no se ejecuta ninguna vez. Por ejemplo:

```
In [2]: ▶ print("Comienzo")
for i in []:
    print("Hola ", end="")
print()
print("Final")
```

Comienzo

Final

3. Si la variable de control no se va a utilizar en el cuerpo del ciclo, como en los ejemplos anteriores, se puede utilizar el guion (__) en vez de un nombre de variable. Esta notación no tiene ninguna consecuencia con respecto al funcionamiento del programa, pero sirve de ayuda a la persona que esté leyendo el código fuente, que así sabe que los valores no se van a utilizar. Por ejemplo:

```
In [3]: ▶ print("Comienzo")
for _ in [0, 1, 2]:
    print("Hola ", end="")
print()
print("Final")
```

Comienzo
Hola Hola Hola
Final

El indicador puede incluir cualquier número de guiones bajos (_, __, ___, etc). Los más utilizados son uno o dos guiones (_ o __).

4. En el ejemplo anterior, la variable de control "_" no se utilizaba en el bloque de instrucciones, pero en muchos casos sí que se utiliza una variable. Cuando se utiliza, hay que tener en cuenta que la variable de control va tomando los valores del elemento que se recorre. Por

ejemplo:

```
In [4]: ▶ print("Comienzo")
for i in [3, 4, 5]:
    print(f"Hola. Ahora i vale {i} y su cuadrado es {i ** 2}")
print("Final")
```

```
Comienzo
Hola. Ahora i vale 3 y su cuadrado es 9
Hola. Ahora i vale 4 y su cuadrado es 16
Hola. Ahora i vale 5 y su cuadrado es 25
Final
```

```
1 print("Comienzo")
2 for i in [3, 4, 5]:
3     print(f"Hola. Ahora i vale {i} y su cuadrado es {i ** 2}")
⇒ 4 print("Final")
```

Print output (drag lower right corner to resize)

```
Comienzo
Hola. Ahora i vale 3 y su cuadrado es 9
Hola. Ahora i vale 4 y su cuadrado es 16
Hola. Ahora i vale 5 y su cuadrado es 25
Final
```

Frames Objects

Global frame
i | 5

5. La lista puede contener cualquier tipo de elementos, no sólo números. El ciclo se repetirá siempre tantas veces como elementos tenga la lista y la variable irá tomando los valores de uno en uno. Por ejemplo:

```
In [5]: ▶ print("Comienzo")
for i in ["Pepe", "Pepa", 47]:
    print(f"Hola. Ahora i vale {i}")
print("Final")
```

```
Comienzo
Hola. Ahora i vale Pepe
Hola. Ahora i vale Pepa
Hola. Ahora i vale 47
Final
```

6. La costumbre más extendida es utilizar la letra i como nombre de la variable de control, pero se puede utilizar cualquier otro nombre válido. Por ejemplo:

```
In [6]: ▶ print("Comienzo")
for numero in [0, 1, 2, 3]:
    print(f"{numero} * {numero} = {numero ** 2}")
print("Final")
```

```
Comienzo
0 * 0 = 0
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
Final
```

7. La variable de control puede ser una variable empleada antes del ciclo. El valor que tuviese la variable no afecta a la ejecución del ciclo, pero cuando termina el ciclo, la variable de control conserva el último valor asignado:

```
In [7]: ▶ i = 10
print(f"El ciclo no ha comenzado. Ahora i vale {i}")
for i in [0, 1, 2, 3, 4]:
    print(f"{i} * {i} = {i ** 2}")
print(f"El ciclo ha terminado. Ahora i vale {i}")
```

```
El ciclo no ha comenzado. Ahora i vale 10
0 * 0 = 0
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
4 * 4 = 16
El ciclo ha terminado. Ahora i vale 4
```

8. Cuando se escriben 2 o más ciclos seguidos, la costumbre es utilizar el mismo nombre de variable puesto que cada ciclo establece los valores de la variable sin importar los valores anteriores:

```
In [8]: for i in [0, 1, 2]:
        print(f"{i} * {i} = {i ** 2}")
        print()

0 * 0 = 0
1 * 1 = 1
2 * 2 = 4
```

```
In [9]: for i in [0, 1, 2, 3]:
        print(f"{i} * {i} * {i} = {i ** 3}")

0 * 0 * 0 = 0
1 * 1 * 1 = 1
2 * 2 * 2 = 8
3 * 3 * 3 = 27
```

8. Cuando se escriben 2 o más ciclos seguidos, la costumbre es utilizar el mismo nombre de variable puesto que cada ciclo establece los valores de la variable sin importar los valores anteriores:

```
In [10]: for i in "AMIGOS":
        print(f"Dame una {i}")
        print("¡AMIGOS!")

Dame una A
Dame una M
Dame una I
Dame una G
Dame una O
Dame una S
¡AMIGOS!
```

```
1 for i in "AMIGOS":
2     print(f"Dame una {i}")
⇒ 3 print("¡AMIGOS!")
```

Print output (drag lower right corner to resize)

```
Dame una A
Dame una M
Dame una I
Dame una G
Dame una O
Dame una S
¡AMIGOS!
```

Frames

Objects

```
Global frame
i "S"
```

10. En los ejemplos anteriores se ha utilizado una lista para facilitar la comprensión del funcionamiento de los ciclos pero, -si es posible hacerlo-, se recomienda utilizar tipos `range()`, entre otros motivos porque durante la ejecución del programa ocupan menos memoria. El siguiente programa es equivalente al programa del ejemplo anterior:

```
In [11]: print("Comienzo")
        for i in range(3):
            print("Hola ", end="")
        print()
        print("Final")

Comienzo
Hola Hola Hola
Final
```

11. Otra de las ventajas de utilizar tipos `range()` es que el argumento del tipo `range()` controla el número de veces que se ejecuta el ciclo. En el ejemplo anterior era suficiente cambiar el argumento para que el programa salude muchas más veces.

```
In [12]: print("Comienzo")
        for i in range(0,10,2):
            print("Hola ", end="")
        print()
        print("Final")

Comienzo
Hola Hola Hola Hola Hola
Final
```

Esto permite que el número de iteraciones dependa del desarrollo del programa.

12. En el ejemplo siguiente es el usuario quien decide cuántas veces se ejecuta el ciclo:

```
In [13]: > veces = int(input("¿Cuántas veces quiere que le salude? "))
for i in range(veces):
    print("Hola ", end="")
print()
print("Adiós")
```

```
¿Cuántas veces quiere que le salude? 3
Hola Hola Hola
Adiós
```

Contadores, variables testigo (o bandera) y acumuladores

En muchos programas se necesitan variables que cuenten cuántas veces ha ocurrido algo (contadores) o que indiquen si simplemente ha ocurrido algo (testigo) o que acumulen valores (acumuladores).

Contador

Se entiende por contador una variable que lleva la cuenta del número de veces que se ha cumplido una condición. El ejemplo siguiente es un ejemplo de programa con contador (en este caso, la variable que hace de contador es la variable cuenta):

```
In [14]: > print("Comienzo")
cuenta = 0
for i in range(1, 6):
    if i % 2 == 0:
        cuenta = cuenta + 1
# cuenta++ no se puede usar
print(f"Desde 1 hasta 5 hay {cuenta} múltiplos de 2")
```

```
Comienzo
Desde 1 hasta 5 hay 2 múltiplos de 2
```

Importante:

- En cada iteración, el programa comprueba si *i* es múltiplo de 2.
- El contador se modifica sólo si la variable de control *i* es múltiplo de 2.
- El contador va aumentando de uno en uno.
- Antes del ciclo se debe inicializar el contador (en este caso, 0)

Variables testigo o bandera

Se entiende por testigo una variable booleana que indica simplemente si una condición se ha cumplido o no. Es un caso particular de contador, pero se suele hacer con variables lógicas en vez de numéricas (en este caso, la variable que hace de testigo es la variable encontrado):

```
In [15]: > print("Comienzo")
encontrado = False
for i in range(1, 6):
    if i % 2 == 0:
        encontrado = True
if encontrado:
    print(f"Entre 1 y 5 hay al menos un múltiplo de 2.")
else:
    print(f"Entre 1 y 5 no hay ningún múltiplo de 2.")
```

```
Comienzo
Entre 1 y 5 no hay ningún múltiplo de 2.
```

Importante:

- En cada iteración, el programa comprueba si *i* es múltiplo de 2.
- El testigo se modifica la primera vez que la variable de control *i* es múltiplo de 2.
- El testigo no cambia una vez ha cambiado.
- Antes del ciclo se debe dar un valor inicial al testigo (en este caso, False)

Acumulador

Se entiende por acumulador una variable que acumula el resultado de una operación. El ejemplo siguiente es un ejemplo de programa con acumulador (en este caso, la variable que hace de acumulador es la variable suma):

```
In [16]: ▶ print("Comienzo")
suma = 0
for i in [1, 2, 3, 4]:
    suma = suma + i
print(f"La suma de los números de 1 a 4 es {suma}")
```

Comienzo
La suma de los números de 1 a 4 es 10

for con else

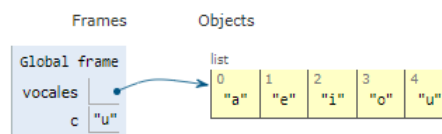
```
In [17]: ▶ vocales = ['a', 'e', 'i', 'o', 'u']
for c in vocales:
    if c == 'b':
        break
else:
    print('No se ha encontrado el carácter b')
```

No se ha encontrado el carácter b

```
1  vocales = ['a', 'e', 'i', 'o', 'u']
2  for c in vocales:
3      if c == 'b':
4          break
5  else:
⇒ 6      print('No se ha encontrado el carácter b')
```

Print output (drag lower right corner to resize)

No se ha encontrado el carácter b



Ciclos anidados

Se habla de ciclos anidados cuando un ciclo se encuentra en el bloque de instrucciones de otro bloque. Al ciclo que se encuentra dentro del otro se le puede denominar ciclo interior o ciclo interno. El otro ciclo sería el ciclo exterior o ciclo externo. Los ciclos pueden tener cualquier nivel de anidamiento (un ciclo dentro de otro ciclo dentro de un tercero, etc.).

Ciclos anidados (variables independientes)

Se dice que las variables de los ciclos son independientes cuando los valores que toma la variable de control del ciclo interno no dependen del valor de la variable de control del ciclo externo. Por ejemplo:

```
In [18]: ▶ for i in [0, 1, 2]:
          for j in [0, 1]:
              print(f"i vale {i} y j vale {j}")
```

```
i vale 0 y j vale 0
i vale 0 y j vale 1
i vale 1 y j vale 0
i vale 1 y j vale 1
i vale 2 y j vale 0
i vale 2 y j vale 1
```

En el ejemplo anterior, el ciclo externo (el controlado por i) se ejecuta 3 veces y el ciclo interno (el controlado por j) se ejecuta 2 veces por cada valor de i. Por ello la instrucción print() se ejecuta en total 6 veces (3 veces que se ejecuta el ciclo externo x 2 veces que se ejecuta cada vez el ciclo interno = 6 veces). El siguiente programa es equivalente al ejemplo anterior:

```
In [19]: ▶ for i in range(3):
          for j in range(2):
              print(f"i vale {i} y j vale {j}")
```

```
i vale 0 y j vale 0
i vale 0 y j vale 1
i vale 1 y j vale 0
i vale 1 y j vale 1
i vale 2 y j vale 0
i vale 2 y j vale 1
```

Al escribir ciclos anidados, hay que prestar atención a la **indentación** de las instrucciones, ya que indica a Python si una instrucción forma parte de un bloque u otro. En los tres siguientes programas la única diferencia es el sangrado de la última instrucción:

1. En este caso, la última instrucción forma parte del cuerpo del ciclo interno. Por tanto el valor de i se escribe cada vez que se ejecuta el ciclo interno.

```
In [20]: ▶ for i in [1, 2, 3]:
          for j in [11, 12]:
              print(i, end=" ")
              print(j, end=" ")
```

1 11 1 12 2 11 2 12 3 11 3 12

2. En este caso, la última instrucción forma parte del cuerpo del ciclo externo, pero no del interno. Por tanto el valor de i se escribe cada vez que se ha terminado de ejecutar el ciclo interno.

```
In [21]: ▶ for i in [1, 2, 3]:
          for j in [11, 12]:
              print(j, end=" ")
          print(i, end=" ")
```

11 12 1 11 12 2 11 12 3

3. En este caso, la última instrucción no forma parte de ningún ciclo. Por tanto el valor de i se escribe una sola vez, al terminarse de ejecutar el ciclo externo.

```
In [22]: ▶ for i in [1, 2, 3]:
          for j in [11, 12]:
              print(j, end=" ")
          print(i, end=" ")
```

11 12 11 12 11 12 3

La costumbre más extendida es utilizar la letra "i" como nombre de la variable de control del ciclo externo y la letra "j" como nombre de la variable de control del ciclo interno (o "k" si hay un tercer nivel de anidamiento), pero se puede utilizar cualquier otro nombre válido.

4. En Python se puede incluso utilizar la misma variable en los dos ciclos anidados porque Python las trata como si fueran dos variables distintas. Por ejemplo:

```
In [23]: ▶ for i in range(3):
          print(f"i (externa) vale {i}")
          for i in range(2):
              print(f"i (interna) vale {i}")
```

i (externa) vale 0
i (interna) vale 0
i (interna) vale 1
i (externa) vale 1
i (interna) vale 0
i (interna) vale 1
i (externa) vale 2
i (interna) vale 0
i (interna) vale 1

De todas formas, este uso no se recomienda porque da lugar a programas difíciles de leer y además no es habitual en otros lenguajes de programación. Se aconseja utilizar siempre nombres de variables distintos.

Ciclos anidados (variables dependientes)

Se dice que las variables de los ciclos son dependientes cuando los valores que toma la variable de control del ciclo interno dependen del valor de la variable de control del ciclo externo. Por ejemplo:

```
In [24]: ▶ for i in [1, 2, 3]:
          for j in range(i):
              print(f"i vale {i} y j vale {j}")
```

i vale 1 y j vale 0
i vale 2 y j vale 0
i vale 2 y j vale 1
i vale 3 y j vale 0
i vale 3 y j vale 1
i vale 3 y j vale 2

En el ejemplo anterior, el ciclo externo (el controlado por i) se ejecuta 3 veces y el ciclo interno (el controlado por j) se ejecuta 1, 2 y 3 veces. Por ello la instrucción print() se ejecuta en total 6 veces.

La variable `i` toma los valores de 1 a 3 y la variable `j` toma los valores de 0 a `i`, por lo que cada vez el ciclo interno se ejecuta un número diferente de veces:

- Cuando `i` vale 1, `range(i)` devuelve la lista `[0]` y por tanto el ciclo interno se ejecuta una sola vez y el programa escribe una sola línea en la que `i` vale 1 (y `j` vale 0).
- Cuando `i` vale 2, `range(i)` devuelve la lista `[0, 1]` y por tanto el ciclo interno se ejecuta dos veces y el programa escribe dos líneas en la que `i` vale 2 (y `j` vale 0 o 1 en cada una de ellas).
- Cuando `i` vale 3, `range(i)` devuelve la lista `[0, 1, 2]` y por tanto el ciclo interno se ejecuta tres veces y el programa escribe tres líneas en la que `i` vale 3 (y `j` vale 0, 1 o 2 en cada una de ellas).

while

Un ciclo `while` permite repetir la ejecución de un grupo de instrucciones mientras se cumpla una condición (es decir, mientras la condición tenga el valor `True`). La sintaxis del ciclo `while` es la siguiente:

`while` condicion:

 cuerpo del ciclo

La ejecución de esta estructura de control `while` es la siguiente:
Python evalúa la condición:

si el resultado es `True` se ejecuta el cuerpo del ciclo. Una vez ejecutado el cuerpo del ciclo, se repite el proceso (se evalúa de nuevo la condición y, si es verdadera, se ejecuta de nuevo el cuerpo del ciclo) una y otra vez mientras la condición sea verdadera.

si el resultado es `False`, el cuerpo del ciclo no se ejecuta y continúa la ejecución del resto del programa.

La variable o las variables que aparezcan en la condición se llaman **variables de control**. Las variables de control deben definirse antes del ciclo `while` y modificarse en el ciclo `while`.

1. Por ejemplo, el siguiente programa escribe los números del 1 al 3:

```
In [25]: ▶ i = 1
while i <= 3:
    print(i)
    i += 1
print("Programa terminado")
```

```
1
2
3
Programa terminado
```

2. El ejemplo anterior se podría haber programado con un ciclo `for`. La ventaja de un ciclo `while` es que la variable de control se puede modificar con mayor flexibilidad, como en el ejemplo siguiente:

```
In [26]: ▶ i = 1
while i <= 50:
    print(i)
    i = 3 * i + 1
print("Programa terminado")
```

```
1
4
13
40
Programa terminado
```

3. Otra ventaja del ciclo `while` es que el número de iteraciones no está definida antes de empezar el ciclo, por ejemplo porque los datos los proporciona el usuario. El ejemplo siguiente ejemplo pide un número positivo al usuario una y otra vez hasta que el usuario lo haga correctamente:

```

Escriba un número positivo: -3
¡Ha escrito un número negativo! Inténtelo de nuevo
Escriba un número positivo: 4
Gracias por su colaboración

```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110
111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138
139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165
166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193
194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248
249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275
276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303
304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330
331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358
359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385
386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440
441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468
469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495
496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523
524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550

3. Se aconseja expresar las condiciones como desigualdades en vez de comparar valores. En el ejemplo siguiente, el programador ha escrito una condición que se cumplirá siempre y el programa imprimirá números consecutivos indefinidamente:

```
In [30]: i = 1
while i != 100:
    print(i, end=" ")
    i += 2
```

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113 115 117 119 121 123 125 127 129 131 133 135 137 139 141 143 145 147 149 151 153 155 157 159 161 163 165 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199 201 203 205 207 209 211 213 215 217 219 221 223 225 227 229 231 233 235 237 239 241 243 245 247 249 251 253 255 257 259 261 263 265 267 269 271 273 275 277 279 281 283 285 287 289 291 293 295 297 299 301 303 305 307 309 311 313 315 317 319 321 323 325 327 329 331 333 335 337 339 341 343 345 347 349 351 353 355 357 359 361 363 365 367 369 371 373 375 377 379 381 383 385 387 389 391 393 395 397 399 401 403 405 407 409 411 413 415 417 419 421 423 425 427 429 431 433 435 437 439 441 443 445 447 449 451 453 455 457 459 461 463 465 467 469 471 473 475 477 479 481 483 485 487 489 491 493 495 497 499 501 503 505 507 509 511 513 515 517 519 521 523 525 527 529 531 533 535 537 539 541 543 545 547 549 551 553 555 557 559 561 563 565 567 569 571 573 575 577 579 581 583 585 587 589 591 593 595 597 599 601 603 605 607 609 611 613 615 617 619 621 623 625 627 629 631 633 635 637 639 641 643 645 647 649 651 653 655 657 659 661 663 665 667 669 671 673 675 677 679 681 683 685 687 689 691 693 695 697 699 701 703 705 707 709 711 713 715 717 719 721 723 725 727 729 731 733 735 737 739 741 743 745 747 749 751 753 755 757 759 761 763 765 767 769 771 773 775 777 779 781 783 785 787 789 791 793 795 797 799 801 803 805 807 809 811 813 815 817 819 821 823 825 827 829 831 833 835 837 839 841 843 845 847 849 851 853 855 857 859 861 863 865 867 869 871 873 875 877 879 881 883 885 887 889 891 893 895 897 899 901 903 905 907 909 911 913 915 917 919 921 923 925 927 929 931 933 935 937 939 941 943 945 947 949 951 953 955 957 959 961 963 965 967 969 971 973 975 977 979 981 983 985 987 989 991 993 995 997 999 1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 1021 1023 1025 1027 1029 1031 1033 1035 1037 1039 1041 1043 1045 1047 1049 1051 1053 1055 1057 1059 1061 1063 1065 1067 1069 1071 1073 1075 1077 1079 1081 1083 1085 1087 1089 1091 1093 1095 1097 1099 1101 1103 1105 1107 1109 1111 1113 1115 1117 1119 1121 1123 1125 1127 1129 1131 1133 1135 1137 1139 1141 1143 1145 1147 1149 1151 1153 1155 1157 1159 1161 1163 1165 1167 1169 1171 1173 1175 1177 1179 1181 1183 1185 1187 1189 1191 1193 1195 1197 1199 1201 1203 1205 1207 1209 1211 1213 1215 1217 1219 1221 1223 1225 1227 1229 1231 1233 1235 1237 1239 1241 1243 1245 1247 1249 1251 1253 1255 1257 1259 1261 1263 1265 1267 1269 1271 1273 1275 1277 1279 1281 1283 1285 1287 1289 1291 1293 1295 1297 1299 1301 1303 1305 1307 1309 1311 1313 1315 1317 1319 1321 1323 1325 1327 1329 1331 1333 1335 1337 1339 1341 1343 1345 1347 1349 1351 1353 1355 1357 1359 1361 1363 1365 1367 1369 1371 1373 1375 1377 1379 1381 1383 1385 1387 1389 1391 1393 1395 1397 1399 1401 1403 1405 1407 1409 1411 1413 1415 1417 1419 1421 1423 1425 1427 1429 1431 1433 1435 1437 1439 1441 1443 1445 1447 1449 1451 1453 1455 1457 1459 1461 1463 1465 1467 1469 1471 1473 1475 1477 1479 1481 1483 1485 1487 1489 1491 1493 1495 1497 1499 1501 1503 1505 1507 1509 1511 1513 1515 1517 1519 1521 1523 1525 1527 1529 1531 1533 1535 1537 1539 1541 1543 1545 1547 1549 1551 1553 1555 1557 1559 1561 1563 1565 1567 1569 1571 1573 1575 1577 1579 1581 1583 1585 1587 1589 1591 1593 1595 1597 1599 1601 1603 1605 1607 1609 1611 1613 1615 1617 1619 1621 1623 1625 1627 1629 1631 1633 1635 1637 1639 1641 1643 1645 1647 1649 1651 1653 1655 1657 1659 1661 1663 1665 1667 1669 1671 1673 1675 1677 1679 1681 1683 1685 1687 1689 1691 1693 1695 1697 1699 1701 1703 1705 1707 1709 1711 1713 1715 1717 1719 1721 1723 1725 1727 1729 1731 1733 1735 1737 1739 1741 1743 1745 1747 1749 1751 1753 1755 1757 1759 1761 1763 1765 1767 1769 1771 1773 1775 1777 1779 1781 1783 1785 1787 1789 1791 1793 1795 1797 1799 1801 1803 1805 1807 1809 1811 1813 1815 1817 1819 1821 1823 1825 1827 1829 1831 1833 1835 1837 1839 1841 1843 1845 1847 1849 1851 1853 1855 1857 1859 1861 1863 1865 1867 1869 1871 1873 1875 1877 1879 1881 1883 1885 1887 1889 1891 1893 1895 1897 1899 1901 1903 1905 1907 1909 1911 1913 1915 1917 1919 1921 1923 1925 1927 1929 1931 1933 1935 1937 1939 1941 1943 1945 1947 1949 1951 1953 1955 1957 1959 1961 1963 1965 1967 1969 1971 1973 1975 1977 1979 1981 1983 1985 1987 1989 1991 1993 1995 1997 1999 2001 2003 2005 2007 2009 2011 2013 2015 2017 2019 2021 2023 2025 2027 2029 2031 2033 2035 2037 2039 2041 2043 2045 2047 2049 2051 2053 2055 2057 2059 2061 2063 2065 2067 2069 2071 2073 2075 2077 2079 2081 2083 2085 2087 2089 2091 2093 2095 2097 2099 2101 2103 2105 2107 2109 2111 2113 2115 2117 2119 2121 2123 2125 2127 2129 2131 2133 2135 2137 2139 2141 2143 2145 2147 2149 2151 2153 2155 2157 2159 2161 2163 2165 2167 2169 2171 2173 2175 2177 2179 2181 2183 2185 2187 2189 2191 2193 2195 2197 2199 2201 2203 2205 2207 2209 2211 2213 2215 2217 2219 2221 2223 2225 2227 2229 2231 2233 2235 2237 2239 2241 2243 2245 2247 2249 2251 2253 2255 2257 2259 2261 2263 2265 2267 2269 2271 2273 2275 2277 2279 2281 2283 2285 2287 2289 2291 2293 2295 2297 2299 2301 2303 2305 2307 2309 2311 2313 2315 2317 2319 2321 2323 2325 2327 2329 2331 2333 2335 2337 2339 2341 2343 2345 2347 2349 2351 2353 2355 2357 2359 2361 2363 2365 2367 2369 2371 2373 2375 2377 2379 2381 2383 2385 2387 2389 2391 2393 2395 2397 2399 2401 2403 2405 2407 2409 2411 2413 2415 2417 2419 2421 2423 2425 2427 2429 2431 2433 2435 2437 2439 2441 2443 2445 2447 2449 2451 2453 2455 2457 2459 2461 2463 2465 2467 2469 2471 2473 2475 2477 2479 2481 2483 2485 2487 2489 2491 2493 2495 2497 2499 2501 2503 2505 2507 2509 2511 2513 2515 2517 2519 2521 2523 2525 2527 2529 2531 2533 2535 2537 2539 2541 2543 2545 2547 2549 2551 2553 2555 2557 2559 2561 2563 2565 2567 2569 2571 2573 2575 2577 2579 2581 2583 2585 2587 2589 2591 2593 2595 2597 2599 2601 2603 2605 2607 2609 2611 2613 2615 2617 2619 2621 2623 2625 2627 2629 2631 2633 2635 2637 2639 2641 2643 2645 2647 2649 2651 2653 2655 2657 2659 2661 2663 2665 2667 2669 2671 2673 2675 2677 2679 2681 2683 2685 2687 2689 2691 2693 2695 2697 2699 2701 2703 2705 2707 2709 2711 2713 2715 2717 2719 2721 2723 2725 2727 2729 2731 2733 2735 2737 2739 2741 2743 2745 2747 2749 2751 2753 2755 2757 2759 2761 2763 2765 2767 2769 2771 2773 2775 2777 2779 2781 2783 2785 2787 2789 2791 2793 2795 2797 2799 2801 2803 2805 2807 2809 2811 2813 2815 2817 2819 2821 2823 2825 2827 2829 2831 2833 2835 2837 2839 2841 2843 2845 2847 2849 2851 2853 2855 2857 2859 2861 2863 2865 2867 2869 2871 2873 2875 2877 2879 2881 2883 2885 2887 2889 2891 2893 2895 2897 2899 2901 2903 2905 2907 2909 2911 2913 2915 2917 2919 2921 2923 2925 2927 2929 2931 2933 2935 2937 2939 2941 2943 2945 2947 2949 2951 2953 2955 2957 2959 2961 2963 2965 2967 2969 2971 2973 2975 2977 2979 2981 2983 2985 2987 2989 2991 2993 2995 2997 2999 3001 3003 3005 3007 3009 3011 3013 3015 3017 3019 3021 3023 3025 3027 3029 3031 3033 3035 3037 3039 3041 3043 3045 3047 3049 3051 3053 3055 3057 3059 3061 3063 3065 3067 3069 3071 3073 3075 3077 3079 3081 3083 3085 3087 3089 3091 3093 3095 3097 3099 3101 3103 3105 3107 3109 3111 3113 3115 3117 3119 3121 3123 3125 3127 3129 3131 3133 3135 3137 3139 3141 3143 3145 3147 3149 3151 3153 3155 3157 3159 3161 3163 3165 3167 3169 3171 3173 3175 3177 3179 3181 3183 3185 3187 3189 3191 3193 3195 3197 3199 3201 3203 3205 3207 3209 3211 3213 3215 3217 3219 3221 3223 3225 3227 3229 3231 3233 3235 3237 3239 3241 3243 3245 3247 3249 3251 3253 3255 3257 3259 3261 3263 3265 3267 3269 3271 3273 3275 3277 3279 3281 3283 3285 3287 3289 3291 3293 3295 3297 3299 3301 3303 3305 3307 3309 3311 3313 3315 3317 3319 3321 3323 3325 3327 3329 3331 3333 3335 3337 3339 3341 3343 3345 3347 3349 3351 3353 3355 3357 3359 3361 3363 3365 3367 3369 3371 3373 3375 3377 3379 3381 3383 3385 3387 3389 3391 3393 3395 3397 3399 3401 3403 3405 3407 3409 3411 3413 3415 3417 3419 3421 3423 3425 3427 3429 3431 3433 3435 3437 3439 3441 3443 3445 3447 3449 3451 3453 3455 3457 3459 3461 3463 3465 3467 3469 3471 3473 3475 3477 3479 3481 3483 3485 3487 3489 3491 3493 3495 3497 3499 3501 3503 3505 3507 3509 3511 3513 3515 3517 3519 3521 3523 3525 3527 3529 3531 3533 3535 3537 3539 3541 3543 3545 3547 3549 3551 3553 3555 3557 3559 3561 3563 3565 3567 3569 3571 3573 3575 3577 3579 3581 3583 3585 3587 3589 3591 3593 3595 3597 3599 3601 3603 3605 3607 3609 3611 3613 3615 3617 3619 3621 3623 3625 3627 3629 3631 3633 3635 3637 3639 3641 3643 3645 3647 3649 3651 3653 3655 3657 3659 3661 3663 3665 3667 3669 3671 3673 3675 3677 3679 3681 3683 3685 3687 3689 3691 3693 3695 3697 3699 3701 3703 3705 3707 3709 3711 3713 3715 3717 3719 3721 3723 3725 3727 3729 3731 3733 3735 3737 3739 3741 3743 3745 3747 3749 3751 3753 3755 3757 3759 3761 3763 3765 3767 3769 3771 3773 3775 3777 3779 3781 3783 3785 3787 3789 3791 3793 3795 3797 3799 3801 3803 3805 3807 3809 3811 3813 3815 3817 3819 3821 3823 3825 3827 3829 3831 3833 3835 3837 3839 3841 3843 3845 3847 3849 3851 3853 3855 3857 3859 3861 3863 3865 3867 3869 3871 3873 3875 3877 3879 3881 3883 3885 3887 3889 3891 3893 3895 3897 3899 3901 3903 3905 3907 3909 3911 3913 3915 3917 3919 3921 3923 3925 3927 3929 3931 3933 3935 3937 3939 3941 3943 3945 3947 3949 3951 3953 3955 3957 3959 3961 3963 3965 3967 3969 3971 3973 3975 3977 3979 3981 3983 3985 3987 3989 3991 3993 3995 3997 3999 4001 4003 4005 4007 4009 4011 4013 4015 4017 4019 4021 4023 4025 4027 4029 4031 4033 4035 4037 4039 4041 4043 4045 4047 4049 4051 4053 4055 4057 4059 4061 4063 4065 4067 4069 4071 4073 4075 4077 4079 4081 4083 4085 4087 4089 4091 4093 4095 4097 4099 4101 4103 4105 4107 4109 4111 4113 4115 4117 4119 4121 4123 4125 4127 4129 4131 4133 4135 4137 4139 4141 4143 4145 4147 4149 4151 4153 4155 4157 4159 4161 4163 4165 4167 4169 4171 4173 4175 4177 4179 4181 4183 4185 4187 4189 4191 4193 4195 4197 4199 4201 4203 4205 4207 4209 4211 4213 4215 4217 4219 4221 4223 4225 4227 4229 4231 4233 4235 4237 4239 4241 4243 4245 4247 4249 4251 4253 4255 4257 4259 4261 4263 4265 4267 4269 4271 4273 4275 4277 4279 4281 4283 4285 4287 4289 4291 4293 4295 4297 4299 4301 4303 4305 4307 4309 4311 4313 4315 4317 4319 4321 4323 4325 4327 4329 4331 4333 4335 4337 4339 4341 4343 4345 4347 4349 4351 4353 4355 4357 4359 4361 4363 4365 4367 4369 4371 4373 4375 4377 4379 4381 4383 4385 4387 4389 4391 4393 4395 4397 4399 4401 4403 4405 4407 4409 4411 4413 4415 4417 4419 4421 4423 4425 4427 4429 4431 4433 4435 4437 4439 4441 4443 4445 4447 4449 4451 4453 4455 4457 4459 4461 4463 4465 4467 4469 4471 4473 4475 4477 4479 4481 4483 4485 4487 4489 4491 4493 4495 4497 4499 4501 4503 4505 4507 4509 4511 4513 4515 4517 4519 4521 4523 4525 4527 4529 4531 4533 4535 4537 4539 4541 4543 4545 4547 4549 4551 4553 4555 4557 4559 4561 4563 4565 4567 4569 4571 4573 4575 4577 4579 4581 4583 4585 4587 4589 4591 4593 4595 4597 4599 4601 4603 4605 4607 4609 4611 4613 4615 4617 4619 4621 4623 4625 4627 4629 4631 4633 4635 4637 4639 4641 4643 4645 4647 4649 4651 4653 4655 4657 4659 4661 4663 4665 4667 4669 4671 4673 4675 4677 4679 4681 4683 4685 4687 4689 4691 4693 4695 4697 4699 4701 4703 4705 4707 4709 4711 4713 4715 4717 4719 4721 4723 4725 4727 4729 4731 4733 4735 4737 4739 4741 4743 4745 4747 4749 4751 4753 4755 4757 4759 4761 4763 4765 4767 4769 4771 4773 4775 4777 4779 4781 4783 4785 4787 4789 4791 4793 4795 4797 4799 4801 4803 4805 4807 4809 4811 4813 4815 4817 4819 4821 4823 4825 4827 4829 4831 4833 4835 4837 4839 4841 4843 4845 4847 4849 4851 4853 4855 4857 4859 4861 4863 4865 4867 4869 4871 4873 4875 4877 4879 4881 4883 4885 4887 4889 4891 4893 4895 4897 4899 4901 4903 4905 4907 4909 4911 4913 4915 4917 4919 4921 4923 4925 4927 4929 4931 4933 4935 4937 4939 4941 4943 4945 4947 4949 4951 4953 4955 4957 4959 4961 4963 4965 4967 4969 4971 4973 4975 4977 4979 4981 4983 4985 4987 4989 4991 4993 4995 4997 4999 5001 5003 5005 5007 5009 5011 5013 5015 5017 5019 5021 5023 5025 5027 5029 5031 5033 5035 5037 5039 5041 5043 5045 5047 5049 5051 5053 5055 5057 5059 5061 5063 5065 5067 5069 5071 5073 5075 5077 5079 5081 5083 5085 5087 5089 5091 5093 5095 5097 5099 5101 5103 5105 5107 5109 5111 5113 5115 5117 511