Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Understanding Data vs. Machine Training

Malte Luttermann – Institute for Humanities-Centered Artificial Intelligence (CHAI)

December 12, 2025

Version from December 11, 2025

Overview of Contents

(1) Programming language Python
  (a) Introduction and first steps
  (b) Basics
  (c) Advanced
(2) Markup languages
  (a) LaTeX
  (b) Markdown
(3) Development environments
  (a) Jupyter notebooks
(4) Version control
  (a) Git and GitHub
(5) Scientific computing
  (a) NumPy and SciPy
(6) Data processing and visualisation
  (a) Pandas, matplotlib, and NLTK
(7) Machine learning (scikit-learn)
  (a) Basics (datasets, analysis)
  (b) Simple methods (clustering, . . .)
(8) Deep learning
  (a) PyTorch

Topics for Today

(1) Data processing
   (a) Python internal
   (b) Pandas
   (c) Natural Language Toolkit (NLTK)
(2) Data visualisation
   (a) Matplotlib

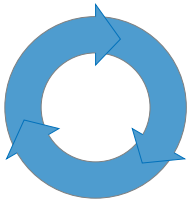
Acknowledgement

- The upcoming slides are taken from the following lecture and have been translated and partially modified:
  - Dr. Magnus Bender: »Python für Machine Learning und Data Science« as part of the German course »Werkzeuge für das wissenschaftliche Arbeiten«

Data Processing Overview

- JSON
- CSV
- NumPy (`.npy`, `.npz`)
- TXT
- ...

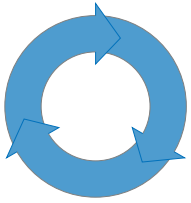
Data Processing Overview

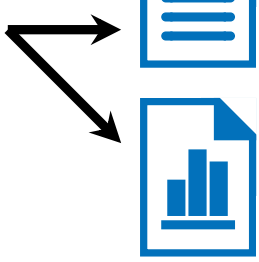


- JSON
- CSV
- NumPy (`.npy`, `.npz`)
- TXT
- ...

Data Processing Overview

- JSON
- CSV
- NumPy (`.npy`, `.npz`)
- TXT
- ...





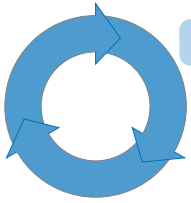
- JSON
- Pickle
- Shelve
- NumPy (`.npy`, `.npz`)

- Tables
- Graphics

Data Processing Overview

- JSON
- CSV
- NumPy (`.npy`, `.npz`)
- TXT
- ...

Data processing



Data processing

Data visualisation

- JSON
- Pickle
- Shelve
- NumPy (`.npy`, `.npz`)

- Tables
- Graphics

JavaScript Object Notation (JSON)

```python
import json

d = {
  "a" : 1,
  "b" : "B",
  "c" : [1.2, 1.3, 1.4],
  "e" : None
}

json_str = json.dumps(d, indent=2)
print(json_str)

d_ = json.loads(json_str)
print(d_)
```

JavaScript Object Notation (JSON)

```python
import json

d = {
   "a" : 1,
   "b" : "B",
   "c" : [1.2, 1.3, 1.4],
   "e" : None
}

json_str = json.dumps(d, indent=2)
print(json_str)

d_ = json.loads(json_str)
print(d_)
```

```
$> python3 script.py

{
  "a": 1,
  "b": "B",
  "c": [
    1.2,
    1.3,
    1.4
  ],
  "e": null
}
{'a': 1, 'b': 'B', 'c': [1.2, 1.3,
1.4], 'e': None}
```

JavaScript Object Notation (JSON)

Supported in (almost) all programming languages

```python
import json

d = {
  "a" : 1,
  "b" : "B",
  "c" : [1.2, 1.3, 1.4],
  "e" : None
}

json_str = json.dumps(d, indent=2)
print(json_str)

d_ = json.loads(json_str)
print(d_)
```

```
$> python3 script.py

{
  "a": 1,
  "b": "B",
  "c": [
    1.2,
    1.3,
    1.4
  ],
  "e": null
}
{'a': 1, 'b': 'B', 'c': [1.2, 1.3,
1.4], 'e': None}
```

JavaScript Object Notation (JSON)

Supported in (almost) all programming languages

```python
import json

d = {
  "a" : 1,
  "b" : "B",
  "c" : [1.2, 1.3, 1.4],
  "e" : None
}

json_str = json.dumps(d, indent=2)
print(json_str)

d_ = json.loads(json_str)
print(d_)
```

```
$> python3 script.py

{
  "a": 1,
  "b": "B",
  "c": [
    1.2,
    1.3,
    1.4
  ],
  "e": null
}
{'a': 1, 'b': 'B', 'c': [1.2, 1.3,
1.4], 'e': None}
```

`json.dumps` supports types `dict`, `list`, `str`, `int`, `float`, `True`, `False`, and `None`

Pickle

```
1  import pickle
2
3  class Triple():
4    def __init__(self, a, b, c):
5      self.a, self.b, self.c = a, b, c
6    def __repr__(self):
7      return str((self.a, self.b, self.c))
8
9  d = [Triple(1, 2, 3), Triple(4, 5, 6), Triple(7, 8, 9)]
10
11 pickle_data = pickle.dumps(d)
12 d_ = pickle.loads(pickle_data)
13 print(d, d_)
14
15 pickle.dump(d, open("d.pickle", "wb"))
16 d__ = pickle.load(open("d.pickle", "rb"))
17 print(d__)
```

Pickle

```
$> python3 script.py

[(1, 2, 3), (4, 5, 6), (7, 8, 9)] [(1, 2, 3), (4, 5, 6), (7, 8, 9)]
[(1, 2, 3), (4, 5, 6), (7, 8, 9)]
```

```python
import pickle

class Triple():
  def __init__(self, a, b, c):
    self.a, self.b, self.c = a, b, c
  def __repr__(self):
    return str((self.a, self.b, self.c))

d = [Triple(1, 2, 3), Triple(4, 5, 6), Triple(7, 8, 9)]

pickle_data = pickle.dumps(d)
d_ = pickle.loads(pickle_data)
print(d, d_)

pickle.dump(d, open("d.pickle", "wb"))
d__ = pickle.load(open("d.pickle", "rb"))
print(d__)
```

Pickle

- Supports all types (implementation of `__repr__` must be available)
- Do not load any Pickles from unknown sources!
- Data is stored in binary format

Pandas – Installation

- Pandas is a Python package
- https://pandas.pydata.org/
- Installation, e.g., via `pip3 install pandas`
- Import via

```
import pandas
```
or
```
import pandas as pd
```

Pandas DataFrame

- Management and manipulation of tables
- Various data types combined
- Fast runtimes due to C code in the background
- Import from CSV, JSON, SQL, Excel, . . .

ID	Name	Age	Species
1	Alice	60	Human
2	Bob	65	Human
3	Charlie	10	Dog
	3	45	2

Pandas DataFrame

```python
import pandas as pd

df = pd.DataFrame({
  "ID" : pd.Index([1, 2, 3]),
  "Name" : ["Alice", "Bob", "Charlie"],
  "Age" : [60, 65, 10],
  "Species" : pd.Categorical(["Human", "Human", "Dog"])
})

print(df)
print(df.head(1))
print(df.tail(2))

print(df.dtypes)
print(df.index)
print(df.columns)

print(df.describe())
print(df.to_numpy())
```

ID	Name	Age	Species
1	Alice	60	Human
2	Bob	65	Human
3	Charlie	10	Dog

Pandas DataFrame

```python
import pandas as pd

df = pd.DataFrame({
  "ID" : pd.Index([1, 2, 3]),
  "Name" : ["Alice", "Bob", "Charlie"],
  "Age" : [60, 65, 10],
  "Species" : pd.Categorical(["Human", "Human", "Dog"])
})

print(df)
print(df.head(1))
print(df.tail(2))

print(df.dtypes)
print(df.index)
print(df.columns)

print(df.describe())
print(df.to_numpy())
```

ID	Name	Age	Species
1	Alice	60	Human
2	Bob	65	Human
3	Charlie	10	Dog

```
$> python3 script.py

    ID      Name   Age  Species
0    1     Alice    60    Human
1    2       Bob    65    Human
2    3   Charlie    10      Dog
    ID      Name   Age  Species
0    1     Alice    60    Human
    ID      Name   Age  Species
1    2       Bob    65    Human
2    3   Charlie    10      Dog
⋮
```

Pandas DataFrame

```python
import pandas as pd

df = pd.DataFrame({
  "ID" : pd.Index([1, 2, 3]),
  "Name" : ["Alice", "Bob", "Charlie"],
  "Age" : [60, 65, 10],
  "Species" : pd.Categorical(["Human", "Human", "Dog"])
})

print(df)
print(df.head(1))
print(df.tail(2))

print(df.dtypes)
print(df.index)
print(df.columns)

print(df.describe())
print(df.to_numpy())
```

ID	Name	Age	Species
1	Alice	60	Human
2	Bob	65	Human
3	Charlie	10	Dog

```
$> python3 script.py
:
ID          int64
Name        object
Age         int64
Species     category
dtype:  object
RangeIndex(start=0, stop=3, step=1)
Index(['ID', 'Name', 'Age',
'Species'], dtype='object')
:
```

Pandas DataFrame

```python
import pandas as pd

df = pd.DataFrame({
  "ID" : pd.Index([1, 2, 3]),
  "Name" : ["Alice", "Bob", "Charlie"],
  "Age" : [60, 65, 10],
  "Species" : pd.Categorical(["Human", "Human", "Dog"])
})

print(df)
print(df.head(1))
print(df.tail(2))

print(df.dtypes)
print(df.index)
print(df.columns)

print(df.describe())
print(df.to_numpy())
```

ID	Name	Age	Species
1	Alice	60	Human
2	Bob	65	Human
3	Charlie	10	Dog

```
$> python3 script.py
:
:
            ID        Age
 count   3.0    3.000000
  mean   2.0   45.000000
   std   1.0   30.413813
   min   1.0   10.000000
   25%   1.5   35.000000
   50%   2.0   60.000000
   75%   2.5   62.500000
   max   3.0   65.000000
[[1 'Alice' 60 'Human']
[2 'Bob' 65 'Human']
[3 'Charlie' 10 'Dog']]
```

Continuation

- Import and export of DataFrames from/to various file formats
- Selection (slices, indexes)
- Operation (statistics, applying functions)
- Joining and modifying DataFrames
- Visualisation

Continuation

- Import and export of DataFrames from/to various file formats
- Selection (slices, indexes)
- Operation (statistics, applying functions)
- Joining and modifying DataFrames
- Visualisation

More details:

- https://pandas.pydata.org/docs/user_guide/10min.html
- https://pandas.pydata.org/docs/user_guide/index.html

Natural Language Processing (NLP)

> *»Enter the link to your repository in Moodle. You will also be asked whether the repository is public or hidden! For hidden repositories, remember to grant the GitHub user* `WerkzeugeWissArbeiten` *access rights to your created Git-repository.«*

- Split sentences
- Create word vector(s)
  - Frequency of words per sentence
  - Words with little meaning

Natural Language Processing (NLP)

> *»Enter the link to your repository in Moodle. You will also be asked whether the repository is public or hidden! For hidden repositories, remember to grant the GitHub user `WerkzeugeWissArbeiten` access rights to your created Git-repository.«*

- Split sentences
- Create word vector(s)
  - Frequency of words per sentence
  - Words with little meaning

Natural Language Processing (NLP)

»Enter the link to your repository in Moodle. You will also be asked whether the repository is public or hidden! For hidden repositories, remember to grant the GitHub user `WerkzeugeWissArbeiten` *access rights to your created Git-repository.«*

- Split sentences
- Create word vector(s)
  - Frequency of words per sentence
  - Words with little meaning

Natural Language Processing (NLP)

»Enter the link to your repository in Moodle. You will also be asked whether the repository is public or hidden! For hidden repositories, remember to grant the GitHub user `WerkzeugeWissArbeiten` *access rights to your created Git-repository.«*

- Split sentences
- Create word vector(s)
  - Frequency of words per sentence
  - Words with little meaning

Natural Language Processing (NLP)

»Enter the link to your repository in Moodle. You will also be asked whether the repository is public or hidden! For hidden repositories, remember to grant the GitHub user `WerkzeugeWissArbeiten` *access rights to your created Git-repository.«*

- Preprocessing is an essential task in NLP
- We consider only a few ideas here

Preprocessing with NLTK I

```
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem.snowball import SnowballStemmer

text = "Enter the link to your repository in Moodle. You will ..."

stop_words = set(stopwords.words("english"))
print(stop_words)

stemmer_en = SnowballStemmer("english")

print(stemmer_en.stem("public"))
print(stemmer_en.stem("publicity"))

print(stemmer_en.stem("repository"))
print(stemmer_en.stem("repositories"))
```

Preprocessing with NLTK I

```python
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem.snowball import SnowballStemmer

text = "Enter the link to your repository in Moodle. You will ..."

stop_words = set(stopwords.words("english"))
print(stop_words)

stemmer_en = SnowballStemmer("english")

print(stemmer_en.stem("public"))
print(stemmer_en.stem("publicity"))

print(stemmer_en.stem("repository"))
print(stemmer_en.stem("repositories"))
```

```
$> python3 script.py

{'both', 'up', "shouldn't", 'ours',
'few', 'herself', 'needn', "they'll",
'to', 'whom', 'ain', ...}
public
public
repositori
repositori
```

Preprocessing with NLTK II

```
text = text.replace("-", " ")
sentences = []
for sentence in sent_tokenize(text, language='english'):
  print(sentence)
  words = []
  for word in word_tokenize(sentence, language='english'):
    print(word)
```

Preprocessing with NLTK II

```
text = text.replace("-", " ")
sentences = []
for sentence in sent_tokenize(text, language='english'):
  print(sentence)
  words = []
  for word in word_tokenize(sentence, language='english'):
    print(word)
```

```
$> python3 script.py

Enter the link to your
repository in Moodle.
Enter
the
link
to
your
repository
in
Moodle
.
```

Preprocessing with NLTK III

```
text = text.replace("-", " ")
sentences = []
for sentence in sent_tokenize(text, language='english'):
  print(sentence)
  words = []
  for word in word_tokenize(sentence, language='english'):
    print(word)
    word = word.lower()
    word = re.sub("[^a-z]", "", word)

    if word not in stop_words:
      word = stemmer_en.stem(word)
      print(word)
```

Preprocessing with NLTK III

```
text = text.replace("-", " ")
sentences = []
for sentence in sent_tokenize(text, language='english'):
  print(sentence)
  words = []
  for word in word_tokenize(sentence, language='english'):
    print(word)
    word = word.lower()
    word = re.sub("[^a-z]", "", word)

    if word not in stop_words:
      word = stemmer_en.stem(word)
      print(word)
```

```
$> python3 script.py

Enter the link to your
repository in Moodle.
Enter
enter
the
link
link
to
your
repository
repositori
in
Moodle
moodl
.
```

Preprocessing with NLTK IV

```
text = text.replace("-", " ")
sentences = []
for sentence in sent_tokenize(text, language='english'):
  words = []
  for word in word_tokenize(sentence, language='english'):
    word = word.lower()
    word = re.sub("[^a-z]", "", word)

    if word not in stop_words:
      word = stemmer_en.stem(word)
      if len(word) > 0:
        words.append(word)

  sentences.append(words)

print(sentences)
```

Preprocessing with NLTK IV

```
text = text.replace("-", " ")
sentences = []
for sentence in sent_tokenize(text, language='english'):
  words = []
  for word in word_tokenize(sentence, language='english'):
    word = word.lower()
    word = re.sub("[^a-z]", "", word)

    if word not in stop_words:
      word = stemmer_en.stem(word)
      if len(word) > 0:
        words.append(word)

  sentences.append(words)

print(sentences)
```

```
$> python3 script.py

[['enter', 'link',
'repositori', 'moodl'],
['also', 'ask',
'whether', 'repositori',
'public', 'hidden'],
['hidden', 'repositori',
'rememb', 'grant',
'github', 'user',
'werkzeugewissarbeiten',
'access', 'right',
'creat', 'git',
'repositori']]
```

Result

- *Bag-of-Words* model
- A document is a bit vector (or frequency vector) of its words
- Texts are now vectors (which we can compare, etc.)

Word	Sentence 0	Sentence 1	Sentence 2
access	0	0	1
also	0	1	0
ask	0	1	0
creat	0	0	1
enter	1	0	0
git	0	0	1
github	0	0	1
grant	0	0	1
hidden	0	1	1
link	1	0	0
moodl	1	0	0
public	0	1	0
rememb	0	0	1
repositori	1	1	2
right	0	0	1
user	0	0	1
werkzeugewissarbeiten	0	0	1
whether	0	1	0

Result as a NumPy Array

```
import numpy as np

sentences = [['enter', 'link', 'repositori', 'moodl'], ['also', 'ask',
     'whether', 'repositori', 'public', 'hidden'], ['hidden', '
    repositori', 'rememb', 'grant', 'github', 'user', '
    werkzeugewissarbeiten', 'access', 'right', 'creat', 'git', '
    repositori']]

id2word = sorted(list(set([ w for s in sentences for w in s ])))
word2id = { word : i for i, word in enumerate(id2word) }

corpus = np.zeros((len(id2word), len(sentences)), dtype=int)
for s_id, sentence in enumerate(sentences):
  for word in sentence:
    corpus[word2id[word], s_id] += 1

print(word2id)
print(corpus)
```

Result as a NumPy Array

```
1  import numpy as np
2
3  sentences = [['enter', 'link', 'repositori', 'moodl'], ['also', 'ask',
       'whether', 'repositori', 'public', 'hidden'], ['hidden', '
     repositori', 'rememb', 'grant', 'github', 'user', '
     werkzeugewissarbeiten', 'access', 'right', 'creat', 'git', '
     repositori']]
4
5  id2word = sorted(list(set([ w for s in sentences for w in s ])))
6  word2id = { word : i for i, word in enumerate(id2word) }
7
8  corpus = np.zeros((len(id2word), len(sentences)), dtype=int)
9  for s_id, sentence in enumerate(sentences):
10   for word in sentence:
11     corpus[word2id[word], s_id] += 1
12
13 print(word2id)
14 print(corpus)
```

```
$> python3 script.py

{'access': 0, 'also': 1, 'ask': 2, 'creat': 3,
'enter': 4, 'git': 5, 'github': 6, 'grant': 7,
'hidden': 8, 'link': 9, 'moodl': 10, 'public': 11,
'rememb': 12, 'repositori': 13, 'right': 14, 'user':
15, 'werkzeugewissarbeiten': 16, 'whether': 17} ...
```

Result as a NumPy Array

```python
import numpy as np

sentences = [['enter', 'link', 'repositori', 'moodl'], ['also', 'ask',
     'whether', 'repositori', 'public', 'hidden'], ['hidden', '
    repositori', 'rememb', 'grant', 'github', 'user', '
    werkzeugewissarbeiten', 'access', 'right', 'creat', 'git', '
    repositori']]

id2word = sorted(list(set([ w for s in sentences for w in s ])))
word2id = { word : i for i, word in enumerate(id2word) }

corpus = np.zeros((len(id2word), len(sentences)), dtype=int)
for s_id, sentence in enumerate(sentences):
  for word in sentence:
    corpus[word2id[word], s_id] += 1

print(word2id)
print(corpus)
```

```
...
[[0 0 1]
[0 1 0]
[0 1 0]
[0 0 1]
[1 0 0]
[0 0 1]
[0 0 1]
[0 0 1]
[0 1 1]
[1 0 0]
[1 0 0]
[0 1 0]
[0 0 1]
[1 1 2]
[0 0 1]
[0 0 1]
[0 0 1]
[0 1 0]]
```

Result as a NumPy Array

- Dictionary that maps words to indices
- NumPy array with word frequencies per sentence
- In general, you would use a library for this

Matplotlib – Installation

matplotlib

- Matplotlib is a Python package
- https://matplotlib.org
- Installation, e.g., via `pip3 install matplotlib`
- Import via

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

Matplotlib – The First Plot

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x = np.arange(-10, 10.5, step=0.5)
5  print(x)
6
7  plt.plot(x, x**2)
8  plt.plot(x, x**3)
9
10 plt.show()
```

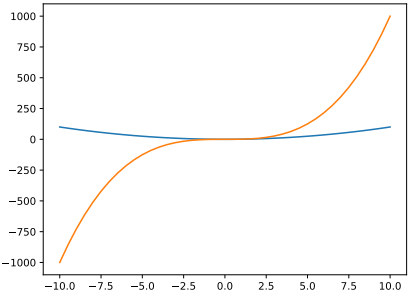
Matplotlib – The First Plot

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

x = np.arange(-10, 10.5, step=0.5)
print(x)

plt.plot(x, x**2)
plt.plot(x, x**3)

plt.show()
```

```
$> python3 script.py

[-10.  -9.5  -9.  -8.5  -8.  -7.5  -7.  -6.5  -6.  -5.5  -5.  -4.5
  -4.  -3.5  -3.  -2.5  -2.  -1.5  -1.  -0.5   0.   0.5   1.   1.5
   2.   2.5   3.   3.5   4.   4.5   5.   5.5   6.   6.5   7.   7.5
   8.   8.5   9.   9.5  10. ]
```

Matplotlib – The First Plot

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

x = np.arange(-10, 10.5, step=0.5)
print(x)

plt.plot(x, x**2)
plt.plot(x, x**3)

plt.show()
```



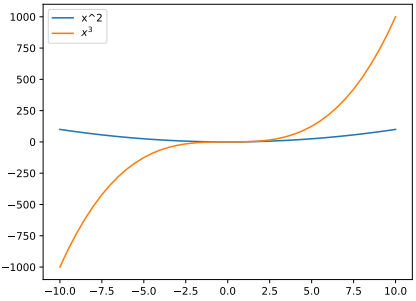

```
$> python3 script.py

[-10.  -9.5  -9.  -8.5  -8.  -7.5  -7.  -6.5  -6.  -5.5  -5.  -4.5
  -4.  -3.5  -3.  -2.5  -2.  -1.5  -1.  -0.5   0.   0.5   1.   1.5
   2.   2.5   3.   3.5   4.   4.5   5.   5.5   6.   6.5   7.   7.5
   8.   8.5   9.   9.5  10. ]
```

PyPlot and Object-Oriented

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x = np.arange(-10, 10.5, step=0.5)
5
6  plt.plot(x, x**2, label='x^2')
7  plt.plot(x, x**3, label=r'$x^3$')
8  plt.legend()
9
10 plt.show()
```

PyPlot and Object-Oriented

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x = np.arange(-10, 10.5, step=0.5)
5
6  plt.plot(x, x**2, label='x^2')
7  plt.plot(x, x**3, label=r'$x^3$')
8  plt.legend()
9
10 plt.show()
```



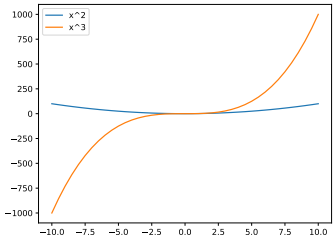
PyPlot and Object-Oriented

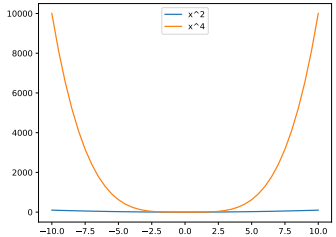
```python
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(-10, 10.5, step=0.5)

plt.plot(x, x**2, label='x^2')
plt.plot(x, x**3, label=r'$x^3$')
plt.legend()

plt.show()
```

```python
fig1, ax1 = plt.subplots()

ax1.plot(x, x**2, label='x^2')
ax1.plot(x, x**3, label='x^3')
ax1.legend()

fig2, ax2 = plt.subplots()

ax2.plot(x, x**2, label='x^2')
ax2.plot(x, x**4, label='x^4')
ax2.legend()

plt.show()
```

PyPlot and Object-Oriented

```
12 fig1, ax1 = plt.subplots()
13
14 ax1.plot(x, x**2, label='x^2')
15 ax1.plot(x, x**3, label='x^3')
16 ax1.legend()
17
18 fig2, ax2 = plt.subplots()
19
20 ax2.plot(x, x**2, label='x^2')
21 ax2.plot(x, x**4, label='x^4')
22 ax2.legend()
23
24 plt.show()
```





PyPlot and Object-Oriented

```
12 fig1, ax1 = plt.subplots()
13
14 ax1.plot(x, x**2, label='x^2')
15 ax1.plot(x, x**3, label='x^3')
16 ax1.legend()
17
18 fig2, ax2 = plt.subplots()
19
20 ax2.plot(x, x**2, label='x^2')
21 ax2.plot(x, x**4, label='x^4')
22 ax2.legend()
23
24 plt.show()
```

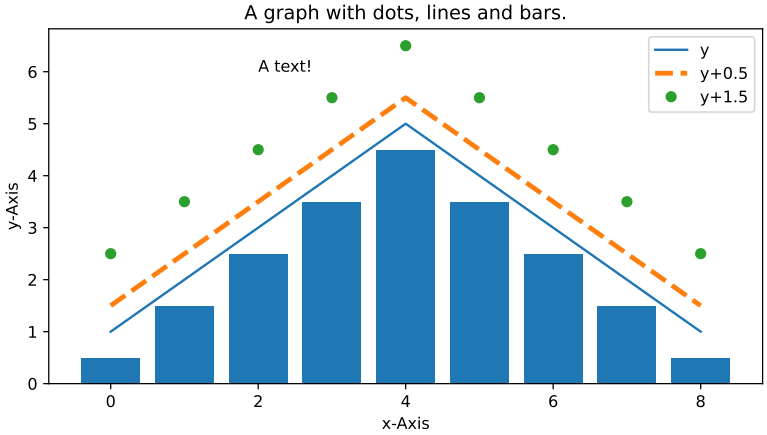
Objector: Creates object and then uses it

Two objects, i.e. a window opens with two plots.

Another Example

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

y = np.array([1, 2, 3, 4, 5, 4, 3, 2, 1])
x = np.arange(9)

fig, ax = plt.subplots(figsize=(8, 4))
ax.plot(x, y,           label='y')
ax.plot(x, y+0.5,       label='y+0.5', linewidth=3, linestyle='--')
ax.plot(x, y+1.5, 'o', label='y+1.5')
ax.bar( x, y-0.5);
ax.legend();

ax.set_xlabel("x-Axis")
ax.set_ylabel("y-Axis")
ax.set_title('A graph with dots, lines and bars.')
ax.text(2, 6, 'A text!')

plt.savefig("plot.png")
```

Another Example – Output



Summary

- Data processing
  - Python internal
  - Pandas
  - Natural Language Toolkit (NLTK)
- Data visualisation
  - Matplotlib