Buy 3 get 1 for free

Have you ever felt you've been cheated by a "buy 3 get 1 for free" action? It's quite all right if shops advertise such an action and interpret it in such a way that if you purchase four items, you get the cheapest one for free. What bothers most customers is that some shops turn this into their advantage as soon as you buy more than four products. Cash registers might be programmed to interpret a "buy 3 get 1 for free" action such that for each n items purchased, you get the $\lfloor\frac{n}{4}\rfloor$ cheaptest items for free. Here, $\lfloor x\rfloor$ is the largest integer not greater than $x \in \mathbb{R}$.

This action means that for each n items purchased, you get the $\lfloor\frac{n}{4}\rfloor$ cheaptest items for free.

Only after customers have paid all their items at once, they realize that they would have made profit if they had passed the cash register several times. They could have arranged the purchased items from the most expensive one to the cheapest one, and passed the cash register with groups of four items. The first time with the four most expensive items (for which they get the cheapest for free), a second time with the next four most expensive items, and so on. If the total number of items purchased is not a multiple of four, they would have to pass the cash register one last time with less than four items, and not get an item for free this time.

Assignment

Say you have purchased a number of items in a shop that runs a "buy 3 get 1 for free" action, and whose cash register has been programmed in favor of the shop as described in the introduction of this assignment. Your task is to determine your own profit, in case you would not purchase all products at once, but pass the cash register multiple times, each time getting the cheapest item for free for the group of the next four most expensive items. In order to do so, you have to write four functions, that each take a sequence (a list or a tuple) of $n \mathbb{N}_0$ *floating point* numbers. These numbers represent the prices of the items purchased.

- Write a function `together` that returns the total amount you would have to pay in case you passed the cash register once to purchase all items together.
- Write a function `group` that returns a list of tuples, where each tuple contains at least one and at most four *floating point* numbers that are arranged from the largest to the smallest. The first tuple must contain the prices of the four most expensive items, the second tuple must contain the prices of the next four most expensive items, and so on. If the number of purchased items is not a multiple of four, the last tuple must contain the prices of the remaining items.
- Write a function `grouped` that returns the total amount you would have to pay in case you passed the cash register multiple times to purchase the items in groups of at most four items, as described in the introduction of this assignment.

- Write a function profit that returns the profit you make if you follow the strategy where you pass the cash register multiple times, compared to the total amount you would have to pay if you only passed the cash register once to purchase all items together.

Example

```
>>> prices = [3.23, 5.32, 8.23, 2.23, 9.98, 7.43, 6.43, 8.23, 4.23]
>>> together(prices)
49.85
>>> group(prices)
[(9.98, 8.23, 8.23, 7.43), (6.43, 5.32, 4.23, 3.23), (2.23,)]
>>> grouped(prices)
44.65
>>> profit(prices)
5.2
```

Heb je je ook al eens bedrogen gevoeld bij een "3+1 gratis" actie? Dat winkels hiermee uitpakken en daarbij in de *kleine lettertjes* schrijven dat je voor elke aankoop van vier producten de goedkoopste gratis krijgt, tot daar aan toe. Waar klanten vooral problemen mee hebben, is dat sommige winkels dit in hun voordeel laten uitdraaien op het ogenblik dat een klant meer dan vier producten aankoopt. Voor "3 + 1 gratis" durven winkels hun kassasysteem vaak zó programmeren dat een klant die n producten aankoopt, daarvan de goedkoopste $\lfloor\frac{n}{4}\rfloor$ producten gratis krijgt. Hierbij is $\lfloor x\rfloor$ het grootste gehele getal dat niet groter is dan $x \in \mathbb{R}$.

(*) Deze actie betekent dat voor elke n producten die je aankoopt, je de goedkoopste $\lfloor\frac{n}{4}\rfloor$ producten gratis krijgt.

Als klanten eenmaal betaald hebben, beseffen ze achteraf pas dat ze er profijt zouden aan gedaan hebben als ze verschillende keren langs de kassa gepasseerd hadden. Ze zouden immers de gekochte producten kunnen rangschikken van het duurste naar het goedkoopste product, en dan per vier producten langs de kassa te passeren. De eerste keer met de vier duurste producten (waarvan ze de goedkoopste gratis krijgen), een tweede keer met de volgende vier duurste producten, enzoverder. Als het aantal aangekochte producten geen veelvoud is van vier, dan zullen ze een laatste keer langs de kassa moeten passeren met minder dan vier producten, en daarbij dus geen product gratis krijgen.

Opgave

Stel dat je een aantal producten gekocht hebt in een winkel waar een "3+1 gratis" loopt, en waar het kassasysteem in het voordeel van de winkel geprogrammeerd is op een manier zoals omschreven in de inleiding van deze opgave. Je opdracht bestaat erin te bepalen hoeveel winst je zal maken als je niet alle producten in één keer betaalt, maar meerdere keren langs de kassa passeert om telkens het goedkoopste product van een groep van maximaal vier producten gratis te krijgen. Hiervoor moet je vier functies schrijven, waaraan telkens een reeks (een lijst of een

tuple) met $n \in \mathbb{N}_0$ *floating point* getallen moet doorgegeven worden. Deze getallen stellen de prijzen van de aangekochte producten voor.

- Schrijf een functie samen die het totaalbedrag teruggeeft dat je zal moeten betalen als je slechts één keer langs de kassa passeert om de gegeven reeks producten samen aan te kopen.
- Schrijf een functie groeperen die een lijst van tuples teruggeeft, waarbij elk tuple minimaal 1 en maximaal 4 *floating point* getallen bevat die gesorteerd zijn van groot naar klein. Het eerste tuple moet de prijzen van de vier duurste van de reeks aangekochte producten bevatten, het tweede tuple de volgende vier duurste producten, enzoverder. Als het aantal aangekochte producten geen veelvoud van vier is, dan moet het laatste tuple de prijzen van de resterende producten bevatten.
- Schrijf een functie gegroepeerd die het totaalbedrag teruggeeft dat je zal moeten betalen als je meerdere keren langs de kassa passeert om de gegeven reeks producten in groepjes van maximaal vier producten aan te kopen, zoals omschreven in de inleiding van deze opgave.
- Schrijf een functie winst die de winst teruggeeft die je maakt als je de strategie gebruikt waarbij je meerdere keren langs de kassa passeert, ten opzichte van de prijs die je zou betalen als je slechts één keer langs de kassa zou passeren.

Voorbeeld

```
>>> prijzen = [3.23, 5.32, 8.23, 2.23, 9.98, 7.43, 6.43, 8.23, 4.23]
>>> samen(prijzen)
49.85
>>> groeperen(prijzen)
[(9.98, 8.23, 8.23, 7.43), (6.43, 5.32, 4.23, 3.23), (2.23,)]
>>> gegroepeerd(prijzen)
44.65
>>> winst(prijzen)
5.2
```