COMP5310: Principles of Data Science

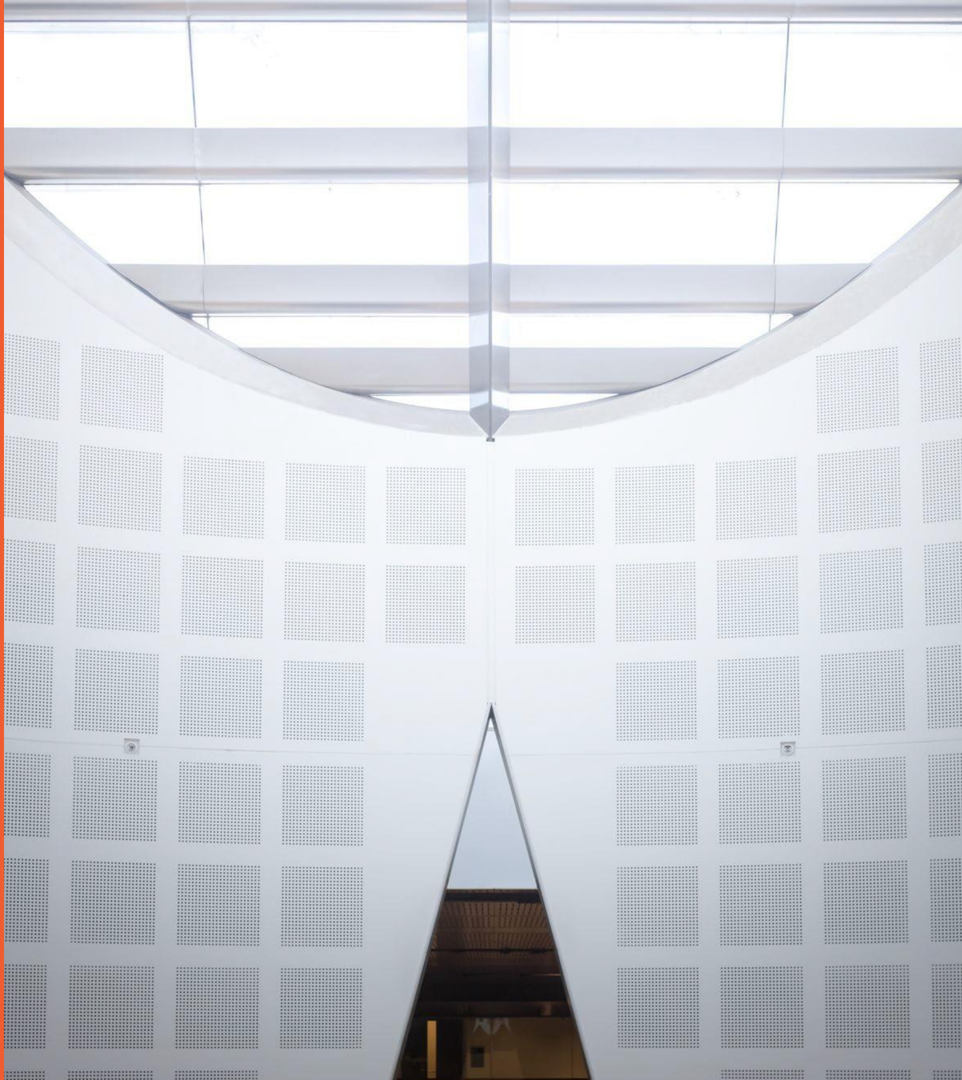
W3: Data Exploration with Python

Presented by

Dr Ali Anaissi

School of Computer Science

THE UNIVERSITY OF SYDNEY



Assignment 1: Project Stage 1

Project stage 1: Data Exploration and Cleaning

Objective

Explore a data set and define a research question based on research/business requirement.

Activities

- Choose a data set
- Explore data set
- Clean and prepare data
- Recomment problem for Stage 2

Output

- See the specification!

Marking

- 5% of overall mark

Marking Criteria

- Requirements for Pass, Distinction, Full marks are defined in the Assignment Specification on Canvas. Please read it carefully!

Suggested timeline for Assignment 1 (Project Stage 1)

- W1: Identify possible topics
- W2: Obtain datasets and metadata
- W3: Load data with Python
- W4: Clean and prepare data
- W5: Assess strengths and limitations of each topic/dataset
- W6: Submit 2-page report

Types of projects to consider

- Discover clusters in data
- Learn association rules
- Train a classifier and evaluate prediction accuracy
- Train a regression model and evaluate prediction accuracy

Overview of Week 3

Today: Data Exploration with Python

Objective

Learn Python tools for exploring a new data set programmatically.

Lecture

- Data types, cleaning, preprocessing
- Descriptive statistics, e.g., median, quartiles, IQR, outliers
- Descriptive visualisation, e.g., boxplots, confidence intervals

Readings

- Data Science from Scratch: Ch 4-5

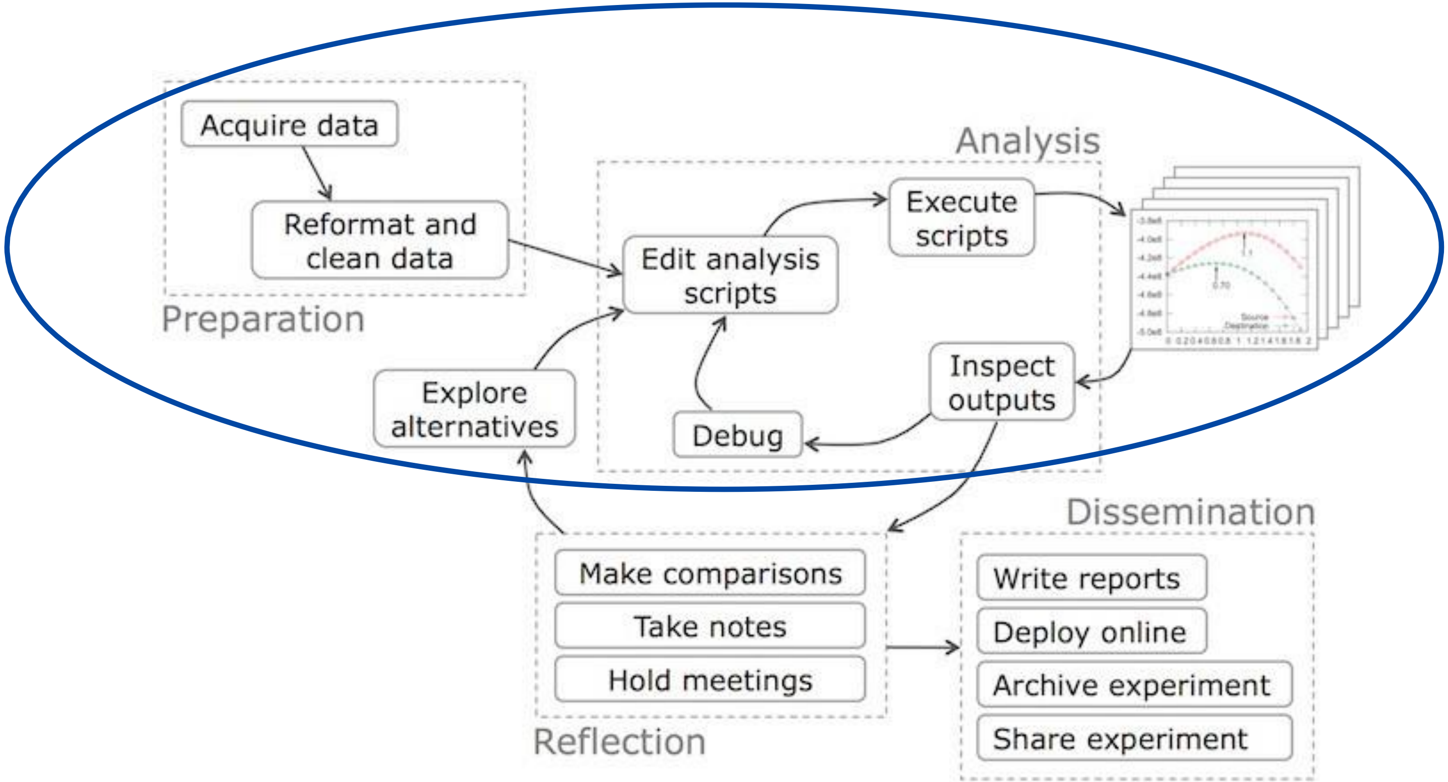
Exercises

- matplotlib: Visualisation
- numpy/scipy: Descriptive stats

TODO in W3

- Grok Python modules 7-9
- Grok SQL modules 3 – 4
- Load project data with Python

Exploratory analysis workflow



We'll revisit some descriptive questions with Python

- What industries do we know? What would we like to go into?
- What areas of data analytics are considered important?
- How do professional/programming experience compare?
- How does programming experience differ across industries?
- What skills do we know? What would we like to learn?

And look at a text question

- Which industries are most desirable? Do past/future differ?
- What areas are considered most important? Reliable?
- What skills co-occur most? How strong is the association?
- Are there natural clusters corresponding to profiles?
- Is there a significant dependence between experience?
- What terms/topics characterise our DS definitions?

Python and Jupyter Notebooks

THE UNIVERSITY OF SYDNEY

Python is great for prototyping

- **Interpreted:** direct execution without compilation
- **Dynamically-typed:** don’t have to declare a static type
- **Readable:** easy-to-understand syntax
- **Deployable:** easy to incorporate in applications

Python Recap

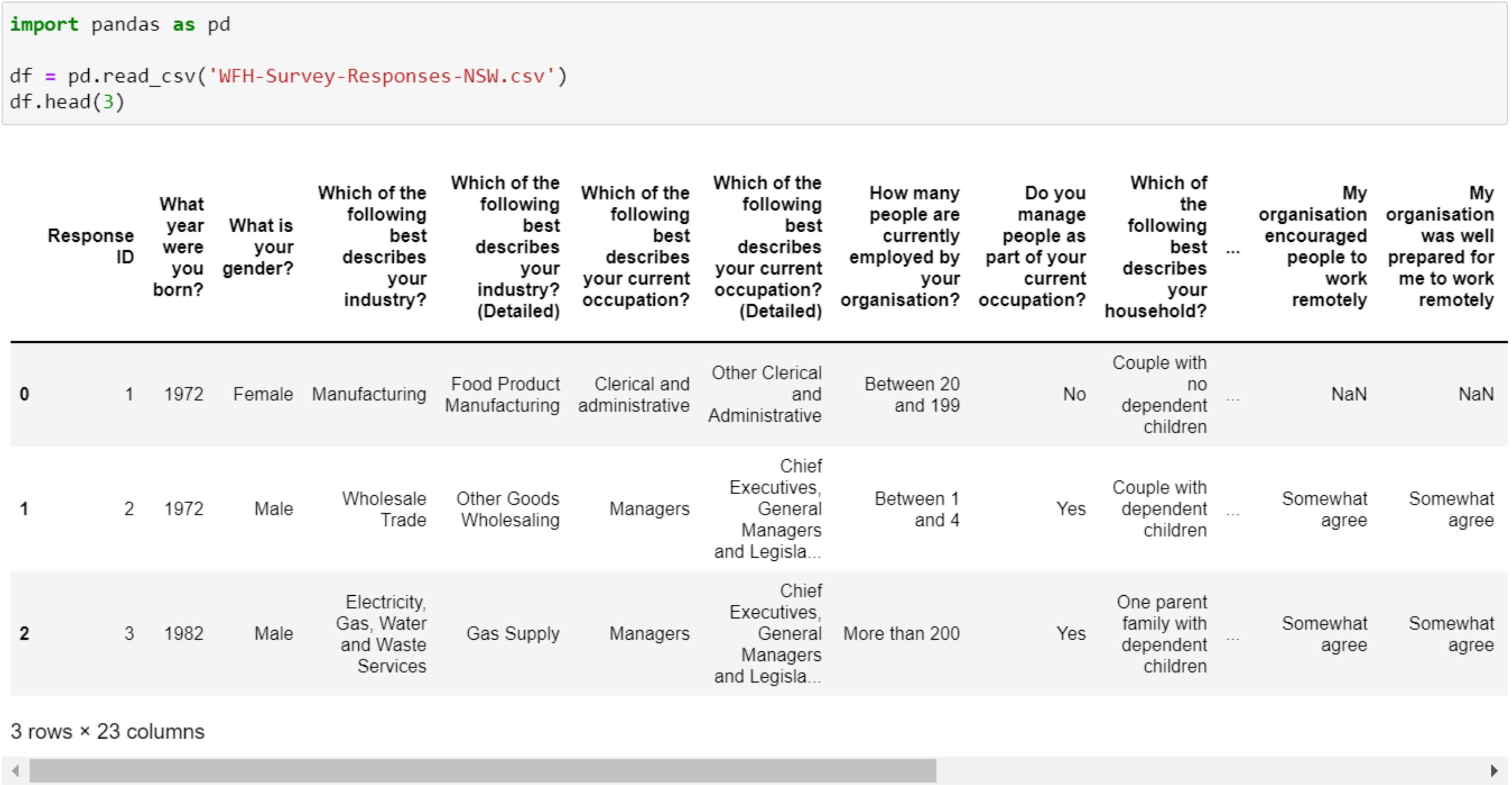
- general program syntax
- variables and types
  - integer and float numbers, string types, type conversion
  - list of values (list, array)
- condition statements (if/elif/else)
- for loops, ranges
- functions
  - input(), print(), len(), lower(), upper(), …
  - nesting of functions; example: print(len(*str*.upper()))

Python Import System

- Grok lessons, so far, concentrated on *built-in* functions
- additional functionality available via **import** statement
  - gives access to classes and functions from various 3rd party modules
  - Example: **pandas:** comma-separated file format support

```
import pandas as pd
```

Read data using Pandas



```python
import pandas as pd

df = pd.read_csv('WFH-Survey-Responses-NSW.csv')
df.head(3)
```

	Response ID	What year were you born?	What is your gender?	Which of the following best describes your industry?	Which of the following best describes your industry? (Detailed)	Which of the following best describes your current occupation?	Which of the following best describes your current occupation? (Detailed)	How many people are currently employed by your organisation?	Do you manage people as part of your current occupation?	Which of the following best describes your household?	...	My organisation encouraged people to work remotely	My organisation was well prepared for me to work remotely
0	1	1972	Female	Manufacturing	Food Product Manufacturing	Clerical and administrative	Other Clerical and Administrative	Between 20 and 199	No	Couple with no dependent children	...	NaN	NaN
1	2	1972	Male	Wholesale Trade	Other Goods Wholesaling	Managers	Chief Executives, General Managers and Legisla...	Between 1 and 4	Yes	Couple with dependent children	...	Somewhat agree	Somewhat agree
2	3	1982	Male	Electricity, Gas, Water and Waste Services	Gas Supply	Managers	Chief Executives, General Managers and Legisla...	More than 200	Yes	One parent family with dependent children	...	Somewhat agree	Somewhat agree

3 rows × 23 columns

Python has excellent open-source data libraries

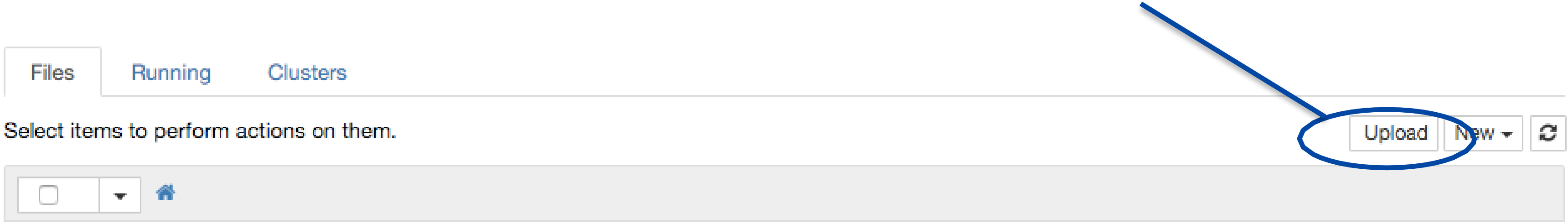
- **scipy:** libraries for scientific and technical computing
- **numpy:** support for large multidimensional arrays and matrices
- **matplotlib:** port of matlab plotting functionality
- **seaborn:** abstraction on top of matplotlib (less flexible but easier to use)
- **scikit-learn:** machine learning library
- **nltk:** natural language toolkit

- **pandas:** R-like data frame and associated manipulations

Exercise: Upload survey data and notebook to Jupyter

1. Download data and notebook from Canvas
2. Open your Jupyter notebook
3. Upload data and notebook to your Jupyter

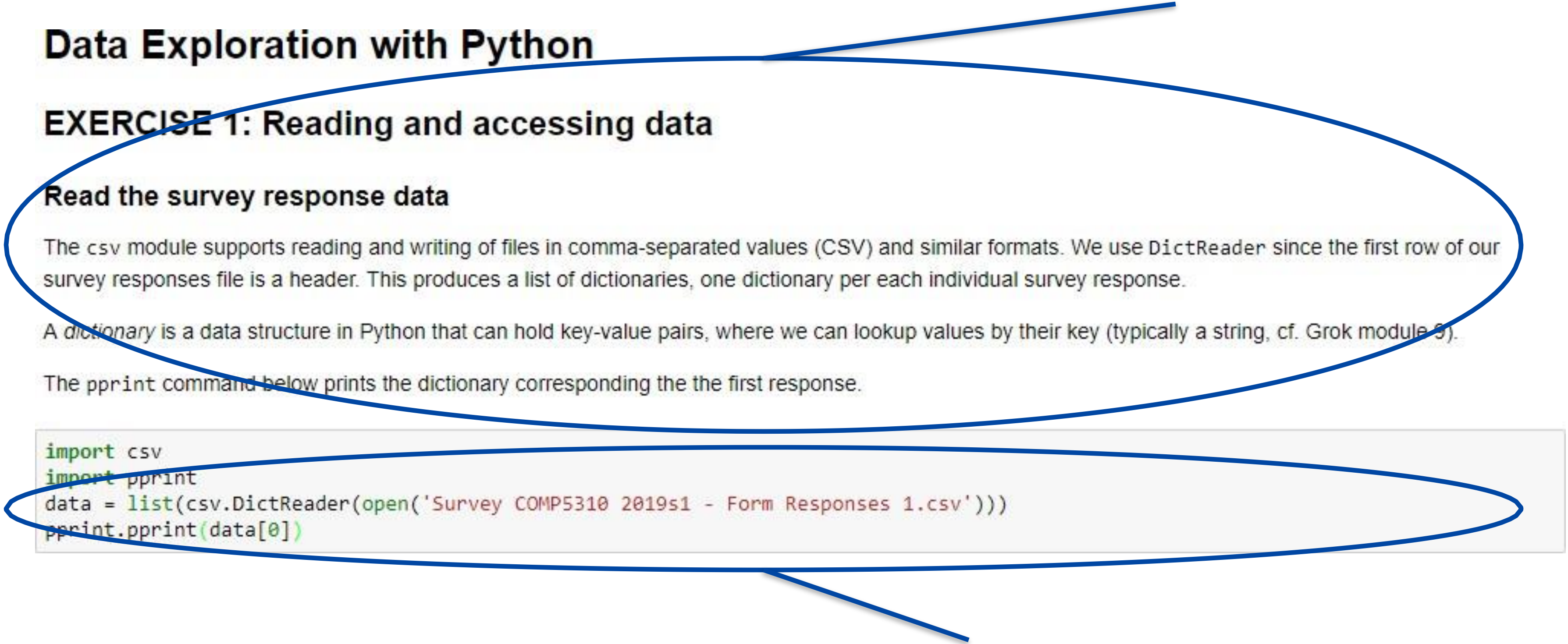
1. **Click here for file open dialogue**
2. **Click upload next to file name**



Select items to perform actions on them.

Jupyter notebook cells

Markdown cell for formatted text



Data Exploration with Python

EXERCISE 1: Reading and accessing data

Read the survey response data

The `csv` module supports reading and writing of files in comma-separated values (CSV) and similar formats. We use `DictReader` since the first row of our survey responses file is a header. This produces a list of dictionaries, one dictionary per each individual survey response.

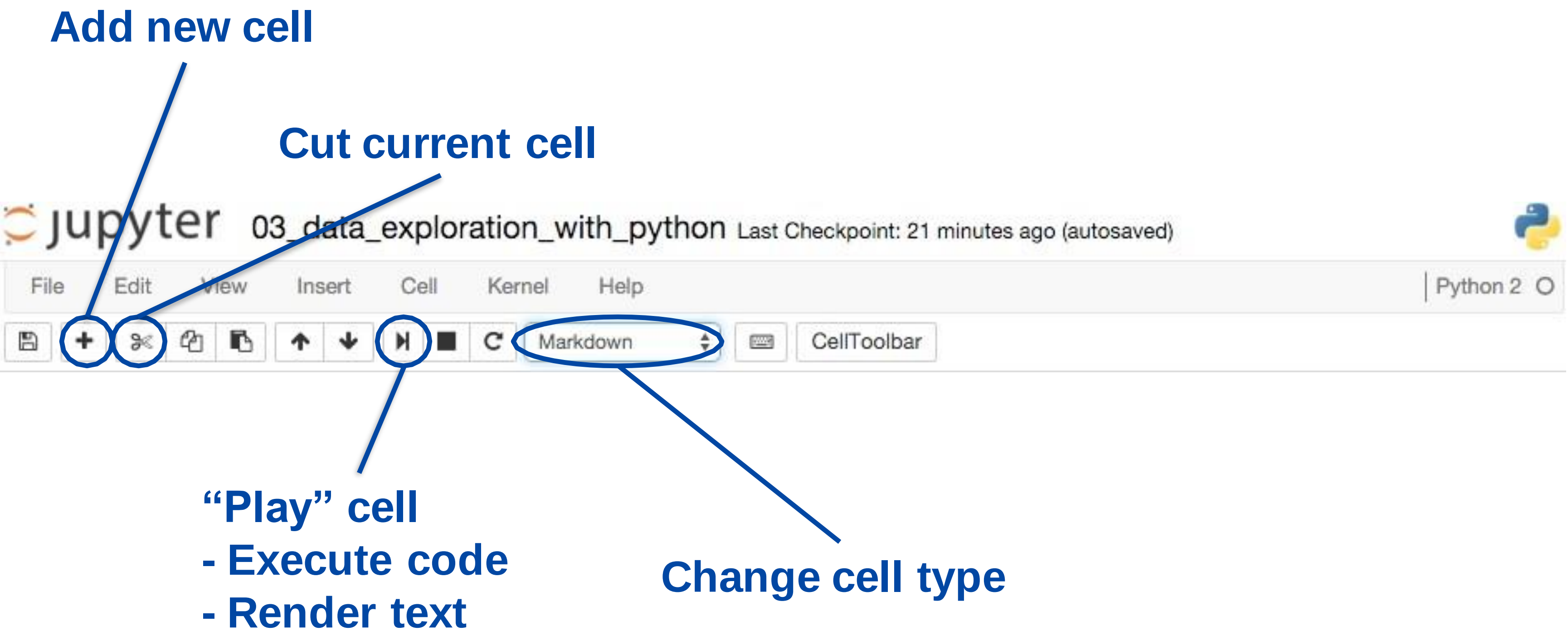
A *dictionary* is a data structure in Python that can hold key-value pairs, where we can lookup values by their key (typically a string, cf. Grok module 9).

The `pprint` command below prints the dictionary corresponding the the first response.

```
import csv
import pprint
data = list(csv.DictReader(open('Survey COMP5310 2019s1 - Form Responses 1.csv')))
pprint.pprint(data[0])
```

Code cell for writing Python commands

Jupyter menu bar


jupyter
03_data_exploration_with_python Last Checkpoint: 21 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Help
Python 2
Markdown
CellToolbar

Add new cell

Cut current cell

"Play" cell
- Execute code
- Render text

Change cell type

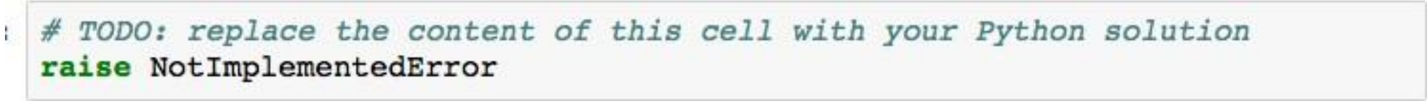
Read data using csv

- Python **csv** module
  - Reads/writes comma-separated values with escaping
  - csv.reader reads rows into arrays
  - csv.DictReader reads rows into ***dictionaries***

EXERCISE 1: Read survey data into notebook

- Execute read cells
  - ⏭ code cell after “Read the survey response data”
  - ⏭ code cell after “Define constants for dictionary keys”
- Access data
  - What Python data types do we get from the csv reader?
  - What is the third respondent’s rating for communication?

- **Tip:** All cells where your input is needed are marked TODO



```
# TODO: replace the content of this cell with your Python solution
raise NotImplementedError
```

Descriptive Statistics

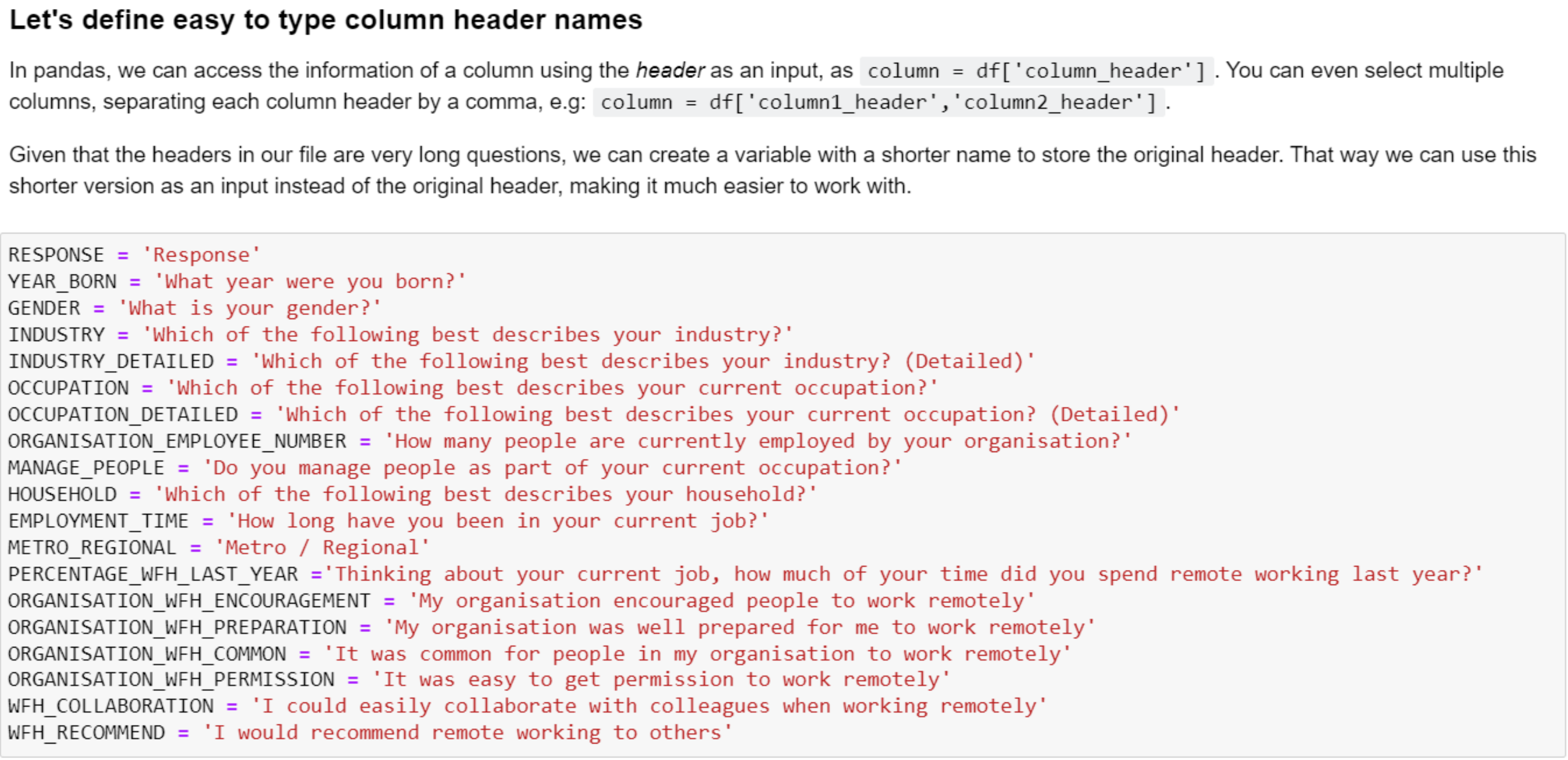
Accessing columns

Now that we have created an easier way to access a column, let's see how it works. Let's select the column that contains the answers to the question *What year were you born?*.

```
df[YEAR_BORN]
```

```
0       1972
1       1972
2       1982
3       1987
4       1991
        ... 
1502    1995
1503    1990
1504    1998
1505    1968
1506    1980
Name: What year were you born?, Length: 1507, dtype: int64
```

Renaming columns



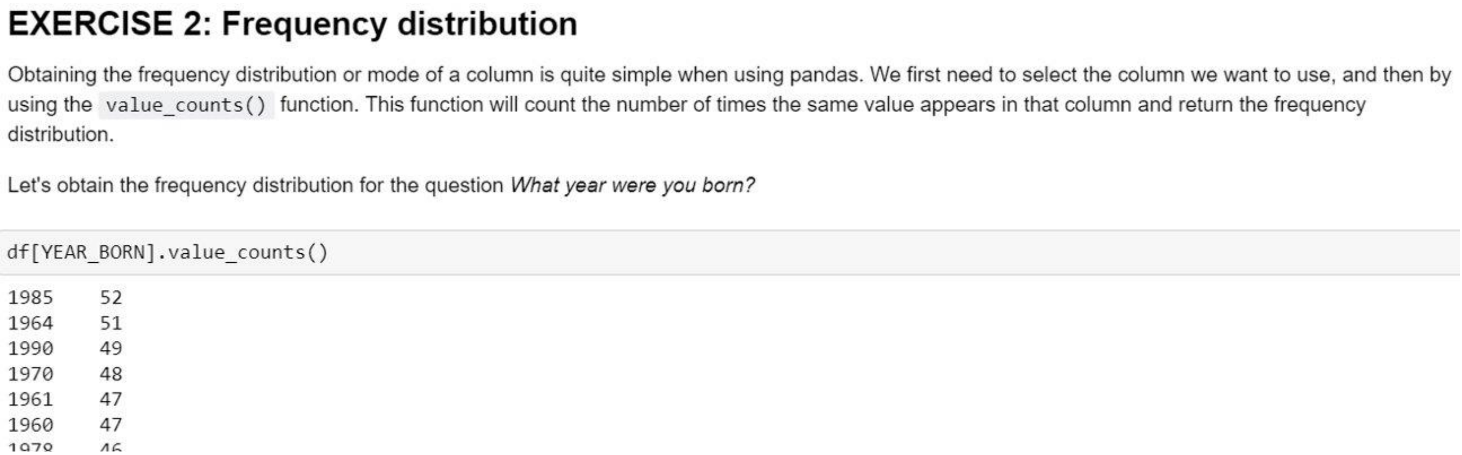
Let's define easy to type column header names

In pandas, we can access the information of a column using the *header* as an input, as `column = df['column_header']`. You can even select multiple columns, separating each column header by a comma, e.g: `column = df['column1_header','column2_header']`.

Given that the headers in our file are very long questions, we can create a variable with a shorter name to store the original header. That way we can use this shorter version as an input instead of the original header, making it much easier to work with.

```
RESPONSE = 'Response'
YEAR_BORN = 'What year were you born?'
GENDER = 'What is your gender?'
INDUSTRY = 'Which of the following best describes your industry?'
INDUSTRY_DETAILED = 'Which of the following best describes your industry? (Detailed)'
OCCUPATION = 'Which of the following best describes your current occupation?'
OCCUPATION_DETAILED = 'Which of the following best describes your current occupation? (Detailed)'
ORGANISATION_EMPLOYEE_NUMBER = 'How many people are currently employed by your organisation?'
MANAGE_PEOPLE = 'Do you manage people as part of your current occupation?'
HOUSEHOLD = 'Which of the following best describes your household?'
EMPLOYMENT_TIME = 'How long have you been in your current job?'
METRO_REGIONAL = 'Metro / Regional'
PERCENTAGE_WFH_LAST_YEAR ='Thinking about your current job, how much of your time did you spend remote working last year?'
ORGANISATION_WFH_ENCOURAGEMENT = 'My organisation encouraged people to work remotely'
ORGANISATION_WFH_PREPARATION = 'My organisation was well prepared for me to work remotely'
ORGANISATION_WFH_COMMON = 'It was common for people in my organisation to work remotely'
ORGANISATION_WFH_PERMISSION = 'It was easy to get permission to work remotely'
WFH_COLLABORATION = 'I could easily collaborate with colleagues when working remotely'
WFH_RECOMMEND = 'I would recommend remote working to others'
```

Frequency distributions and mode



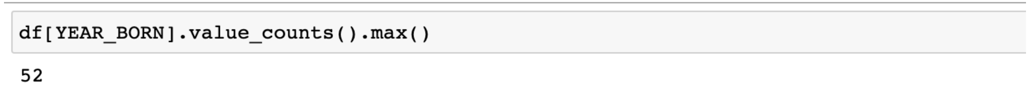
EXERCISE 2: Frequency distribution

Obtaining the frequency distribution or mode of a column is quite simple when using pandas. We first need to select the column we want to use, and then by using the `value_counts()` function. This function will count the number of times the same value appears in that column and return the frequency distribution.

Let's obtain the frequency distribution for the question *What year were you born?*

```
df[YEAR_BORN].value_counts()
```

```
1985    52
1964    51
1990    49
1970    48
1961    47
1960    47
1978    46
```



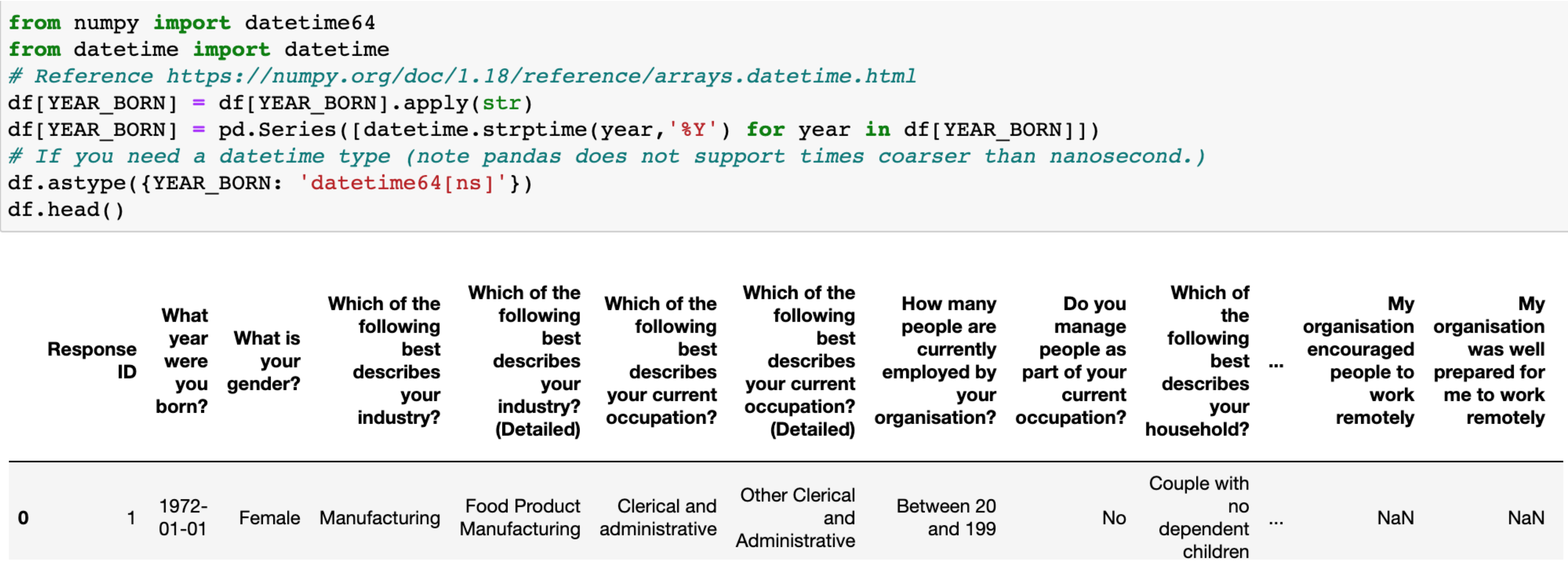
```
df[YEAR_BORN].value_counts().max()
```

```
52
```

Cleaning data: convert to correct types

- The Python csv module reads everything as string types
- Need to convert as appropriate (e.g., int, float, timestamp)
  - int() creates integer objects, e.g., -1, 101
  - float() creates floating point object, e.g., 3.14, 2.71
  - datetime.strptime() creates datetime objects from strings
- Pandas will guess types
- Need to convert as appropriate (e.g., int, float, timestamp)
  - pandas.DataFrame.dtype
  - DataFrame.astype

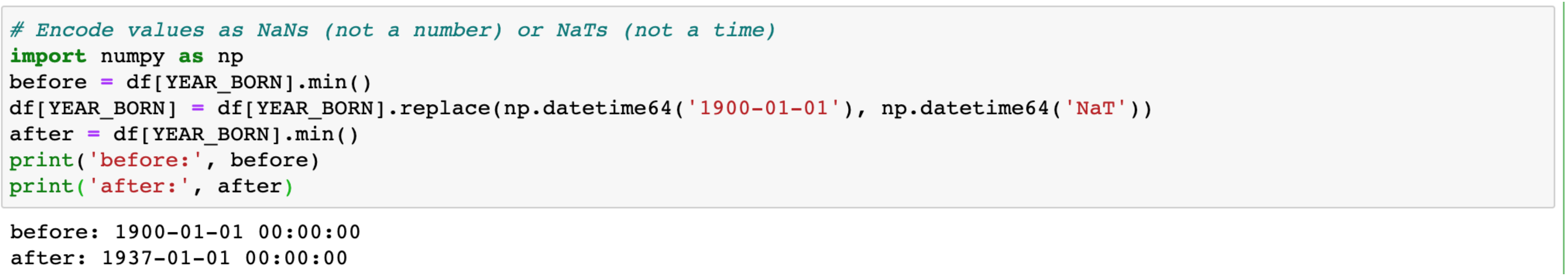
Fixing types



```
from numpy import datetime64
from datetime import datetime
# Reference https://numpy.org/doc/1.18/reference/arrays.datetime.html
df[YEAR_BORN] = df[YEAR_BORN].apply(str)
df[YEAR_BORN] = pd.Series([datetime.strptime(year,'%Y') for year in df[YEAR_BORN]])
# If you need a datetime type (note pandas does not support times coarser than nanosecond.)
df.astype({YEAR_BORN: 'datetime64[ns]'})
df.head()
```

	Response ID	What year were you born?	What is your gender?	Which of the following best describes your industry?	Which of the following best describes your industry? (Detailed)	Which of the following best describes your current occupation?	Which of the following best describes your current occupation? (Detailed)	How many people are currently employed by your organisation?	Do you manage people as part of your current occupation?	Which of the following best describes your household?	...	My organisation encouraged people to work remotely	My organisation was well prepared for me to work remotely
0	1	1972-01-01	Female	Manufacturing	Food Product Manufacturing	Clerical and administrative	Other Clerical and Administrative	Between 20 and 199	No	Couple with no dependent children	...	NaN	NaN

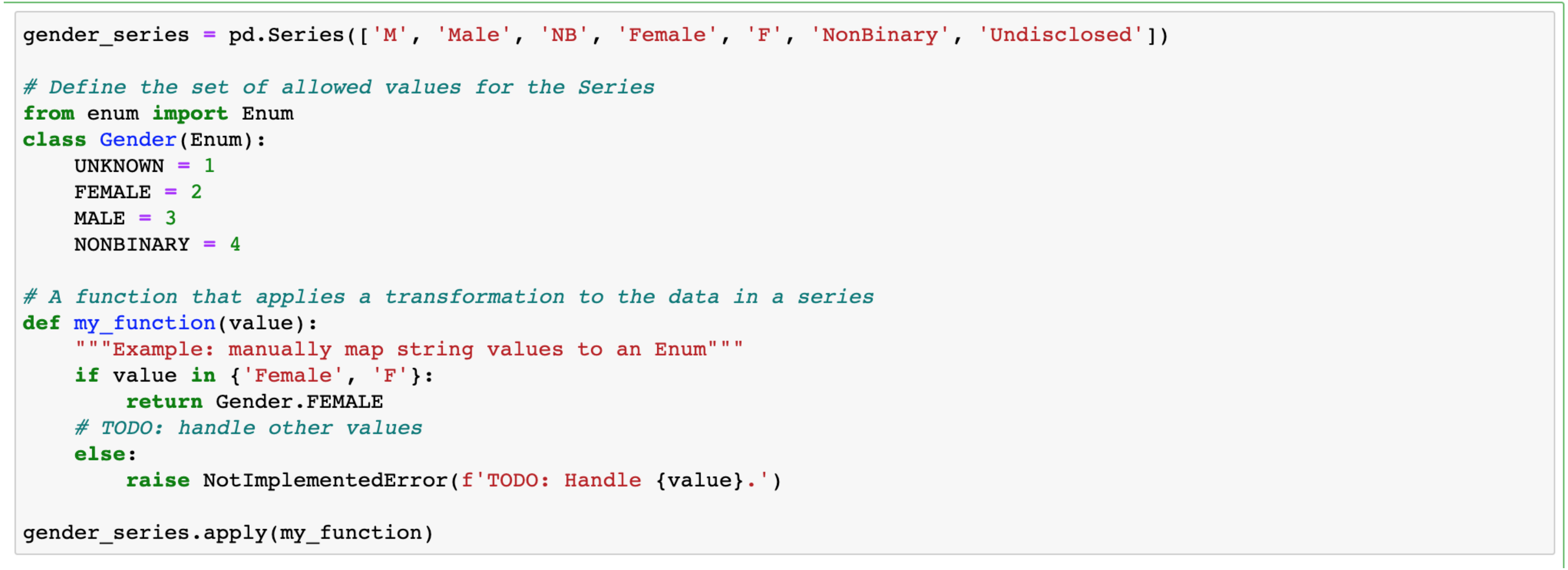
Encoding NaNs



```
# Encode values as NaNs (not a number) or NaTs (not a time)
import numpy as np
before = df[YEAR_BORN].min()
df[YEAR_BORN] = df[YEAR_BORN].replace(np.datetime64('1900-01-01'), np.datetime64('NaT'))
after = df[YEAR_BORN].min()
print('before:', before)
print('after:', after)
```

```
before: 1900-01-01 00:00:00
after: 1937-01-01 00:00:00
```

Defining custom functions to clean a column



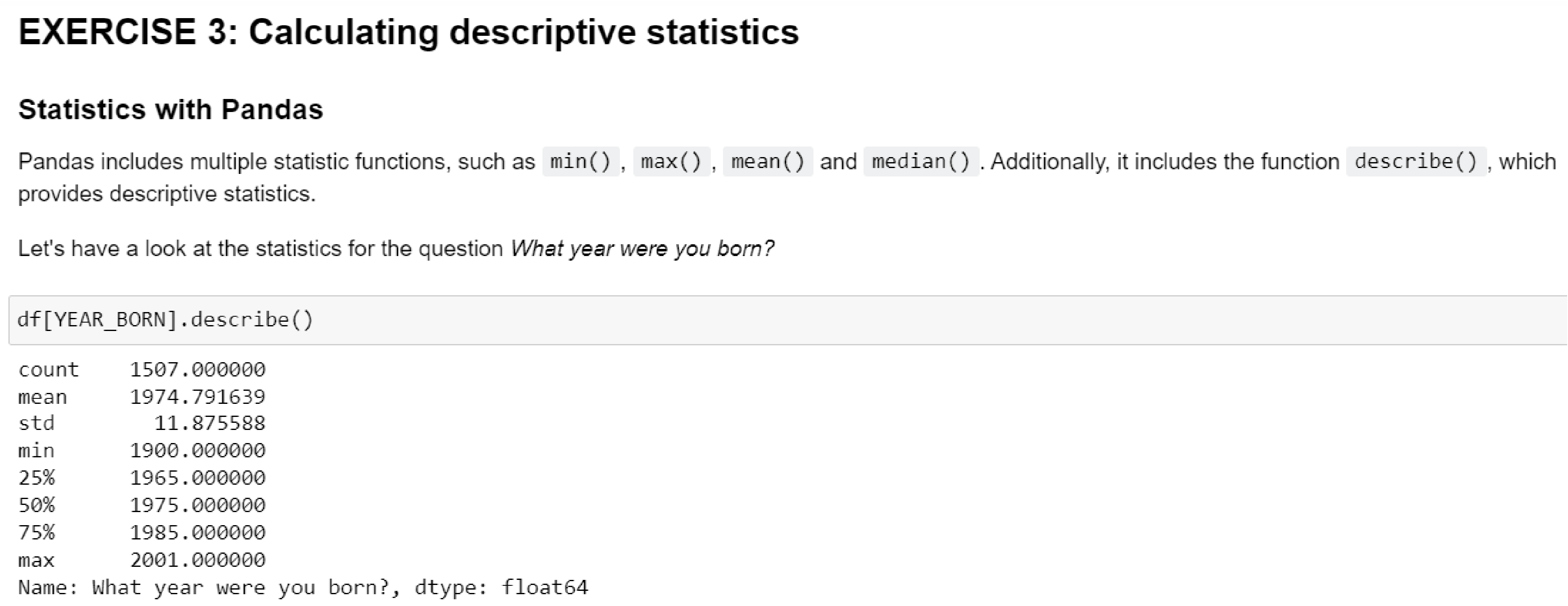
```
gender_series = pd.Series(['M', 'Male', 'NB', 'Female', 'F', 'NonBinary', 'Undisclosed'])

# Define the set of allowed values for the Series
from enum import Enum
class Gender(Enum):
    UNKNOWN = 1
    FEMALE = 2
    MALE = 3
    NONBINARY = 4

# A function that applies a transformation to the data in a series
def my_function(value):
    """Example: manually map string values to an Enum"""
    if value in {'Female', 'F'}:
        return Gender.FEMALE
    # TODO: handle other values
    else:
        raise NotImplementedError(f'TODO: Handle {value}.')

gender_series.apply(my_function)
```

Central tendency and dispersion with numpy

- **Numpy** provides various statistics for numeric data
- Median, percentiles, mean, standard deviation, etc
- nan* versions calculate same statistics, ignoring NaN values

- Reference page for numpy statistics:
  http://docs.scipy.org/doc/numpy/reference/routines.statistics.html

Central tendency and dispersion with pandas



EXERCISE 3: Calculating descriptive statistics

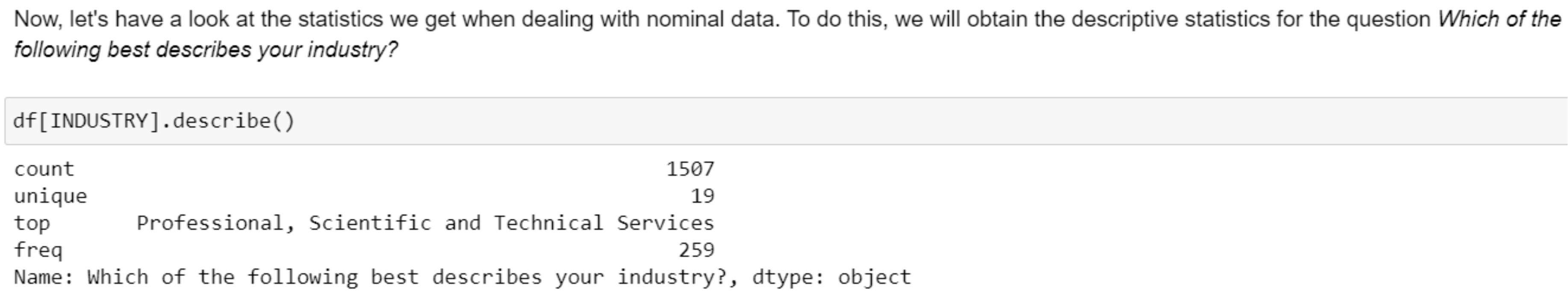
Statistics with Pandas

Pandas includes multiple statistic functions, such as `min()`, `max()`, `mean()` and `median()`. Additionally, it includes the function `describe()`, which provides descriptive statistics.

Let's have a look at the statistics for the question *What year were you born?*

```
df[YEAR_BORN].describe()
```

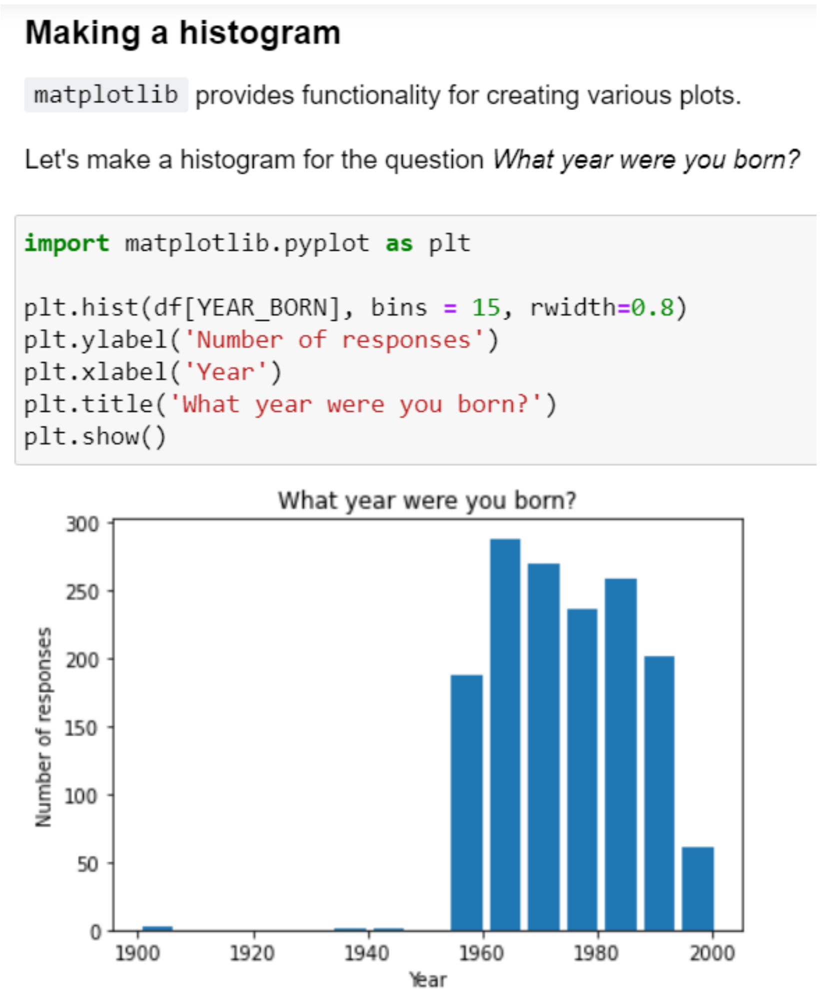
```
count    1507.000000
mean     1974.791639
std        11.875588
min      1900.000000
25%      1965.000000
50%      1975.000000
75%      1985.000000
max      2001.000000
Name: What year were you born?, dtype: float64
```



Now, let's have a look at the statistics we get when dealing with nominal data. To do this, we will obtain the descriptive statistics for the question *Which of the following best describes your industry?*

```
df[INDUSTRY].describe()
```

```
count                                               1507
unique                                                19
top       Professional, Scientific and Technical Services
freq                                                 259
Name: Which of the following best describes your industry?, dtype: object
```

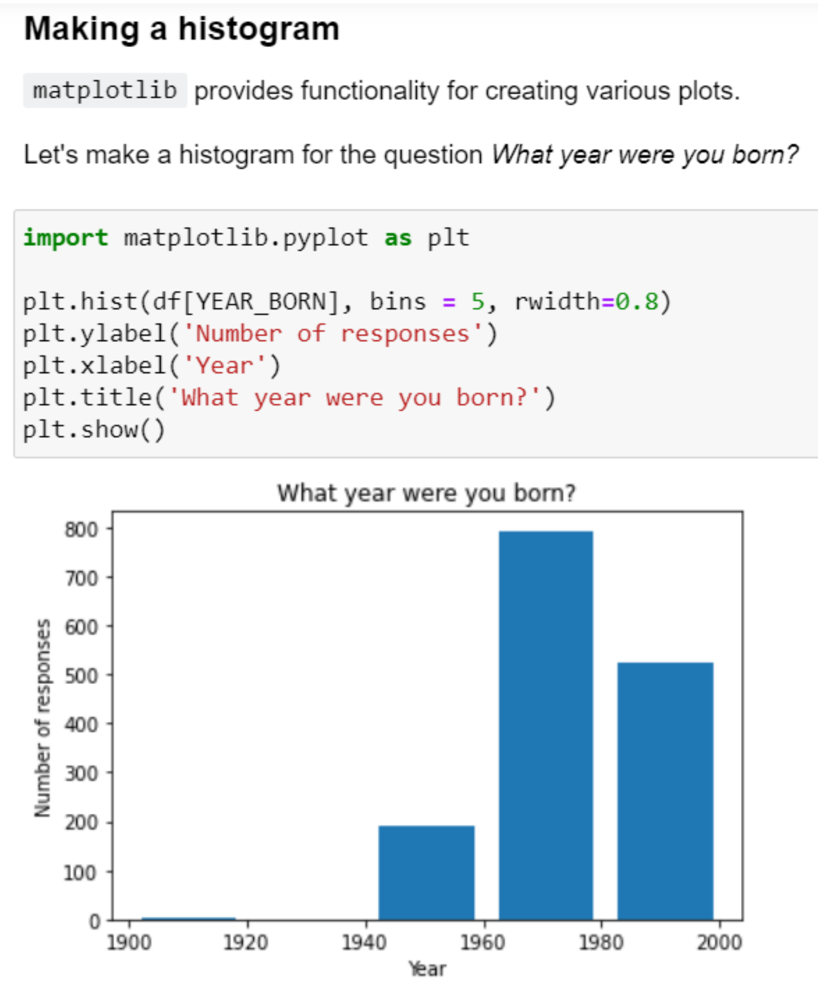


Making a histogram

`matplotlib` provides functionality for creating various plots.

Let's make a histogram for the question *What year were you born?*

```
import matplotlib.pyplot as plt

plt.hist(df[YEAR_BORN], bins = 15, rwidth=0.8)
plt.ylabel('Number of responses')
plt.xlabel('Year')
plt.title('What year were you born?')
plt.show()
```



Making a histogram

`matplotlib` provides functionality for creating various plots.

Let's make a histogram for the question *What year were you born?*

```
import matplotlib.pyplot as plt

plt.hist(df[YEAR_BORN], bins = 5, rwidth=0.8)
plt.ylabel('Number of responses')
plt.xlabel('Year')
plt.title('What year were you born?')
plt.show()
```

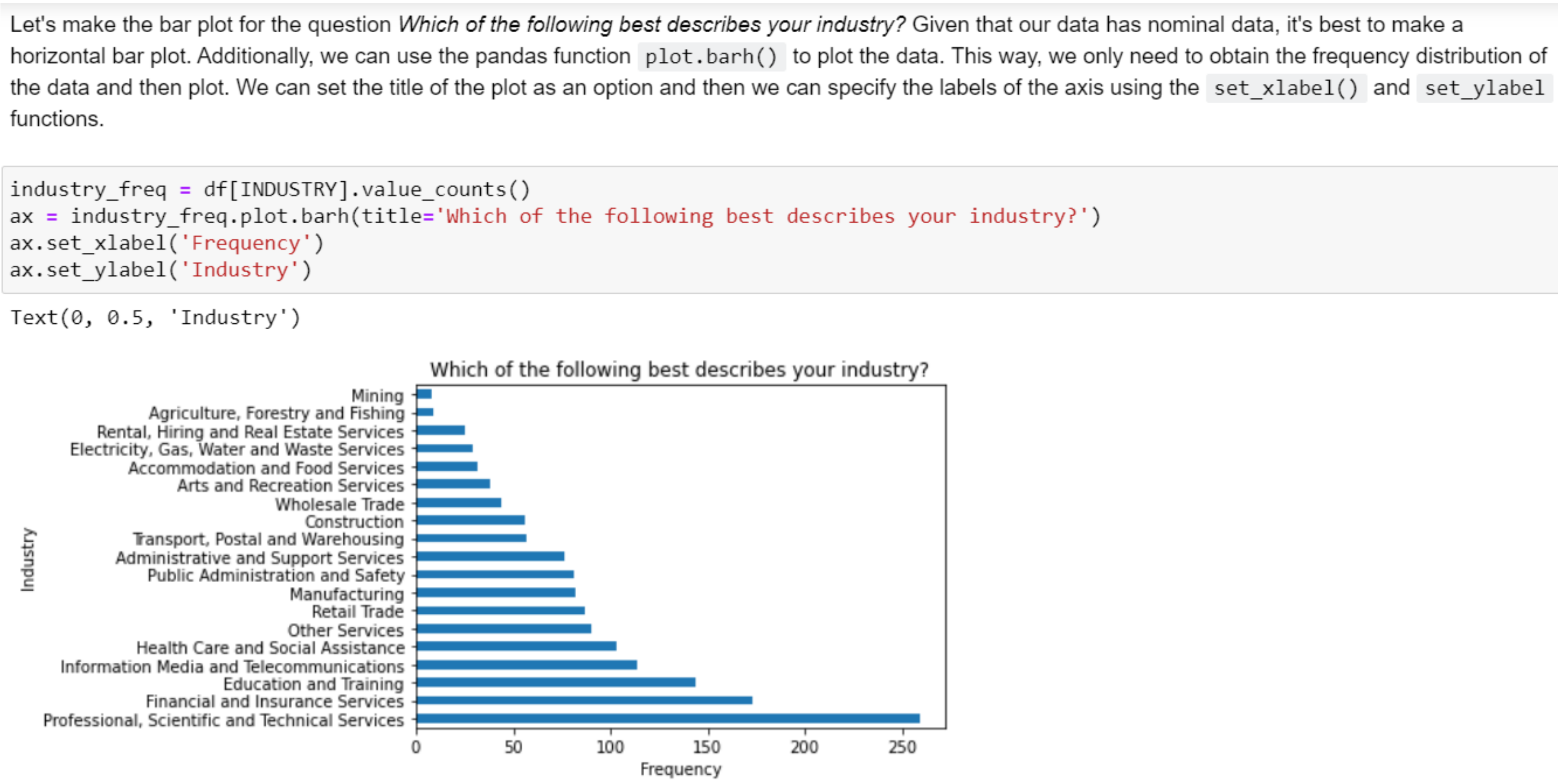
Visualisation

Visualising data with matplotlib

- Matplotlib provides functionality for creating various plots
- Bar charts, line charts, scatter plots, etc

- Reference page for pyplot: http://matplotlib.org/api/pyplot_api.html
- Documentation: http://matplotlib.org/contents.html

Creating a bar chart



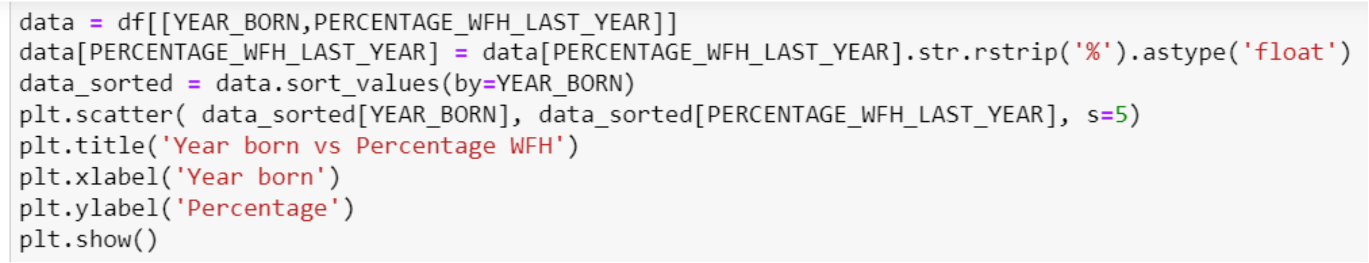
Let's make the bar plot for the question *Which of the following best describes your industry?* Given that our data has nominal data, it's best to make a horizontal bar plot. Additionally, we can use the pandas function `plot.barh()` to plot the data. This way, we only need to obtain the frequency distribution of the data and then plot. We can set the title of the plot as an option and then we can specify the labels of the axis using the `set_xlabel()` and `set_ylabel` functions.

```
industry_freq = df[INDUSTRY].value_counts()
ax = industry_freq.plot.barh(title='Which of the following best describes your industry?')
ax.set_xlabel('Frequency')
ax.set_ylabel('Industry')
```

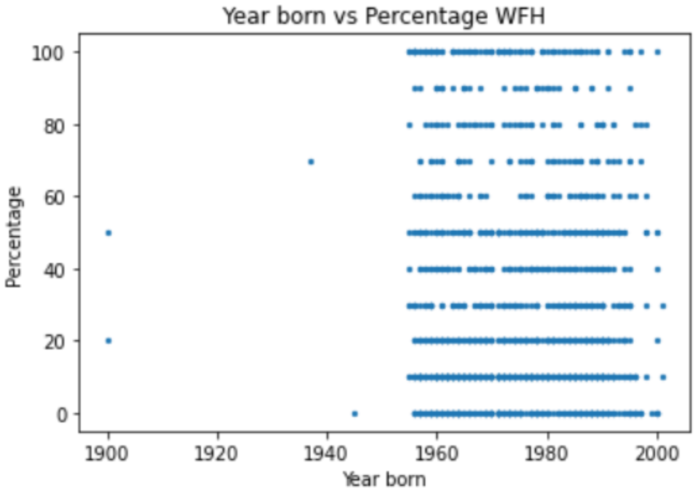
```
Text(0, 0.5, 'Industry')
```

Which of the following best describes your industry?
Industry
Mining
Agriculture, Forestry and Fishing
Rental, Hiring and Real Estate Services
Electricity, Gas, Water and Waste Services
Accommodation and Food Services
Arts and Recreation Services
Wholesale Trade
Construction
Transport, Postal and Warehousing
Administrative and Support Services
Public Administration and Safety
Manufacturing
Retail Trade
Other Services
Health Care and Social Assistance
Information Media and Telecommunications
Education and Training
Financial and Insurance Services
Professional, Scientific and Technical Services
0
50
100
150
200
250
Frequency

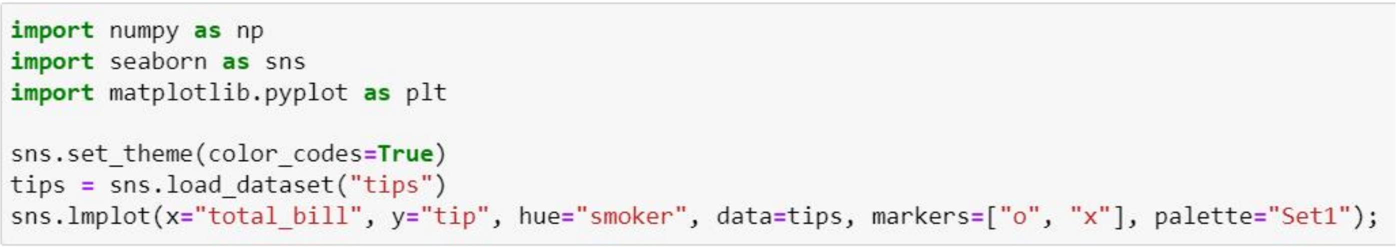
Creating a scatter plot



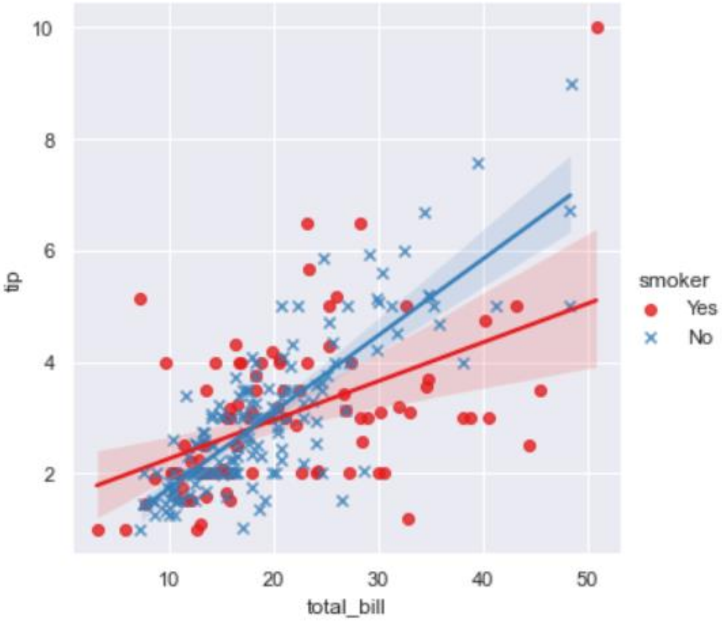
```
data = df[[YEAR_BORN,PERCENTAGE_WFH_LAST_YEAR]]
data[PERCENTAGE_WFH_LAST_YEAR] = data[PERCENTAGE_WFH_LAST_YEAR].str.rstrip('%').astype('float')
data_sorted = data.sort_values(by=YEAR_BORN)
plt.scatter( data_sorted[YEAR_BORN], data_sorted[PERCENTAGE_WFH_LAST_YEAR], s=5)
plt.title('Year born vs Percentage WFH')
plt.xlabel('Year born')
plt.ylabel('Percentage')
plt.show()
```



Creating a scatter plot

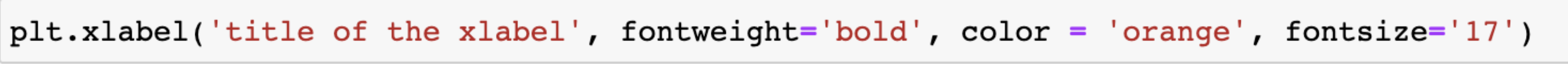


```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_theme(color_codes=True)
tips = sns.load_dataset("tips")
sns.lmplot(x="total_bill", y="tip", hue="smoker", data=tips, markers=["o", "x"], palette="Set1");
```



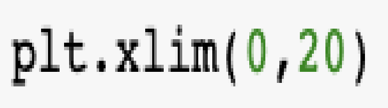
Customise plots

- Customise Axis Titles: xlabel(), ylabel() function



```
plt.xlabel('title of the xlabel', fontweight='bold', color = 'orange', fontsize='17')
```

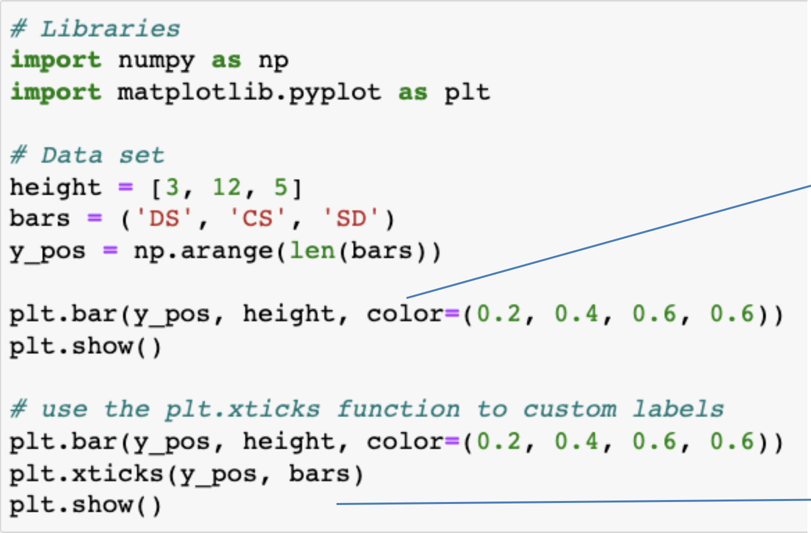
- Change Axis Limits: xlim(), ylim()



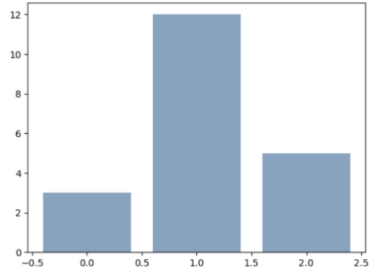
```
plt.xlim(0,20)
```

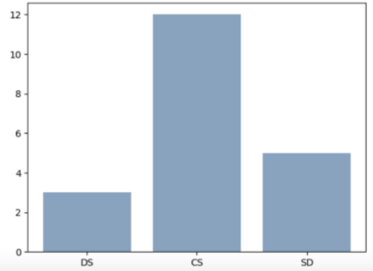
Customise plots (add x-ticks, change axis limits)

- Customize Axis Tick Labels: x_ticks() and y_ticks() function



```
# Libraries
import numpy as np
import matplotlib.pyplot as plt

# Data set
height = [3, 12, 5]
bars = ('DS', 'CS', 'SD')
y_pos = np.arange(len(bars))

plt.bar(y_pos, height, color=(0.2, 0.4, 0.6, 0.6))
plt.show()

# use the plt.xticks function to custom labels
plt.bar(y_pos, height, color=(0.2, 0.4, 0.6, 0.6))
plt.xticks(y_pos, bars)
plt.show()
```



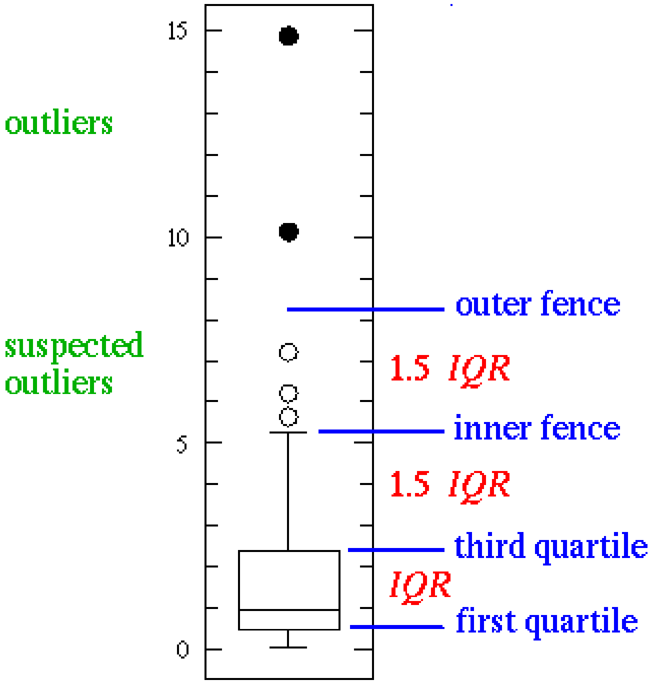


Box plots and correlation

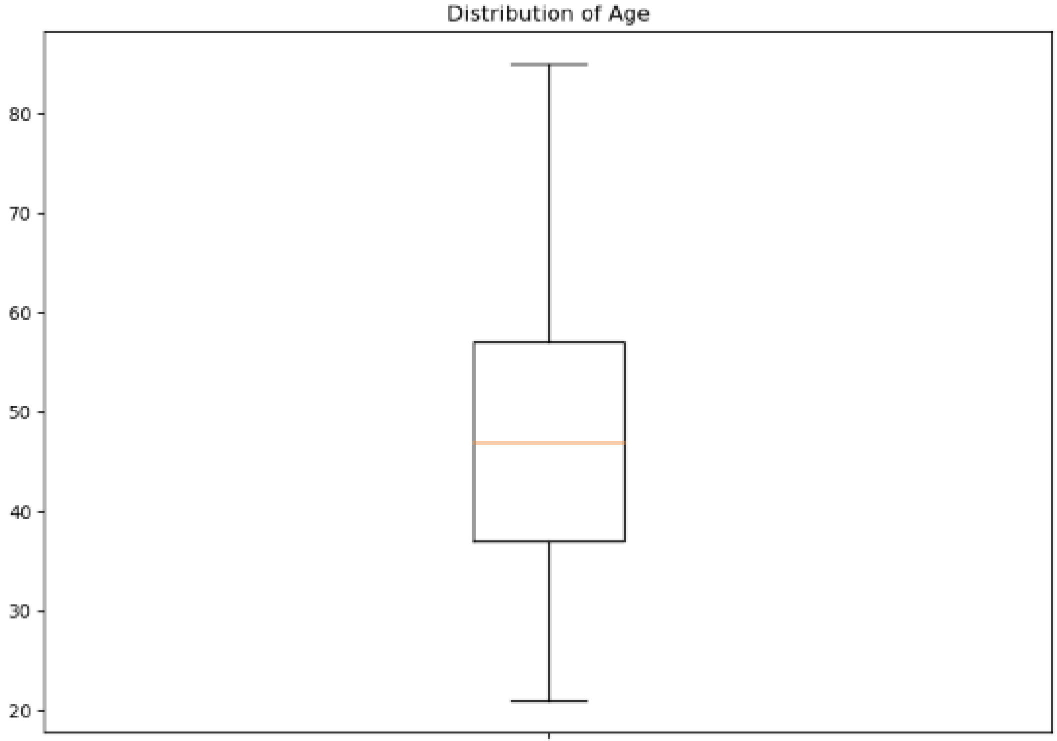
Using boxplots to compare distributions

- Mean and stdev are not informative when data is skewed
- **Box plots** summarise data based on 5 numbers:
  - Lower inner fence – Q1–1.5*IQR
  - First quartile (Q1) – equivalent to 25th percentile
  - Median (Q2) – equivalent to 50th percentile
  - Third quartile (Q3) – equivalent to 75th percentile
  - Upper inner fence – Q3+1.5*IQR
- Values outside fences are outliers
- Sometimes include outer fences at 3*IQR

Box Plots illustrated



A box plot for the age distribution



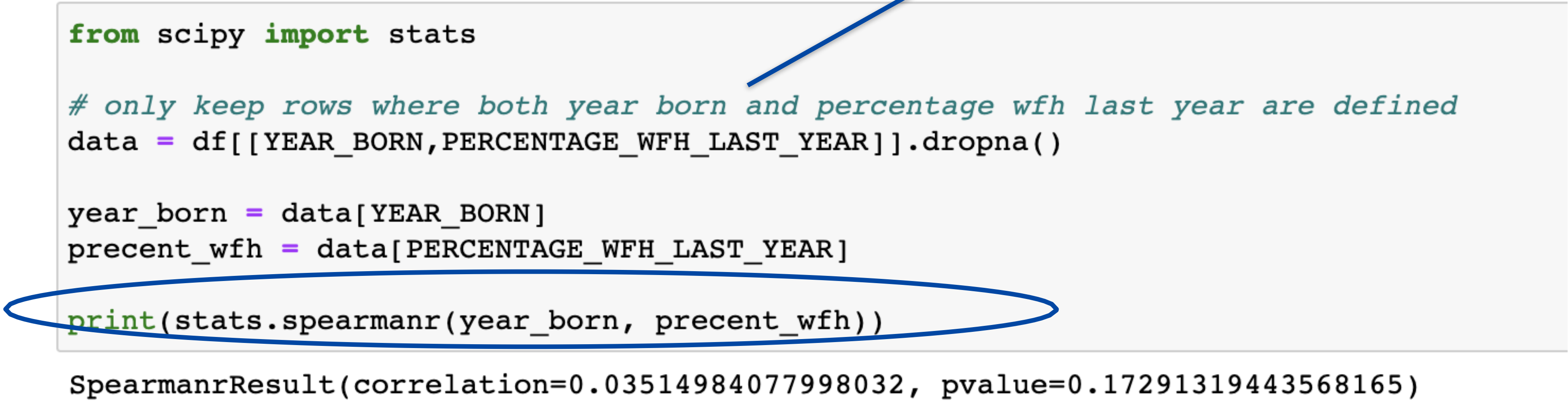
Using correlation statistics to measure dependence

- Scipy includes various correlation statistics
  - Pearson's *r* for two normally distributed variables
  - Spearman's *rho* for ratio data, ordinal data, etc (rank-order correlation)
  - Kendall's *tau* for ordinal variables

- List of various scipy statistics including correlation coefficients: http://docs.scipy.org/doc/scipy-0.14.0/reference/stats.html

Calculating correlation

Since correlation is paired, grab values where both variables are defined



```
from scipy import stats

# only keep rows where both year born and percentage wfh last year are defined
data = df[[YEAR_BORN,PERCENTAGE_WFH_LAST_YEAR]].dropna()

year_born = data[YEAR_BORN]
precent_wfh = data[PERCENTAGE_WFH_LAST_YEAR]

print(stats.spearmanr(year_born, precent_wfh))
```

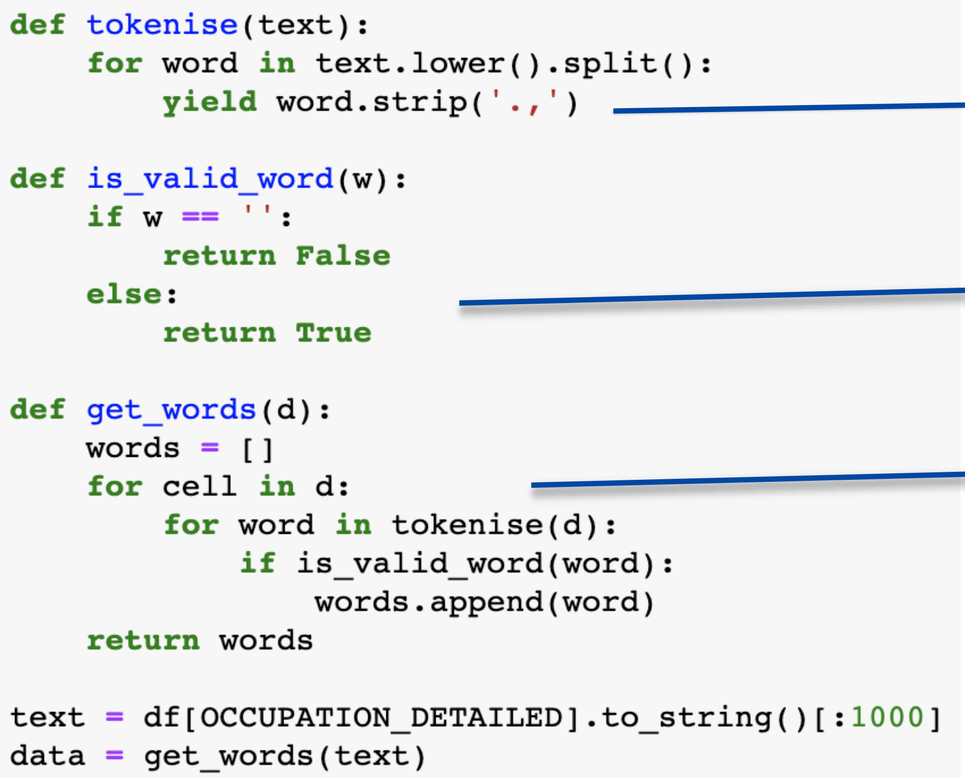
```
SpearmanrResult(correlation=0.03514984077998032, pvalue=0.1729131944356816
5)
```

Calculate Spearman’s rho

Text Data

THE UNIVERSITY OF
SYDNEY

A simple whitespace tokeniser



```python
def tokenise(text):
    for word in text.lower().split():
        yield word.strip('.,')

def is_valid_word(w):
    if w == '':
        return False
    else:
        return True

def get_words(d):
    words = []
    for cell in d:
        for word in tokenise(d):
            if is_valid_word(word):
                words.append(word)
    return words

text = df[OCCUPATION_DETAILED].to_string()[:1000]
data = get_words(text)
```

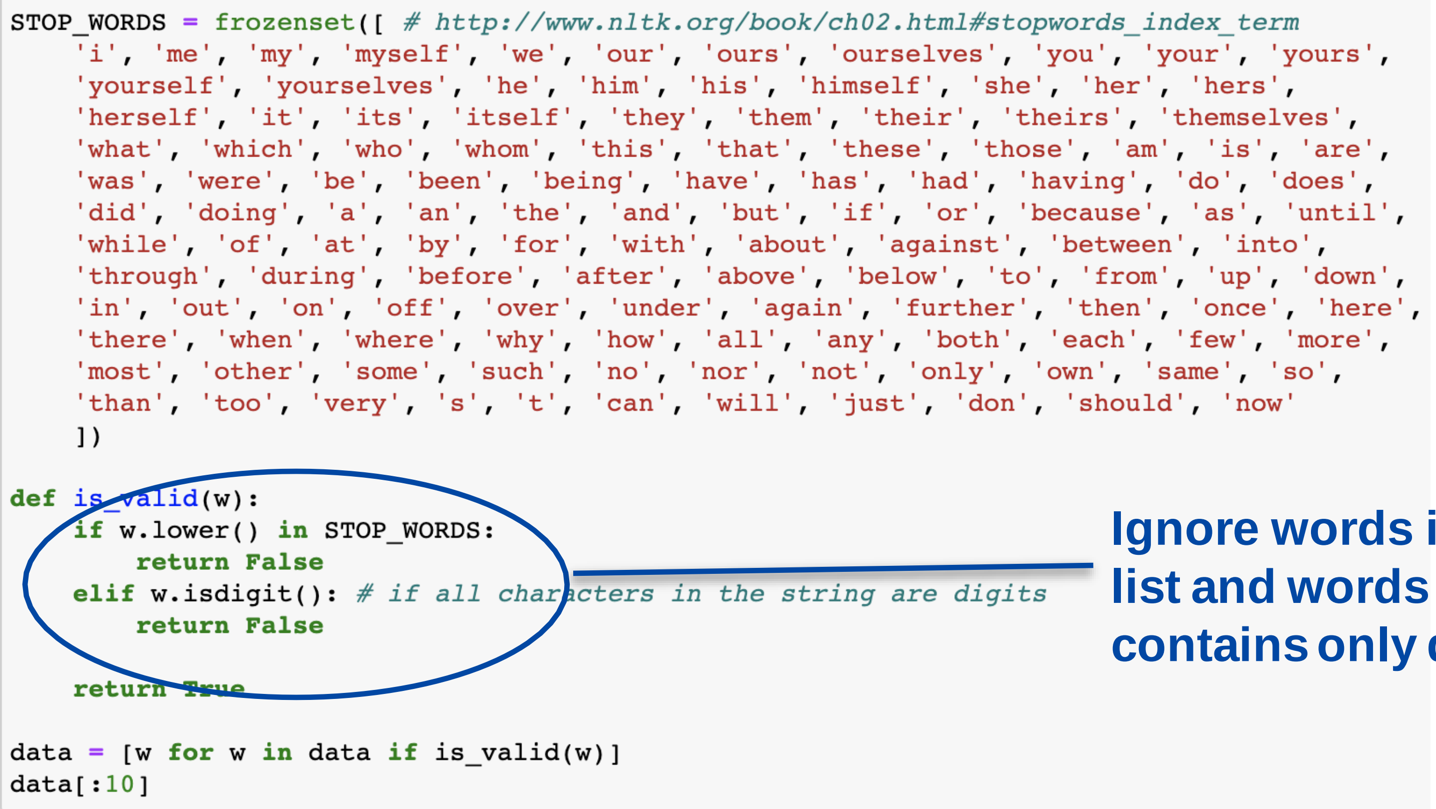
Convert text string to lower case and split on whitespace

Remove leading/trailing ‘.’ and ‘,’

Ignore empty strings

Get each word token from each definition

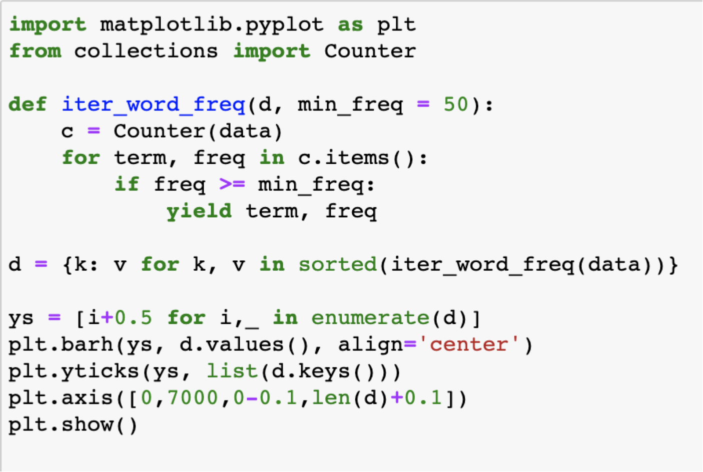
Removing stop words



```
STOP_WORDS = frozenset([ # http://www.nltk.org/book/ch02.html#stopwords_index_term
    'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours',
    'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
    'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
    'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
    'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
    'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
    'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
    'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
    'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
    'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
    'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
    'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now'
    ])

def is_valid(w):
    if w.lower() in STOP_WORDS:
        return False
    elif w.isdigit(): # if all characters in the string are digits
        return False

    return True

data = [w for w in data if is_valid(w)]
data[:10]
```

Ignore words in stop list and words contains only digits

Plotting most frequent words

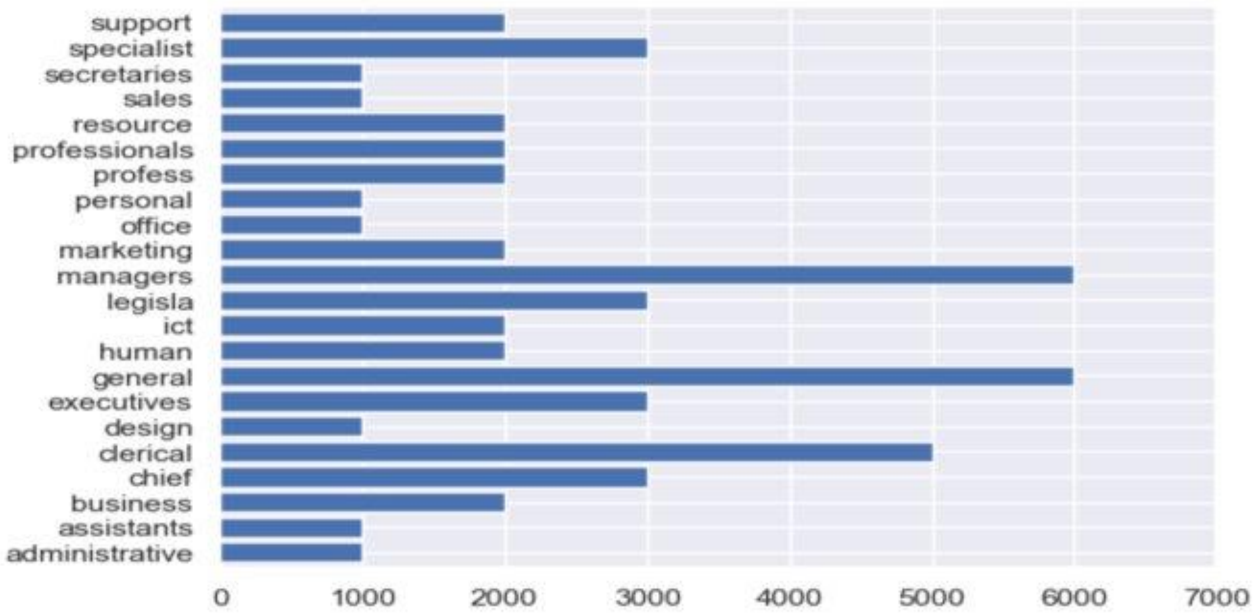


```
import matplotlib.pyplot as plt
from collections import Counter

def iter_word_freq(d, min_freq = 50):
    c = Counter(data)
    for term, freq in c.items():
        if freq >= min_freq:
            yield term, freq

d = {k: v for k, v in sorted(iter_word_freq(data))}

ys = [i+0.5 for i,_ in enumerate(d)]
plt.barh(ys, d.values(), align='center')
plt.yticks(ys, list(d.keys()))
plt.axis([0,7000,0-0.1,len(d)+0.1])
plt.show()
```

Yield words and their frequencies if they occur 4 or more times

Create a horizontal bar chart

A term frequency bar chart



Review

THE UNIVERSITY OF
SYDNEY

Notes

- Python a good example of a scripting language for DS
- programmatic approaches allow for more powerful / flexible data preparation and analysis,
  - and more control on the visualisations
- Many useful support libraries available in the Python ecosystem
  - numpy, scipy, matplotlib

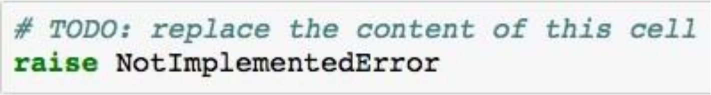
Additional reading (not examinable)

- matplotlib API reference
  http://matplotlib.org/api/pyplot_api.html
- NumPy and SciPy documentation
  http://docs.scipy.org/doc/
- Data Analysis with Python (O'Reilly)
  http://shop.oreilly.com/product/0636920023784.do

Participation

Requirements

- Submit code at end of each week
- **Jupyter Notebooks:**
  - The various exercises have placeholder cells marked as TODO:



```
# TODO: replace the content of this cell
raise NotImplementedError
```

  - The content of these cells needs to be replaced with your own solution => basis for participation marking

Output

- Code/spreadsheets from exercises

Marking

- 10% of overall mark

TODO until Monday next week:

Export your Jupyter notebooks as HTML and upload to Canvas

Next Time

THE UNIVERSITY OF SYDNEY

Next week: Cleaning and storing data

Objective

How to clean and prepare a data set for effective analysis and for storage in a database.

Lecture

- ETL: extract, transform and load
- CSV and SQL

Readings

- Data Science from Scratch: Ch 10 (start) + Ch 23

Exercises

- Preparing CSV data for storage
- Storing data in a database

TODO in W3

- Grok Python modules 10-12
- Grok SQL modules 3 and 4
- Explore and select project data

Questions?