THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

DEPARTMENT OF LAND SURVEYING AND GEO-INFORMATICS
土地測量及地理資訊學系

The Hong Kong Polytechnic University

Department of Land Surveying and Geo-informatics

LSGI3322 Satellite Positioning Systems

GPS Positioning Project

(Intermediate 1 – Reading 2 RINEX files)

Subject Lecturer: Prof. George Liu

Student Name: Waterdragen

Student ID: 24[redacted]D

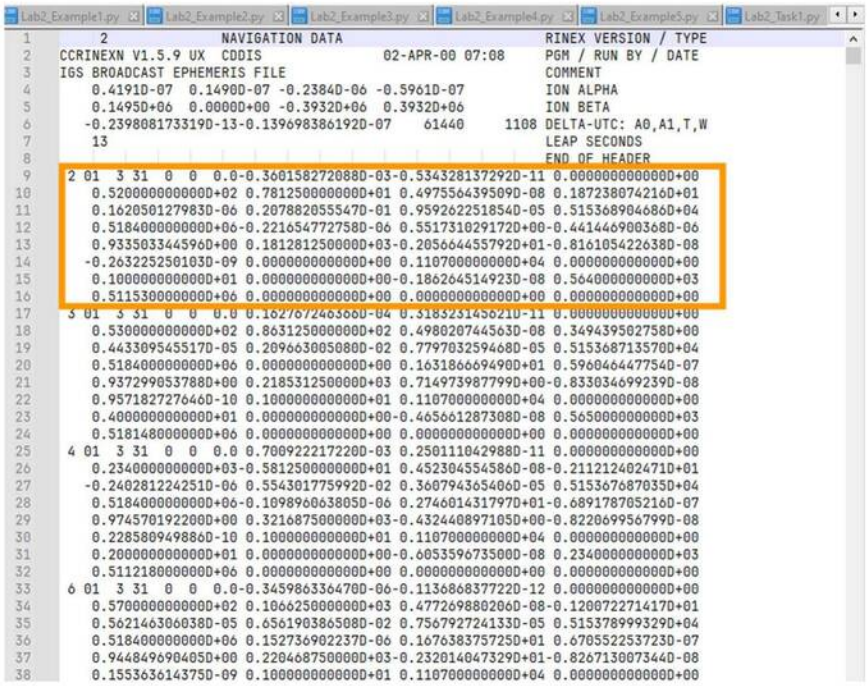
Year 3 student of Land Surveying and Geo-informatics

Task

In this exercise I have developed a computer program to open the files for navigation data (.xxn) and observation data (.xxo) and store them in two .csv files.

Interpretation

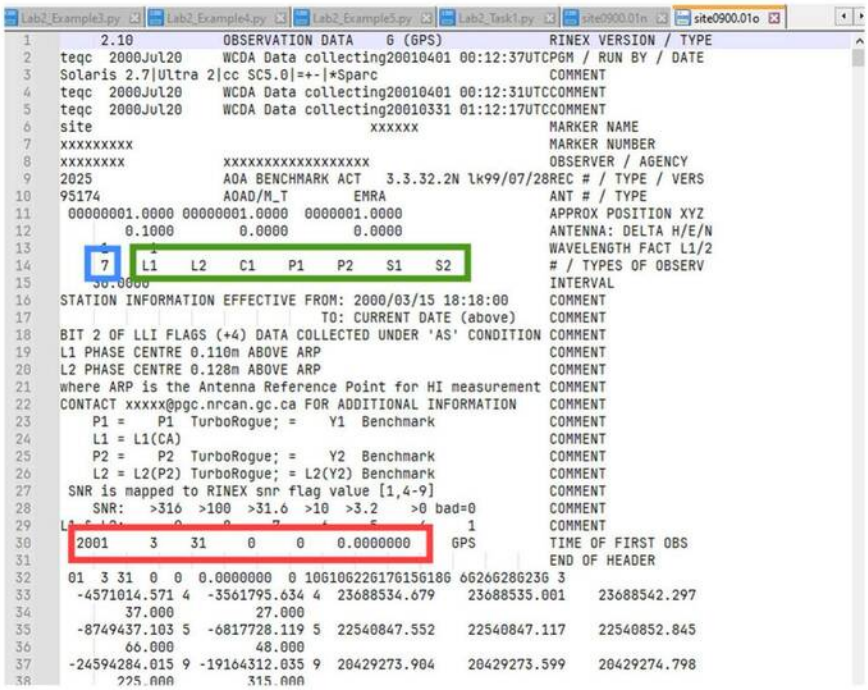
The Receiver Independent Exchange Format (RINEX) defines two formats for two files, .xxn file for navigation data, and .xxo file for observation data. The following uses site0900.01n and site0900.01o as an example for our interpretation.



```
     2           NAVIGATION DATA                     RINEX VERSION / TYPE
CCRINEXN V1.5.9 UX  CDDIS               02-APR-00 07:08     PGM / RUN BY / DATE
IGS BROADCAST EPHEMERIS FILE                        COMMENT
    0.4191D-07  0.1490D-07 -0.2384D-06 -0.5961D-07          ION ALPHA
    0.1495D+06  0.0000D+00 -0.3932D+06  0.3932D+06          ION BETA
   -0.239808173319D-13-0.1396983861920D-07    61440     1108 DELTA-UTC: A0,A1,T,W
    13                                                      LEAP SECONDS
                                                            END OF HEADER
 2 01  3 31  0  0  0.0-0.360158272088D-03-0.534328137292D-11 0.000000000000D+00
    0.520000000000D+02 0.781250000000D+01 0.497556439509D-08 0.187238074216D+01
    0.162050127983D-06 0.207882055547D-01 0.959262251854D-05 0.515368904686D+04
    0.518400000000D+06-0.221654772758D-06 0.551731029172D+00-0.441446900368D-06
    0.933503344596D+00 0.181281250000D+03-0.205664455792D+01-0.816105422638D-08
   -0.263225250103D-09 0.000000000000D+00 0.110700000000D+04 0.000000000000D+00
    0.100000000000D+01 0.000000000000D+00-0.186264514923D-08 0.564000000000D+03
    0.511530000000D+06 0.000000000000D+00 0.000000000000D+00 0.000000000000D+00
 3 01  3 31  0  0  0.0 0.162767246566D-04 0.318523145621D-11 0.000000000000D+00
    0.530000000000D+02 0.863125000000D+02 0.498020744563D-08 0.349439502758D+00
    0.443309545517D-05 0.209663005080D-02 0.779703259468D-05 0.515368713570D+04
    0.518400000000D+06 0.000000000000D+00 0.163186669490D+01 0.596046447754D-07
    0.937299053788D+00 0.218531250000D+03 0.714973987799D+00-0.833034699239D-08
    0.957182727646D-10 0.100000000000D+01 0.110700000000D+04 0.000000000000D+00
    0.400000000000D+01 0.000000000000D+00-0.465661287308D-08 0.565000000000D+03
    0.518148000000D+06 0.000000000000D+00 0.000000000000D+00 0.000000000000D+00
 4 01  3 31  0  0  0.0 0.700922217220D-03 0.250111042988D-11 0.000000000000D+00
    0.234000000000D+03-0.581250000000D+01 0.452304554586D-08-0.211212402471D+01
   -0.240281224251D-06 0.554301775992D-02 0.360794365406D-05 0.515367687035D+04
    0.518400000000D+06-0.109896063805D-06 0.274601431797D+01-0.689178705216D-07
    0.974570192200D+00 0.321687500000D+03-0.432440897105D+00-0.822069956799D-08
    0.228580949886D-10 0.100000000000D+01 0.110700000000D+04 0.000000000000D+00
    0.200000000000D+01 0.000000000000D+00-0.605359673500D-08 0.234000000000D+03
    0.511218000000D+06 0.000000000000D+00 0.000000000000D+00 0.000000000000D+00
 6 01  3 31  0  0  0.0-0.345986336470D-06-0.113686837722D-12 0.000000000000D+00
    0.570000000000D+02 0.106625000000D+03 0.477269880206D-08-0.120072271417D+01
    0.562146306038D-05 0.656190386508D-02 0.756792724133D-05 0.515378999329D+04
    0.518400000000D+06 0.152736902237D-06 0.167638375725D+01 0.670552253723D-07
    0.944849690405D+00 0.220468750000D+03-0.232014047329D+01-0.826713007344D-08
    0.155363614375D-09 0.100000000000D+01 0.110700000000D+04 0.000000000000D+00
```

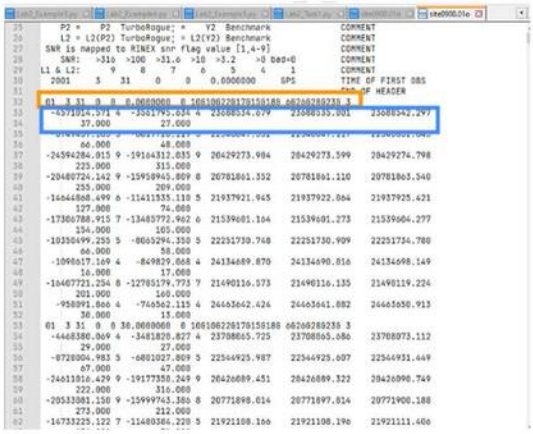
According to the RINEX version 2 by Gurtner (1993), and the detailed descriptions by Corcetto et al (1997), the area in orange frame represents each individual navigation data and their descriptions can be interpreted as in the following table:

prn	year, month, day, hour, minute, second (space delimited)	Sv (satellite) clock bias	Sv (satellite) clock drift	Sv (satellite) clock drift rate
IODE (Issue of Data, Ephemeris)		Crs (correction terms to orbital radius)	Δn (mean motion difference)	M0 (mean anomaly)
Cuc (latitude argument correction terms)		Eccentricity	Cus (latitude argument correction terms)	$\sqrt{a}$ (square root of the semi-major axis)
t0 (or TOE, Time of Ephemeris)		Cic (inclination correction terms)	Ω0 (right ascension)	Cis (inclination correction terms)
i0 (inclination)		Crc (orbital radius correction terms)	Ω (perigee argument)	dΩ/dt (or omega dot, rate of right ascension)
di/dt (rate of inclination)		Codes for L2 channels	Week number	L2 P Data flag
Sv accuracy		Sv health	Tgd	IODC (Issue of Data, clock)
Transmission time of message		Fit interval	Spare	Spare



```
     2.10           OBSERVATION DATA    G (GPS)             RINEX VERSION / TYPE
teqc  2000Jul20     WCDA Data collecting20010401 00:12:37UTCPGM / RUN BY / DATE
Solaris 2.7|Ultra 2|cc SC5.0|=+-|*Sparc                     COMMENT
teqc  2000Jul20     WCDA Data collecting20010401 00:12:31UTCCOMMENT
teqc  2000Jul20     WCDA Data collecting20010331 01:12:17UTCCOMMENT
site                                xxxxxx                  MARKER NAME
xxxxxxxxx                                                   MARKER NUMBER
xxxxxxxx            xxxxxxxxxxxxxxxxxx                      OBSERVER / AGENCY
2025                AOA BENCHMARK ACT   3.3.32.2N lk99/07/28REC # / TYPE / VERS
95174               AOAD/M_T        EMRA                    ANT # / TYPE
 00000001.0000 00000001.0000  0000001.0000                  APPROX POSITION XYZ
       0.1000        0.0000        0.0000                   ANTENNA: DELTA H/E/N
    1     1                                                 WAVELENGTH FACT L1/2
    7   L1    L2    C1    P1    P2    S1    S2              # / TYPES OF OBSERV
      30.0000                                               INTERVAL
STATION INFORMATION EFFECTIVE FROM: 2000/03/15 18:18:00     COMMENT
                                TO: CURRENT DATE (above)    COMMENT
BIT 2 OF LLI FLAGS (+4) DATA COLLECTED UNDER 'AS' CONDITION COMMENT
L1 PHASE CENTRE 0.110m ABOVE ARP                            COMMENT
L2 PHASE CENTRE 0.128m ABOVE ARP                            COMMENT
where ARP is the Antenna Reference Point for HI measurement COMMENT
CONTACT xxxxx@pgc.nrcan.gc.ca FOR ADDITIONAL INFORMATION    COMMENT
    P1 =    P1  TurboRogue; =    Y1  Benchmark              COMMENT
    L1 = L1(CA)                                             COMMENT
    P2 =    P2  TurboRogue; =    Y2  Benchmark              COMMENT
    L2 = L2(P2) TurboRogue; = L2(Y2) Benchmark              COMMENT
 SNR is mapped to RINEX snr flag value [1,4-9]              COMMENT
    SNR:   >316  >100  >31.6  >10  >3.2    >0 bad=0         COMMENT
L1 & L2:    9     8     7     6     5     4     1           COMMENT
  2001     3    31     0     0    0.0000000     GPS         TIME OF FIRST OBS
                                                            END OF HEADER
 01  3 31  0  0  0.0000000  0 10610622617615618G 66266286236 3
  -4571014.571 4  -3561795.634 4  23688534.679    23688535.001    23688542.297
        37.000          27.000
  -8749437.103 5  -6817728.119 5  22540847.552    22540847.117    22540852.845
        66.000          48.000
 -24594284.015 9 -19164312.035 9  20429273.904    20429273.599    20429274.798
      225.000         315.000
```

As described in the RINEX specification (1993) and the label, the number in blue frame is the number of observations, and the array of strings in green, delimited by space, represents the types of observations. Here we have 7 observations, which are L1, L2, C1, P1, P2, S1, and S2.



As described in the RINEX specification (1993), the first line in orange frame stores the epoch, epoch flag, PRN count, and the PRNs. The subsequent rows in blue frame represent individual observations of each type, in this case, L1, L2, C1, P1, P2, S1, S2 respectively. In this exercise, I am only reading the signal from C1 (Coarse/Acquisition) code, which is modulated on the L1 signal. If a line reaches 80 characters, it goes on to the next line. Here is a table for more detailed interpretation:

year (yy), month, day, hour, minute, second (space delimited)			epoch flag	PRN count and PRNs		
L1 (single digit signal strength, optional)	L2 (single digit signal strength, optional)	C1 (single digit signal strength, optional)	P1 (single digit signal strength, optional)	P2 (single digit signal strength, optional)	S1 (single digit signal strength, optional)	S2 (single digit signal strength, optional)
L1	L2	C1	P1	P2	S1	S2
(Repetition until the row number reaches the PRN number)						

Other notes:

- Epoch flag 0: OK, 1: power failure between previous and current epoch, >1: Event flag.

- Single strength ranges from 1-9, the higher the stronger. 0 or blank means not known or don't care.

- The first number of the PRN string is the PRN count. Subsequent numbers following the PRN count are the PRNs. For example "10G10G22G17G15G18G 6G26G28G23G 3" represents 10 PRNs: 10, 22, 17, 15, 18, 6, 26, 28, 23, 3.

Computer program

I used Python program for this exercise, and it requires at least version 3.7. The main.py file should be in the same directory as the site0900.01n and site0900.01o.

Methodology

In my program I make use of the labels and make helper functions to advance the file object to the line as shown below.

```
from dataclasses import dataclass
from os import path

OBS_FILE = "site0900.01o"
NAV_FILE = "site0900.01n"
OBS_CSV_FILE = "site0900-01o.csv"
NAV_CSV_FILE = "site0900-01n.csv"

if not path.exists(OBS_FILE):
    print(f"Please make sure '{OBS_FILE}' is placed in the same directory as main.py, aborting...")
    exit(1)
if not path.exists(NAV_FILE):
    print(f"Please make sure '{NAV_FILE}' is placed in the same directory as main.py, aborting...")
    exit(1)

FIELD1 = slice(3, 22)
FIELD2 = slice(22, 41)
FIELD3 = slice(41, 60)
FIELD4 = slice(60, 79)
```

Importing necessary libraries, defining constants for file names and slices for string slicing when reading navigation files, and ensuring the .xxn and .xxo files exists.

```
def jump_to_suffix_line(file_obj, suffix: str) -> str:
    suffix += "\n"
    line: str
    for line in file_obj:
        if line.endswith(suffix):
            return line.rstrip()
    raise RuntimeError("Cannot find suffix " + suffix)

def jump_to_containing_line(file_obj, s: str) -> str:
    for line in file_obj:
        if s in line:
            return line.rstrip()
    raise RuntimeError("Cannot find text " + s)
```

Helper functions to advance the file stream object to a specific line. `jump_to_suffix_line` can advance the file object to the line ending with the suffix, and similarly, `jump_to_containing_line` can advance it to the line containing the string. This can be useful for locating the lines with string labels.

```python
def parse_sat_prn(s: str) -> list[int]:
    # collect the numbers in the string into a list
    # for example: "10G10G22G17G15G18G 6G26G28G23G 3" becomes
    # [10, 22, 17, 15, 18, 6, 26, 28, 23, 3]
    # and there are 10 satellites
    num_satellites = int(s[30:32])
    return [int(s[33 + i * 3: 35 + i * 3]) for i in range(num_satellites)]
```

This helper function can parse a satellite prn (Pseudorandom Noise) string into a list of prn integers.

```python
def parse_time(s: str, four_digit_year=False) -> tuple[int, int, int, int, int, float]:
    # helper function to parse a space delimited time
    tme = s.split()
    year = int(tme[0])
    if not four_digit_year:
        year += 1900 if year >= 80 else 2000
    return (year, int(tme[1]), int(tme[2]),
            int(tme[3]), int(tme[4]), float(tme[5]))
```

This helper function can parse the date string, delimited by space, into a list of numbers. We need to add 1900 or 2000 to the year if it is only two digits.

```python
def parse_sci_float(s: str) -> float:
    return float(s[:15]) * 10 ** int(s[16:])
```

This helper function can parse the scientific notations numbers from the navigation file with string slicing because the length is fixed.

```python
@dataclass
class FirstObsTime:
    year: int
    month: int
    day: int
    hour: int
    minute: int
    second: float
```

Dataclass for storing the first observation time

```python
class KeyValueCSV:
    @classmethod
    def header_row(cls) -> str:
        return ",".join(cls.__annotations__.keys()) + '\n'

    def values_row(self):
        return ",".join(str(i) for i in vars(self).values()) + '\n'
```

This base class can convert class fields into csv headers and value rows. Dataclasses instead of tuples are used because of clarity. The dataclasses for navigation and observation files derive from this base class and the dataclass metaclass generates the constructor method.

```python
@dataclass
class ObsData(KeyValueCSV):
    prn: int
    year: int
    month: int
    day: int
    hour: int
    minute: int
    second: float
    epoch_flag: int
    c1: float
```

Dataclass for storing observation data

```python
@dataclass
class NavData(KeyValueCSV):
    prn: int
    year: int
    month: int
    day: int
    hour: int
    minute: int
    second: float
    sv_clock_bias: float
    sv_clock_drift: float
    sv_clock_drift_rate: float

    iode: float     # Issue of Data, Ephemeris
    crs: float      # correction terms to orbital radius
    delta_n: float  # mean motion difference
    m0: float       # mean anomaly

    cuc: float      # latitude argument correction terms
    ecc: float      # eccentricity
    cus: float      # latitude argument correction terms
    sqrt_a: float   # sqrt of semi-major axis
```

```
    t0: float       # or toe, Time of Ephemeris
    cic: float      # inclination correction terms
    omega_0: float  # right ascension
    cis: float      # inclination correction terms

    i0: float       # inclination
    crc: float      # orbital radius correction terms
    omega: float    # perigee argument
    d_omega: float  # or "dΩ/dt", "omega dot", rate of right ascension

    d_i: float      # or "di/dt", rate of inclination
    code_l2: float  # codes for L2 channels
    week_no: float
    l2_p_data_flag: float

    sv_accuracy: float
    sv_health: float
    tgd: float
    iodc: float     # Issue of Data, Clock

    transmission_time_of_msg: float
    fit_interval: float
    spare_1: float
    spare_2: float
```

Dataclass for storing navigation data

```
@classmethod
def parse(cls, f):
    line = next(f)
    prn = int(line[:2])
    year, month, day, hour, minute, second = parse_time(line[2:22])

    sv_clock_bias = parse_sci_float(line[FIELD2])
    sv_clock_drift = parse_sci_float(line[FIELD3])
    sv_clock_drift_rate = parse_sci_float(line[FIELD4])

    misc = []
    for _ in range(7):
        line = next(f)
        misc.append([
            parse_sci_float(line[FIELD1]),
            parse_sci_float(line[FIELD2]),
            parse_sci_float(line[FIELD3]),
            parse_sci_float(line[FIELD4]),
        ])
    return cls(
        prn, year, month, day, hour, minute, second, sv_clock_bias, sv_clock_drift, sv_clock_drift_rate,
        misc[0][0], misc[0][1], misc[0][2], misc[0][3],  # broadcast orbit 1
        misc[1][0], misc[1][1], misc[1][2], misc[1][3],  # broadcast orbit 2
        misc[2][0], misc[2][1], misc[2][2], misc[2][3],  # broadcast orbit 3
        misc[3][0], misc[3][1], misc[3][2], misc[3][3],  # broadcast orbit 4
        misc[4][0], misc[4][1], misc[4][2], misc[4][3],  # broadcast orbit 5
        misc[5][0], misc[5][1], misc[5][2], misc[5][3],  # broadcast orbit 6
        misc[6][0], misc[6][1], misc[6][2], misc[6][3],  # broadcast orbit 7
    )
```

Core implementation for navigation data. It first parses the line containing the time, then iterates the number of broadcast orbits (7), the rest of the matrix corresponds to each of the fields in NavData.

```
def read_obs():
    with open(OBS_FILE) as f, open(OBS_CSV_FILE, 'w') as table:
        table.write(ObsData.header_row())

        text_types_of_satellites = jump_to_suffix_line(f, "# / TYPES OF OBSERV")
        type_obs_count = int(text_types_of_satellites[1:6])
        type_obs_names = [t for t in text_types_of_satellites[6:60].split()]
        c1_index = type_obs_names.index("C1")

        text_time_of_first_obs = jump_to_suffix_line(f, "TIME OF FIRST OBS")
        first_obs_time = FirstObsTime(*parse_time(text_time_of_first_obs, four_digit_year=True))

        print(first_obs_time)

        _ = jump_to_suffix_line(f, "END OF HEADER")

        current_line = f.readline()
        while current_line != "":
            # Check the epoch flag is 0 (OK)
            epoch_flag = current_line[27:30]
            if epoch_flag != " 0 ":
                current_line = f.readline()
                continue

            year, month, day, hour, minute, second = parse_time(current_line[:26])

            sat_prn = parse_sat_prn(current_line)

            for prn in sat_prn:
                lines = ""
                new_line = f.readline()
                lines += new_line
```

```
                while len(new_line) == 79:
                    new_line = f.readline()
                    # Add an arbitrary character to make it 80, multiple of 16
                    lines += new_line + "\n"

                obs = [float(lines[i * 16: i * 16 + 14]) for i in range(type_obs_count)]

                obs_data = ObsData(prn, year, month, day, hour, minute, second,
                                   int(epoch_flag),
                                   obs[c1_index])
                table.write(obs_data.values_row())

            current_line = f.readline()

    print(f"Successfully wrote observation data to {OBS_CSV_FILE}")
```

Core implementation for reading observation data. Here are the steps:

1. go to the line that ends with “# / TYPES OF OBSERV”. Parse the type count and type names. Find the index of “C1” of the list

2. go to the line that ends with “TIME OF FIRST OBS”. Parse the first observation time and print it to console.

3. jump to “END OF HEADER”

4. start on a line containing an epoch flag == 0, then parse the time and list of prn. For each prn, read space delimited lines until there are no lines reaching 79 characters. Concatenate the lines and slice them into a list of floats, and index with the c1 index previously found.

5. write to csv then repeat until there are no more lines left.

```python
def read_nav():
    with open(NAV_FILE) as f, open(NAV_CSV_FILE, 'w') as table:
        table.write(NavData.header_row())

        _ = jump_to_containing_line(f, "END OF HEADER")
        try:
            while True:
                nav_data = NavData.parse(f)
                table.write(nav_data.values_row())
        except StopIteration:
            pass

    print(f"Successfully wrote navigation data to {NAV_CSV_FILE}")


def main():
    read_nav()
    read_obs()

if __name__ == '__main__':
    main()
```

Controller functions read_nav and main. In read_nav, we write the csv header, then we jump to the line containing the label “END OF HEADER” and for each line parse the line into NavData and write the values to the csv until the iterator ends. The main function calls read_nav and read_obs.

How to run

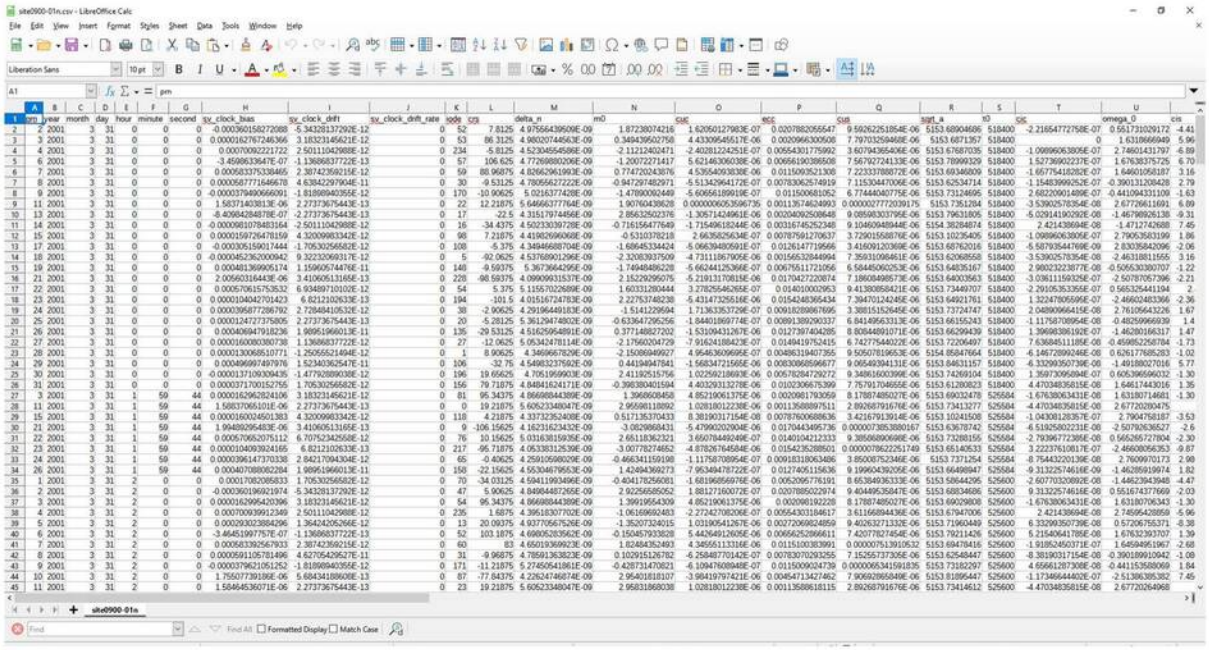
use the command `python3 main.py` in the directory of main.py. Use `python` for windows operating system. This Python program requires at least Python 3.7. Make sure that `site0900.01n` and `site0900.01o` are in the same directory as main.py. `main.py` file has been attached in the .zip file.

Expected results

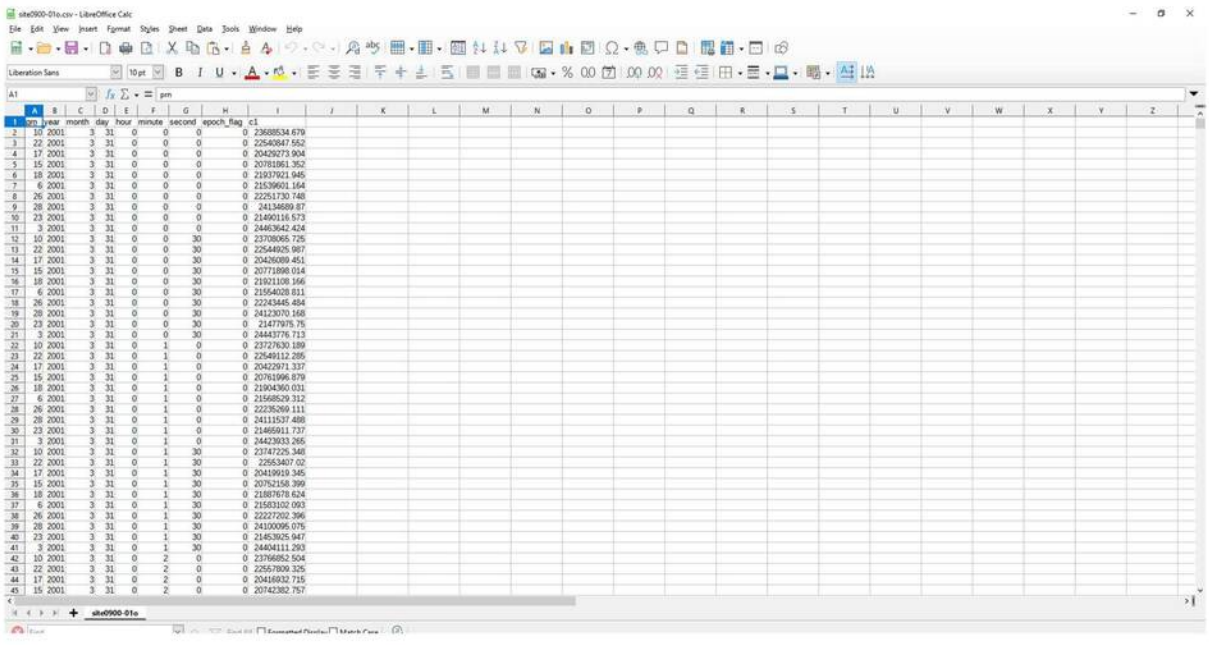
```
PS C:\[redacted]\Desktop\Satellite Positioning> python main.py
Successfully wrote navigation data to site0900-01n.csv
FirstObsTime(year=2001, month=3, day=31, hour=0, minute=0, second=0.0)
Successfully wrote observation data to site0900-01o.csv
PS C:\[redacted]\Desktop\Satellite Positioning>
```

the program should print out success messages for writing the csvs and the first observation time.

Expected CSV results



opening the csv containing the navigation data in LibreOffice



opening the csv containing the observation data in LibreOffice

References

Calais E. *GPS Geodesy – Lab 5 From GPS ephemerides to ECEF satellite positions.* Retrieved from https://www.geologie.ens.fr/~ecalais/teaching/gps-geodesy/lab_5.pdf

Crocetto N., Gatti M. & Perfetti N. (1997) *The GPS Single Point Positioning: A Data Processing Program for Tutorial Purposes.* ISPRS Commision VI, Working Group 3. 1-5 Retrieved from https://www.isprs.org/proceedings/XXXII/6-W1/131_XXXII-6-W1.pdf

Gurtner W. & Mader G. (1990) *RINEX: The Receiver Independent Exchange Format Version 2.* CSTG GPS Bulletin. Vol. 3. Retrieved from https://www.ngs.noaa.gov/CORS/RINEX-2.txt

Mussio L., Crippa B. & Vettore A. (1997) *International Society for Photogrammetry and Remote Sensing* ISPRS 32:1-11. Retrieved from https://www.researchgate.net/profile/Gabriella-Caroti/publication/260934448_Kinematic_GNSS_SurveyExperiences_to_be_Transferred/links/58f759d50f7e9be34b3426db/Kinematic-GNSS-SurveyExperiences-to-be-Transferred.pdf