Analyse af Løbsfil

Projektorienteret python programmeringsopgave

Formål

Du skal lave en lidt mere omfattende opgave hvor selv skal vælge værktøjer og fremgangsmåde for at lave en løsning på en opgave hvor dine valg har stor betydning for løsningens udformning.

Du skal selv vælge hvilke komponenter, i form af programbiblioteker (libraries) eller anden kode, du henter fra andre, og hvor meget du skriver selv. Her er det lige så vigtigt at vælge klogt, så du får en stærk og fleksibel løsning, som kan udvides i en eventuel fremtid, som det er vigtigt at vise at du selv kan skrive programkode overskueligt, dokumenteret og effektivt.

Datagrundlaget udleveres, og er dermed entydigt og givet.

Løsningen skal være et program der beregner en bestemt problemstilling, og måske nogle flere.
Noget af svaret har således et helt bestemt output, andre måske et lidt mere valgfrit udseende.

Du skal altså arbejde med at

- analysere problemfeltets usecases,
- datastrukturer der benyttes,
- forstå brugerens domæne (godt nok til at løse opgaven),
- søge viden om problemfelt, brugerens domæne og relevante datastrukturer og brugbare komponenter,
- vælge komponenter (aka libraries),
- integrere komponenter i programmet,
- programmere,
- og dokumentere

Det forventes at du løser opgaven med almindelig python syntakss, med loops og flow-control strukturer. Nogenlunde som du har gjort i starten af ugen.

Case-beskrivelse

Jeg forsøger at lære orienteringsløb. Derfor deltager jeg i nogle orienteringsløb af og til. Til sådanne løb er jeg udfordret på flere måder; jeg skal kunne løbe rimeligt hurtigt, jeg skal kunne læse kort og finde vej hen til posterne, og jeg skal kunne gøre begge dele samtidigt. Det er en udfordring at løbe i terrænet, som oftest er en skov, hvor der er

træer, underskov, grene, grøfter og alt muligt skrammel. Nogle løb afholdes i byområder, og her der stier, fortove, hegn, trapper, porte, huse, gader, biler, cykler, fodgængere og alt muligt andet, man skal uden om.

Et orienteringsløb foregår ved at jeg får udleveret et kort lige inden start. På kortet er et område med indtegnede cirkler, der er en post inden for hver cirkel. Når tiden starter, må jeg vende kortet, som jeg altså ikke har set inden start. Herefter gælder det så om at finde og checke ind på posterne i den rækkefølge de er på kortet. Et supplement til kortet er en liste med postdefinitioner, som både er på arket med kortet og på et ekstra papir, som udleveres inden, og jeg har i en holder på armen. Hver post checkes af med en elektronisk brik jeg har på hånden. på hver post findes en kontrolstation. Når jeg checker ind på posten, registreres brik id, kontrol id, og tidspunkt. Faktisk lagres det på brikken, så det kan aflæses umiddelbart efter jeg kommer i mål. Når er tilbage i mål, kontrolleres om jeg har checket alle poster, og den samlede tid, tæller i konkurrencen.

Jeg vil gerne have noget viden om hvor jeg kan gøre mit løb bedre, så jeg tænker at det kan jeg få ved at analysere det data der opsamles under løbet.
Nogen steder lykkes det ikke at løbe med hastigheden kommer ned i gang-hastighed, enten pga. udfordringer i terrænet eller fordi jeg er tvivl om retningen. Nogen gange står jeg stille og kigger på kortet, og nogen gange på landskabet, for at genkende kort og landskab.

Af og til løber jeg ikke direkte frem mod posten, enten fordi jeg vælger en rute hen til posten, som går uden om forhindringer (f.eks. en sø eller en bygning), en rute der følger en vej eller sti, hvor jeg kan løbe hurtigere, eller jeg orienterer mig simpelthen forkert eller kan ikke finde posten. Her kan det være interessant at vide noget om ruten går en omvej i god hastighed, eller lige forbi posten, eller i en helt anden retning...

Produktmål Jeg vil gerne vide hvor meget af løbet reelt er foregået i løb, hvor meget i gang og hvor meget jeg har stået stille. Her skal der bruges nogle grænseværdier; løb defineres som tempo hurtigere end 10 minutter pr. kilometer,

$løb : tempo < 10\frac{min}{km}$

$gang : 10\frac{min}{km} < tempo < 50\frac{min}{km}$

$stå : tempo > 50\frac{min}{km}$

Bemærk at løbere ofte beskriver farten som tempo forstået som minutter pr kilometer [1]. Se afsnittet x under z.

Delmål

1. Skriv en funktion som omregner fra tempo (min/km) til hastighed (m/s)
2. indlæs løbsfilen xxx.fit
3. Skriv en funktion som beregner distance og tid mellem hvert målepunkt, i en ny liste af linjestykker.
4. Beregn den samlede fordeling mellem tempo-zoner

    A. I en funktion som beregner hastighed mellem hvert målepunkt. Egentlig i listen med linjestykker.
    B. I en funktion som
        a. beregner sum af afstand og sum af tid i hver af de tre tempo-zoner; løb, gang og stå
        b. beregn hvor mange procent af tiden, der samlet foregår hver tempo-zone
5. Udskriv på skærmen; tid, afstand og procent i udskrift i pæn udskrift som tabel 1
6. Lav en ny funktion som, ud fra tidspunktet for post-kontrollerne i tabel x,
    A. beregn afstanden i fugleflugt, mellem posterne
    B. opdel listen af målepunkter i lister med punkter for stræk mellem hver post
    C. beregner afstanden som faktisk er tilbagelagt på hvert stræk mellem posterne
7. Ligesom i punkt 4, lav en ny funktion som beregner fordelingen mellem tempo-zoner, i hvert stræk
    A. beregn hastighed mellem hvert målepunkt
    B. beregn sum af afstand og sum af tid i hver af de tre tempo-zoner; løb, gang og stå
    C. beregn hvor mange procent af tiden, der samlet foregår hver tempo-zone
8. beregn forholdet mellem afstand på hvert stræk, og den reelt tilbagelagte afstand på strækket, også en funktion
9. beregn gennemsnitshastigheden (og tempo) på hvert stræk
10. beregn den teoretiske gennemsnitshastighed hvis strækket for foregået i fugleflugtslinje
11. beregn forholdet mellem gennemsnitshastighederne i pkt 10 og 11
12. udskriv beregningerne for 8-12 for hvert stræk, på en pæn måde, i stil med punkt 5.

Datagrundlaget

Det væsentligeste datagrundlag er `fit`-filen. Her er hele turen rundt tracket. Desuden er der en fil med tiderne jeg har "klippet" på kontrollerne.

Løbsfilen

Du får udleveret med fil med data fra mit løbeur. Filen indeholder data om løbeturen optaget undervejs. På turen samples tid, position, højde over havet, puls og skridtfrekvens. Filen kan enten overføres direkte fra uret med et USB kabel, eller fra GARMINs medfølgende app og website. Fra appen er det letteste format GPX, mens det på uret ligger i formatet FIT. .fit Filformatet FIT er udviklet af Garmin[2], og det format som løbedata gemmes i direkte på Garmin Uret. Andre ure har sikkert andre formater. En fil fil kan læses eller konverteres med forskellige programmer, f.eks. FIT ConVerter[3], eller python modulerne fit2gpx[4], fitdecode[5], fitparse[6].

Du får også udlevet et python modul `fit-file` til at læse `fit`-filen.

I artiklen “Parsing fitness tracker data with Python” forklarer Alan Bunbury, hvordan forskellige løbefilformater kan læses [7], jeg har brugt hans eksempel (med få ændringer) til at dumpe indholdet af en .FIT fil [8].

Herunder ser du et eksempel på hvordan fit filen er indlæst, og en `dict` fra listen er udlæst:

In [7]:
```
from fit_file import read
fname = "data/hok_klubmesterskab_2022/CA8D1347.FIT"
points = read.read_points(fname)
print(len(points))
print(points[300])
```

```
1210
{'latitude': 55.91865699738264, 'longitude': 11.664552418515086, 'altitude': -
8.0, 'timestamp': datetime.datetime(2022, 10, 8, 13, 32, 58, tzinfo=<DstTzInfo
'Europe/Copenhagen' CEST+2:00:00 DST>), 'heart_rate': 162, 'cadence': 76}
```

Tempo vs hastighed

Som nævnt beskrives farten i løb ofte, af løbere, som minutter pr. kilometer. Jeg antager at det er lettere at overskue, så at sige, i farten. Hvor mange minutter er der gået siden sidste kilometersten eller omgang.

Casper Mikkelsen redegør udmærket for mange af de værdier og enheder der bruges til at beskrive løb mv, artiklen Pace Beregner: Din Hastighed I Km/t, M/s & Min/km (Beregner) [1]

Tempobegrebet er anderledes end hastighed, på den måde at et lille tempo-tal udtryk for høj fart, mens ved højt tempo-tal er langsomt.

Her kan du se hvordan *tempo* i min/km, omregnes til *hastighed* i m/s, hvor p er pace, dvs. tempoet i $\frac{min}{km}$:

$$p \cdot \frac{min}{km}$$

$$= p\frac{60 \cdot s}{1000 \cdot m} = \frac{\frac{1000 \cdot m}{60 \cdot s}}{p} = \frac{\frac{1000}{60}}{p} \cdot \frac{m}{s}$$

$$= \frac{16.667}{p} \cdot \frac{m}{s}$$

Så, i matematisk format, ser funktionen sådan her ud:

$$v(p) = \frac{(\frac{1000}{60})}{p}$$

Altså bliver de to grænseværdier:

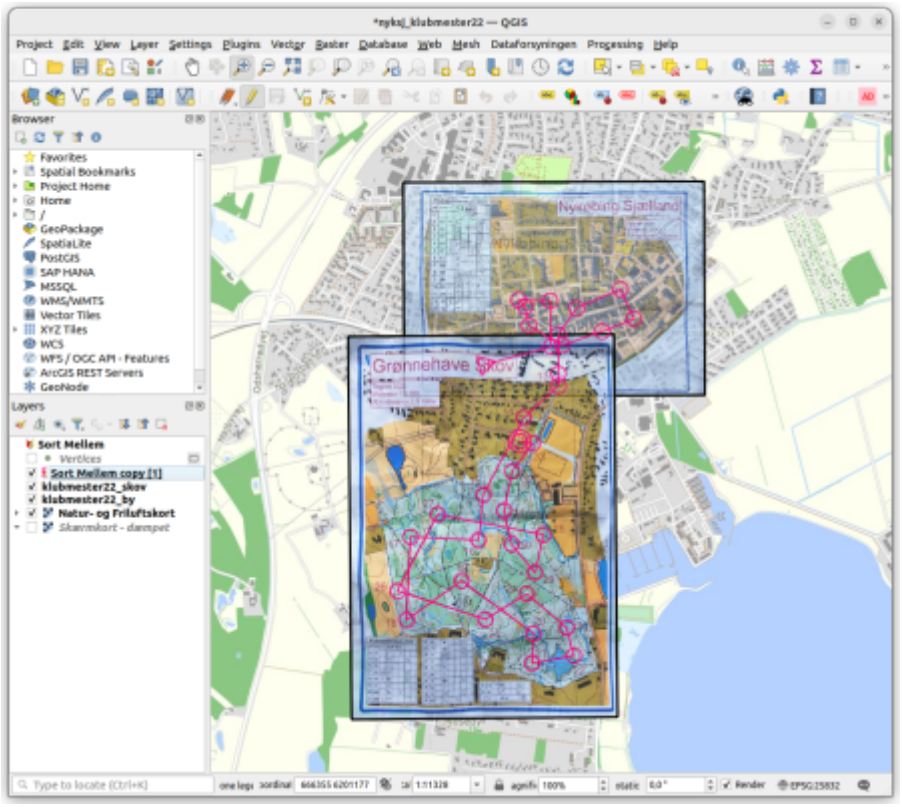
- løb: $10\frac{min}{km} = 1.666667 \cdot \frac{m}{s}$
- gang: $50\frac{min}{km} = 0,333333 \cdot \frac{m}{s}$

Du skal selv implementere funktionen `pace2velocity()`.

```
10 min/km = 1.666667 m/s
50 min/km = 0.333333 m/s
```

Poster

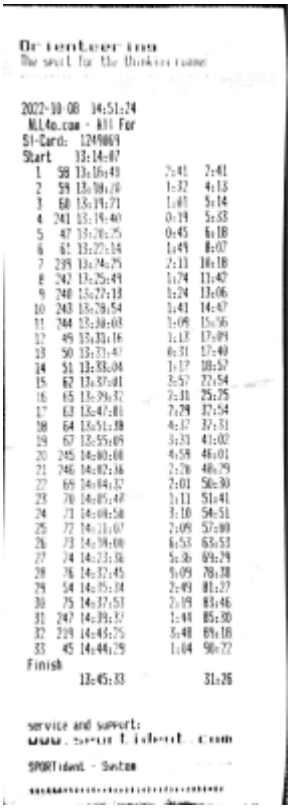
I .fit-løbsfilen kan man ikke se om jeg på turen kommer til en post. Man kan ikke se om posten checkes af (elektronisk), eller om jeg løber forbi, uden at se den. Derfor er der to dataset mere i opgaven. En liste af posternes positioner, og en udskrift fra SportIdent, med tider for hvornår jeg har været ved hver enkelt post.



Posternes positioner er estimeret fra kortet, i gis-programmet QGIS, Se billede. Positionerne er eksporteret til filen poster.csv. (note: poster.cvs, mangler ...)

En af måderne at indlæse en kommasepareret fil (.csv), er at bruge pythons csv mudul [9]

Den anden fil er tiderne fra printet herunder. I filen kontroltider.csv er posternes nr og den tid der er noteret på hver linje. For at indlæse csv-filen, og konvertere tiderne fra iso formatet til pythons datetime, kan du bruge nedenstående kode:



Billet herover er en kopi af print fra sportident systemet, med kontroltider på alle posterne. Bemærk at der er tidsstempel på alle posterne, som bør kunne sammenlignes med tiderne i løbsfilen.

In [12]:
```
import csv
import datetime

with open('data/hok_klubmesterskab_2022/kontroltider.csv', 'r',
          encoding='utf-8', newline='') as csvfile:
    reader = csv.DictReader(csvfile)
    postkontroller = [{ 'nr':row['nr'],
            'timestamp': datetime.datetime.fromisoformat(row['timestamp']) }
          for row in reader]
```

Find delstræk mellem poster

Du kan finde hvilke punkter det ligger mellem posterne, ved at sammenligne tiden på punkterne med tiden hvor postkontrollen er checket.

In [15]:
```
st_1 = [p for p in points if p['timestamp'].astimezone() < postkontroller[1]['ti
# her ses postkontrollens timestamp, og det sidste punkt fra fit filen der er fø
postkontroller[1]['timestamp'], st_1[-1]
```

Out[15]:
```
(datetime.datetime(2022, 10, 8, 13, 16, 48, tzinfo=datetime.timezone(datetime.t
imedelta(seconds=7200))),
 {'latitude': 55.921604661270976,
  'longitude': 11.666395431384444,
  'altitude': 4.800000000000011,
  'timestamp': datetime.datetime(2022, 10, 8, 13, 16, 40, tzinfo=<DstTzInfo 'Eu
rope/Copenhagen' CEST+2:00:00 DST>),
  'heart_rate': 155,
  'cadence': 56})
```

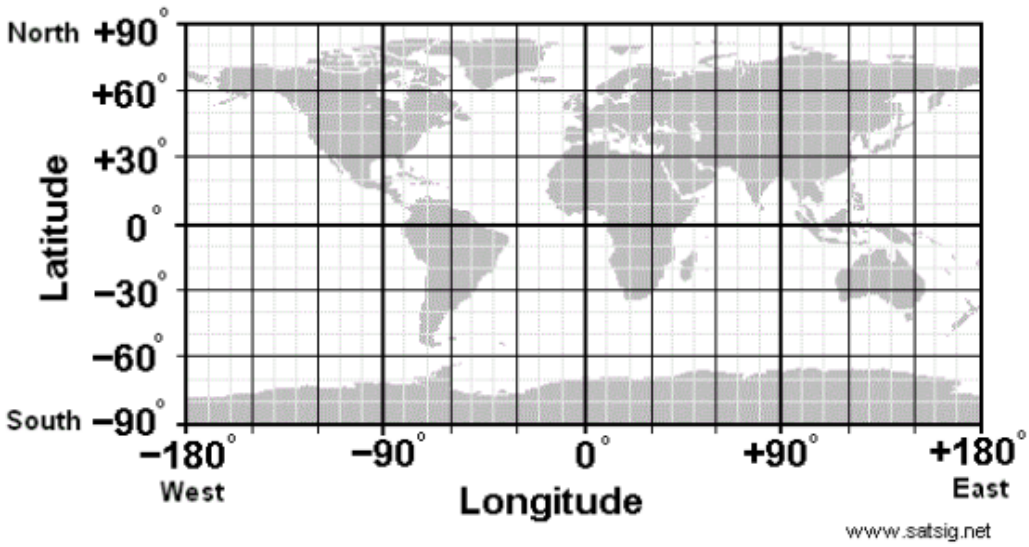
Så skal du altså "bare" lave et loop som gennemløber alle postkontrollerne og finder alle det punkter der ligger imellem. Det mellem liggende punkter skal indsættes i 34 nye lister (eller en liste med lister) så du kan beregne den samlede reele afstand og tid.

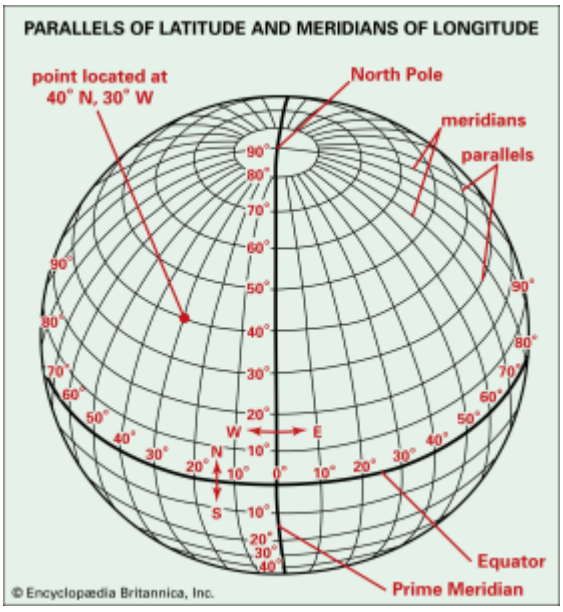
Koordinater i lat/lon og afstande

Positionerne på posterne og positionerne på målepunkterne i løbsfilen angives som longitude og latitude, eller på dansk længdegrad og breddegrad.

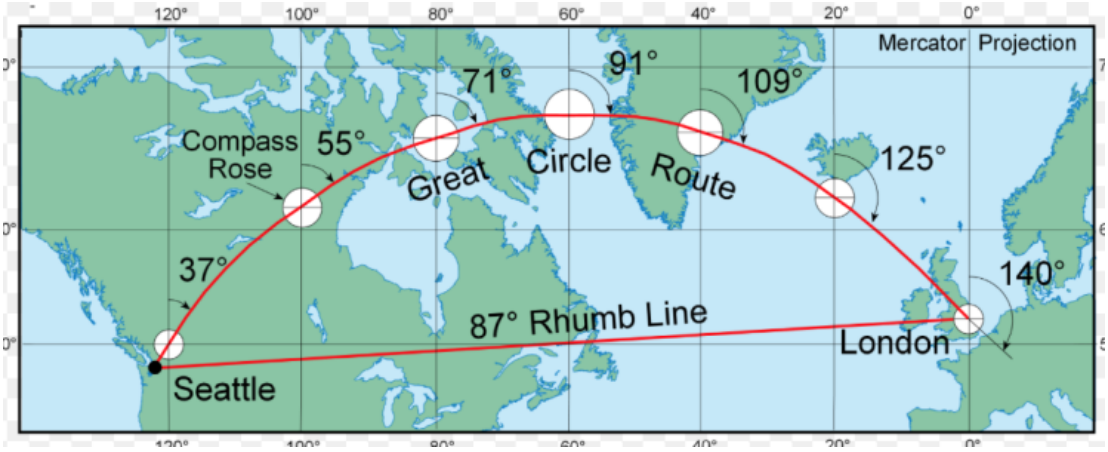
- stammer fra søkort
  - vinkeltro
  - ikke arealtro
  - ikke afstandstro
- Den korteste vej er ikke en ret linje
- god til kompasretninger
- godt til at bestemme position med sekstant og ur

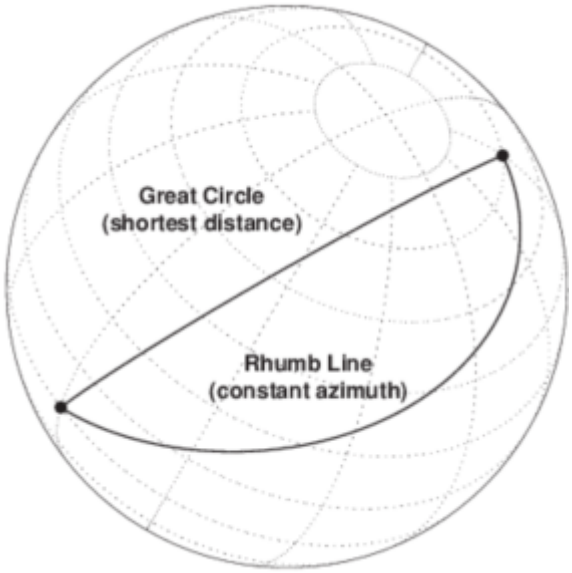
Se encyklopædierne [10], [11]





Den korteste linje er ikke en ret linje, på et søkort (med Mercantor projektion)[12]





Brug geopy!

Fordi det er ret krævende matematisk at beregne den reelle afstend mellem koordinater angivet i længde og breddegrad (latetude/longitude), så vælger jeg at bruge funktionen `distance` fra modulet `geopy` til dette:

In [ ]:
```
from geopy.distance import distance
lines = []
for i, p in enumerate(points[1:]):
    # bemærk at i starter på 0 selv om vi slicer med [1:]
    pp = points[i]                # previous_point
    dt = (p['timestamp'] - pp['timestamp']).seconds
    dd = distance( (pp['latitude'], pp['longitude']), (p['latitude'], p['longitu
    v = dd/dt

    line = {
        'start'             : pp,
        'end'               : p,
        'delta_time'        : dt,
        'delta_distance'    : dd,

# ... mere kode her ...
```

Udskrifter

Jeg vil gerne vide hvor godt jeg gør det, på løbet. F.eks. vil jeg gerne vide om jeg kan holde et tempo i løb eller jeg "går ned i gang", når det bliver svært at finde vej, eller tærrenet bliver hårdt. Derfor vil jeg gerne have en optælling af alle linjerne (mellem punkterne), så jeg kan se hvor meget der henholdsvis løb, gang, og stilstand, både opmålt i afstand, tid og procent (af tid).

Det skal stilles lidt pænt op, og kan f.eks. se sådan her ud:

```
          meters seconds % of time
Run:      5854.6    2251     39.9%
Walk:     3047.4    2977     52.8%
Idle        69.1     415      7.4%
Total:    8971.1    5643    100.0%
```

Når du har fundet de punkter der ligger mellem hver post, vil jeg gerne have oplysninger udskerevet for hvert delstræk, på samme måde som oven for, så jeg kan se hvilke stræk der gik godt og hvilke der er gået mindre godt.

Bedømmelse

Der er rigtigt mange delopgaver i denne opgave, så det er ikke sikkert at du kan nå at lave dem alle sammen.

I bedømmelsen, indgår både

- hvor mange opgaver der er løst fuktionelt korrekt,
- hvor læselig og forståelig koden er, logisk opbygget,
- og dokumenteret.

Desuden skal du også lave en lille beskrivelse og din løsning, som også kan fungere som en slags læse vejledningtil hvor jeg skal starte med at læse jeres program, hvordan men stater det. Beskriv også hvordan det er opbygget. Med funktioner og datastrukturer.

Hvis en gruppe afleverer sammen, indgår det i bedømmelsen at jeg forventer at en gruppe kan nå at lave mere.

Kilder

[1] C. Mikkelsen, “Pace Beregner: Din Hastighed I Km/t, M/s & Min/km (Beregner),” Nettofitness, Feb. 28, 2022. https://nettofitness.dk/a/pace-beregner (accessed Oct. 17, 2022).
[2] “FIT Protocol | FIT SDK | Garmin Developers.” https://developer.garmin.com/fit/protocol/ (accessed Oct. 16, 2022).

[3] C. P. UK, “FIT 2 GPX.” https://pinns.co.uk/osm/fit.html (accessed Oct. 16, 2022).
[4] dodo-saba, “fit2gpx -- convert .fit to .gpx.” Sep. 29, 2022. Accessed: Oct. 16, 2022. [Online]. Available: https://github.com/dodo-saba/fit2gpx
[5] JC, “fitdecode.” Oct. 16, 2022. Accessed: Oct. 16, 2022. [Online]. Available: https://github.com/polyvertex/fitdecode
[6] D. Cooper, “python-fitparse.” Oct. 14, 2022. Accessed: Oct. 16, 2022. [Online]. Available: https://github.com/dtcooper/python-fitparse
[7] A. Bunbury, “Parsing fitness tracker data with Python,” Medium, Jan. 23, 2021. https://towardsdatascience.com/ parsing-fitness-tracker-data-with-python-a59e7dc17418 (accessed Oct. 18, 2022).
[8] bunburya, “bunburya/fitness_tracker_data_parsing.” Jul. 20, 2022. Accessed: Oct. 18, 2022. [Online]. Available: https://github.com/bunburya/fitness_tracker_data_parsing/blob/8d71ff496d874f34ed806e8e
[9] “csv — CSV File Reading and Writing — Python 3.10.8 documentation.” https://docs.python.org/3/library/csv.html (accessed Oct. 18, 2022).
[10] “kortprojektioner | lex.dk,” Den Store Danske. https://denstoredanske.lex.dk/kortprojektioner (accessed Oct. 17, 2022).
[11] “latitude and longitude | Definition, Examples, Diagrams, & Facts | Britannica.” https://www.britannica.com/science/latitude (accessed Oct. 17, 2022).
[12] S. Piloten, “Hvorfor flyver man over Grønland på en tur fra København til New York? Er det ikke en omvej?,” Spørg Piloten, Sep. 10, 2017. https://spoerg-piloten.dk/hvorfor-flyver-man-over-groenland-paa-en-tur-fra-koebenhavn-til-new-york-er-det-ikke-en-omvej/ (accessed Oct. 17, 2022).
[13] “Haversine formula,” Wikipedia. Jun. 01, 2022. Accessed: Oct. 17, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Haversine_formula&oldid=1090892412
[14] “Haversine.” Mapado, Oct. 08, 2022. Accessed: Oct. 17, 2022. [Online]. Available: https://github.com/mapado/haversine
[15] “Welcome to GeoPy’s documentation! — GeoPy 2.2.0 documentation.” https://geopy.readthedocs.io/en/stable/index.html?highlight=distance#module-geopy.distance (accessed Oct. 17, 2022).