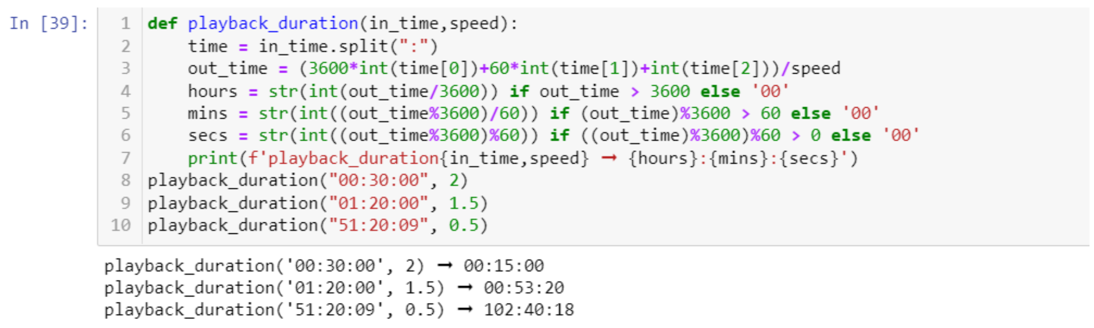
1. YouTube offers different playback speed options for users. This allows users to increase or decrease the speed of the video content. Given the actual duration and playback speed of the video, calculate the playback duration of the video.

Examples:
playback_duration("00:30:00", 2) ➞ "00:15:00"
playback_duration("01:20:00", 1.5) ➞ "00:53:20"
playback_duration("51:20:09", 0.5) ➞ "102:40:18"

Ans:



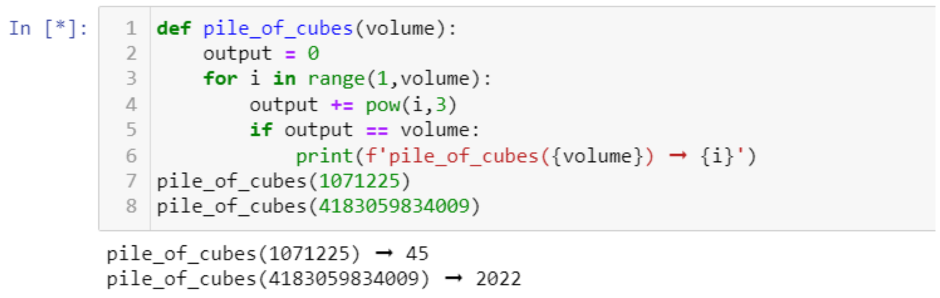
```
In [39]:
def playback_duration(in_time,speed):
    time = in_time.split(":")
    out_time = (3600*int(time[0])+60*int(time[1])+int(time[2]))/speed
    hours = str(int(out_time/3600)) if out_time > 3600 else '00'
    mins = str(int((out_time%3600)/60)) if (out_time)%3600 > 60 else '00'
    secs = str(int((out_time%3600)%60)) if ((out_time)%3600)%60 > 0 else '00'
    print(f'playback_duration{in_time,speed} ➞ {hours}:{mins}:{secs}')
playback_duration("00:30:00", 2)
playback_duration("01:20:00", 1.5)
playback_duration("51:20:09", 0.5)
```

```
playback_duration('00:30:00', 2) ➞ 00:15:00
playback_duration('01:20:00', 1.5) ➞ 00:53:20
playback_duration('51:20:09', 0.5) ➞ 102:40:18
```

2. We need your help to construct a building which will be a pile of n cubes. The cube at the bottom will have a volume of n^3, the cube above will have volume of (n-1)^3 and so on until the top which will have a volume of 1^3.

Given the total volume m of the building, can you find the number of cubes n required for the building?
In other words, you have to return an integer n such that: n^3 + (n-1)^3 + ... + 1^3 == m
Return None if there is no such number.
Examples:
pile_of_cubes(1071225) ➞ 45
pile_of_cubes(4183059834009) ➞ 2022
pile_of_cubes(16) ➞ None

Ans:



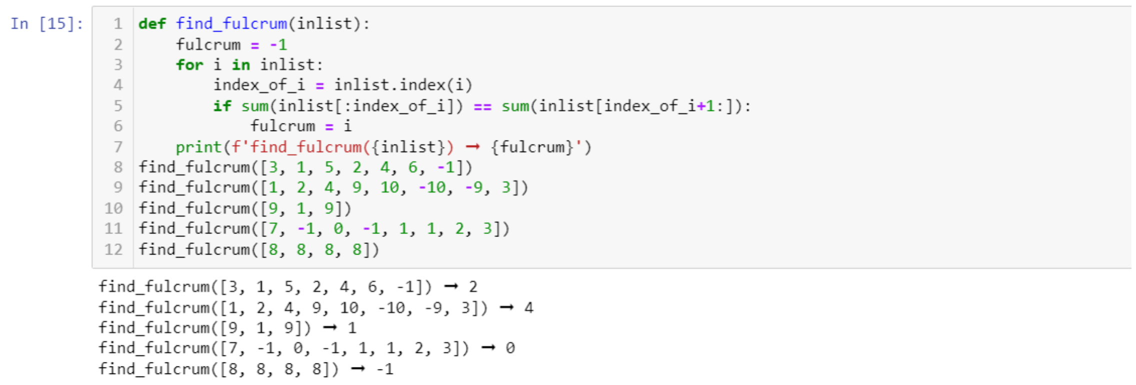
```
In [*]:
def pile_of_cubes(volume):
    output = 0
    for i in range(1,volume):
        output += pow(i,3)
        if output == volume:
            print(f'pile_of_cubes({volume}) ➞ {i}')
pile_of_cubes(1071225)
pile_of_cubes(4183059834009)
```

```
pile_of_cubes(1071225) ➞ 45
pile_of_cubes(4183059834009) ➞ 2022
```

3. A fulcrum of a list is an integer such that all elements to the left of it and all elements to the right of it sum to the same value. Write a function that finds the fulcrum of a list.

To illustrate:
find_fulcrum([3, 1, 5, 2, 4, 6, -1]) ➞ 2 // Since [3, 1, 5] and [4, 6, -1] both sum to 9
Examples:
find_fulcrum([1, 2, 4, 9, 10, -10, -9, 3]) ➞ 4
find_fulcrum([9, 1, 9]) ➞ 1
find_fulcrum([7, -1, 0, -1, 1, 1, 2, 3]) ➞ 0
find_fulcrum([8, 8, 8, 8]) ➞ -1

Ans:



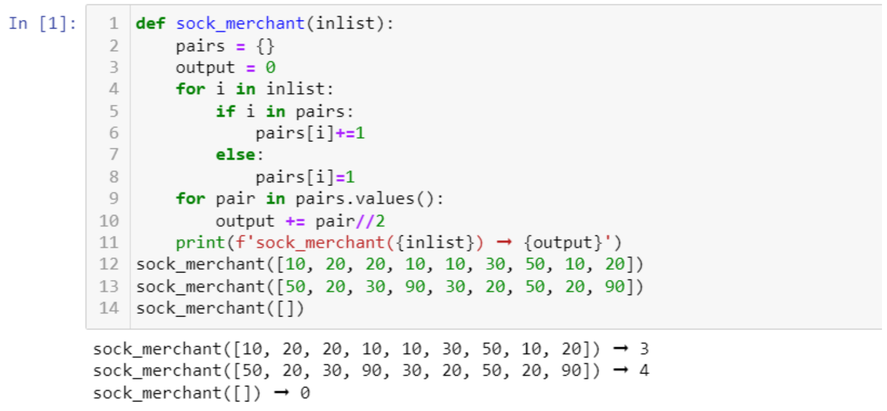
In [15]:

```python
def find_fulcrum(inlist):
    fulcrum = -1
    for i in inlist:
        index_of_i = inlist.index(i)
        if sum(inlist[:index_of_i]) == sum(inlist[index_of_i+1:]):
            fulcrum = i
    print(f'find_fulcrum({inlist}) ➞ {fulcrum}')
find_fulcrum([3, 1, 5, 2, 4, 6, -1])
find_fulcrum([1, 2, 4, 9, 10, -10, -9, 3])
find_fulcrum([9, 1, 9])
find_fulcrum([7, -1, 0, -1, 1, 1, 2, 3])
find_fulcrum([8, 8, 8, 8])
```

```
find_fulcrum([3, 1, 5, 2, 4, 6, -1]) ➞ 2
find_fulcrum([1, 2, 4, 9, 10, -10, -9, 3]) ➞ 4
find_fulcrum([9, 1, 9]) ➞ 1
find_fulcrum([7, -1, 0, -1, 1, 1, 2, 3]) ➞ 0
find_fulcrum([8, 8, 8, 8]) ➞ -1
```

4. Given a list of integers representing the color of each sock, determine how many pairs of socks with matching colors there are. For example, there are 7 socks with colors [1, 2, 1, 2, 1, 3, 2]. There is one pair of color 1 and one of color 2. There are three odd socks left, one of each color. The number of pairs is 2.

Create a function that returns an integer representing the number of matching pairs of socks that are available.
Examples:
sock_merchant([10, 20, 20, 10, 10, 30, 50, 10, 20]) ➞ 3
sock_merchant([50, 20, 30, 90, 30, 20, 50, 20, 90]) ➞ 4
sock_merchant([]) ➞ 0
Ans:



In [1]:

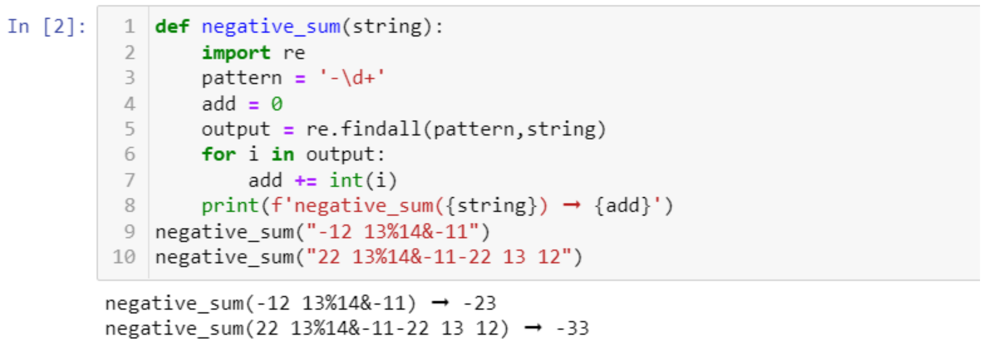
```python
def sock_merchant(inlist):
    pairs = {}
    output = 0
    for i in inlist:
        if i in pairs:
            pairs[i]+=1
        else:
            pairs[i]=1
    for pair in pairs.values():
        output += pair//2
    print(f'sock_merchant({inlist}) ➞ {output}')
sock_merchant([10, 20, 20, 10, 10, 30, 50, 10, 20])
sock_merchant([50, 20, 30, 90, 30, 20, 50, 20, 90])
sock_merchant([])
```

```
sock_merchant([10, 20, 20, 10, 10, 30, 50, 10, 20]) ➞ 3
sock_merchant([50, 20, 30, 90, 30, 20, 50, 20, 90]) ➞ 4
sock_merchant([]) ➞ 0
```

5. Create a function that takes a string containing integers as well as other characters and return the sum of the negative integers only.

Examples:
negative_sum("-12 13%14&-11") ➞ -23
# -12 + -11 = -23
negative_sum("22 13%14&-11-22 13 12") ➞ -33
# -11 + -22 = -33

Ans:



```
In [2]:
def negative_sum(string):
    import re
    pattern = '-\d+'
    add = 0
    output = re.findall(pattern,string)
    for i in output:
        add += int(i)
    print(f'negative_sum({string}) ➞ {add}')
negative_sum("-12 13%14&-11")
negative_sum("22 13%14&-11-22 13 12")
```

```
negative_sum(-12 13%14&-11) ➞ -23
negative_sum(22 13%14&-11-22 13 12) ➞ -33
```