Divers log

Preparation

In Python functions can be defined that one can call with an arbitrary number of arguments. These arguments are bundled into tuple and assigned to the parameter that is preceded by an asterisk (*) in the definition of the function. Zero or more normal arguments may occur for these variable arguments. This is illustrated in the Python session below.

```
>>> def sum(*terms):
...     result = 0
...     for term in terms:
...         result += term
...     return result
...
>>> sum()
0
>>> sum(1, 2, 3, 4, 5)
15
```

The reverse situation may also occur. Namely that arguments are already in a list or tuple but need to be unpacked before calling a function that wants them as separate positional arguments. The unpacking can be done by placing an asterisk (*) for the list or tuple that is given to the function as an argument. This is illustrated in the Python session below.

```
>>> terms = [1, 2, 3, 4, 5]
>>> sum(*terms)
15
>>> sum(*range(101))
5050
```

Look at the Python manual for more examples of *tuple packing* and *tuple unpacking*.

Problem

Water creates extra pressure when diving. The regulator in a tank has been specially designed to provide oxygen with the same pressure as the surrounding water pressure. That means that if a diver fills his lungs at a depth of 33 feet (1 foot = 0.305 metres) — where the pressure is about 2 psi (*pound-force per square inch*) — he uses twice as much air as he would at the surface. A full scuba tank at 33 feet (about 10 meters) deep last half as long as it would at the surface. A bottle that would last an hour at the surface, will be empty after 20 minutes at a depth of 66 feet, where the pressure is 3 psi.

The SAC ratio (*Surface Air Consumption*) is an important concept among scuba divers ("SCUBA" was originally an acronym for *self contained underwater breathing apparatus*, but is now commonly used as a word in itself) in order to estimate how much oxygen they will use at a certain depth. The amount of air a diver breathes depends on several factors: water temperature, physical condition, manner of breathing, experience of the diver, level of effort and amount of stress. In cold conditions and low visibility, an inexperienced diver will have a significantly higher SAC ratio than an experienced diver in the same conditions.

After each dive, the amount of oxygen a diver consumes at a certain depth is normalized to his

oxygen consumption at the surface. In order to do so, the following formula is used $$ \text{SAC} = \frac{33(s - f)}{t(d + 33)} $$ where d represents the depth (in feet) to which was plunged , s is the pressure in the scuba tank (in psi) for the dive, f is the pressure in the scuba tank (in psi) after the dive and t is the length of the dive (in minutes). Typically, the SAC-ratio of an individual diver is levelled over multiple dives.

Assignment

Define a class `Dive` of which the objects have various attributes that record the data of a single dive, such as the pressure before and after the dive, the start time and end time of the dive, and the depth of the dive. Typical values are an initial pressure of 3000 psi, an ultimate pressure of 700 psi, a depth of 30 to 80 feet, and a duration of 30 minutes (at 80 feet) to 60 minutes (at 30 feet). The SAC-ratio is typically between 10 and 20. The class `Dive` must also have a method `Dive.SACratio()`, which can be used to calculate the SAC ratio of that dive. The class Dive must support the following methods:

`Dive.__init__(self, initialpressure, ultimatepressure, starttime, endtime, depth)`

The initializaing method `__init__` must ensure that objects of the class `Dive` are initialized with attributes that indicate the pressure at the beginning and at the end of the dive (given as floats expressed in psi), the start time and end time of the dive, and the depth of the dive (given as floats, expressed in feet). To specify data from the dive more easily, the times are given as a string of the format `hh:mm`. The initializing method will therefore have to use string functions in order to cut the number of hours and the number of minutes (after midnight) from the string. To determine the duration of a dive, the method must normalize time to a number of minutes since midnight based on the calculation $uu \times 60 + mm$. If the start and end times were converted to a number of minutes since midnight, we can subtract the two times to determine the duration of the dive (in minutes). You may assume here that there are no dives over midnight. It is also permitted to add a helper method `Dive.determineDuration()` to the class Dive, in which this calculation for the duration of a dive is executed.

`Dive.SACratio(self)`

The method `SACratio` prints the SAC ratio of the dive, based on the circumstances.

To represent a sequence of consecutive dives, you will also define a class `Log`. This log can be initialized with the data from the sequence dive, and afterwards, individual dives can also be added to the log. Based on the log, you must be able to calculated the mean SAC ratio of the diver. The class `Log` must support the following methods:

`Log.__init__(self, *dive)`

The initializing method `__init__` must ensure that objects from the class Log can be initialized with a list of dives (objects from the class `Dive`). To add these dives as positional parameters to the initializing method, the parameter `*dives`, which allows parameters to be bundled in a list.

`Log.newDive(self, dive)`

The method `newDive` adds a given dive (object of the class `Dive`) in the back of the log.

`Log.meanSACratio(self)`

The method meanSACratio calculates the mean SAC ratio of a diver over all dives in his log, and print the mean result.

Example

```
>>> dive = Dive(3100, 1300, '11:52', '12:45', 35)
>>> dive.SACratio()
16.4816870144
>>> dive = Dive(2700, 1000, '11:16', '12:06', 40)
>>> dive.SACratio()
15.3698630137
>>> dive = Dive(2800, 1200, '11:26', '12:06', 60)
>>> dive.SACratio()
14.1935483871
>>> dive = Dive(2800, 1150, '11:54', '12:16', 95)
>>> dive.SACratio()
19.3359375

>>> log = Log(
...     Dive(3100, 1300, '11:52', '12:45', 35),
...     Dive(2700, 1000, '11:16', '12:06', 40),
...     Dive(2800, 1200, '11:26', '12:06', 60)
... )
>>> log.meanSACratio()
15.3483661384
>>> log.newDive(Dive(2800, 1150, '11:54', '12:16', 95))
>>> log.meanSACratio()
16.3452589788
```

Voorbereiding

In Python kunnen functies gedefinieerd worden die men kan aanroepen met een willekeurig aantal argumenten. Deze argumenten worden gebundeld in tuple en toegekend aan de parameter die in de definitie van de functie wordt voorafgegaan door een sterretje (*). Voor deze variabele argumenten kunnen nul of meer gewone argumenten voorkomen. Dit wordt geïllustreerd in onderstaande Python sessie.

```
>>> def som(*termen):
...     resultaat = 0
...     for term in termen:
...         resultaat += term
...     return resultaat
...
>>> som()
0
>>> som(1, 2, 3, 4, 5)
15
```

De omgekeerde situatie kan ook voorkomen. Namelijk dat argumenten reeds in een lijst of een tuple zitten, maar moeten uitgepakt worden voor het aanroepen van een functie die ze als afzonderlijke positionele argumenten wil meekrijgen. Dit uitpakken kan gebeuren door een sterretje (*) voor de lijst of het tuple te plaatsen dat als argument aan de functie wordt doorgegeven. Dit wordt geïllustreerd in onderstaande Python sessie.

```
>>> termen = [1, 2, 3, 4, 5]
>>> som(*termen)
```

```
15
>>> som(*range(101))
5050
```

Zie de Python handleiding voor meer voorbeelden van *tuple packing* en *tuple unpacking*.

Probleemomschrijving

Water zorgt voor extra druk bij het duiken. De ademautomaat in een duikfles is dan ook speciaal ontworpen om zuurstof te leveren aan dezelfde druk als de omringende waterdruk. Dat betekent dat als een duiker de longen vult op een diepte van 33 voet (1 voet = 0,305 meter) — waar de druk ongeveer 2 psi (*pound-force per square inch*) bedraagt — hij tweemaal zoveel lucht gebruikt als aan de oppervlakte. Een volle duikfles zal op 33 voet (ongeveer 10 meter) diep dus slechts half zo lang meegaan als aan de oppervlakte. Een fles die aan de oppervlakte pas na 1 uur zou opgebruikt zijn, is op 66 voet diep — waar de druk 3 psi bedraagt — al na 20 minuten leeg.

De SAC-verhouding (*Surface Air Consumption*) is een belangrijk begrip onder scubaduikers ("SCUBA" was oorspronkelijk een acroniem voor *self contained underwater breathing apparatus*, maar wordt nu algemeen gebruikt als een woord op zichzelf) om te kunnen inschatten hoeveel zuurstof ze zullen gebruiken op een bepaalde diepte. De hoeveelheid lucht die een duiker inademt is immers afhankelijk van verschillende factoren: watertemperatuur, lichamelijke conditie, manier van ademen, ervaring van de duiker, mate van inspanning en hoeveelheid stress. Een onervaren duiker zal in koude omstandigheden en bij slecht zicht een aanzienlijk hogere SAC-verhouding hebben dan een ervaren duiker in dezelfde omstandigheden.

Na elke duik wordt de hoeveelheid zuurstof die een duiker op een bepaalde diepte verbruikt heeft genormaliseerd naar zijn zuurstofverbruik aan de oppervlakte. Hiervoor wordt gebruik gemaakt van de volgende formule $$ \text{SAC} = \frac{33(s - f)}{t(d + 33)} $$ waarbij d de diepte (in voet) voorstelt waarop werd gedoken, s de druk in de duikfles (in psi) voor de duik, f de druk in de duikfles (in psi) na de duik en t de lengte van de duik (in minuten). Doorgaans wordt de SAC-verhouding van een individuele duiker uitgemiddeld over verschillende duiken heen.

Opgave

Definieer een klasse `Duik` waarvan de objecten verschillende attributen hebben die de gegevens van één enkele duik bijhouden, zoals de druk voor en na de duik, het aanvangsuur en einduur van de duik, en de diepte waarop gedoken werd. Typische waarden zijn een startdruk van 3000 psi, een einddruk van 700 psi, een diepte van 30 tot 80 voet en een duur van 30 minuten (bij 80 voet) tot 60 minuten (bij 30 voet). De SAC-verhouding ligt typisch tussen 10 en 20. De klasse `Duik` moet ook een methode `Duik.SACverhouding()` hebben, waarmee de SAC-verhouding voor die duik kan berekend worden. De klasse Duik moet dus ondersteuning bieden voor volgende methoden:

`Duik.__init__(self, startdruk, einddruk, starttijd, eindtijd, diepte)`

De initialisatiemethode `__init__` moet ervoor zorgen dat objecten van de klasse `Duik` geïnitialiseerd worden met attributen die de druk aan het begin en het einde van de duik aangeven (gegeven als floats uitgedrukt in psi), de aanvangstijd en eindtijd van de duik, en de diepte waarop werd gedoken (gegeven als floats, uitgedrukt in voet). Om makkelijker de gegevens van de duik te kunnen opgeven, worden de tijden doorgegeven als een string in het formaat `uu:mm`. De initialisatiemethode zal dus stringfuncties moeten gebruiken om het aantal uren en het aantal

minuten (na middernacht) uit de string te knippen. Om de duur van een duik te bepalen, moet de methode dus de tijd normaliseren naar een aantal minuten sinds middernacht op basis van de berekening $uu \times 60 + mm$. Als de begin- en eindtijd werden omgezet naaar een aantal minuten sinds middernacht, dan kunnen we deze twee tijden van elkaar aftrekken om de duur van de duik (in minuten) te bepalen. Je mag er hierbij van uitgaan dat er nooit over middernacht heen gedoken wordt. Het is ook toegelaten om een hulpmethode Duik.bepaalDuur() toe te voegen aan de klasse Duik, waarin deze berekening voor de duur van een duik wordt uitgevoerd.

Duik.SACverhouding(self)

De methode SACverhouding geeft de SAC-verhouding van de duik terug, op basis van de omstandigheden waarin gedoken werd.

Om een reeks opeenvolgende duiken te kunnen voorstellen, definieer je ook een klasse Logboek. Een dergelijk logboek kan geïnitialiseerd worden met de gegevens van een reeks duiken, en achteraf kunnen ook nog individuele duiken aan het logboek toegevoegd worden. Op basis van het logboek moet de gemiddelde SAC-verhouding van de duiker kunnen berekend worden. De klasse Logboek moet dus ondersteuning bieden aan de volgende methoden:

Logboek.__init__(self, *duiken)

De initialisatiemethode __init__ moet ervoor zorgen dat objecten van de klasse Logboek kunnen geïnitialiseerd worden met een lijst van duiken (objecten van de klasse Duik). Om deze duiken als positionele parameters te kunnen meegeven aan de initialisatiemethode wordt gebruik gemaakt van de parameter *duiken, waardoor de parameters in een lijst gebundeld worden.

Logboek.nieuweDuik(self, duik)

De methode nieuweDuik voegt een gegeven duik (object van de klasse Duik) toe achteraan het logboek.

Logboek.gemiddeldeSACverhouding(self)

De methode gemiddeldeSACverhouding berekent de gemiddelde SAC-verhouding van een duiker over alle duiken in diens logboek heen, een geeft dit gemiddelde als resultaat terug.

Voorbeeld

```
>>> duik = Duik(3100, 1300, '11:52', '12:45', 35)
>>> duik.SACverhouding()
16.4816870144
>>> duik = Duik(2700, 1000, '11:16', '12:06', 40)
>>> duik.SACverhouding()
15.3698630137
>>> duik = Duik(2800, 1200, '11:26', '12:06', 60)
>>> duik.SACverhouding()
14.1935483871
>>> duik = Duik(2800, 1150, '11:54', '12:16', 95)
>>> duik.SACverhouding()
19.3359375

>>> logboek = Logboek(
...     Duik(3100, 1300, '11:52', '12:45', 35),
...     Duik(2700, 1000, '11:16', '12:06', 40),
```

```
...     Duik(2800, 1200, '11:26', '12:06', 60)
... )
>>> logboek.gemiddeldeSACverhouding()
15.3483661384
>>> logboek.nieuweDuik(Duik(2800, 1150, '11:54', '12:16', 95))
>>> logboek.gemiddeldeSACverhouding()
16.3452589788
```