**6 kyu**

**Abundant Numbers**

11393% of 3047 of80[user2821739](https://www.codewars.com/users/user2821739)

Python

* [TRAIN AGAIN](https://www.codewars.com/kata/abundant-numbers/train/python)
* [NEXT KATA](https://www.codewars.com/trainer/python)

Details

[Solutions](https://www.codewars.com/kata/abundant-numbers/solutions/python)

[Discourse (14)](https://www.codewars.com/kata/abundant-numbers/discuss/python)

* Add to Collection
* |
* Share this kata:

In number theory, an **[abundant](https://en.wikipedia.org/wiki/Abundant_number" \t "_blank)** number or an **[excessive](https://en.wikipedia.org/wiki/Abundant_number" \t "_blank)** number is one for which the sum of it's **[proper divisors](http://mathworld.wolfram.com/ProperDivisor.html" \t "_blank)** is greater than the number itself. The integer **12** is the first abundant number. Its proper divisors are 1, 2, 3, 4 and 6 for a total of **16**. The amount by which the sum exceeds the number is the **abundance**. The number **12** has an abundance of **4**, for example. Other initial abundant numbers are : 12, 18, 20, 24, 30, 36, 40, 42, 48, 54 etc . **Infinitely** many **odd** and **even** abundant numbers exist.  
As you should have guessed by now, in this kata your function will take a positive integer **h** as range input and return a nested array/list that will contain the following informations-

* Highest available **odd** or **even** abundant number in that range
* It's **abundance**

Examples

A few examples never hurt nobody, right???

abundant(15) = [[12], [4]]

abundant(19) = [[18], [3]]

abundant(100) = [[100], [17]]

abundant(999) = [[996], [360]]

Tips

The problem involves some pretty big random numbers. So try to optimize your code for performance as far as you can. And yes, the input argument will always be positive integers. So no need to check there.

Good luck!

<https://www.onlinegdb.com/online_python_compiler>

def SumDivisores( n):

s = 0

i = 1

while(i\*i <= n):

if(n%i==0):

s +=i

if(n / i != i):

if(n/i != n):

s += (n/i)

i+=1

return s

def abundant(h):

# your code here

ans = []

for i in range(h, 0, -1):

#print(i)

s = SumDivisores(i)

if(s > i):

x = []

x.append(i)

ans.append(x)

y = []

y.append(int(s- i))

ans.append(y)

break

return ans

print(abundant(19))

-------------------

[Chris\_Rands](https://www.codewars.com/users/Chris_Rands)

**def abundant(h):**

**for n in range(h,0,-1):**

**s = sum(i for i in range(1,n) if n % i == 0)**

**if s > h:**

**return [[n],[s-n]]**