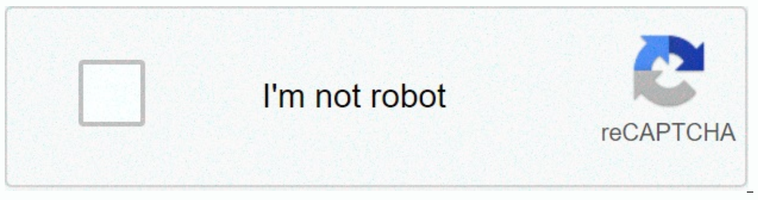

Continue

Samsung galaxy s6 camera apk

Last update 26 February 2020 08:09:19 (UTC / GMT + 8 hours) Write the Python program to find those numbers that are divisible by 7 and multiple 5, between 1500 and 2700 (both included). Illustrated presentation: Sample solution: Python code: nl=[] x range (1500, 2701): if (x%7==0) and (x%5==0): nl.append(str(x)) print (','.join(nl)) Sample Output: 1505,1540,1575,1610,164 15,1680,1715,1750,1785,1820,1855,1890,1925,1960,1995,2030,2065,2100,2135,2170,2205,2240, 2275 2310 2345 2380 2415 2450 2485 2520 2555 2590,2 2625,2660,2695 Flowchart: Visualize python code execution: the following tool to visualize it, what your computer does step by step, because it executes the above program: Python Code Editor: Have another way to solve this solution? Promote your code (and comments) using Disqus. Previous: Python Conditional Notifications and Loops Exercises Home Next: Write a Python program to convert the temperature to and from celsius, fahrenheit. What is the level of difficulty in this exercise? Test your Python skills with w3resource's quiz Python: Reversed Function & Reverse Method) Reversed function and reversed function and reversed method can only be used to change the object of Python. But between the two: the inverse function is a big difference, and it can change and iterable object and returns the inverse object as a data type. the reversed method can only be used with lists, because it is only a list method. lst=[Keyboard,Monitors,Printer,Mouse] lst.reverse() print(lst) Output: <list_reverseiterator object= at= 0x7f187576d668=>['Mouse', 'Printer', 'Monitor', 'Keyboard'] lst=[Keyboard,Monitor,Printer,Mouse] x=reversed(lst) print(x) print(list(x)) Output: <list_reverseiterator object= at= 0x7f187576d68=>['Mouse', 'Printer', 'Monitor', 'Keyboard'] str=Python' a=reversed(str) print(.join(a))) Output: nohtyP Based on a range of numbers, the task is to write a Python program to find numbers to be divisible by 7 and divisible 5. Example: Input: Enter at least 100 Enter a maximum output of 200: 105 is divisible by 7 and 5.140 is divisible by 7 and 5.175 is divisible by 7 and 5. Input:Input:Enter at least 29 Enter a maximum output of 36: 35 is divisible by 7 and 5. You can check whether the numbers can have dividibility by 7 and 5 by performing the modulated operation of the number, respectively, and then checking the remainder. This can be done as follows: start_num = int (29) end_num = int (36) cnt = start_num <= end_num:= if= cnt= %= 0= and= cnt= %= 5== 0:= print(cnt,= = is= divisible= by= 7= and= 5.) = cnt= +=1 output:= 35= is= divisible= by= by= and= 5.= this= can= also= be= done= by= check= if= number= is= divisible= by= 35,= since= the= lcm= of= 7= and= 5= is= 35= and= any= number= divisible= by= 35= is= divisible= by= 7= and= 5= and= vice= versa= also. = start_num=int(68) end_num=int(167) cnt=start_num while= cnt=></=> <= end_num: if(cnt % 35 0): print(cnt, end_num:= if(cnt= %= 35== 0):= print(cnt,=></= end_num: if(cnt % 35 == 0): print(cnt, > bet cnt</list_reverseiterator> </list_reverseiterator> </list_reverseiterator> divided by 7 and 5.) cnt += 1 Result: 70 is divisible by 7 and 5.105 is divisible by 7 and 5.140 is divisible by 7 and 5. Attention geek! Strengthen your basics with python programming foundation course and learn the basics. To begin with, your interview preparations improve your data structure concepts with python DS Course. Recommended Posts: If you like GeeksforGeeks and would like to promote, you can also write an article using contribute.geeksforgeeks.org or post your article contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks

main page and help other geeks.Please improve this article if you find something wrong by clicking the Enhance Article button below. Python program to print numbers to be divisible by 3 and 5 using the loopPython program to print numbers divisible by 7, using loop Python program to print the first n numbers to be divisible by 5 using, but loop start = int (input (Enter start number:)) end = int (input (Enter last number:)) in range i (start, end +1): if((i%3==0) & end + 1): if ((i% 3 = 0) & (i%5==0)): print(i) Output Enter start number: 1 Enter last number: 30 15 30 Python program to print numbers divisible by 7 using for loop # Python program to print numbers divisible (i %50 == 0)): print (i) Output Enter start number: 1 Enter last number: 30 15 30 Python program print numbers dividing by 7 using loop #python program print number divisible with 7 using loop start = int (input (Enter start number:)) end = int (input (Enter last number:)) in range i (start, end + 1): if (i%7 = 0) : print(s) Output Enter start number: 1 Enter last number: 50 7 14 21 28 35 42 49 Python program print first n numbers divisibility with 5 using, but loop #python program print numbers divisible by 7 using , but loop start = int (input (Enter start number :)) end = int (input (Enter last number:)), but (start< = end): if (start%5 = = 0): print (start) start += 1 Output Enter the starting number: 1 Enter the last number: 50 5 10 15 20 20 30 35 40 45 50, if you don't want to do 7 counts and mathematically do it here is how can you n= 7 i = 0 j = 100 # largest_multiple_of_n_smaller_than_j = (j - j%n) # 100 = 100% 7 = > 100 -2 - > 98 # 98//7 --> 14 # so we can say 7 multiply i (where I go from 0 - > 14) will be less than 100 multiples_of_n = [n * i i range (0 , ((j - j%n) // n) +1)] # 0 7 14 21 .... 98 #17 27 37.. (don't pick 77 here).). 97 and 71 72 73...77..79 contains_n = [i*10+n for i in range(1,10) if i != n] + [n*10+x for x in range(1,10) if (n*10+x) % n !=0] '' or use this contains_n = [x for x in range(10+n,j,10) if x != n*11] + [x for x in range(n*10+1,n*10+10) if x % n !=0] "mult_and_contain_n = sorted(multiples_of_n + contains_n) print(mult_and_contain_n Output) n = 7 7, 14, 17, 21, 27, 28, 35, 37, 42, 47, 49, 56, 57, 63, 67, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 84, 87, 91, 97, 98] n = 5 [0, 5, 10, 15, 15 , 20, 25, 25, 30, 35, 35, 40, 45, 45 , 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 65, 65, 70, 75, 75, 80, 85, 85, 90, 95, 95, 100 Numbers (kas ir 1 līdz 1000), un mums ir drukāt visus numurus, kas ir dalāms are 7 un nav dalāms ar 5 python. Given a range (1 to 1000), we printed all the numbers that can be dicted by 7 on python days and not by 5. Piemērs: Example: Ievade: Dotais ievades diapazons ir 1 līdz 1000 Izvade: 7, 14, 21, 28, 42, 49, 56, ... Loģika: Logic: Lai īstenotu loģiku, mēs izmantos un cilpa ar range () metodi. Diapazona () metodes priekšraksts ar minimālo līdz maksimālo diapazonu ir diapazons (sākt, beigt +1). To implement this logic, we will use the and ar loop) methods. The range from the maximum day range() method day statement is the range (begin, end, plus 1). Un, pārbaudiet nosacījumu, ka vērtība ir dalāms ar 7 un nedrīkst būt dalāms ar 5 (piemēram kods: ((cnt% 7 s 0) un (cnt% 5! s 0))) and, check conditions, the value should be divided by 7 and not by 5 (example code: (c% 7 x 0) and (cnt% 5!=0)). Ja nosacījums ir patiess, izdrukājiet skaitļus. If the condition is true, the number is printed. Programma: s definēt diapazonu mainīgie s tā, ka mēs varam mainīt tos jebkurā laikā sākas s 1 beigas s 1000 s cilpa, lai pārbaudītu un izdrukātu numurus s kas ir dalāmas ar 7 un nav s dalāmību ar 5 cnt diapazonā (sākas, end:ja (cnt% 7 s 0 un cnt% 5!=0): drukāt cnt, s komanda pēc cnt būs drukāt telpu Output output 7 14 21 28 42 49 56 63 7 7 84 91 98 112 119 126 133 147 154 161 168 182 189 196 203 217 22 4 231 2 38 252 259 266 273 287 294 301 308 322 329 336 343 357 364 371 378 392 399 406 413 427 434 441 448 462 469 4 76 483 497 504 511 518 532 539 546 553 567 574 581 588 602 609 616 623 637 644 651 658 672 679 686 693 777 14 721 728 742 749 749 756 763 777 784 791 798 812 819 826 833 847 854 861 868 882 889 896 903 917 924 92991 38 952 959 966 973 987 994 Translated from: python Digital Dictation Likes Collections Follow one-click triple likes Mark follow the blogger, stay up to date with TA Day's latest blog ©2020 CSDN Skin Theme: Programming Studio Designer: CSDN Official Blog Back home Pirmais attiecībā Uzlu. Python ir oficiāls stils-guide, PEP8, kas iesaka, kam nav nevajadzīgas atstarpes. Tas uzlabo lasāmību daudz, ja jums izvairīties no tukša līnija starp katru līniju. Izmantojiet tos taupīgi, lai nošķirtu lielākus loģiskos blokus. Standarta veids, kas ir visefektīvākais, lai pārbaudītu dalāmību, ir modulācijas operators %. Tādā veidā jums nav jāpaļaujas uz vesela skaitļa dalīšanas vienmēr ir aptuveni (kas tas nebūs, tas ir noņemts ar pilnu sadalījumu python 3.x). Jūs varat arī tieši atgriezt salīdzinājuma rezultātu: def is_divisible_by_7(num): atgriež num % 7 == 0 Jums jāievieto kods f __name__ == __main__ aizsargs un funkcijās, lai varētu atkārtoti izmantot jūsu kodu: def main(): N = raw_input() Q = int (raw_input ()) par _ in。 a, b = map(int, raw_input()split()) num = int(N[a - 1:b]) print if yes num % 7 == 0 else NO, if __name__ == __main__: main() Here I inlined the function is_divisible_by_7, used _ as a place-holder variable that is not used in the loop as usual Python and uses three expressions to print the result. You also don't need to use a split, python 2.x card returns a list that can be directly used for tuple-allocation. Alternatively, if a and b appear repeatedly, caching can help: def memodict(f): Memoization decorator function, taking a single argument class memodict(dict): def __missing__(self, key): ret = self [key] = f (key) return return memodict() .__getitem__ @memodict def is_divisible_by_7 (indexes): a, b = map (int, indexes.split()) returns NO if int(N[a - 1:b]) % 7 other YES N = raw_input() Q = int(raw_input()) par _in xrange(Q): print is_divisible_by_7(raw_input()) As the third alternative, we could not cache the indexes, but the numbers for which we twosibility test. It's worth it if they repeat and are large. @memodict def is_divisible_by_7(n): return int(n) % 7 == 0 N = raw_input() Q = int(raw_input()) for _ in xrange(Q): a, b = map(int, raw_input().split()) print YES, if is_divisible_by_7(N[a - 1:b]) other NO with the same memorizer class as above. You could do this a little faster without printing each line, but summing the results first: N = raw_input() Q = int(raw_input()) out = [] for _ in xrange(Q): a, b = card (int, raw_input (is_divisible_by_7). Not very nice, but I did find this one: Double the last digit and subtract it from the number by other digits. The result must be divisible by 7. (We may apply this rule to this response again) (source) Therefore, we want to recursively check the number. But since it can be up to 10**5 digits, we have hope that tires for small numbers first, otherwise we will run into python's recursion limit. However, we can increase the number of digits of these limits, N. import system MAX_INT = 2 ** 31 - 1 @memodict def is_divisible_by_7 (n_str): if n < = MAX_INT: # mod must be fast enough below this limit return int(n_str) % 7 == 0 return is_divisible_by_7 (str (int (n_str n_str <3> <7> [:-1]) - 2*int(n_str[-1]))) N = raw_input() sys.setrecursionlimit(MAX_INT) # allow processing number Q = int(raw_input()) _ in xrange(Q): a, b = map (int, raw_input().split()) print YES if is_divisible_by_7 (N[a - 1: b]) another NO It's a lot more processing, especially the conversion between str and int. But maybe along with caching it could do some good. Good.