Trail

If you look up a route between two points p_0 and p_n with Google Maps, the itinerary is given based on a number of intermediate points $$p_0, p_1, p_2, \ldots, p_n$$ If we write the distance between two consecutive points p_i and p_{i+1} as $d(p_i, p_{i+1})$, the total distance of the route equals $$\sum_{i=0}^{n - 1} d(p_i, p_{i+1})$$ If the route ends back in its starting point, we have to add the distance $d(p_n, p_0)$ at the end.

In this assignment, we represent the co-ordinates of a point p as a tuple (x, y). Based on this representation, the distance between two points $p_0 = (x_0, y_0)$ and $p_1 = (x_1, y_1)$ can be defined in different ways. The most intuitive distance is the *Euclidean distance*, that corresponds with the distance from a bird's-eye view. In practice, however, this distance isn't always as useful.

The *Manhattan-distance* is an alternative distance that gets its name from the typical street pattern of Manhattan. There, all streets either stand perpendicular with regard to each other or they are parallel. In this case, the distance is given as the sum of the subtraction of the x co-ordinates and the subtraction of the y co-ordinates. This is equal to the road you would take from one point to another if you were only allowed to walk parallel with the x and y axis.

The *Chebyshev distance* or the *chessboard distance* gets its name from the fact that it corresponds with the amount of moves a chess piece king needs to cross a discrete grid. This number is given by the maximum of the subtraction of the x co-ordinates and the subtraction of the y co-ordinates.

In the table below you will find an overview of the various formulas for the distance between two points (x_0, y_0) en (x_0, y_0) as defined by the linear measures above.

name	formula
Euclidian distance	$\sqrt{(x_0-x_1)^2 + (y_0-y_1)^2}$
Manhattan distance	$\|x_0-x_1\| + \|y_0-y_1\|$
Chess board distance	$\max(\|x_0-x_1\|,\|y_0-y_1\|)$

Preparation

In Python, functions are objects themselves, which is the reason why you can use them like any other object. In particular, you can appoint functions to variables or pass them as an argument to other function. Look at the three functions below as an example.

```
def repeat(value, function, amount):
    for i in range(amount):
        value = function(amount)
    return value

def increment(value):
```

```
    return value + 1

def decrement(value):
    return value - 1
```

Below you will find an example of how these functions can be used. Verify how Python reacts if you consecutively execute the following instruction within the interactive Python session in which you first have the functions above defined. Make sure you understand why you get a certain value and what happens exactly in the interactive session before moving on to the actual assignment.

```
>>> increment(5)
>>> decrement(101)
>>> repeat(5, increment, 3)
>>> repeat(5, decrement, 3)
>>> repeat(5, decrement, increment(3))
```

Assignment

In this assignment, we represent a point as a tuple of two real numbers the indicate the x and y co-ordinates of a point in a surface. Asked:

1. Write three functions `euclidean`, `manhattan` and `chessboard` that correspond with the concepts of the same name for the concept distance as they are defined in the table. To each of these functions two points should be given as an argument. The functions must print as a result the distance between the points given, in accordance with the distance formula.
2. Write a function track. This function has an obligatory parameter `points` to which a list of points must be given the function must print the total distance if we travel from one point on the list to another. Apart from the obligatory parameter, the function also has two optional parameters:
    - `cycle`: a Boolean value can be given to this parameter, indicating whether the end of the tour is equal to its starting point (`True`) or not (`False`). The standard value of this parameter is `False`.
    - `distance`: a distance function can be given to this parameter which is used for the calculation of the total distance between two consecutive points. If no value has been given for this distance, the Euclidean distance is used as a standard.

Example

```
>>> euclidean((42.36, 56.78), (125.65, 236.47))
198.05484139500354
>>> manhattan((42.36, 56.78), (125.65, 236.47))
262.98
>>> chessboard((42.36, 56.78), (125.65, 236.47))
179.69

>>> track([(6.59, 6.73), (4.59, 5.54), (5.33, -13.98)])
21.861273201261746
>>> track(cycle=True, points=[(6.59, 6.73), (4.59, 5.54), (5.33, -13.98)])
42.60956710702662
>>> track([(6.59, 6.73), (4.59, 5.54), (5.33, -13.98)], distance=manhattan)
23.45
>>> track([(6.59, 6.73), (4.59, 5.54), (5.33, -13.98)], cycle=True, distance=manhattan)
45.42
```

Als je een route tussen twee punten p_0 en p_n opzoekt via Google Maps, dan wordt de routebeschrijving gegeven op basis van een aantal tussenpunten $$p_0, p_1, p_2, \ldots, p_n$$ Als we de afstand tussen twee opeenvolgende punten p_i en p_{i+1} noteren als $d(p_i, p_{i+1})$, dan is de totale afstand van de route gelijk aan $$\sum_{i=0}^{n - 1} d(p_i, p_{i+1})$$ Als de route terug eindigt op zijn beginpunt, dan moeten we hier finaal nog de afstand $d(p_n, p_0)$ bij optellen.

In deze opgave stellen we de coördinaten van een punt p voor als een tuple (x, y). Op basis van deze voorstelling kan de afstand tussen twee punten $p_0 = (x_0, y_0)$ en $p_1 = (x_1, y_1)$ op verschillende manieren gedefinieerd worden. De meest intuitieve afstand is de *Euclidische afstand*, die overeenkomt met de afstand in vogelvlucht. In de praktijk is deze afstand echter niet altijd even bruikbaar.

De *Manhattan-afstand* is een alternatieve afstand die zijn naam dankt aan het typische stratenpatroon van Manhattan. Daar staan alle straten ofwel loodrecht op elkaar of lopen ze evenwijdig met elkaar. In dit geval wordt de afstand gegeven als de som van het verschil van de x-coördinaten en het verschil van de y-coördinaten. Dit komt overeen met de weg die je zou afleggen om van het ene punt naar het andere punt te gaan als je enkel evenwijdig met de x-as en de y-as zou mogen wandelen.

De *Chebyshev-afstand* of de *schaakbordafstand* dankt zijn naam aan het feit dat ze op een discreet rooster overeenkomt met het aantal zetten die het schaakstuk koning nodig heeft om van het ene punt naar het andere punt te gaan. Dit aantal wordt gegeven door het maximum van het verschil van de x-coördinaten en het verschil van de y-coördinaten.

In onderstaande tabel vind je een overzicht van de verschillende formules voor de afstand tussen twee punten (x_0, y_0) en (x_0, y_0) zoals die wordt gedefinieerd door bovenvernoemde afstandsmaten.

naam	formule
Euclidische afstand	$\sqrt{(x_0-x_1)^2 + (y_0-y_1)^2}$
Manhattan-afstand	$\|x_0-x_1\| + \|y_0-y_1\|$
schaakbordafstand	$\max(\|x_0-x_1\|,\|y_0-y_1\|)$

Voorbereiding

In Python zijn functies zelf ook objecten, waardoor je ze kunt gebruiken zoals alle andere objecten. In het bijzonder kun je functies toekennen aan variabelen of doorgeven als argument aan andere functies. Bekijk bijvoorbeeld onderstaande drie functies.

```
def herhalen(waarde, functie, aantal):
    for i in range(aantal):
        waarde = functie(waarde)
    return waarde

def verhogen(waarde):
```

```
    return waarde + 1

def verlagen(waarde):
    return waarde - 1
```

Hieronder staat een voorbeeld van hoe deze functies gebruikt kunnen worden. Ga na hoe Python reageert als je achtereenvolgens de volgende instructies uitvoert binnen een interactieve Python sessie waarin je eerst zorgt dat bovenstaande functies gedefinieerd werden. Verzeker jezelf ervan dat je goed begrijpt waarom je een bepaalde waarde krijgt en wat er juist gebeurt in de interactieve sessie vooraleer je verder gaat met de eigenlijke opgave.

```
>>> verhogen(5)
>>> verlagen(101)
>>> herhalen(5, verhogen, 3)
>>> herhalen(5, verlagen, 3)
>>> herhalen(5, verlagen, verhogen(3))
```

Opgave

In deze opgave stellen we een punt voor als een tuple van twee reële getallen die de x- en de y-coördinaat van een punt in het vlak voorstellen. Gevraagd wordt:

1. Schrijf drie functies `euclidisch`, `manhattan` en `schaakbord` die corresponderen met de gelijknamige concepten voor het begrip afstand zoals die in bovenstaande tabel gedefinieerd werden. Aan elk van deze functies moeten twee punten als argument doorgegeven worden. De functies moeten als resultaat telkens de afstand teruggeven tussen de twee gegeven punten, overeenkomstig de corresponderende afstandsformule.
2. Schrijf een functie `parcours`. Deze functie heeft een verplichte parameter `punten` waaraan een lijst van punten moet doorgegeven worden. De functie moet de totale afstand teruggeven die afgelegd wordt als men achtereenvolgens alle punten uit de lijst aandoet. Naast deze verplichte parameter, heeft de functie ook nog twee optionele parameters:
    - `cycle`: aan deze parameter kan een Booleaanse waarde doorgegeven worden die aangeeft of men op het einde van de rit terugreist naar het beginpunt (`True`) of niet (`False`). De standaardwaarde van deze parameter is `False`.
    - `afstand`: aan deze parameter kan een afstandsfunctie doorgegeven worden, die bij de berekening van de totale afstand gebruikt wordt om de afstand tussen twee opeenvolgende punten te bepalen. Als geen waarde wordt doorgegeven voor deze afstand, dan wordt standaard de Euclidische afstand gebruikt.

Voorbeeld

```
>>> euclidisch((42.36, 56.78), (125.65, 236.47))
198.05484139500354
>>> manhattan((42.36, 56.78), (125.65, 236.47))
262.98
>>> schaakbord((42.36, 56.78), (125.65, 236.47))
179.69

>>> parcours([(6.59, 6.73), (4.59, 5.54), (5.33, -13.98)])
21.861273201261746
>>> parcours(cycle=True, punten=[(6.59, 6.73), (4.59, 5.54), (5.33, -13.98)])
42.60956710702662
>>> parcours([(6.59, 6.73), (4.59, 5.54), (5.33, -13.98)], afstand=manhattan)
```

```
23.45
>>> parcours([(6.59, 6.73), (4.59, 5.54), (5.33, -13.98)], cycle=True, afstand=manhattan)
45.42
```