

DEDA Digital Economy & Decision Analytics

Cathy Yi-Hsuan Chen

Wolfgang Karl Härdle

Ladislaus von Bortkiewicz Professor of Statistics

C.A.S.E.-Center for Applied Statistics and
Economics

International Research Training Group

Humboldt-Universität zu Berlin

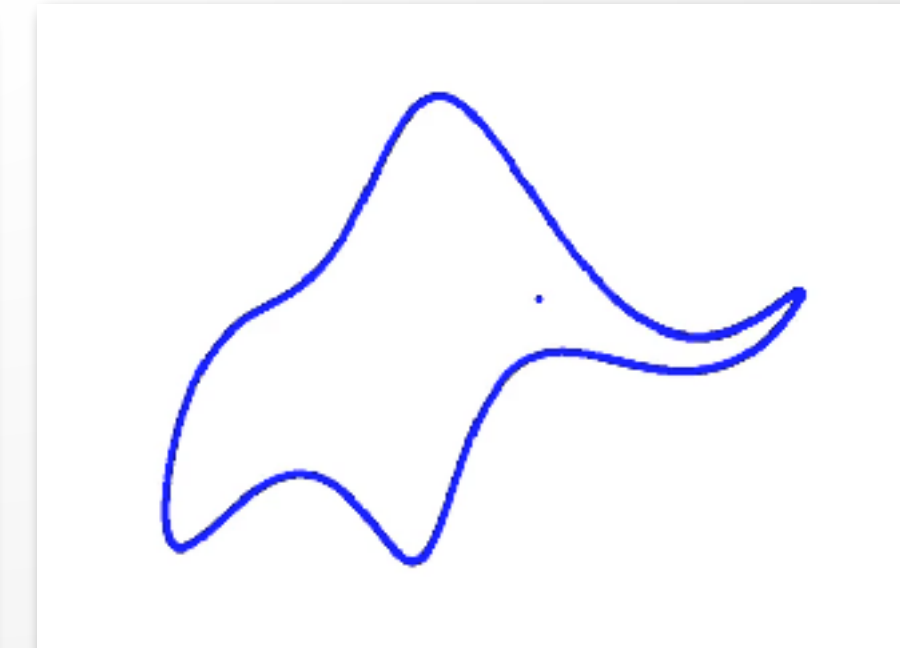
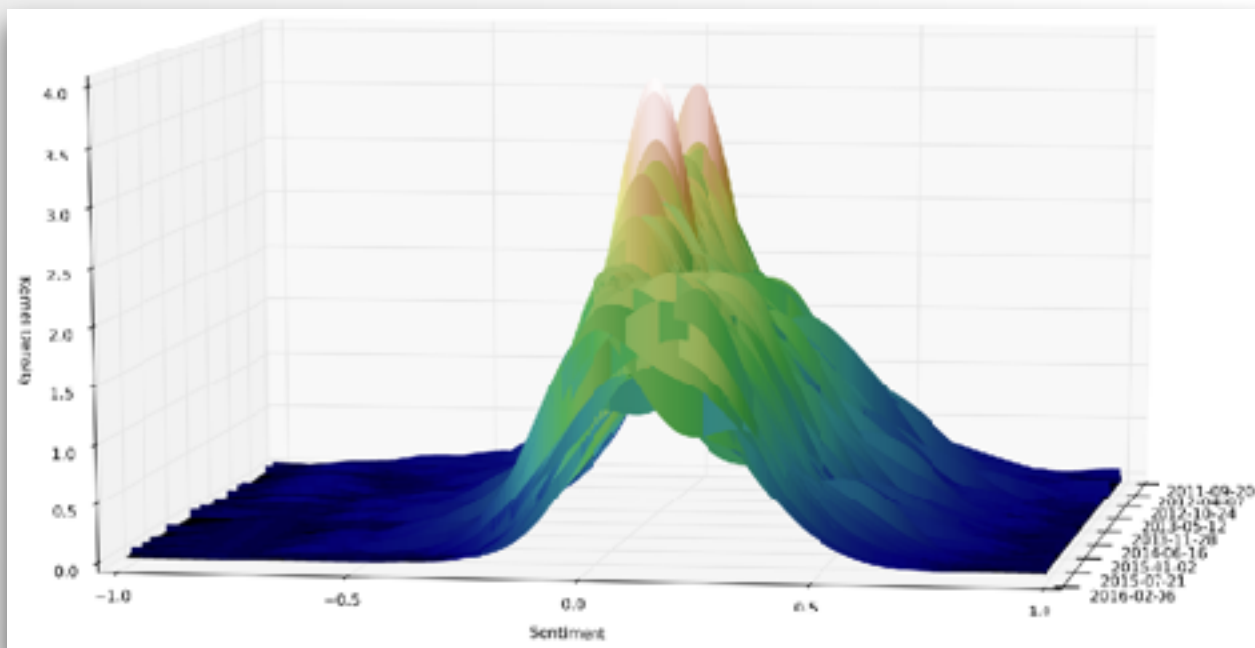
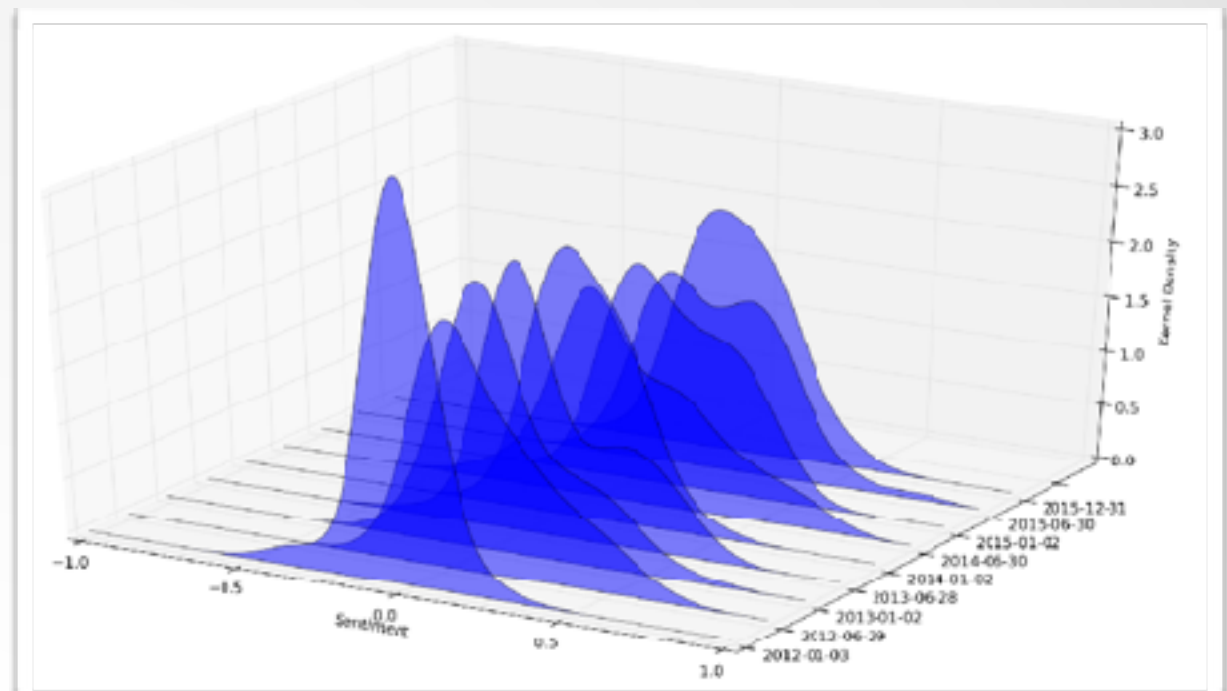
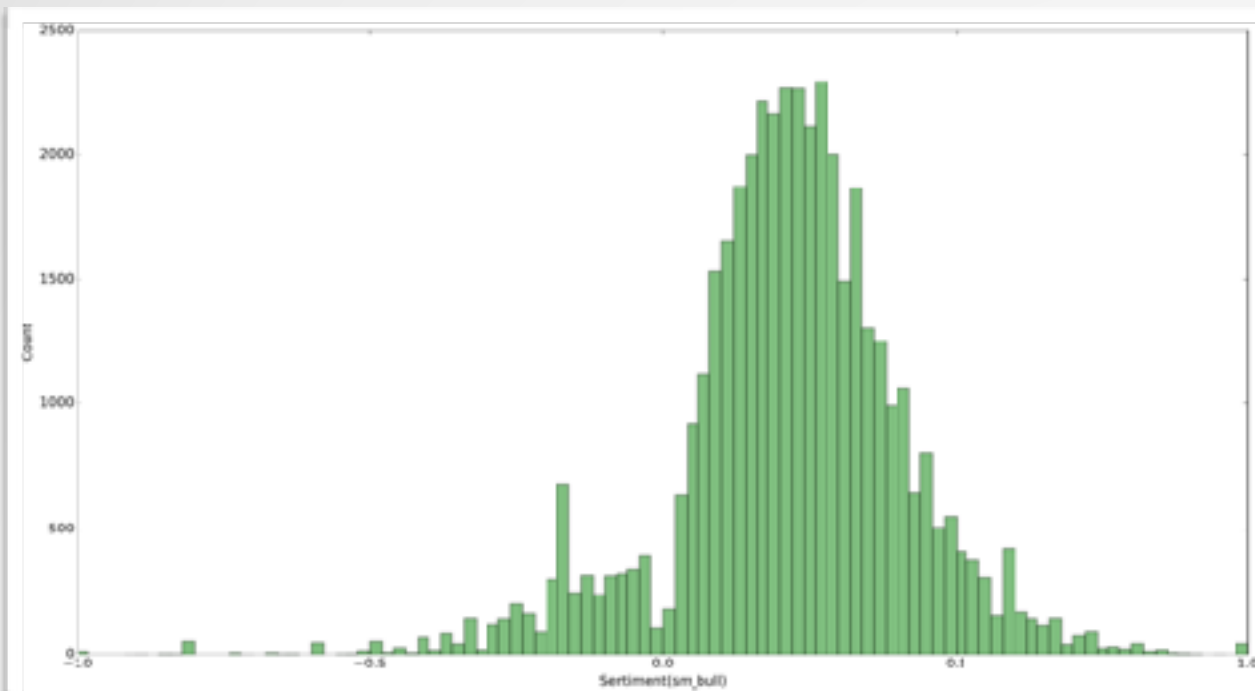
lvb.wiwi.hu-berlin.de

www.case.hu-berlin.de

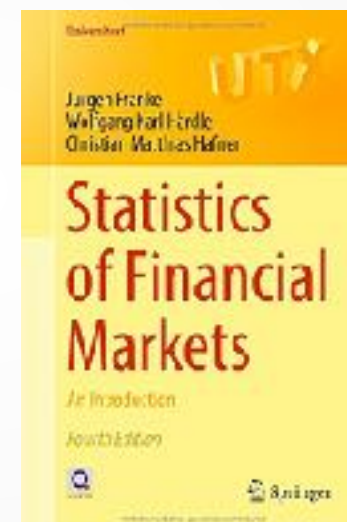
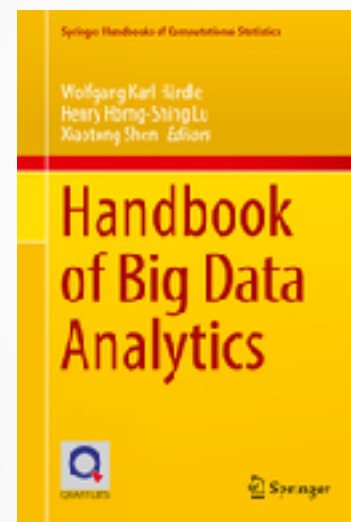
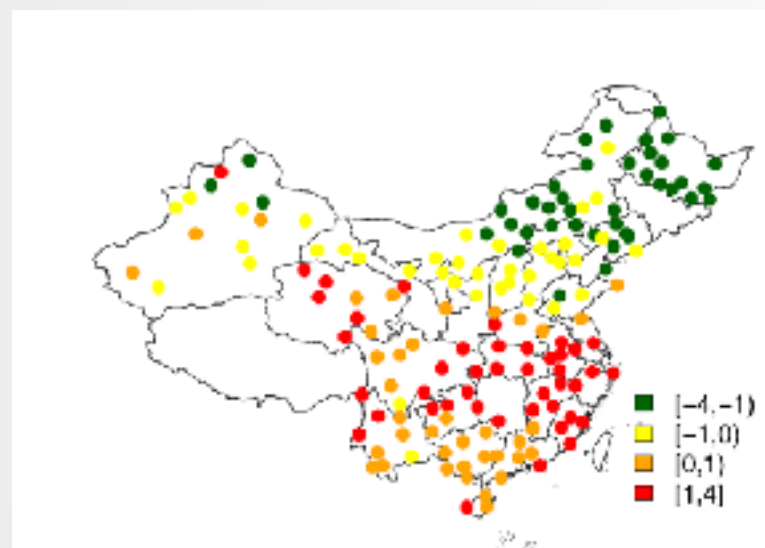
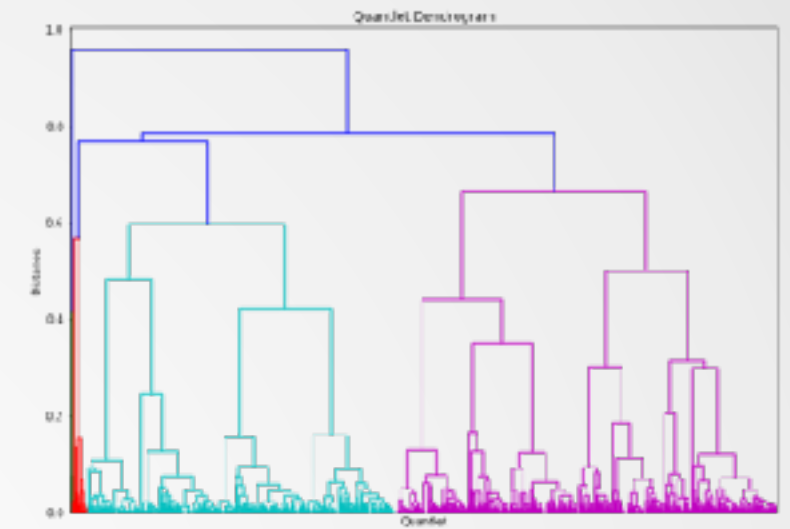
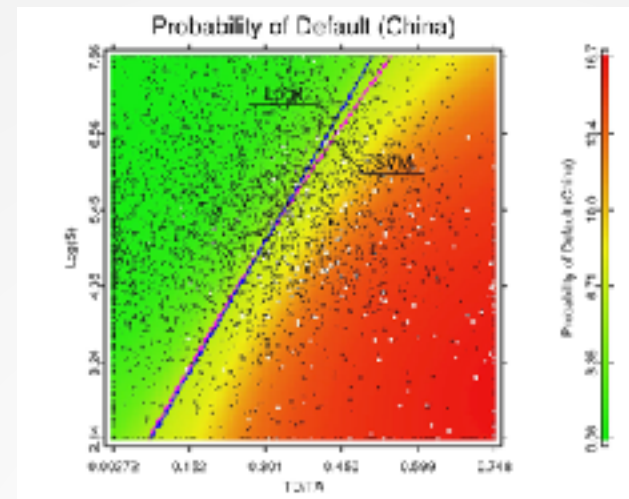
irtg1792.hu-berlin.de



Smart Data Analytics



Digital Economy & Decision Analytics



DT Decision Tree, RF Random Forest, Bagging, SVM Support Vector Machine, Clustering, Smoothing, ML Machine Learning, Variable Selection, Q Quantlets, LDA Latent Dirichlet Analysis, DTM Dynamic Topic Modeling, P2P Peer to Peer lending, PCA Principle Component Analysis, Spectral Clustering, CRIX Crypto currency Index, Sentiment Analysis, Web Scraping, Opinion Mining



Digital Economy & Decision Analytics



欢迎来到我的网页 <https://gla.cathychen.info>

陈怡璇 University of Glasgow 大學教授
International Research Training Group 1792 成員
新加坡管理大學訪問學者



欢迎来到我的网页 hu.berlin/wkh !

沃夫冈是柏林洪堡大学经济商学院的终身教授
统计与计量研究所以及数据研究中心主任
同时兼任IRTG项目的总负责人
厦门大学的外籍专家教授



A word cloud titled "Machine Learning Data Analytics Computer Science". The words are arranged in a circular pattern, with the largest words being "Machine Learning Data Analytics" and "Computer Science". Other prominent words include "Python", "SQL", "Hadoop", "Big Data", "Spark", "Java", "Scala", "R", "Pandas", "TensorFlow", "Keras", "PyTorch", "Neural Networks", "Decision Trees", "Support Vector Machine", "Bayesian Modeling", "K-Nearest Neighbors", "Random Forest", "Gradient Boosting", "XGBoost", "LightGBM", "CatBoost", "Cox Regression", "Logistic Regression", "Linear Regression", "Multivariate Regression", "Principal Component Analysis", "Partial Least Squares", "Canonical Correlation Analysis", "Discriminant Analysis", "Naive Bayes", "Markov Chain Monte Carlo", "Variational Autoencoders", "Generative Adversarial Networks", "Reinforcement Learning", "Deep Q-Networks", "Convolutional Neural Networks", "Recurrent Neural Networks", "Long Short-Term Memory", "Gated Recurrent Units", "Autoencoders", "Generative Models", "Variational Models", "Bayesian Networks", "Hidden Markov Models", "Markov Decision Processes, and many others. The words are in various colors including red, blue, green, yellow, and purple.

4. Machine Learning Methods

Introduction to ML Machine Learning

Python in Finance

Statistics and Finance

DEDA Schedule and Course Outline

C YH Chen, WK Härdle

1. How to install and run Python. Only MAC instructions are shown.
2. How to make an elephant with 4 params and wiggle its trunk
3. Note that all code is on QuantNet www.quantlet.de
4. https://github.com/QuantLet/DEDA_Class_2017
5. https://github.com/QuantLet/DEDA_Class_2017/tree/master/DEDA_Class_2017_Python_Introduction
6. Unit_2 : Packages, etc...



Python Programming Reference Books

[1] Charles Russell Severance. **Python for everyone**[M]. CreateSpace Independent Publishing Platform (2016)

Free PDF: <https://www.py4e.com/book>

[2] Shaw Z A. **Learn Python 3 the Hard Way** [J]. Addison-Wesley Professional (2017)

[3] McKinney W. **Python for data analysis: Data wrangling with Pandas, NumPy, and IPython**[M]. O'Reilly Media, Inc. (2012)

[4] Hilpisch Y. **Python for Finance: Analyze Big Financial Data**[M]. O'Reilly Media, Inc., (2014)

[5] Jones B., Beazley D. **Python Cookbook, 3rd Edition**[M]. O'Reilly Media, Inc., (2013)



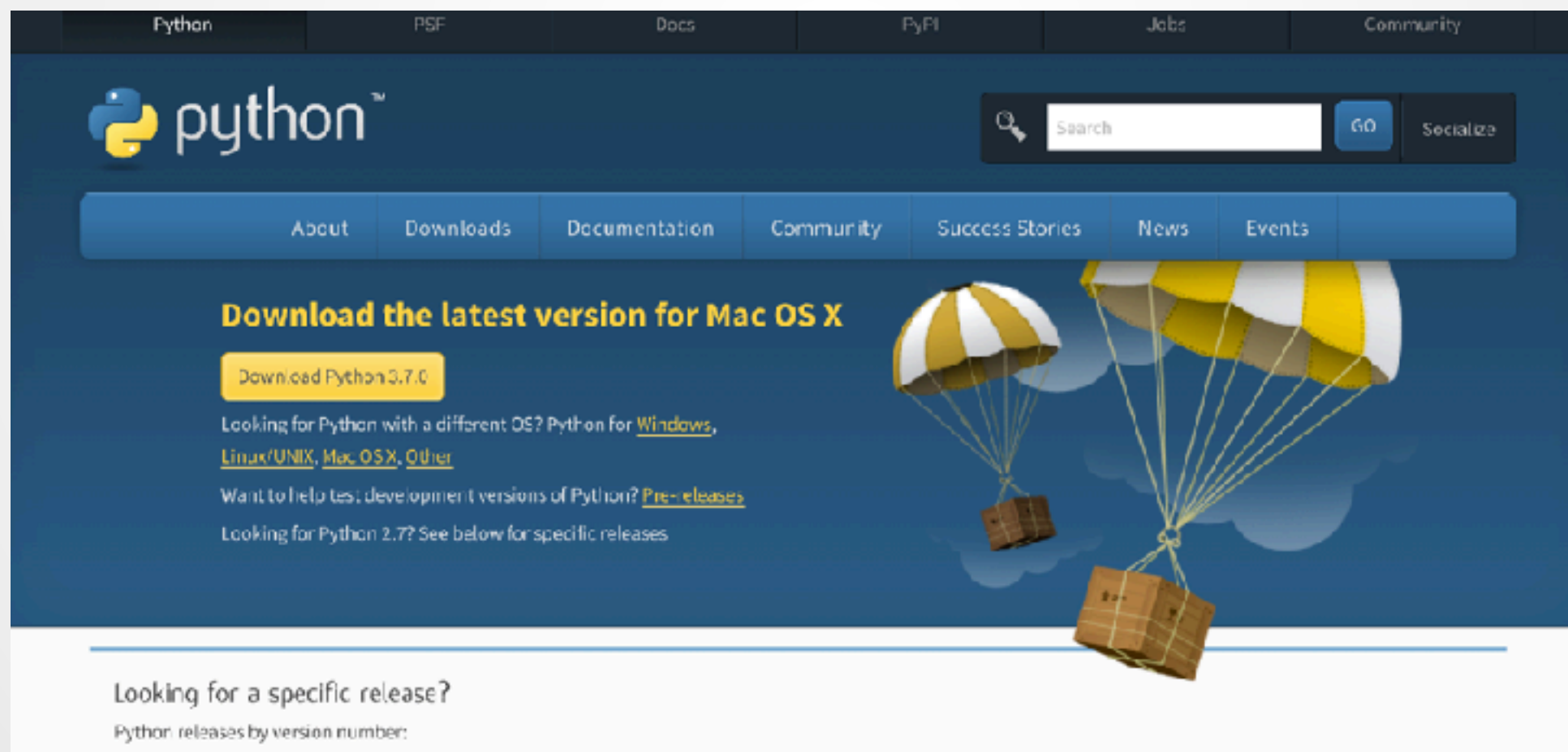
Python Installation and Environment

Open the Python official web site and download python install package:

<https://www.python.org/>

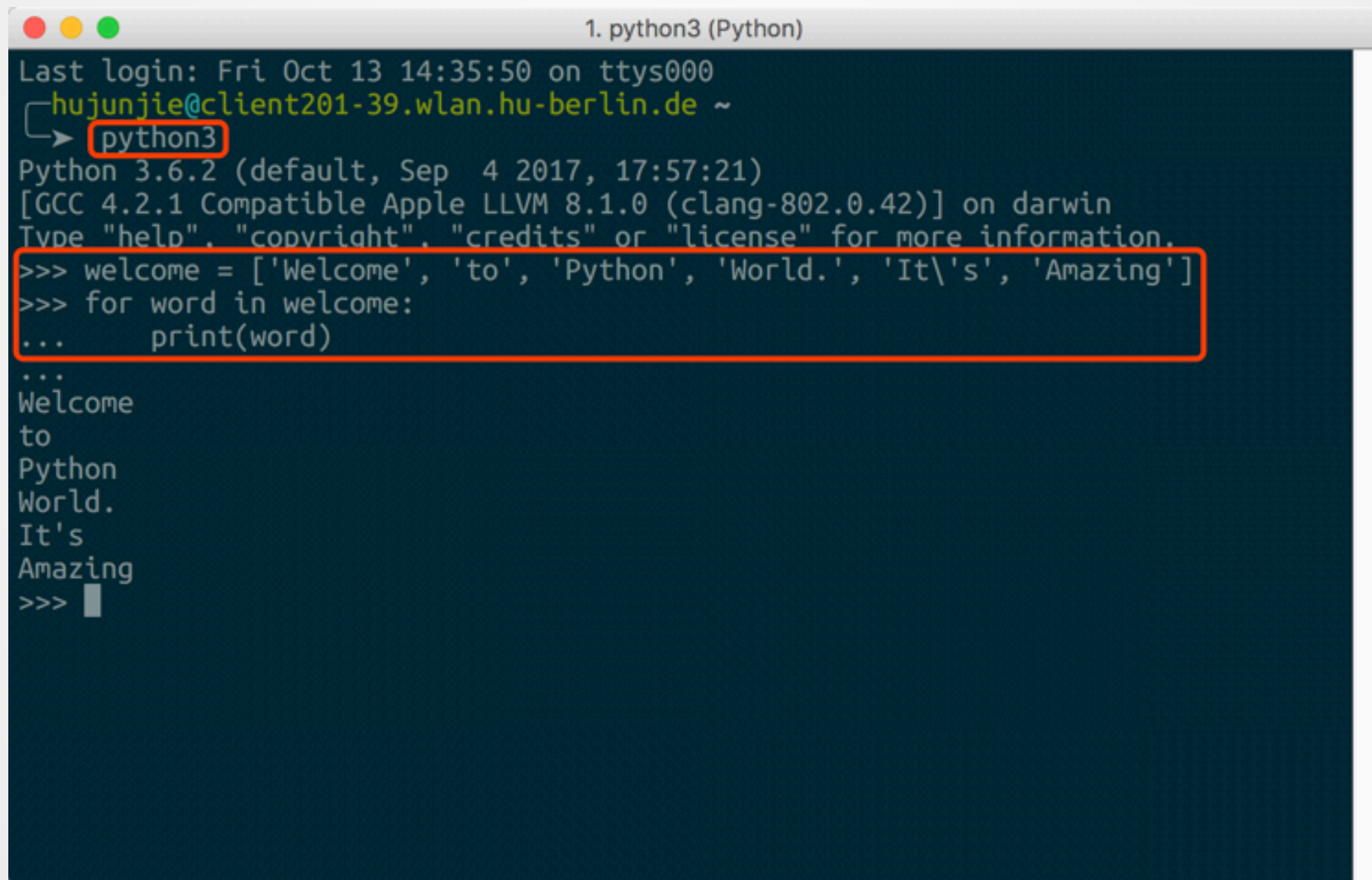
Python 3 is recommended,

Python 2 is retiring soon (Support stops 1st Jan. 2020), download Python 3.7 (or higher version) if not for special reason. Alternatively download Anaconda containing Python 3 already.



Python Installation and Environment

After installation, using "terminal" (mac os) / "cmd" (windows) to enter the python interactive interface. Simply typing "python3".

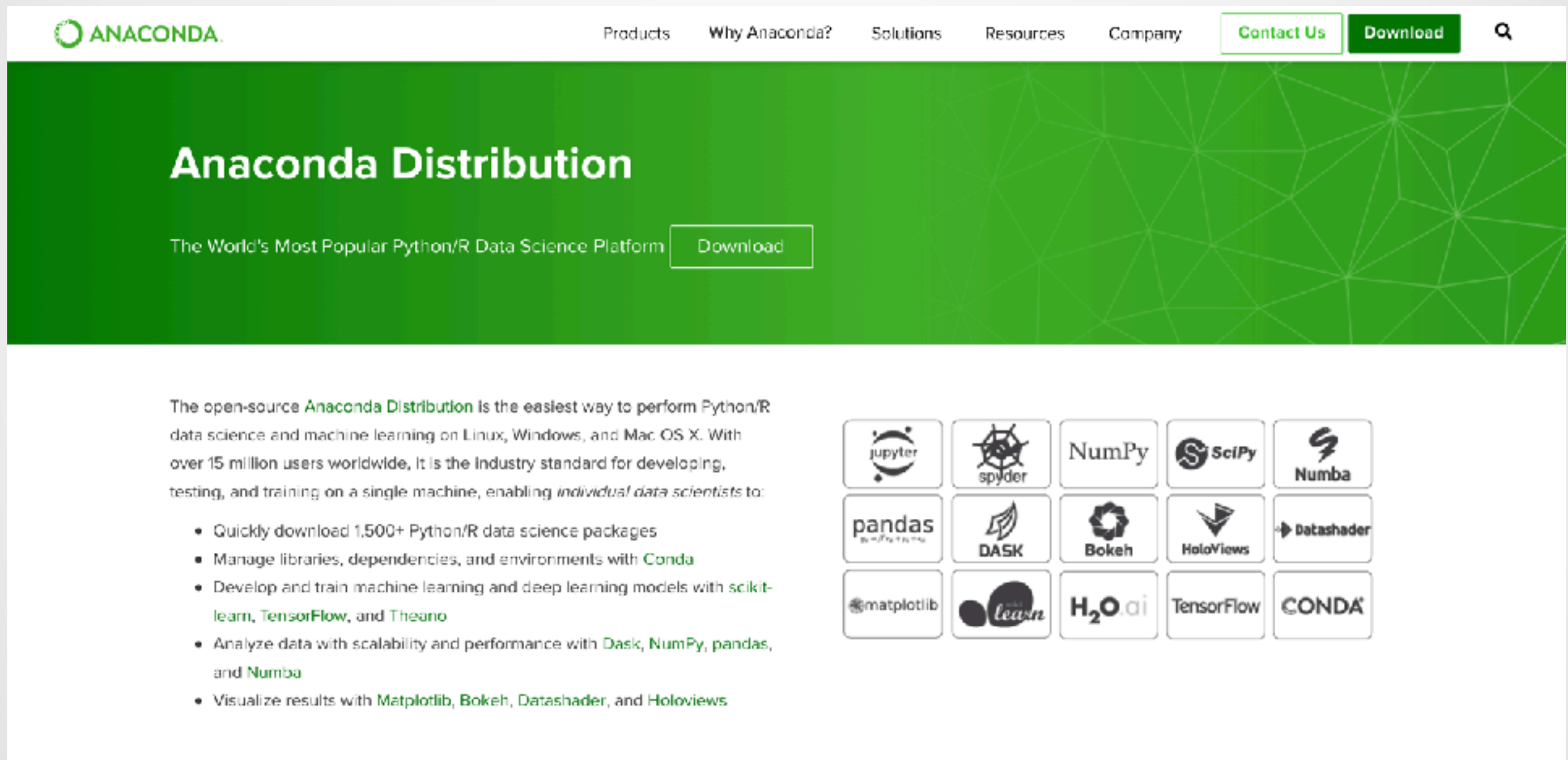


```
1. python3 (Python)
Last login: Fri Oct 13 14:35:50 on ttys000
hujunjie@client201-39.wlan.hu-berlin.de ~
➔ python3
Python 3.6.2 (default, Sep  4 2017, 17:57:21)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> welcome = ['Welcome', 'to', 'Python', 'World.', 'It\'s', 'Amazing']
>>> for word in welcome:
...     print(word)
...
Welcome
to
Python
World.
It's
Amazing
>>>
```



Anaconda - A Data Science Platform

Download Anaconda from: <https://www.anaconda.com/download/>



The screenshot shows the Anaconda Distribution website. The header includes the Anaconda logo and navigation links: Products, Why Anaconda?, Solutions, Resources, Company, Contact Us, and Download. The main section features the title "Anaconda Distribution" and the tagline "The World's Most Popular Python/R Data Science Platform" with a "Download" button. Below this, a paragraph describes the open-source distribution as the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. A list of benefits follows, including quick download of 1,500+ packages, management of libraries and environments with Conda, development and training of machine learning models with scikit-learn, TensorFlow, and Theano, analysis of data with scalability and performance using Dask, NumPy, pandas, and Numba, and visualization of results with Matplotlib, Bokeh, Datashader, and Holoviews. To the right, a grid of 15 logos represents various data science and machine learning libraries and tools.

Anaconda Distribution

The World's Most Popular Python/R Data Science Platform [Download](#)

The open-source **Anaconda Distribution** is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with **Conda**
- Develop and train machine learning and deep learning models with **scikit-learn**, **TensorFlow**, and **Theano**
- Analyze data with scalability and performance with **Dask**, **NumPy**, **pandas**, and **Numba**
- Visualize results with **Matplotlib**, **Bokeh**, **Datashader**, and **Holoviews**

Logos displayed in the grid:

- jupyter, spyder, NumPy, SciPy, Numba
- pandas, DASK, Bokeh, HoloViews, Datashader
- matplotlib, scikit-learn, H2O.ai, TensorFlow, CONDA

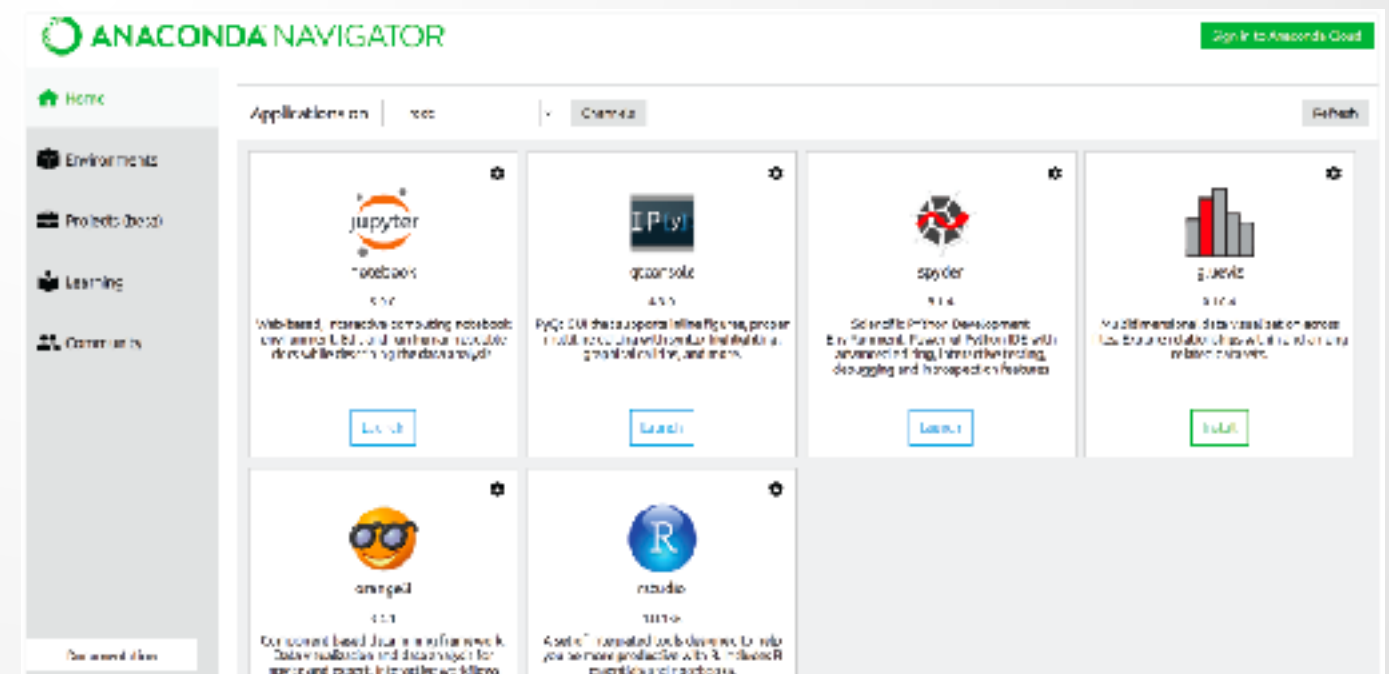
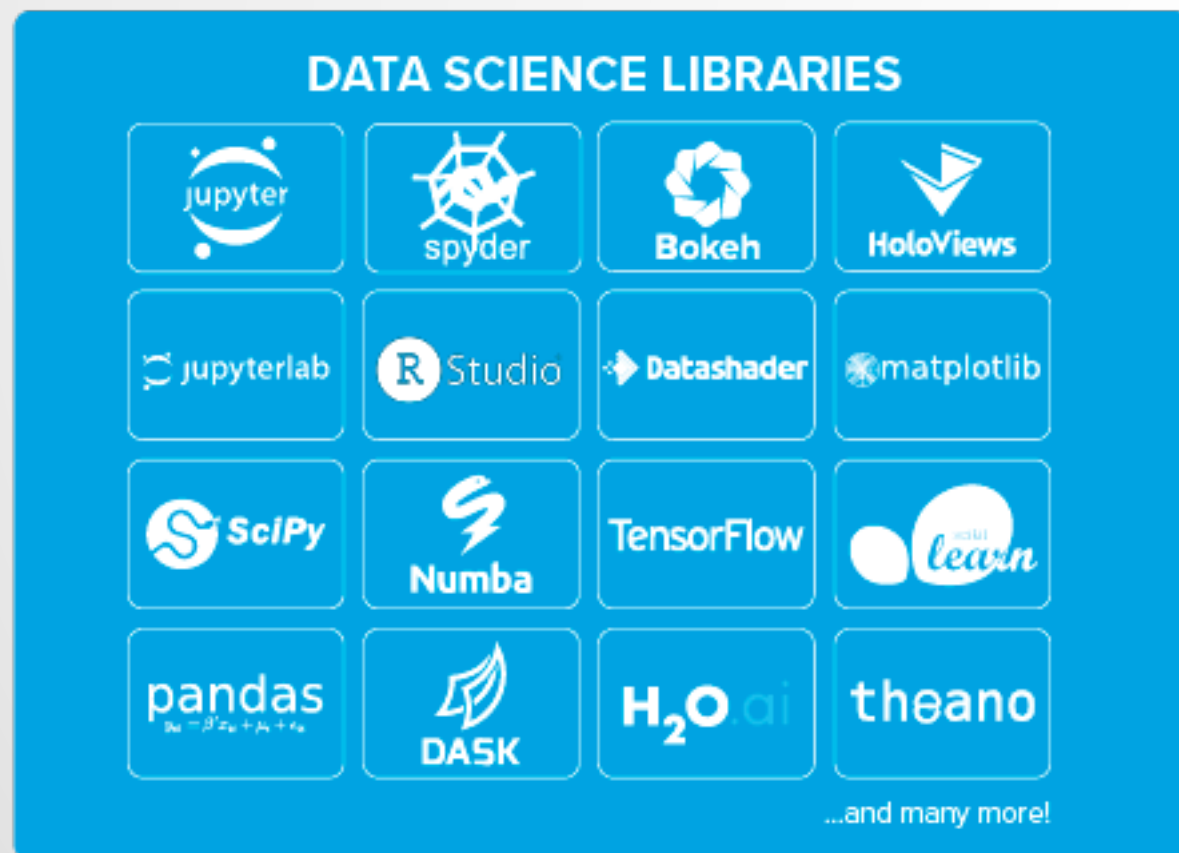


Anaconda - A Data Science Platform

Download Anaconda from: <https://www.anaconda.com/download/>

Anaconda is a package that integrates python environment, IDEs (spyder, jupyter, R studio), package management system (conda) and a lot of useful packages.

In the nutshell, Anaconda is an open-to-use platform for data science. It makes people easily access python.



More Recommendations

▣ Customizing coding environment:

▶ IDEs:

PyCharm, Visual Studio

▶ Package & environment management:

pip, Virtualenv, pipenv, pyenv

▶ Version Control:

Git (GitHub, GitLab)

▶ More tools for smoother coding:

brew (for mac), Sublime3, Atom, Vim, Emacs and etc. *

** Some tools may have sharp learning curve.*



Basic Syntax _ numeric

```
"""
```

Python provides a straight way for numerical operations

Try the basics:

+, -, *, /, %, **, //

```
"""
```

```
a = 5
```

```
b = 3
```

```
a *= 2 # 10
```

```
round(a / b, 4) # 3.3333
```

```
c = '10'
```

```
d = '20'
```

```
e = c + d # '1020'
```

```
f = int(c) + int(d) # 30
```

```
"""
```

try comparison operations:

==, <, <=, >, >=, !=

```
"""
```

Try this
cool
link!



Basic Syntax _ comparison

```
"""
```

Python provides a straight way for numerical operations

Try comparison operations

`==, <, <=, >, >=, !=`

```
"""
```

```
a = 5
```

```
a == 5 # True
```

```
a <= 5 # True
```

```
a < 5 # False
```

```
a = 'A'
```

```
b = 'B'
```

```
a == b # False
```

```
a != b # True
```



Basic Syntax _ string

```
"""
```

String is a basic type in python, it's commonly used and very powerful

```
"""
```

```
welcome_list = ['Welcome', 'to', 'Python', 'World.', 'It\'s', 'Amazing']
```

using back slash to escape the single quote.

using for loop to iterate all elements in the list.

```
for word in welcome_list:
```

```
    print(word)
```

a string is also an object, using the join method to connect all the words in the list.

```
welcome_sentence = ' '.join(welcome_list)
```

slicing string by indices. Check carefully how many characters you are slicing

```
welcome_sliced = welcome_sentence[0:10]
```

try changing the indices to negative.



Basic Syntax _ string

```
"""
```

String is a basic type in python, it's commonly used and very powerful

```
"""
```

see other methods of string object, like:

```
welcome_upper_case = welcome_sentence.upper()
```

by using dir() function, or help() function; you may see all ops on the str object

```
dir(str)
```

or a str instance

```
dir(welcome_upper_case)
```

likewise,

```
help(str)
```

formatting string

```
greeting = 'Hallo'
```

```
name = 'Jon'
```

using format method, -> Hallo, JON. Welcome to Python World. It's amazing

```
welcome_jon = '{}. {}'.format(greeting, name.upper()) + welcome_sentence
```

using f string, you can write variable names inside the brackets, directly.

```
welcome_jon_f = f'{greeting}, {name.upper()}. ' + welcome_sentence
```



Basic Syntax _ list

```
"""
```

```
List is a versatile Python data type to group values.  
Lists can contain different types, e.g. strings, numbers,  
functions, lists, ...
```

```
"""
```

```
p = [2,3,5,7,11]  
p # [2, 3, 5, 7, 11]
```

```
# indexing
```

```
p[0] # 2  
p[-1] # 11
```

```
# slicing
```

```
p[:2] # [2, 3]  
p[-3:] # [5, 7, 11]
```

```
# appending
```

```
p.append(13) # [2, 3, 5, 7, 11, 13]  
p.extend([17,19]) # [2, 3, 5, 7, 11, 13, 17, 19]
```

```
l = list('hallo') # ['h', 'a', 'l', 'l', 'o']
```

```
l.sort() # ['a', 'h', 'l', 'l', 'o']
```



Basic Syntax _ dictionary

```
"""
```

Dictionary is indexed by keys, in general strings.

```
"""
```

```
course = dict(name='DEDA', unit=0) # {'name': 'DEDA', 'unit': 0}
```

alternative

```
course = {'name':'DEDA', 'unit':0} # {'name': 'DEDA', 'unit': 0}
```

accessing

```
course['unit'] # 0
```

```
course['unit'] = 1
```

get keys

```
course.keys() # ['name', 'unit']
```

```
course.values() # ['DEDA', 1]
```

ATTENTION: Output type varies with version of Python: {'Python2.7': list, 'Python3.7': 'its own data type'}

adding values

```
course.update({'lecturers':['Chen','Härdle']})
```

{'lecturers': ['Chen', 'Härdle'], 'name': 'DEDA', 'unit': 1}



Basic Syntax _ if

```
"""
Control Flow Tools: if/elif/else
"""
x = 10
if x < 0:
    print('Negative value')
elif x == 0:
    print('Zero')
else:
    print('Positive value')
# Positive value

# Conditions can be combined or altered with: and, or, not, is, is not
# Comparison statements can be used
p = [2,3,5,7,11]
3 in p # True
3 in p and 4 in p # False
3 in p or 4 in p # True
if x is not None:
    print('Value is not None')
# Alternative
if not x is None:
    print('Value x is not None')
```



Basic Syntax _ for loop

```
"""
For Loop can iterate over all iterables.
"""

l = list([1,2,3,4,5])
for i in l:
    print(i*2, end=' ')
# 2 4 6 8 10

for i in range(6):
    if i == 3:
        continue
    print(i*2, end=' ')
# 0 2 4 8 10

for i in 'DEDA':
    print(i, end=' ')
# D E D A

d = dict(a=1,b=2)
for k,v in d.items():
    print('{} has value {}'.format(k,v))
# a has value 1
# b has value 2
```



Basic Syntax _ while loop

```
"""
```

While loop

There is no Do-While-Loop

```
"""
```

Fibonacci series: sum of two preceding numbers defines next number

```
from __future__ import print_function
```

```
a, b = 0, 1
```

```
while b < 100:
```

```
    print(b, end=' ')
```

```
    a, b = b, b+a
```

1 1 2 3 5 8 13 21 34 55 89

ATTENTION: Make sure not to have infinite loop (loop with tautology in condition)

Do While Loop

```
fib = [0,1]
```

```
while True:
```

```
    fib.append(sum(fib[-2:]))
```

```
    if fib[-1] > 100:
```

```
        break
```

```
fib # [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
```



Basic Syntax _ function

```
"""
Function definition
"""

def square_numeric(x):
    """ Squares numeric x"""
    return x**2

def square_iterable(x):
    """ Squares numerics in iterable x"""
    ret = []
    for i in x:
        ret.append(square_numeric(i))
    return ret

def square_iterabel_short(x):
    """ Squares numerics in iterable x"""
    return [square_numeric(i) for i in x]

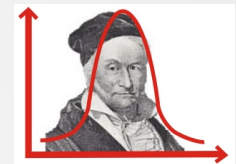
x = [1,2,3,4,5]
square_iterable(x) # [1, 4, 9, 16, 25]
square_iterabel_short(x) # [1, 4, 9, 16, 25]
```



Fitting an elephant with 4 params

- ▣ “*With four parameters I can fit an elephant, and with five I can make him wiggle his trunk*”

John von Neumann (1903-1957) on BBI:

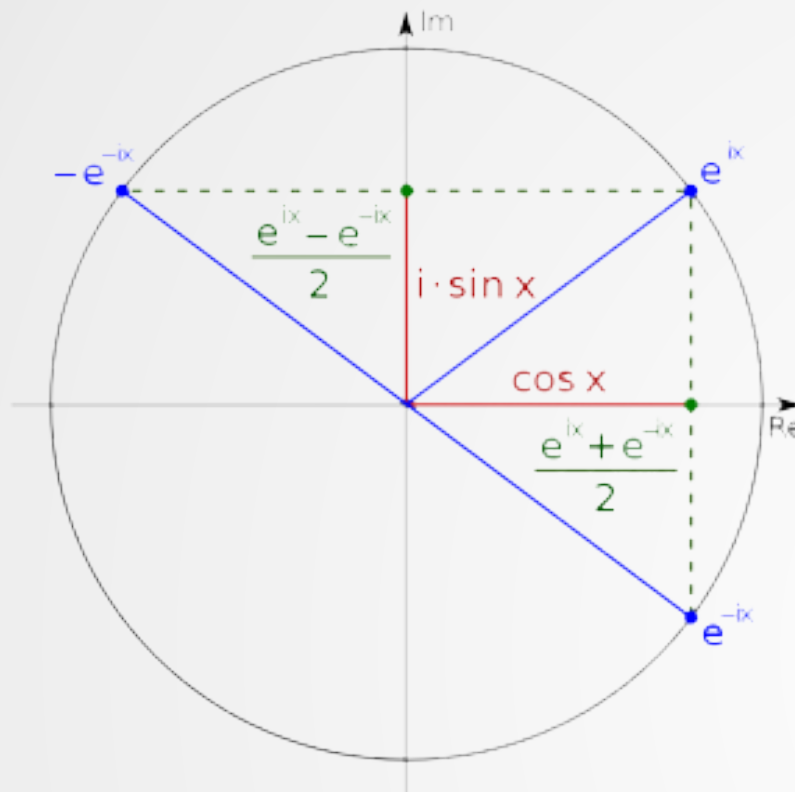


- ▣ Complex models with many params fits any data set. JvN employed Fourier coordinate expansion to describe a 2-dim contour in the form of an elephant.
- ▣ The Fourier expansion has many applications in engineering, finance although in the beginning it was called “a monster” by Henri Poincaré

Fitting Elephant



Complex Numbers



$$(a, b)(d, c) = (ac - bd, ad + bc)$$

$$\bar{z} = a - ib \quad |z| = \sqrt{a^2 + b^2}$$

$$z + \bar{z} = 2 \operatorname{Re} z$$

$$\operatorname{Im} z = \frac{z - \bar{z}}{2i}$$

There is NO order in \mathbb{C} , i.e.

$$\cancel{z > w, z < w} \quad \cancel{-i > 0} \quad \cancel{i > 0}$$

$$(\cos \phi + i \sin \phi)^n = \cos n\phi + i \sin n\phi$$

$$\cos \phi = \frac{e^{i\phi} + e^{-i\phi}}{2} \quad \sin \phi = \frac{e^{i\phi} - e^{-i\phi}}{2i}$$



The Fourier coordinate expansion

- ▣ The Fourier expansion is the opposite of the FFT
- ▣ Note that we do not have the complex unit j anymore!
- ▣ The coefficients come from the four input parameters

$p_1 = 50 - 30j$	$C_x[1] = p_1.\text{Re} \cdot j = 50j$	$C_y[1] = p_4.\text{Im} + p_1.\text{Im} \cdot j = -60 - 30j$
$p_2 = 18 - 8j$	$C_x[2] = p_2.\text{Re} \cdot j = 18j$	$C_y[2] = p_2.\text{Im} \cdot j = 8j$
$p_3 = 12 - 10j$	$C_x[3] = p_3.\text{Re} = 12$	$C_y[3] = p_3.\text{Im} \cdot j = -10j$
$p_4 = -14 - 60j$	$C_x[2] = p_4.\text{Re} = -14$	

$$A = C.\text{Re} , \quad B = C.\text{Im}$$

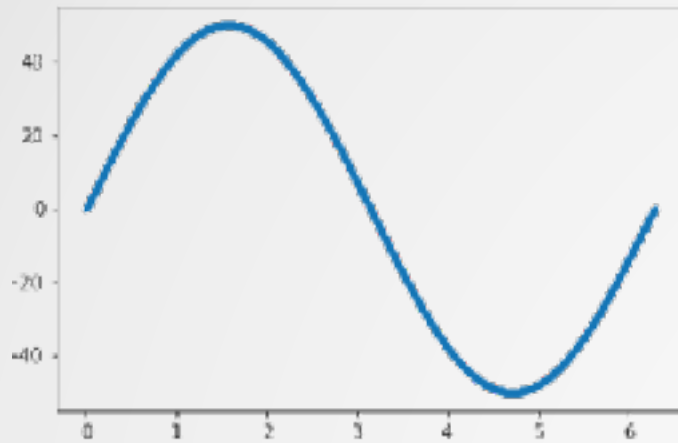
$$f_x(t) = \sum_{k=0}^5 \left(A_x[k] \cos(kt) + B_x[k] \sin(kt) \right) = 50 \sin(t) + 18 \sin(2t) + 12 \cos(3t) - 14 \cos(5t)$$

$$f_y(t) = \sum_{k=0}^5 \left(A_y[k] \cos(kt) + B_y[k] \sin(kt) \right) = -60 \cos(t) - 30 \sin(t) + 8 \sin(2t) - 10 \sin(3t)$$

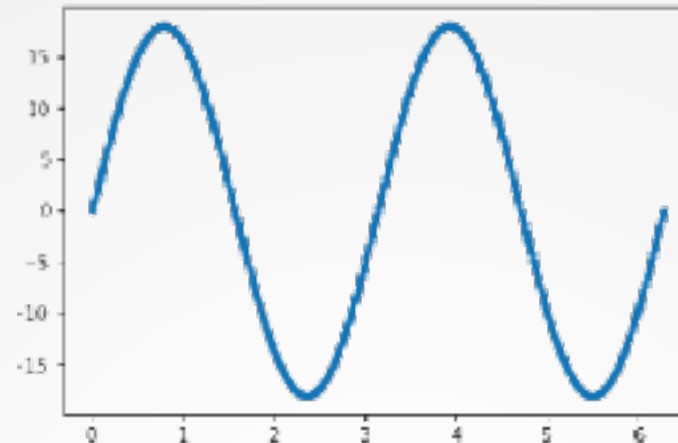


The elephant x-coordinate

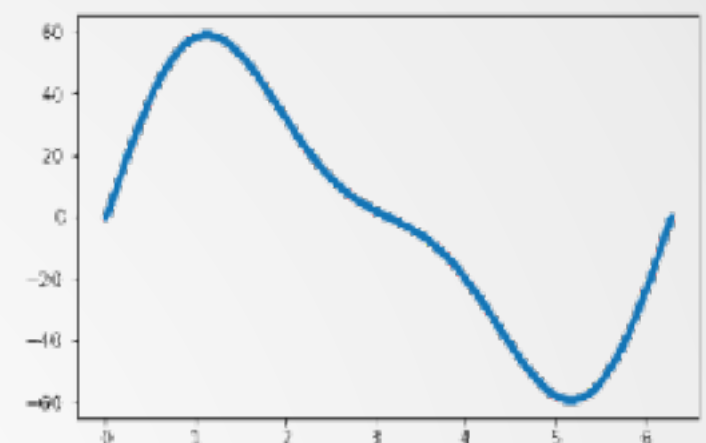
$50\sin(t)$



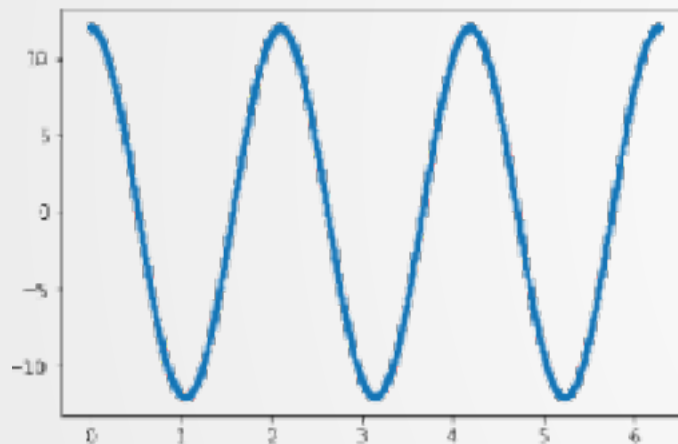
$18\sin(t)$



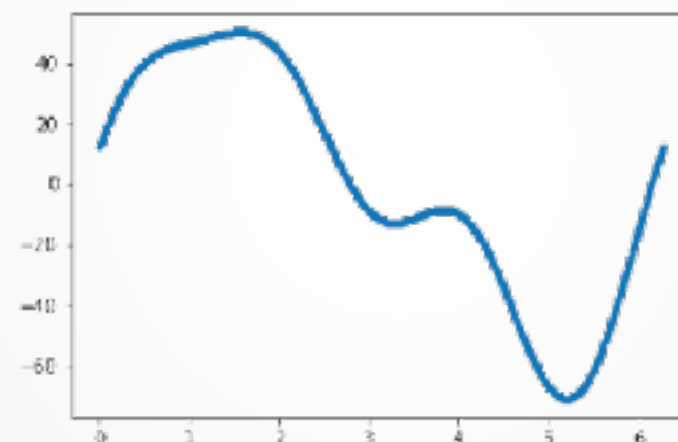
$50\sin(t) + 18\sin(2t)$



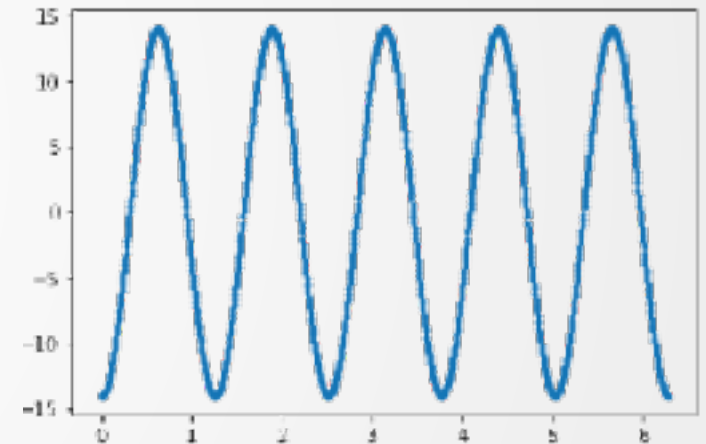
$12\cos(3t)$



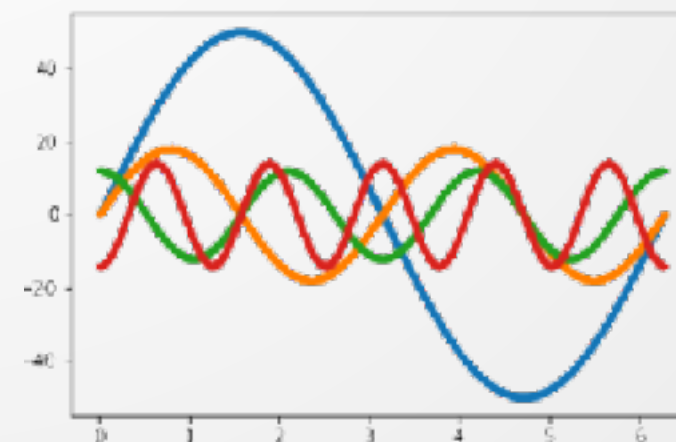
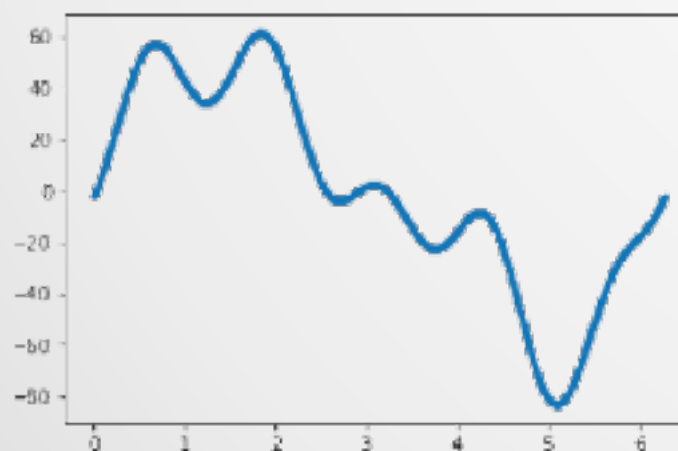
$50\sin(t) + 18\sin(2t) + 12\cos(3t)$



$-14\cos(5t)$



$50\sin(t) + 18\sin(2t) + 12\cos(3t) - 14\cos(5t)$

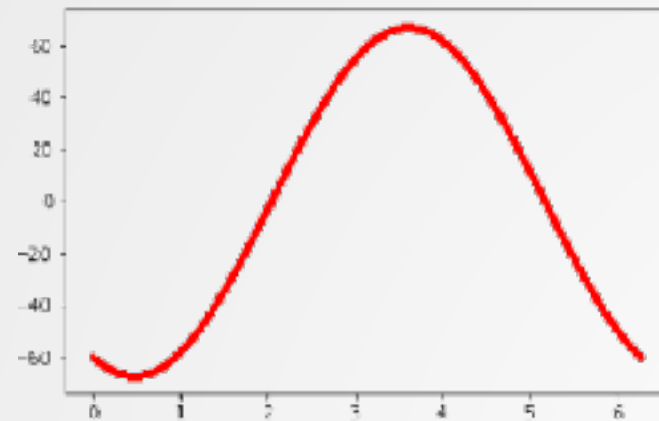


50 sin(t)
18 sin(2t)
12 cos(3t)
-14 cos(5t)

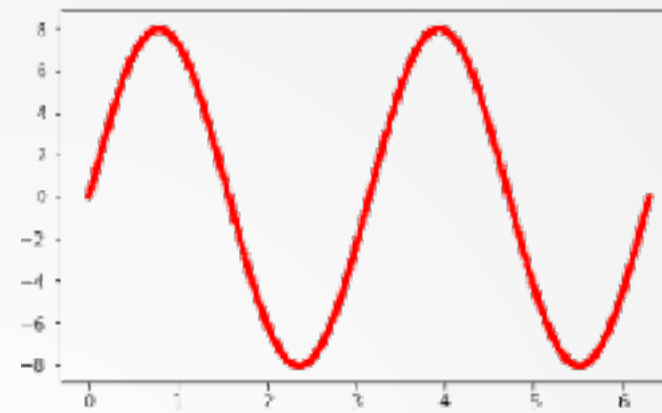


The elephant y-coordinate

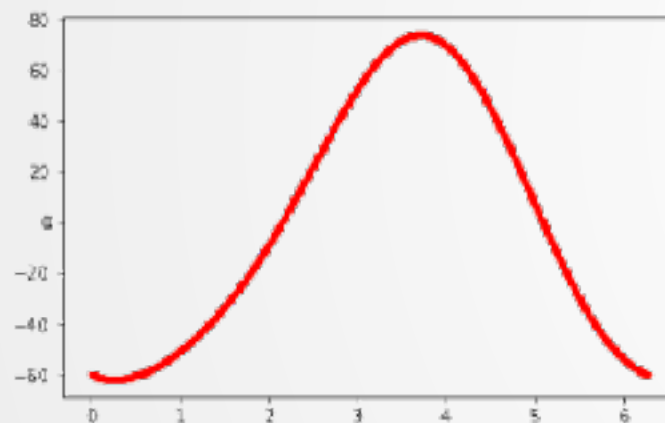
$$-60\cos(t) - 30\sin(t)$$



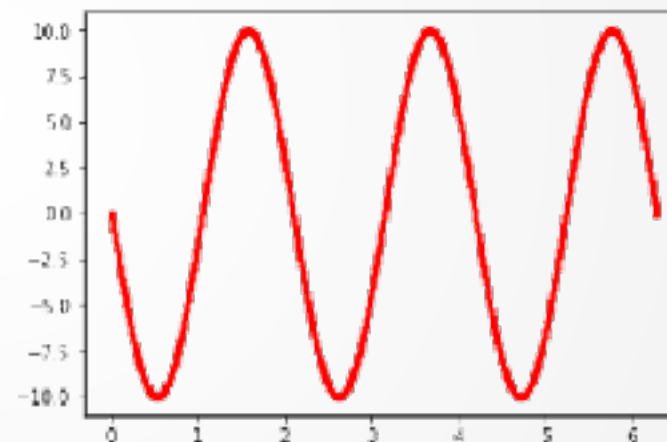
$$8\sin(2t)$$



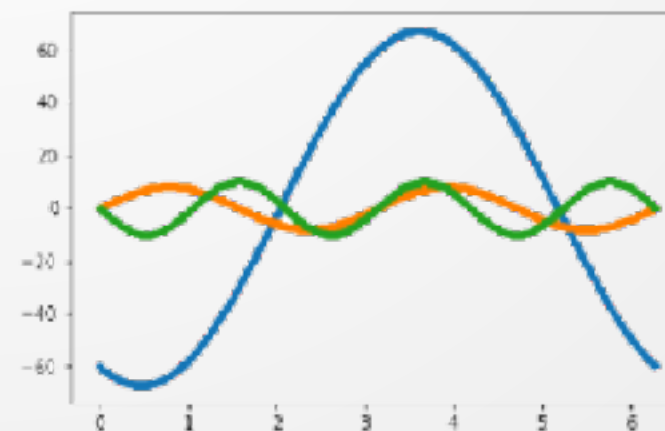
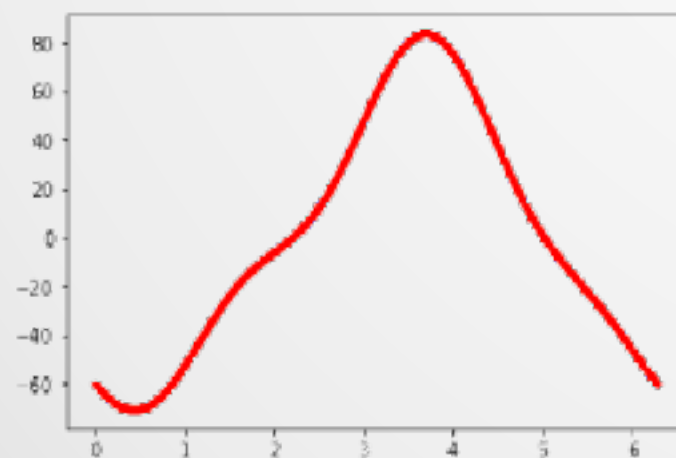
$$-60\cos(t) - 30\sin(t) + 8\sin(2t)$$



$$-10\sin(3t)$$



$$-60\cos(t) - 30\sin(t) + 8\sin(2t) - 10\sin(3t)$$

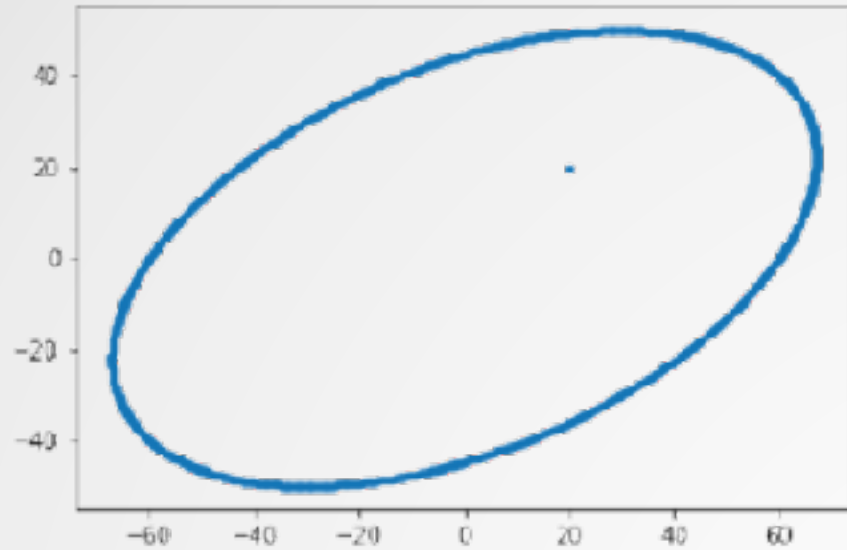


-60 cos(t) - 30 sin(t)

-10 sin(3t)

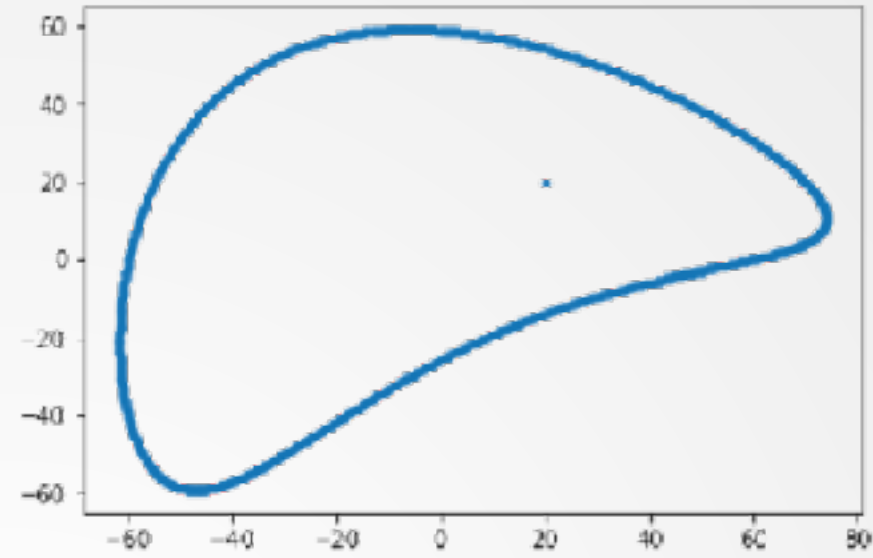
8 sin(2t)





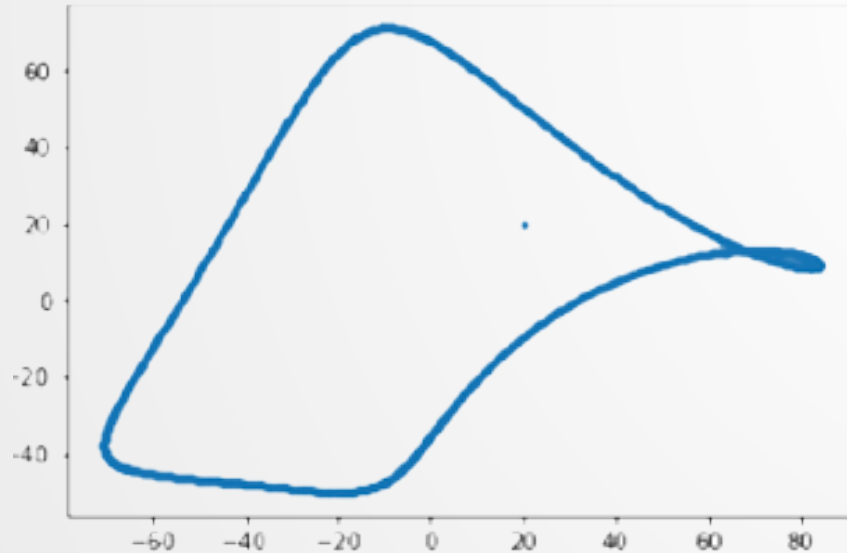
$$f(t) = \sum_{k=0}^1 \left(A[k] \cos(kt) + B[k] \sin(kt) \right)$$

$$(50 \sin(t), -60 \cos(t) - 30 \sin(t))$$



$$f(t) = \sum_{k=0}^2 \left(A[k] \cos(kt) + B[k] \sin(kt) \right)$$

$$(50 \sin(t) + 18 \sin(2t), -60 \cos(t) - 30 \sin(t) + 8 \sin(2t))$$

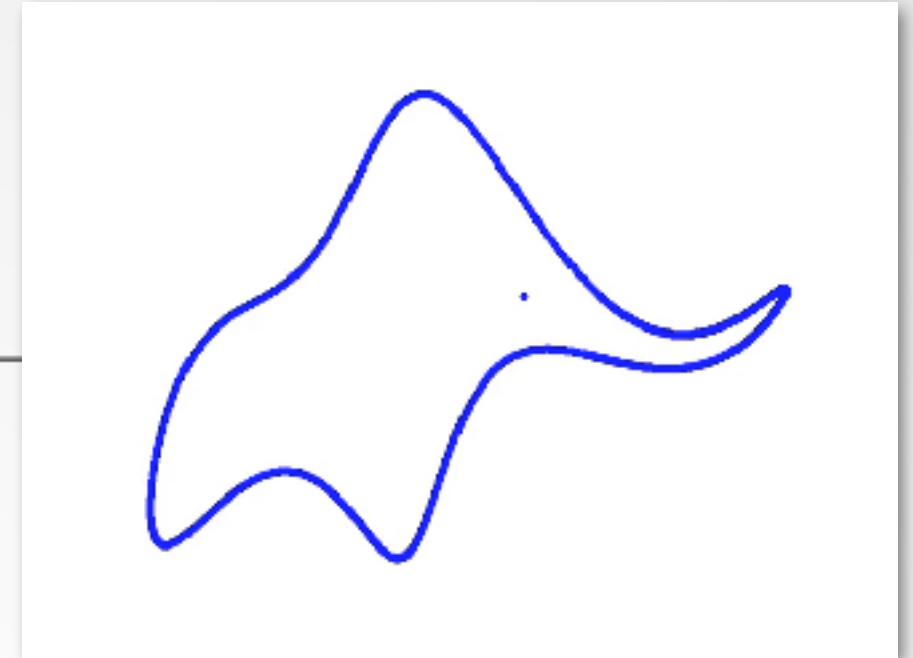
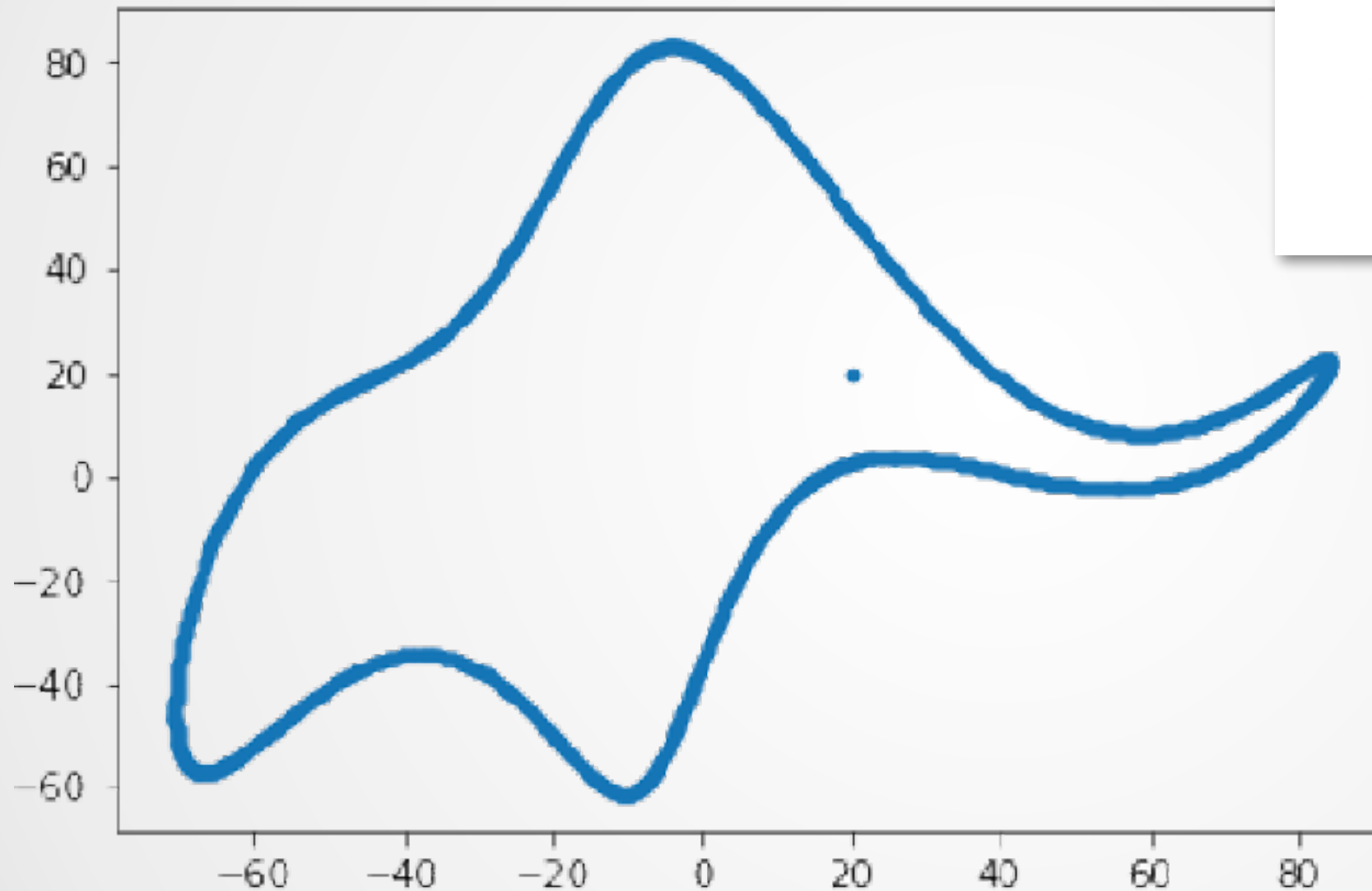


$$f(t) = \sum_{k=0}^3 \left(A[k] \cos(kt) + B[k] \sin(kt) \right)$$

$$(50 \sin(t) + 18 \sin(2t) + 12 \cos(3t), -60 \cos(t) - 30 \sin(t) + 8 \sin(2t) - 10 \sin(3t))$$



Finally the elephant



$$f(t) = \sum_{k=0}^5 \left(A[k] \cos(kt) + B[k] \sin(kt) \right)$$

$$(50\sin(t) + 18\sin(2t) + 12\cos(3t) - 14\cos(5t), -60\cos(t) - 30\sin(t) + 8\sin(2t) - 10\sin(3t))$$



Using complex numbers allows low number of params

```

from numpy import append, cos, linspace, pi, sin, zeros

# elephant parameters PLEASE NOTE IN SPYDER YOU SHOULD DISABLE THE ACTIVE SUPPORT in PREFs
parameters = [50 - 30j, 18 + 8j, 12 - 10j, -14 - 60j, 20 + 20j]

def fourier(t, C):
    f = zeros(t.shape)
    for k in range(len(C)):
        f += C.real[k] * cos(k * t) + C.imag[k] * sin(k * t)
    return f

def elephant(t, p):
    npar = 6
    Cx = zeros((npar,), dtype='complex')
    Cy = zeros((npar,), dtype='complex')
    Cx[1] = p[0].real * 1j
    Cy[1] = p[3].imag + p[0].imag * 1j
    Cx[2] = p[1].real * 1j
    Cy[2] = p[1].imag * 1j
    Cx[3] = p[2].real
    Cy[3] = p[2].imag * 1j
    Cx[5] = p[3].real
    x = append(fourier(t, Cy), [p[4].imag])
    y = -append(fourier(t, Cx), [-p[4].imag])
    return x, y

def init_plot():
    # draw the body of the elephant
    # create trunk
    x, y = elephant(linspace(2.9 * pi, 0.4 + 3.3 * pi, 1000), parameters)
    for ii in range(len(y) - 1):
        y[ii] -= sin(((x[ii] - x[0]) * pi / len(y))) * sin(float(0)) * parameters[4].real
    trunk.set_data(x, y)
    return trunk,

```



..and move the trunk

```
import matplotlib
matplotlib.use('TKAgg')
from matplotlib import animation
import matplotlib.pyplot as plt

def move_trunk(i):
    x, y = elephant(linspace(2.9 * pi, 0.4 + 3.3 * pi, 1000), parameters)
    for ii in range(len(y) - 1):
        y[ii] -= sin(((x[ii] - x[0]) * pi / len(y))) * sin(float(i)) * parameters[4].real
    trunk.set_data(x, y)
    return trunk,

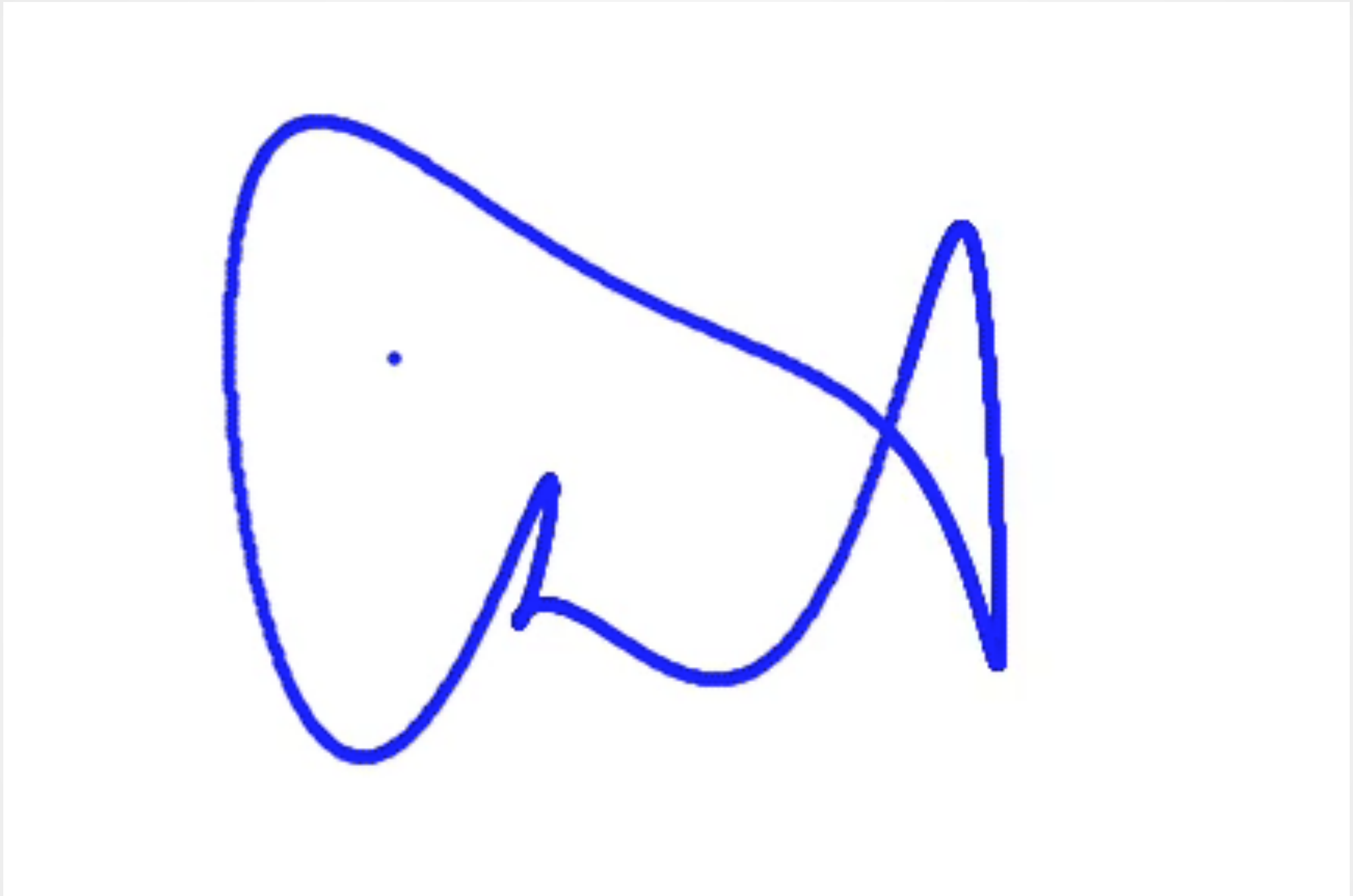
fig, ax = plt.subplots()
# initial the elephant body
x, y = elephant(t=linspace(0.4 + 1.3 * pi, 2.9 * pi, 1000), p=parameters)
plt.plot(x, y, 'b.')
plt.xlim([-75, 90])
plt.ylim([-70, 87])
plt.axis('off')
trunk, = ax.plot([], [], 'b.') # initialize trunk

ani = animation.FuncAnimation(fig=fig,
                              func=move_trunk,
                              frames=1000,
                              init_func=init_plot,
                              interval=500,
                              blit=False,
                              repeat=True)

plt.show()
```



Patrick's happy spermwhale



$$f(t) = \sum_{k=0}^5 \left(A[k] \cos(kt) + B[k] \sin(kt) \right)$$

$$(30\sin(t) + 20\sin(2t) + 40\cos(3t) + 20\cos(5t), -50\cos(t) - 10\sin(t) + 20\sin(2t) + 10\sin(3t))$$



Philipp's flying swan

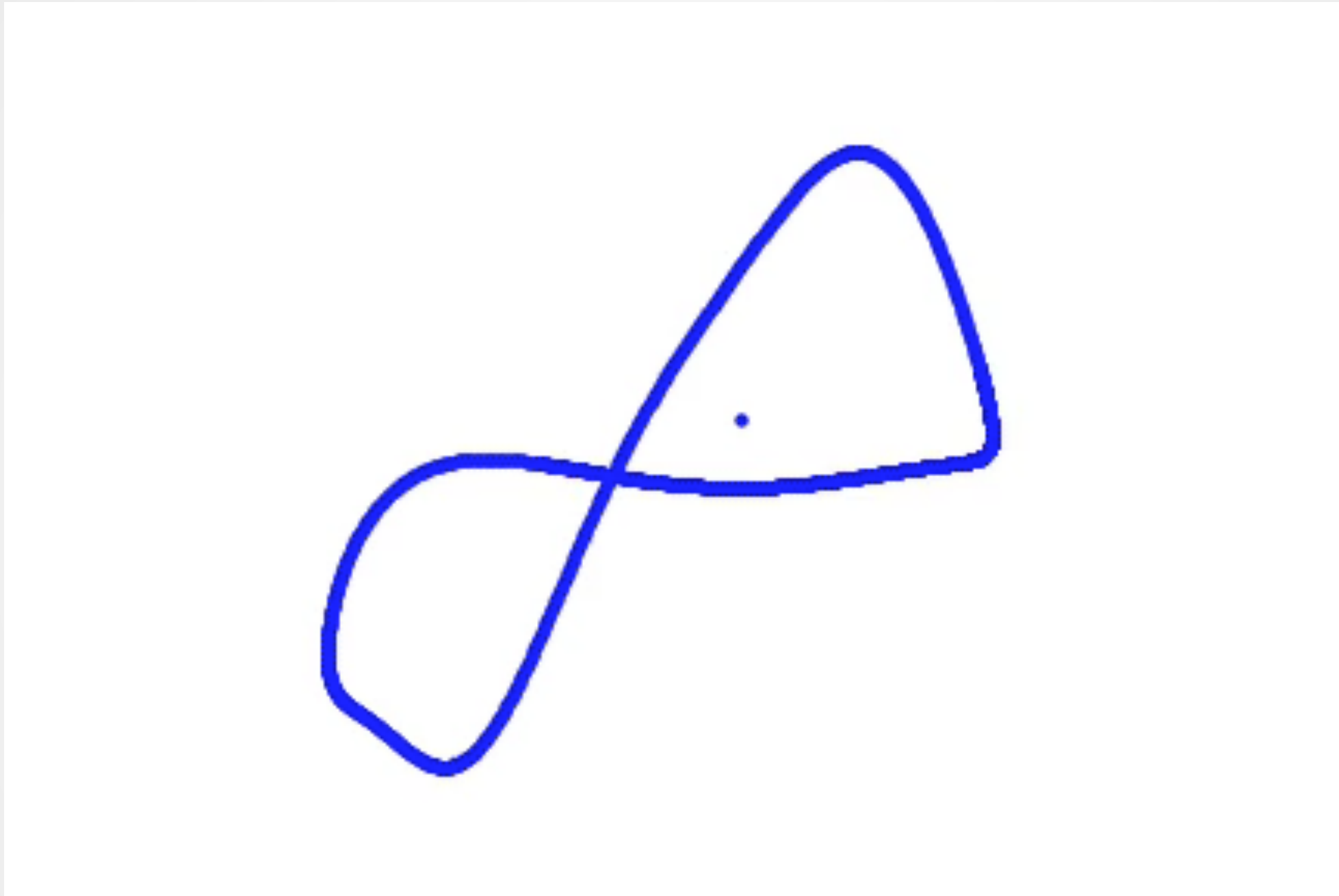


$$f(t) = \sum_{k=0}^5 \left(A[k] \cos(kt) + B[k] \sin(kt) \right)$$

$$(1\sin(t) + 9\sin(2t) + 1\cos(3t) + 9\cos(5t), + 9\cos(t) - 2\sin(t) + 9\sin(2t) - 2\sin(3t))$$



Kathrin's hungry animal



$$f(t) = \sum_{k=0}^5 \left(A[k] \cos(kt) + B[k] \sin(kt) \right)$$

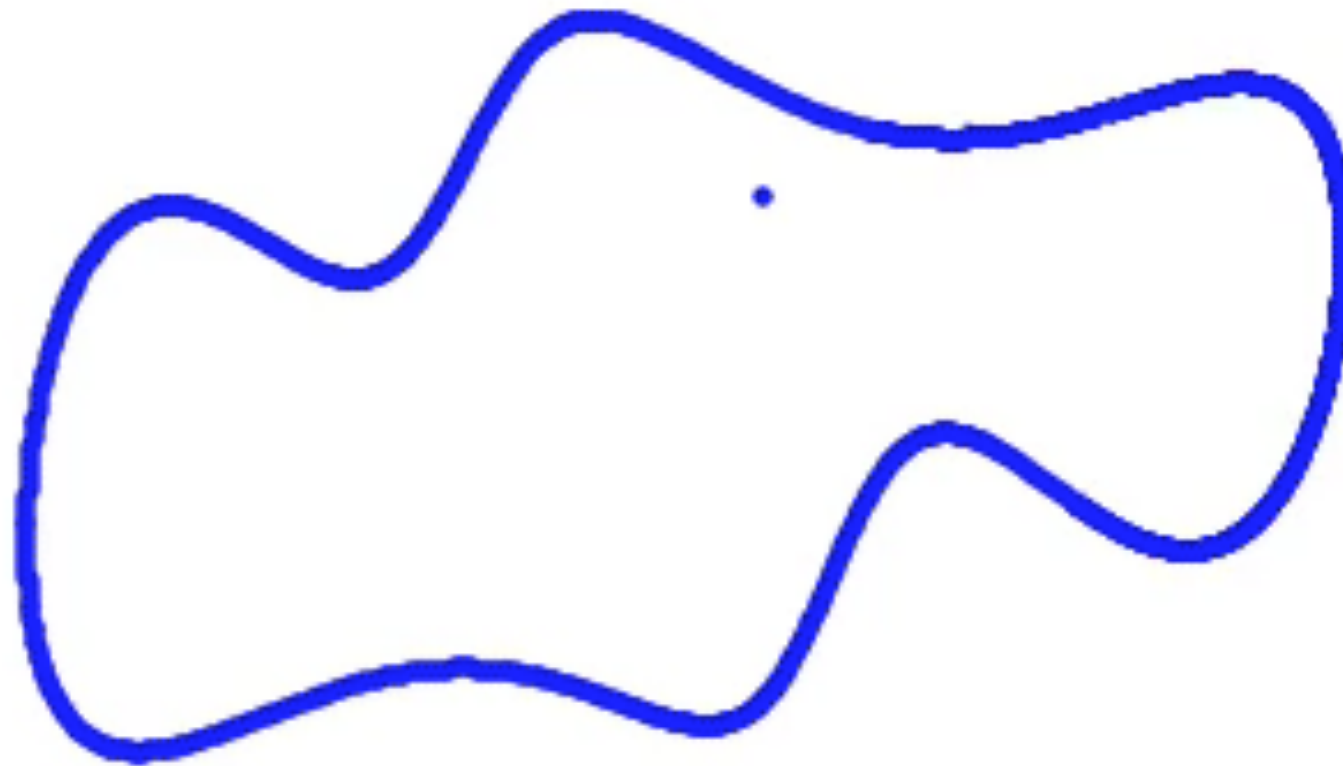
$$(50\sin(t) + 30\sin(2t) + 5\cos(3t) - 5\cos(5t), -6\cos(t) - 50\sin(t) + 10\sin(2t) - 2\sin(3t))$$



Anna's happy hippo

#parameters for the hippo:

```
parameters = [50 - 15j, 5 + 2j, -10 - 10j, -14 - 60j, -15 + 30j]
```



$$f(t) = \sum_{k=0}^5 \left(A[k] \cos(kt) + B[k] \sin(kt) \right)$$

$$(50\sin(t) + 5\sin(2t) - 10\cos(3t) - 14\cos(5t), -60\cos(t) - 15\sin(t) + 2\sin(2t) - 10\sin(3t))$$



DEDA Digital Economy & Decision Analytics

Cathy Yi-Hsuan Chen

Wolfgang Karl Härdle

Ladislaus von Bortkiewicz Professor of Statistics

C.A.S.E.-Center for Applied Statistics and
Economics

International Research Training Group

Humboldt-Universität zu Berlin

lvb.wiwi.hu-berlin.de

www.case.hu-berlin.de

irtg1792.hu-berlin.de

