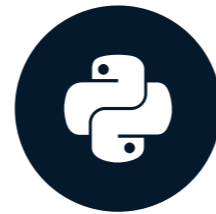


原生类型文件

Introduction to other file types

INTRODUCTION TO IMPORTING DATA IN PYTHON



Hugo Bowne-Anderson
Data Scientist at DataCamp

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

```
import pickle
with open('pickled_fruit.pkl', 'rb') as file:
    data = pickle.load(file)
print(data)
```

```
{'peaches': 13, 'apples': 4, 'oranges': 11}
```

Importing Excel spreadsheets

```
import pandas as pd
file = 'urbanpop.xlsx'
data = pd.ExcelFile(file)
print(data.sheet_names)
```

```
['1960-1966', '1967-1974', '1975-2011']
```

```
df1 = data.parse('1960-1966') # sheet name, as a string
df2 = data.parse(0) # sheet index, as a float
```

You'll learn:

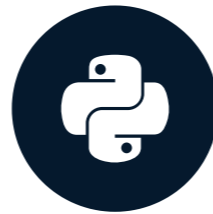
- How to customize your import
- Skip rows
- Import certain columns
- Change column names

Let's practice!

INTRODUCTION TO IMPORTING DATA IN PYTHON

Importing SAS/Stata files using pandas

INTRODUCTION TO IMPORTING DATA IN PYTHON



Hugo Bowne-Anderson
Data Scientist at DataCamp

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

```
import pandas as pd 原生类型
from sas7bdat import SAS7BDAT
with SAS7BDAT('urbanpop.sas7bdat') as file:
    df_sas = file.to_data_frame()
```

Importing Stata files

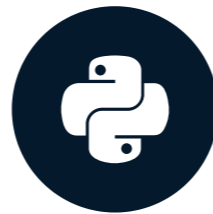
```
import pandas as pd  
data = pd.read_stata('urbanpop.dta')
```

Let's practice!

INTRODUCTION TO IMPORTING DATA IN PYTHON

Importing HDF5 files

INTRODUCTION TO IMPORTING DATA IN PYTHON



Hugo Bowne-Anderson
Data Scientist at DataCamp

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

```
import h5py
filename = 'H-H1_LOSC_4_V1-815411200-4096.hdf5'
data = h5py.File(filename, 'r') # 'r' is to read
print(type(data))
```

```
<class 'h5py._hl.files.File'>
```

The structure of HDF5 files

```
for key in data.keys():  
    print(key)
```

```
meta  
quality  
strain
```

```
print(type(data['meta']))
```

```
<class 'h5py._hl.group.Group'>
```

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

```
for key in data['meta'].keys():  
    print(key)
```

```
Description  
DescriptionURL  
Detector  
Duration  
GPSstart  
Observatory  
Type  
UTCstart
```

```
print(np.array(data['meta']['Description']), np.array(data['meta']['Detector']))
```

```
b'Strain data time series from LIGO' b'H1'
```

The HDF Project

- Actively maintained by the HDF Group



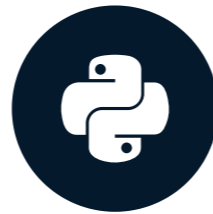
- Based in Champaign, Illinois

Let's practice!

INTRODUCTION TO IMPORTING DATA IN PYTHON

Importing MATLAB files

INTRODUCTION TO IMPORTING DATA IN PYTHON



Hugo Bowne-Anderson
Data Scientist at DataCamp

MATLAB

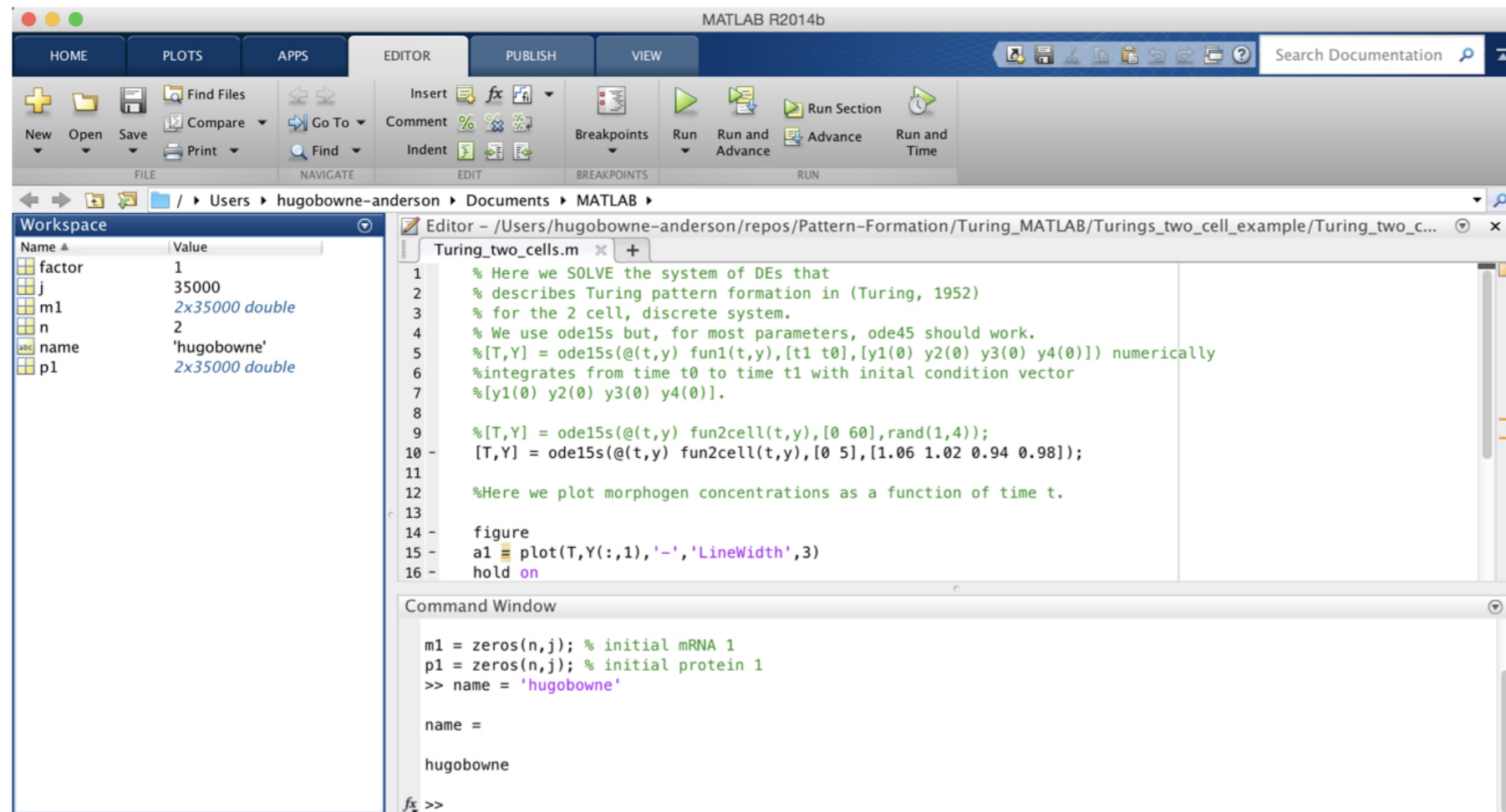
- “Matrix Laboratory”
- Industry standard in engineering and science
- Data saved as .mat files



SciPy to the rescue!

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

What is a .mat file?



The image shows the MATLAB R2014b interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. Below the menu bar is a toolbar with icons for file operations (New, Open, Save, Find Files, Compare, Print), navigation (Go To, Find), editing (Insert, Comment, Indent), breakpoints, and running (Run, Run and Advance, Run Section, Run and Time). The main window is divided into three panes: Workspace, Editor, and Command Window.

Workspace:

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

Editor - Turing_two_cells.m:

```
1 % Here we SOLVE the system of DEs that
2 % describes Turing pattern formation in (Turing, 1952)
3 % for the 2 cell, discrete system.
4 % We use ode15s but, for most parameters, ode45 should work.
5 %[T,Y] = ode15s(@(t,y) fun1(t,y),[t1 t0],[y1(0) y2(0) y3(0) y4(0)]) numerically
6 %integrates from time t0 to time t1 with initial condition vector
7 %[y1(0) y2(0) y3(0) y4(0)].
8
9 %[T,Y] = ode15s(@(t,y) fun2cell(t,y),[0 60],rand(1,4));
10 [T,Y] = ode15s(@(t,y) fun2cell(t,y),[0 5],[1.06 1.02 0.94 0.98]);
11
12 %Here we plot morphogen concentrations as a function of time t.
13
14 figure
15 a1 = plot(T,Y(:,1),'-', 'LineWidth',3)
16 hold on
```

Command Window:

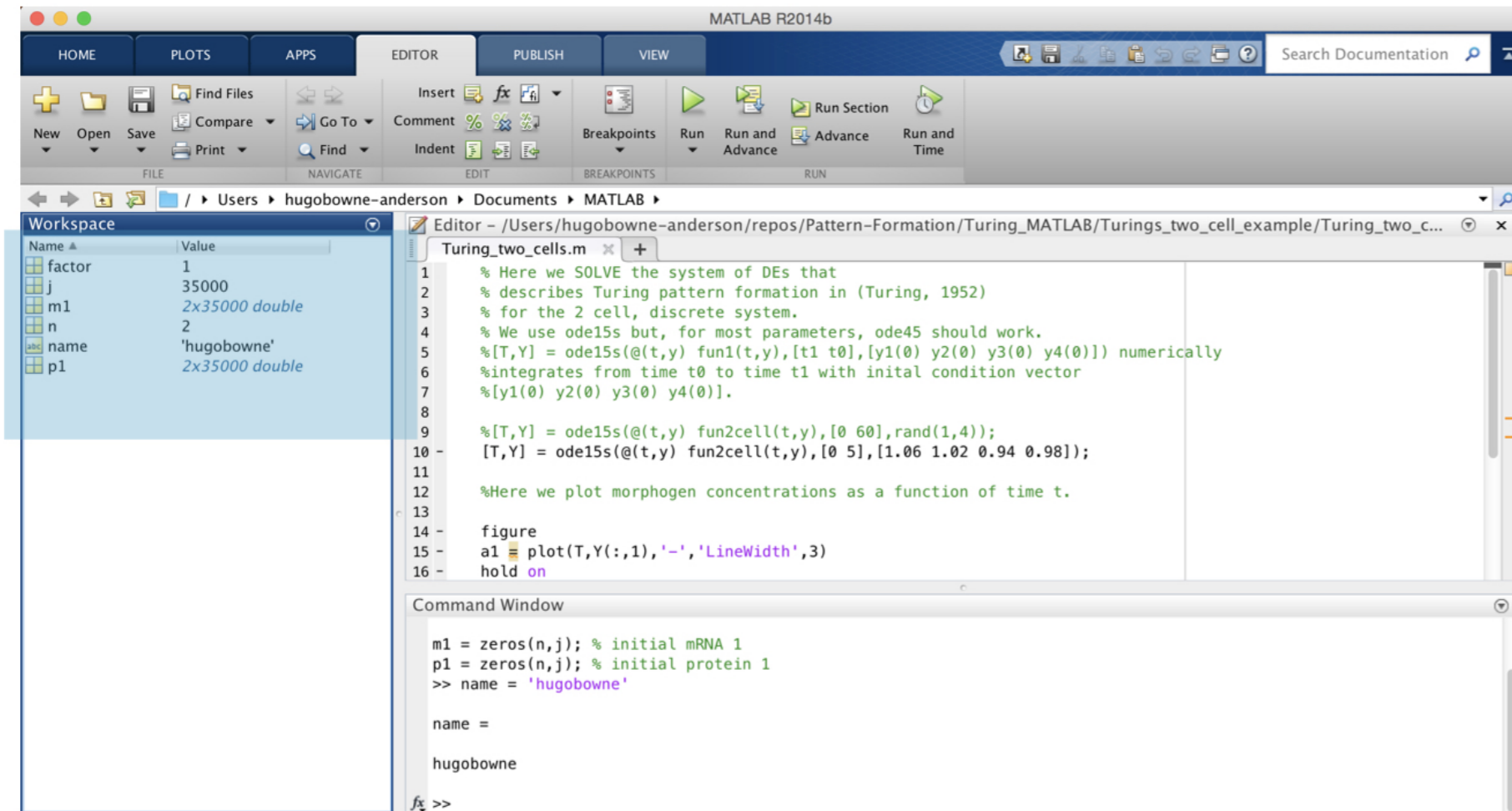
```
m1 = zeros(n,j); % initial mRNA 1
p1 = zeros(n,j); % initial protein 1
>> name = 'hugobowne'

name =

hugobowne

fx >>
```

What is a .mat file?



The screenshot displays the MATLAB R2014b environment. The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. Below the menu is a toolbar with icons for file operations (New, Open, Save, Find Files, Compare, Print), navigation (Go To, Find), editing (Insert, Comment, Indent), breakpoints, and execution (Run, Run and Advance, Run Section, Run and Time). The current directory is `Users > hugobowne-anderson > Documents > MATLAB`. The workspace on the left shows variables: `factor` (1), `j` (35000), `m1` (2x35000 double), `n` (2), `name` ('hugobowne'), and `p1` (2x35000 double). The editor window shows the script `Turing_two_cells.m` with the following code:

```
1 % Here we SOLVE the system of DEs that
2 % describes Turing pattern formation in (Turing, 1952)
3 % for the 2 cell, discrete system.
4 % We use ode15s but, for most parameters, ode45 should work.
5 %[T,Y] = ode15s(@(t,y) fun1(t,y),[t1 t0],[y1(0) y2(0) y3(0) y4(0)]) numerically
6 %integrates from time t0 to time t1 with initial condition vector
7 %[y1(0) y2(0) y3(0) y4(0)].
8
9
10 [T,Y] = ode15s(@(t,y) fun2cell(t,y),[0 60],rand(1,4));
11 [T,Y] = ode15s(@(t,y) fun2cell(t,y),[0 5],[1.06 1.02 0.94 0.98]);
12
13 %Here we plot morphogen concentrations as a function of time t.
14
15 figure
16 a1 = plot(T,Y(:,1),'-','LineWidth',3)
17 hold on
```

The Command Window shows the execution of the script:

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

name =
hugobowne
fx >>
```

Importing a .mat file

```
import scipy.io
filename = 'workspace.mat'
mat = scipy.io.loadmat(filename)
print(type(mat))
```

```
<class 'dict'>
```

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

```
print(type(mat['x']))
```

```
<class 'numpy.ndarray'>
```

Let's practice!

INTRODUCTION TO IMPORTING DATA IN PYTHON