IMPORTING DATA IN PYTHON I

Introduction to other file types

Other file types

- Excel spreadsheets
- MATLAB files
- SAS files
- Stata files
- HDF5 files

Pickled files

- File type native to Python
- Motivation: many datatypes for which it isn't obvious how to store them
- Pickled files are serialized
- Serialize = convert object to bytestream

Pickled files

```
In [1]: import pickle

In [2]: with open('pickled_fruit.pkl', 'rb') as file:
   ...:     data = pickle.load(file)

In [3]: print(data)
{'peaches': 13, 'apples': 4, 'oranges': 11}
```

Code_1

Importing Excel spreadsheets

Code_2

```
In [1]: import pandas as pd

In [2]: file = 'urbanpop.xlsx'

In [3]: data = pd.ExcelFile(file)

In [4]: print(data.sheet_names)
['1960-1966', '1967-1974', '1975-2011']

In [5]: df1 = data.parse('1960-1966')

In [6]: df2 = data.parse(0)
```

sheet name, as a string

sheet index, as a float

You’ll learn:

- How to customize your import
    - Skip rows
    - Import certain columns
    - Change column names

Let’s practice!

Importing SAS/Stata files using pandas

SAS and Stata files

- SAS: Statistical Analysis System
- Stata: “Statistics” + “data”

- SAS: business analytics and biostatistics
- Stata: academic social sciences research

SAS files

- Used for:
    - Advanced analytics
    - Multivariate analysis
    - Business intelligence
    - Data management
    - Predictive analytics
- Standard for computational analysis

Importing SAS files

```
In [1]: import pandas as pd

In [2]: from sas7bdat import SAS7BDAT

In [3]: with SAS7BDAT('urbanpop.sas7bdat') as file:
   ...:     df_sas = file.to_data_frame()
```

Importing Stata files

Code_4

```
In [1]: import pandas as pd

In [2]: data = pd.read_stata('urbanpop.dta')
```

Let’s practice!

Importing HDF5 files

HDF5 files

- Hierarchical Data Format version 5
- Standard for storing large quantities of numerical data
- Datasets can be hundreds of gigabytes or terabytes
- HDF5 can scale to exabytes

Importing HDF5 files

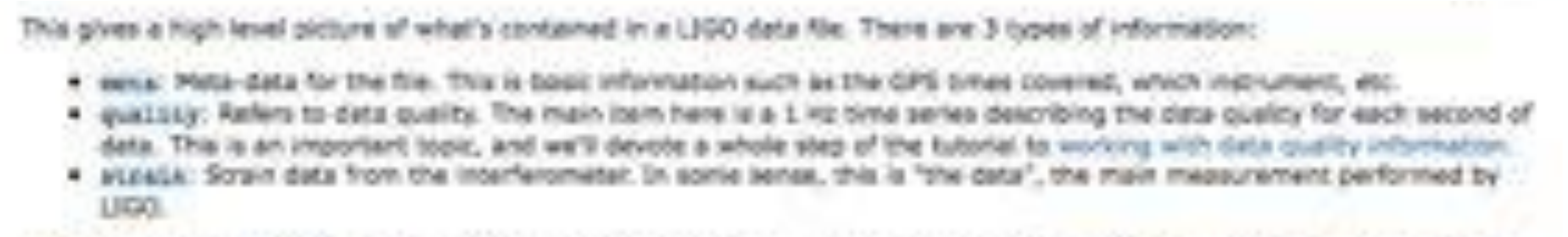
Code_5

```
In [1]: import h5py

In [2]: filename = 'H-H1_LOSC_4_V1-815411200-4096.hdf5'

In [3]: data = h5py.File(filename, 'r') # 'r' is to read

In [4]: print(type(data))
<class 'h5py._hl.files.File'>
```

The structure of HDF5 files

```
In [5]: for key in data.keys():
   ...:     print(key)
meta
quality
strain

In [6]: print(type(data['meta']))
<class 'h5py._hl.group.Group'>
```

Code_6



This gives a high level picture of what's contained in a LIGO data file. There are 3 types of information:

- `meta`: Meta-data for the file. This is basic information such as the GPS times covered, which instrument, etc.
- `quality`: Refers to data quality. The main item here is a 1 Hz time series describing the data quality for each second of data. This is an important topic, and we'll devote a whole step of the tutorial to working with data quality information.
- `strain`: Strain data from the interferometer. In some sense, this is "the data", the main measurement performed by LIGO.

The structure of HDF5 files

```
In [7]: for key in data['meta'].keys():
   ...:         print(key)
Description
DescriptionURL
Detector
Duration
GPSstart
Observatory
Type
UTCstart

In [8]: print(data['meta']['Description'].value, data['meta']
['Detector'].value)
b'Strain data time series from LIGO' b'H1'
```

Code_7

The HDF Project

- Actively maintained by the HDF Group

The HDF Group

- Based in Champaign, Illinois

Let's practice!

Importing MATLAB files

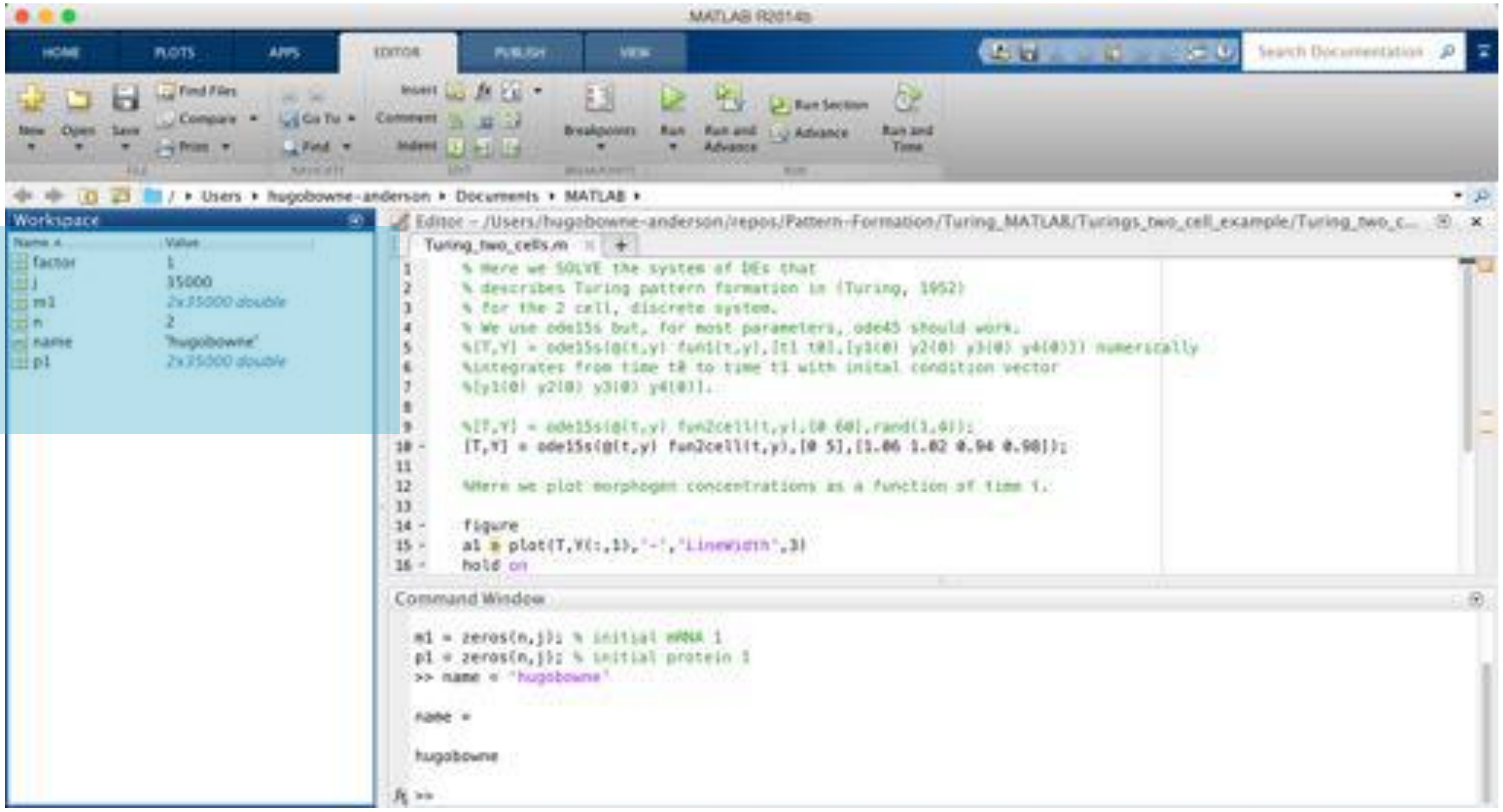
MATLAB

- “Matrix Laboratory”
- Industry standard in engineering and science
- Data saved as .mat files

SciPy to the rescue!

- `scipy.io.loadmat()` - read .mat files
- `scipy.io.savemat()` - write .mat files

What is a .mat file?



Name ▲	Value
factor	1
j	35000
m1	2x35000 double
n	2
name	'hugobowne'
p1	2x35000 double

```
% Here we SOLVE the system of DEs that
% describes Turing pattern formation in (Turing, 1952)
% for the 2 cell, discrete system.
% We use ode15s but, for most parameters, ode45 should work.
%[T,Y] = ode15s(@(t,y) fun1(t,y),[t1 t0],[y1(0) y2(0) y3(0) y4(0)]) numerically
%integrates from time t0 to time t1 with inital condition vector
%[y1(0) y2(0) y3(0) y4(0)].

%[T,Y] = ode15s(@(t,y) fun2cell(t,y),[0 60],rand(1,4));
[T,Y] = ode15s(@(t,y) fun2cell(t,y),[0 5],[1.06 1.02 0.94 0.98]);

%Here we plot morphogen concentrations as a function of time 1.

figure
a1 = plot(T,Y(:,1),'-','LineWidth',3)
hold on
```

```
m1 = zeros(n,j); % initial mRNA 1
p1 = zeros(n,j); % initial protein 1
>> name = 'hugobowne'

name =

hugobowne

>>
```

Importing a .mat file

Code_8

```
In [1]: import scipy.io

In [2]: filename = 'workspace.mat'

In [3]: mat = scipy.io.loadmat(filename)

In [4]: print(type(mat))
<class 'dict'>

In [5]: print(type(mat['x']))
<class 'numpy.ndarray'>
```

\

- keys = MATLAB variable names
- values = objects assigned to variables

Let’s practice!