# Day 04 - Working with `datetime`

Sept. 17, 2020

# Administrative

- **Getting help**:
    - Y'all are making good use of office hours and Slack
    - Office hours are posted on the GR site and D2L
- **Individual check-ins**:
    - Danny will be sending each person an email to check in on how they are doing with online classes and asking for feedback.
- **Group work**:
    - In some groups, not everyone is moving through the activity at the same pace.
    - Make sure to check in with everyone on where they are and share screens to help each other stay together.
    - Remember: You do not have to complete tehe whole assignment to get credit.
- **Extra breakout rooms**:
    - We are going to open 4 additional breakout rooms for one-on-one meetings as needed.

# Any questions?

# From Pre-Class Assignment

## Challenging bits

- Reading the sunspots file
- Knowing what to do with the date information (instructions were unclear for some)
- Converting the date information to `datetime`
- Using `.assign()`

**You will get more practice with this today.**

In [1]:

```
import datetime

# We can store date information as integers
birthyear = 1982
birthmonth = 5
birthday = 1

print('Danny\'s Birthday is:\n', birthmonth, '/', birthday, '/', birthyear)
print(type(birthyear))
```

```
Danny's Birthday is:
 5 / 1 / 1982
<class 'int'>
```

In [2]:
```
# That is less useful for doing math with dates
# For example, how old am i?

today = datetime.datetime.now()
print(today)

# We can convert information to datetime
birthdate = datetime.datetime(year = birthyear, month = birthmonth, day = birthday
)
print(birthdate)
```

```
2020-09-17 09:24:40.438286
1982-05-01 00:00:00
```

In [3]:

```
# datetime objects can have math done on them
age = today - birthdate
print('Age: ', age)

# Notice that the types are different
print(type(today))
print(type(age))
```

```
Age:  14019 days, 9:24:40.438286
<class 'datetime.datetime'>
<class 'datetime.timedelta'>
```

In [4]:

```
# What about working with data?

import pandas as pd

column_names = ["year",
                "month",
                "day",
                "date",
                "count",
                "std",
                "obs",
                "provisional"]

# Read in the date file and specify columns
df = pd.read_csv('data/sunspots.txt', delim_whitespace = True, names = column_name
s)

# Type for each column
print(df.dtypes)
```

```
year             int64
month            int64
day              int64
date           float64
count            int64
std            float64
obs              int64
provisional     object
dtype: object
```

In [5]:
```
# Let's look at the DataFrame
df.head()
```

Out[5]:

| | year | month | day | date | count | std | obs | provisional |
|---|---|---|---|---|---|---|---|---|
| **0** | 1981 | 1 | 1 | 1981.001 | 218 | 12.4 | 9 | NaN |
| **1** | 1981 | 1 | 2 | 1981.004 | 194 | 14.7 | 7 | NaN |
| **2** | 1981 | 1 | 3 | 1981.007 | 168 | 10.8 | 7 | NaN |
| **3** | 1981 | 1 | 4 | 1981.010 | 155 | 9.1 | 11 | NaN |
| **4** | 1981 | 1 | 5 | 1981.012 | 129 | 6.2 | 9 | NaN |

In [6]:
```
# We want year, month, and day to be put together into a datetime Series
# pandas has a tool for that called .to_datetime()

# We can first create a DataFrame with just this information
date_info = df[["year", "month", "day"]]
date_info.head()
```

Out[6]:

| | year | month | day |
|---|---|---|---|
| **0** | 1981 | 1 | 1 |
| **1** | 1981 | 1 | 2 |
| **2** | 1981 | 1 | 3 |
| **3** | 1981 | 1 | 4 |
| **4** | 1981 | 1 | 5 |

In [7]:
```
# Then convert the date_info DataFrame into a datetime Series using .to_datetime()
# Notice the order of the columns in date_info matches the format ordering
df_date = pd.to_datetime(date_info, format = '%Y %m %d')
df_date.head()
```

Out[7]:
```
0   1981-01-01
1   1981-01-02
2   1981-01-03
3   1981-01-04
4   1981-01-05
dtype: datetime64[ns]
```

In [8]:
```
# Finally we need to assign the new series and drop the old columns

# assign() will create a new column with the series you pass it
# it must be returned to a variable or the assignment is temporary
df.assign(datetime = df_date)      # doesn't store the result!
df = df.assign(datetime = df_date) # stores the result!

# drop() will drop columns from a DataFrame
# again, it must be turned to a variable
dropped_columns = ["year", "month", "day", "date"]
df = df.drop(columns = dropped_columns)
df.head()
```

Out[8]:

| | count | std | obs | provisional | datetime |
|---|---|---|---|---|---|
| **0** | 218 | 12.4 | 9 | NaN | 1981-01-01 |
| **1** | 194 | 14.7 | 7 | NaN | 1981-01-02 |
| **2** | 168 | 10.8 | 7 | NaN | 1981-01-03 |
| **3** | 155 | 9.1 | 11 | NaN | 1981-01-04 |
| **4** | 129 | 6.2 | 9 | NaN | 1981-01-05 |

# Questions, Comments, Concerns?