Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **00**

Previously, on Python 3 Beginner…

release 1.0.0

37/39 Avenue Ledru Rollin – CS 11237 – 75750 PARIS CEDEX 12 - info@valnaos.com – www.valnaos.com Tél : 01 86 95 20 48

Q&A

Where? Who?

Class Material

➠ GitHub: https://github.com/cstar-industries/python-3-beginner

Instructor

➠ Charles Francoise <charles@cstar.io>

Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **01**

HTTP with Requests

release 1.0.0

Requests

- Requests is a library that makes HTTP requests very simple, while still providing powerful in-depth options
- "HTTP for Humans"
- Perfect for downloading/uploading data, or communicating with a web service API

Requests

➠ HTTP in a nutshell

```
GET / HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: www.google.com
User-Agent: HTTPie/1.0.2
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Encoding: gzip
Content-Type: text/html; charset=ISO-8859-1
Date: Tue, 12 May 2020 08:34:48 GMT
Expires: -1
Server: gws
Set-Cookie: 1P_JAR=2020-05-12-08; expires=Thu, 11-Jun-2020 08:34:48 GMT; path=/;
domain=.google.com; Secure
Transfer-Encoding: chunked

<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="fr"><head>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta
content="/images/branding/googleg/1x/googleg_standard_color_128dp.png"
itemprop="image"><title>Google</title>...
```

Requests

- `requests.get` sends a GET request
- Returns a `Response` object with info about the data

```
res = requests.get('https://google.com')
print(res.ok)
print(res.status_code)
print(res.reason)
print(res.elapsed)
```

```
True
200
OK
0:00:00.085902
```

Requests

➠ The response content is available in several forms

```
type(res.text)
type(res.content)
```

```
str
bytes
```

➠ Parse a JSON response directly to a `dict`

```
type(res.json())
```

```
dict
```

Requests

➠ Read the response headers in `dict` form

```
print(res.headers)
```

```
{'Date': 'Tue, 12 May 2020 10:00:57 GMT',
 'Expires': '-1',
 'Cache-Control': 'private, max-age=0',
 'Content-Type': 'text/html; charset=ISO-8859-1
...
```

➠ Parse a JSON response directly to a `dict`

```
type(res.json())
```

```
dict
```

Requests

➠ Raise a Python error on HTTP errors

```
res = requests.get('https://google.com/toto')
print(res.ok)
print(f'{res.status_code} {res.reason}')
res.raise_for_status()
```

```
False
404 Not Found
HTTPError: 404 Client Error: Not Found for url: https://google.com/toto
```

➠ Other HTTP methods are available

```
res = requests.post('https://httpbin.org/post', data=...
res = requests.put('https://httpbin.org/post', data=...
res = requests.delete(...
res = requests.head(...
res = requests.request('GET', 'https://...
```

Requests

➠ Send query variables with `params`

```
res = requests.get('https://www.google.com/search',
                   params={'q':'Hello World'})
print(res.url)
```

```
https://www.google.com/search?q=Hello+World
```

➠ Send data in request body with `data`

```
res = requests.post('https://httpbin.org/post',
                    data=b'Hello World!')
res = requests.post('https://httpbin.org/post',
                    data={'hello': 'world', 'search': 'python'})
print(res.request.body)
```

```
hello=world&search=python
```

Requests

➠ Encode request data as JSON

```
res = requests.post('https://httpbin.org/post',
                    json={'hello': 'world', 'python': None})
print(res.request.body)
```

```
b'{"hello": "world", "python": null}'
```

➠ Send file attachments

```
with open('profile_pic.png', 'rb') as f:
  res = requests.post('https://httpbin.org/post',
                      files={'upload': f})
len(res.request.body)
```

```
669525
```

Requests

➠ Customize request headers

```
res = requests.get('https://httpbin.org/post',
                   headers={'Authorization': 'alibaba:opensesame'})
print(res.request.body)
```

```
b'{"hello": "world", "python": null}'
```

➠ Set request timeout

```
res = requests.get('https://www.google.com', timeout=0.001)
```

```
ConnectTimeout: 'Connection to www.google.com timed out. (connect
timeout=0.001)'))
```

Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **02**

Web Applications with Flask

release 1.0.0

Flask

- Flask is a micro-framework for building web applications
- Use Flask to build web servers simply and efficiently without worrying about low-level details
- Flask is perfect to build web applications with a JSON-based REST API
- Use Flask wherever you would use PHP, Node.js or other non-Python backend technology

Requests

```
from flask import Flask
```

➠ Write a simple Flask server

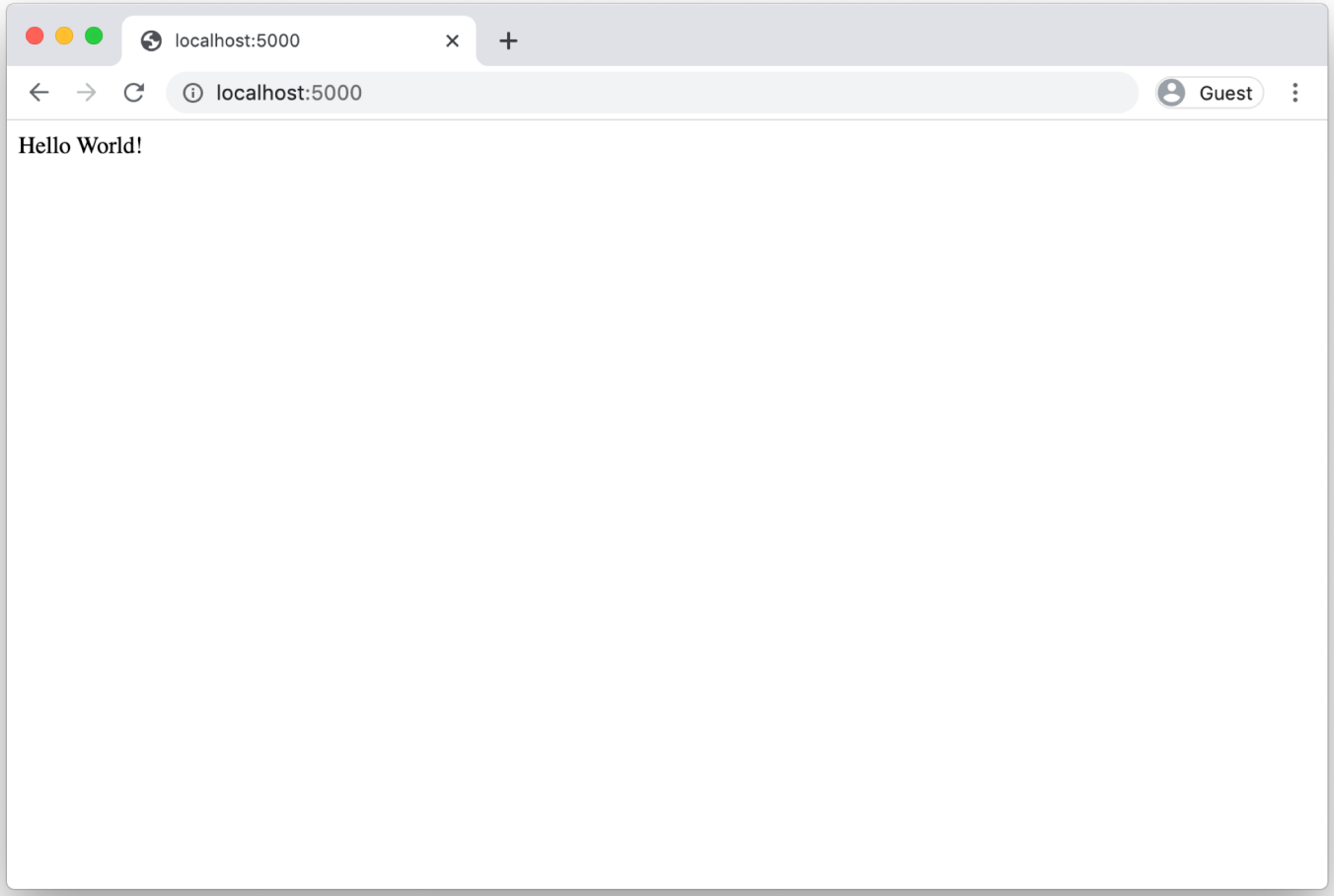
```
app = Flask(__name__)

@app.route('/')
def index():
    return 'Hello World!'
```

➠ Run the server with the Flask CLI

```
FLASK_APP=main.py flask run
```

```
 * Serving Flask app "main.py"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Flask




```
127.0.0.1 - - [12/May/2020 17:35:32] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/May/2020 17:35:32] "GET /favicon.ico HTTP/1.1" 404 -
```

Requests

- Run Flask in debug mode:
  - Extra info in response on errors
  - Automatic reload whan a file changes

```
FLASK_ENV=development FLASK_APP=main.py flask run
```

- By default, only `GET` is supported by your route definitions
- Use `methods` argument in the decorator

```
@app.route('/', methods=['GET', 'POST'])
def index():
    return 'Hello World!'
```

Requests

- Run Flask in debug mode:
  - Extra info in response on errors
  - Automatic reload whan a file changes

```
FLASK_ENV=development FLASK_APP=main.py flask run
```

- By default, only `GET` is supported by your route definitions
- Use `methods` argument in the decorator

```
@app.route('/', methods=['GET', 'POST'])
def index():
    return 'Hello World!'
```

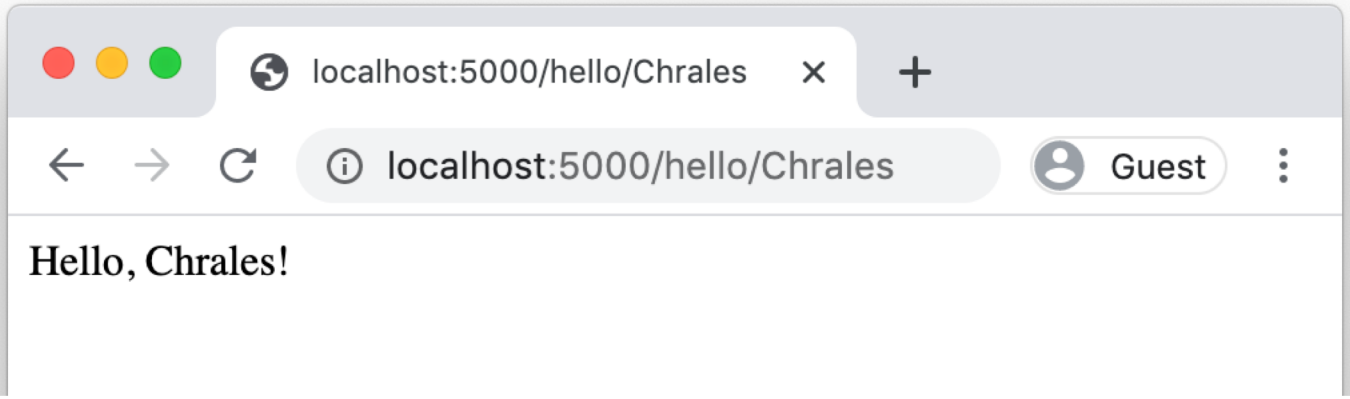
Requests: Defining routes

➠ Define more routes with the `route` decorator

```
@app.route('/hello')
def hello():
    return '<html><body>Hello</body></html>'
```

➠ Capture variables from the address

```
@app.route('/hello/<name>')
def hello(name):
    return f'<html><body>Hello, {name}!</body></html>'
```

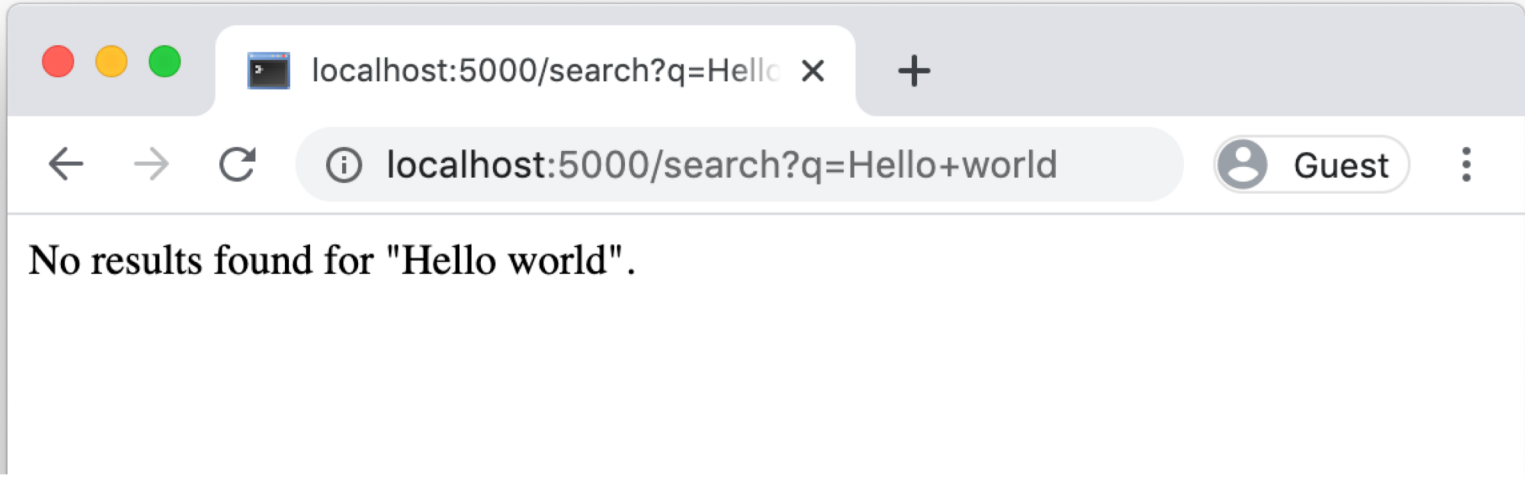


Requests: Retrieving data from request

```
from flask import Flask, request
```

➠ To access parameters submitted in the URL (`?key=value`), use the `args` attribute:

```
@app.route('/search')
def hello():
    search_params = request.args.get('q')
    return f'No results found for "{q}".'
```



Requests: Retrieving data from request

- The raw request body can be read inside the handler

```
req_body = request.data
```

- Request data can be parsed automatically

```
request.form     # Form-encoded request data, parsed to a dict
request.json     # JSON-encoded request data, parsed to a dict
request.files    # A dict of files from multipart form-encoded data
request.headers  # A dict of headers from the client request
request.cookies  # A dict of cookies from the client request
...
```

Requests: Customizing the response

➠ The raw request body can be read inside the handler

```
req_body = request.data
```

➠ Request data can be parsed automatically

```
request.form     # Form-encoded request data, parsed to a dict
request.json     # JSON-encoded request data, parsed to a dict
request.files    # A dict of files from multipart form-encoded data
request.headers  # A dict of headers from the client request
request.cookies  # A dict of cookies from the client request
...
```

Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **03**

Manipulating images with Pillow

release 1.0.0

Pillow

- Pillow is a powerful library for opening, manipulating and creating all types of image files
- Open-source fork of a historical Python library: PIL
- Opens all types of images including JPEG, PNG, TIFF, BMP, TGA, PDF...
- Allows geometrical transformations, pixel manipulation, channel operations, drawing, writing text...

Pillow: Opening an image file

```
from PIL import Image
```

➠ Open an existing file with `Image.open`

```
img = Image.open('flag.png')
img.show()
```

➠ Get information on the file

```
print(f'{img.format} {img.mode} {img.size}')
```

```
PNG RGB (128, 128)
```

Pillow: Geometrical transformations

➠ Rotating

```
img = img.rotate(45, fillcolor=(255, 255, 255))
```

➠ Scaling

```
img = img.resize((256, 128))
```

Pillow: More tranformations

➠ Blur

```
img = img.filter(ImageFilter.BLUR)
```

➠ Crop

```
img = img.crop((128, 0, 256, 128))
```

Pillow: Saving a file to disk

- Converting mode

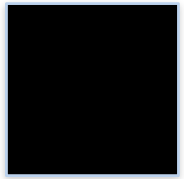
```
img = img.convert('L')
```

- Save an image with `save`
  - Image format is chosen from extension

```
img.save('broken-flag.jpg')
```

Pillow: Create a new image

```
img = Image.new('RGB', (128, 128))
```



- Access pixel data with `load`
- Pixels are `tuples` with 1, 3 or 4 values (depending on the mode)

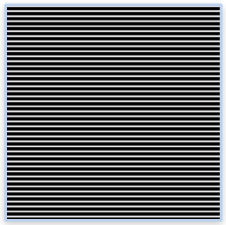
```
px = img.load()
print(px[64, 64])
```

```
(0, 0, 0)
```

Pillow: Pixel modification

➠ `PixelAccess` is mutable

```
for x in img.width:
  for y in img.height:
    if (y % 3) == 0:
      px[x, y] = (255, 255, 255)
```



Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **04**

Math with Numpy

release 1.0.0

Numpy

- Numpy is an advanced mathematics library
- Offers a powerful multi-dimensional array type, and advanced numeric types
- Basic math functionality, random number generation, linear algebra, statistics, Fourier transform...
- Powerful broadcasting algorithms for performing math over large arrays
- Advantageously replaces MATLAB in most situations

Numpy

```
import numpy as np
```

➠ Numpy offers basic math constants and functions

```
print(np.pi)
print(np.e)
print(np.sqrt(2))
print(np.cos(np.pi))
print(np.log(2))
```

```
3.141592653589793
2.718281828459045
1.414213562373095
-1.0
0.6931471805599453
```

Numpy

➠ The `array` class is the heart of Numpy

```
a = np.array([1, 2, 3, 4, 5])
print(a)
```

```
[1 2 3 4 5]
```

```
print(a.dtype)          # datatype
print(a.size)           # number of elements
print(a.ndim)           # number of dimensions
print(a.shape)          # length of the array in each dimension
```

```
int64
5
1
(5,)
```

Numpy

➠ The array class is the heart of Numpy

```
a = np.array([1, 2, 3, 4, 5])
print(a)
```

```
[1 2 3 4 5]
```

```
print(a.dtype)          # datatype
print(a.size)           # number of elements
print(a.ndim)           # number of dimensions
print(a.shape)          # length of the array in each dimension
```

```
int64
5
1
(5,)
```

Numpy

- arrays can be 2-dimensional (or more!)
- many available datatypes

```
a = np.array([[1, 2, 3], [4, 5, 6]], dtype=np.float)
```

```
[[1. 2. 3.]
 [4. 5. 6.]]
```

```
print(a.dtype)          # datatype
print(a.size)           # number of elements
print(a.ndim)           # number of dimensions
print(a.shape)          # length of the array in each dimension
```

```
float64
6
2
(2, 3)
```

Numpy

- Many array creation facilities exist

```
np.zeros((3, 3))     # 3x3 zero-filled array
np.ones((3, 3))      # 3x3 one-filled array
np.eye(3)            # 3x3 identity matrix (ones in the main diagonal)
np.random.random(size=(3, 3))
                     # 3x3 matrix of random 0 <= n < 1 values
```

- Change arrays without changing content

```
a = np.arange(4)                          # array([0, 1, 2, 3])
print(a.shape, a.size, a.dtype)
b = a.reshape((2, 2)).astype(np.float)
print(b)
```

```
(4,) 4 int64
[[0. 1.]
 [2. 3.]]
```

Numpy

➠ Perform array/matrix mathematics

```
a = np.array([[0., 1.], [2., 3.]])
b = np.array([[0., 2.], [-1. ,0.]])
```

➠ Add a scalar

```
print(a + 3.0)
```

```
[[3. 4.]
 [5. 6.]]
```

➠ Multiply by a scalar

```
print(a * 3.0)
```

```
[[0. 3.]
 [6. 9.]]
```

Numpy

➠ Add arrays together

```
print(a + b)
```

```
[[0. 3.]
 [1. 3.]]
```

➠ Element-wise multiplication

```
print(a * b)
```

```
[[ 0.  2.]
 [-2.  0.]]
```

➠ Matrix product

```
print(a.dot(b))             # == a @ b
```

```
[[-1.  0.]
 [-3.  4.]]
```

Numpy

➠ Broadcast functions over arrays

```
print(np.sqrt(a))
```

```
[[0.         1.        ]
 [1.41421356 1.73205081]]
```

➠ Various operations

```
print(a.min(), a.max(), a.mean(), a.std())
print(a.sum(), a.prod())
print(a.sum(axis=0))
print(a.flatten())
```

```
0.0 3.0 1.5 1.118033988749895
6.0 0.0
[2. 4.]
[0., 1., 2., 3.]
```

Numpy

➠ Get eigenvalues and eigenvectors

```
w, v = np.linalg.eig(np.array([[0., 1.], [1., 0.]]))
```

```
[ 1. -1.]
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
```

➠ Solve equation sets

```
x = np.array([[3., 2.], [1., 1.]])
y = np.array([3., 0.])
np.linalg.solve(x, y)
```

```
array([ 3., -3.])
```

Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **05**

Visualizing data with Matplotlib

release 1.0.0

Matplotlib

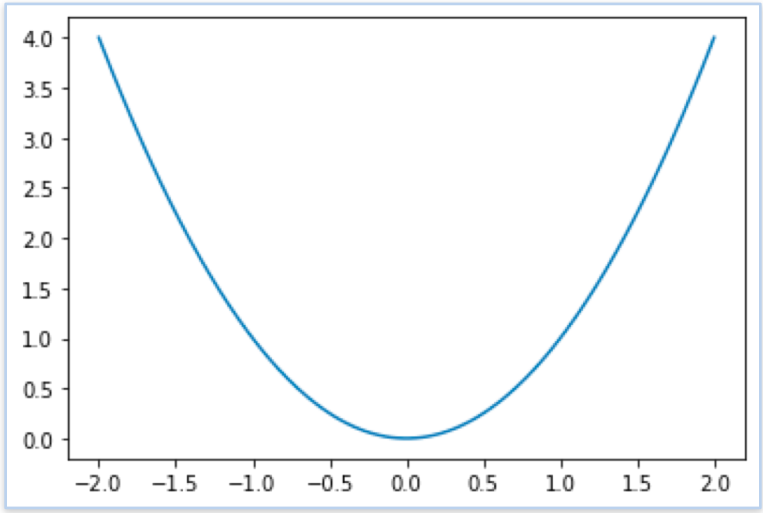
- Matplotlib is the reference graphing and data visualization library for Python data science
- It is most often used with the stateful `matplotlib.pyplot` module
- Similar to MATLAB
- Pyplot is based on creating *figures* and plotting data inside those figures

Matplotlib

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

➠ Plot the function y = x2 for x in [-2, 2]

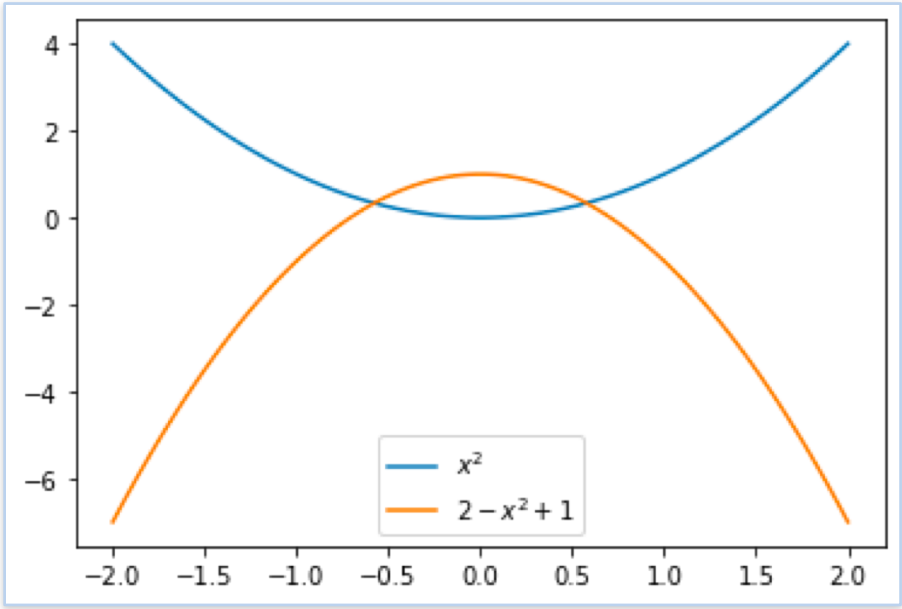
```
x = np.linspace(-2, 2, 100)
y = x * x
plt.plot(x, y)
plt.show()
```



Matplotlib

➠ Plot several series and add a legend

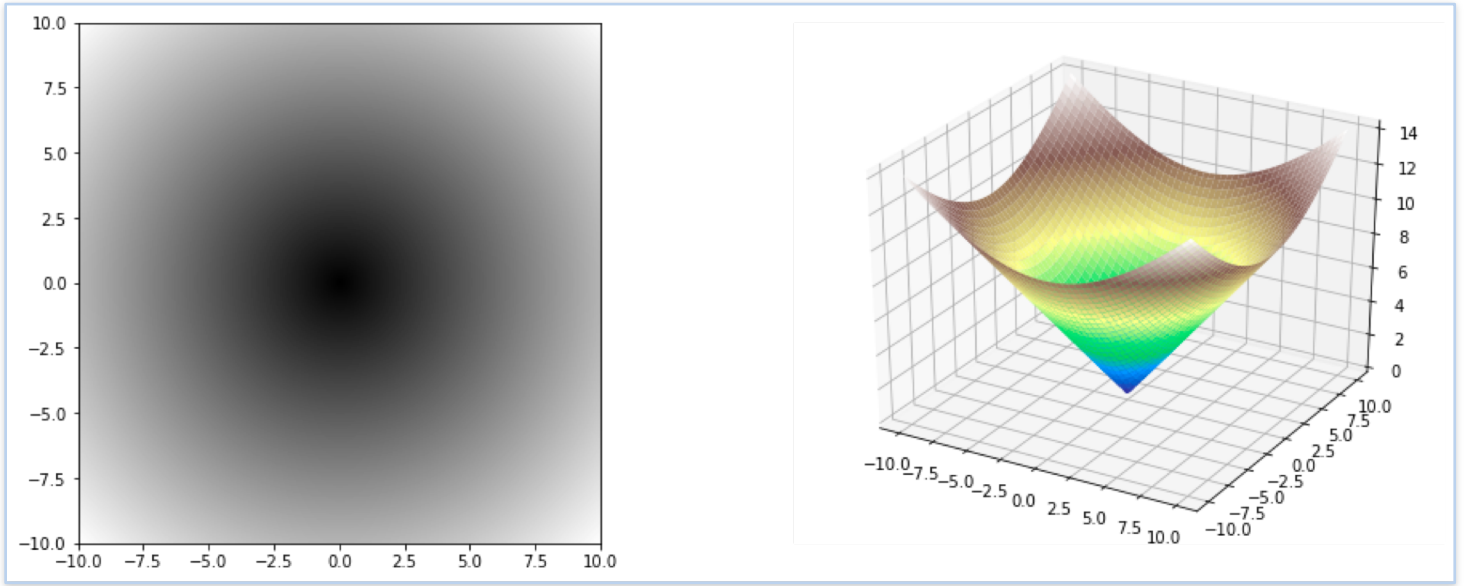
```
y0 = x * x
y1 = -2 * x * x + 1
plt.plot(x, y0)
plt.plot(x, y1)
plt.legend(['$x^2$', '$2-x^2+1$'])
plt.show()
```



Matplotlib

➠ Draw heatmaps and surfaces

```
plt.figure(figsize=(16, 6))
plt.subplot(1, 2, 1)
plt.imshow(z, cmap='gray', extent=[-10, 10, -10, 10])
ax = plt.subplot(1, 2, 2, projection='3d')
ax.plot_surface(xv, yv, z, cmap='terrain')
plt.show()
```



Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **06**

Handling data sets with Pandas

release 1.0.0

Pandas

- Pandas is a library to handle, prepare and process data sets
- Reads many data formats like Excel, CSV, HDF5, Feather, SQLite...
- Handle row-based or column-based data in the form of *series* and *dataframes*
- "Excel for Python"
- Often the first step to data exploration and analysis

Pandas

```
import pandas as pd
```

- Data comes in *series*
- Series can be imported from files or created from scratch
- Series are *indexed*

```
s = pd.Series([6., 5., 6., 3.])
print(s)
```

```
0    6.0
1    5.0
2    6.0
3    3.0
dtype: float64
```

Pandas

- A collection of series with a shared index is a *data frame*
- Similar to a database table or a spreadsheet

```
df = pd.DataFrame(np.random.random((6, 4)),
                  columns=['Col1', 'Col2', 'Col3', 'Col4'])
print(df)
```

```
       Col1      Col2      Col3      Col4
0  0.979237  0.982279  0.562715  0.773005
1  0.502712  0.953881  0.720175  0.529885
2  0.138129  0.186640  0.471914  0.458796
3  0.536048  0.258074  0.004619  0.305234
4  0.577828  0.004287  0.095598  0.217330
5  0.665738  0.605397  0.219344  0.766313
```

Pandas

- A column or a row of a data frame is a series
- Columns can be referenced by name, rows by index with `loc`

```
print(df['Col1'])
print(df.loc[0])
```

```
0    0.979237                    Col1    0.979237
1    0.502712                    Col2    0.982279
2    0.138129                    Col3    0.562715
3    0.536048                    Col4    0.773005
4    0.577828                    Name: 0, dtype: float64
5    0.665738
Name: Col1, dtype: float64
```

Pandas

➠ Get sub-dataframes with indexing

```
print(df[['Col2','Col3']])
print(df[1:-1])
```

```
       Col2      Col3
0  0.982279  0.562715
1  0.953881  0.720175
2  0.186640  0.471914
3  0.258074  0.004619
```

```
       Col1      Col2      Col3      Col4
1  0.502712  0.953881  0.720175  0.529885
2  0.138129  0.186640  0.471914  0.458796
3  0.536048  0.258074  0.004619  0.305234
4  0.577828  0.004287  0.095598  0.217330
```

➠ Add a column by label (like `dict`)

```
df['Names'] = ['Alice', 'Bob', 'Charlie', 'Daniel', 'Eve', 'Fiona']
```

```
       Col1      Col2      Col3      Col4    Names
0  0.979237  0.982279  0.562715  0.773005    Alice
1  0.502712  0.953881  0.720175  0.529885      Bob
2  0.138129  0.186640  0.471914  0.458796  Charlie
```

Pandas

➠ Read a data file into a dataframe

```
df = pd.read_csv('tesla-stock-price.csv')
df.head()
```

```
         date   close      volume    open    high       low
0  2018/10/15  259.59   6189026.0  259.06  263.28  254.5367
1  2018/10/12  258.78   7189257.0  261.00  261.99  252.0100
2  2018/10/11  252.23   8128184.0  257.53  262.25  249.0300
3  2018/10/10  256.88  12781560.0  264.61  265.51  247.7700
4  2018/10/09  262.80  12037780.0  255.25  266.77  253.3000
```

Pandas

➠ Edit index and columns

```
df = df.set_index(pd.DatetimeIndex(df['date'])) # Use date col as index
df = df.loc[:, df.columns != 'date']            # Drop date col
df.head()
```

date	close	volume	open	high	low
2018-10-15	259.59	6189026.0	259.06	263.28	254.5367
2018-10-12	258.78	7189257.0	261.00	261.99	252.0100
2018-10-11	252.23	8128184.0	257.53	262.25	249.0300
2018-10-10	256.88	12781560.0	264.61	265.51	247.7700
2018-10-09	262.80	12037780.0	255.25	266.77	253.3000

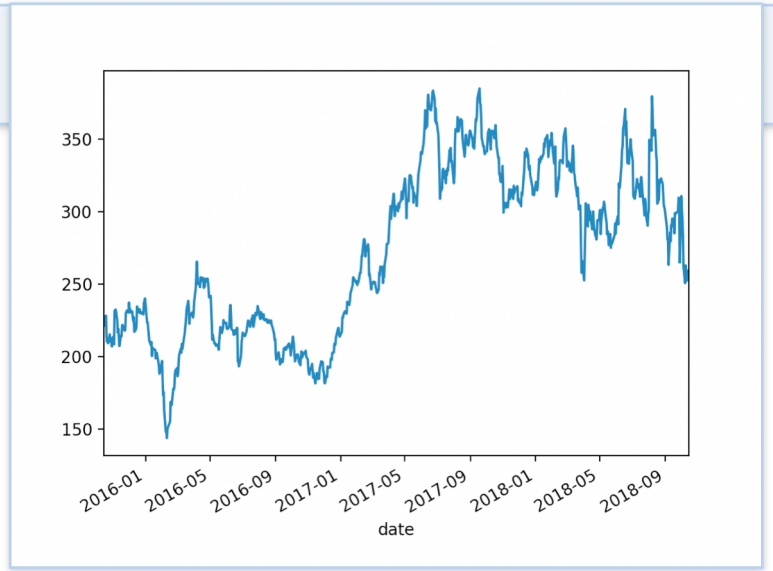
Pandas

➠ Interact with other data science libraries

```
a = df.to_numpy()
type(a)
print(a.shape)
```

```
numpy.ndarray
(756, 5)
```

```
df['close'].plot()
```



Q&A

Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Thank you for your attention!

