Simple
& Brilliant | DSci

Field of Data Science

Paul Julitz

Notizen

03.01.2020

Inhaltsverzeichnis

Teil I

Artifical Intelligence

Kapitel 1

Azure ML Usage

1 Azure ML Python Development

1.1 Understanding ipykernel

1.1.1 Overview Python Interpreter

Reference: Using the Python Interpreter, What Is a Python Interpreter?

High-Level Language The Python language is a high-level programming language. This means, that the programming language is more similar to human language then to machine language, which is written in strings of bits - ones and zeros. The advantage of it is that it is better understood by humans, but the same can not be said for the machines. Writing commands (instruction) in lines of zeros and ones is more difficult, that using higher level concepts. The gap between this, gets closes by the **Python Interpreter**.

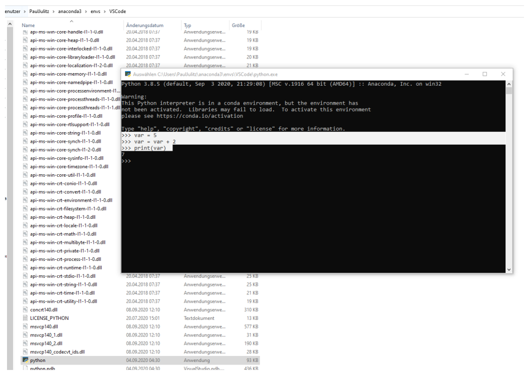
The Python Interpreter reads the command written by the Python programmer, evaluates them, and returns the output. If the *python.exe* application/ interpreter is opened, either through the command-line interpreter (cmd) , for example

```
$ python3.12
///Python 3.12 (default, April 4 2022, 09:25:04)
///[GCC 10.2.0] on linux
///Type "help", "copyright", "credits" or "license" for more
    information.
>>>
```

the commands

```
>>> the_world_is_falt = True
>>> if the_world_is_flat:
    print("Be careful not to fall off!")
/// Be careful not to fall off!
```

or by opening the application directly.



```
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> var = 5
>>> var = var + 2
>>> print(var)
7
>>>
```

Abbildung 1.1: Using the command line with the python application/ interpreter

In both cases, the Python Interpreter receives the commands and executes them. In the case where the Integrated development environment (IDE) is to write a the Python script (aka. module or application), the scripts gets passed to Python Interpreter.

Difference between interpreter and a compiler Both the interpreter and compiler transforms the source code into binary machine code. The difference arise through the way they are doing it differently: The interpreter translate the source code one statement at a time. The compiler on the other hand first scans the entire programm and then translate the whole program into machine code. For more detail, see What Is a Python Interpreter? or Best Python Compiler

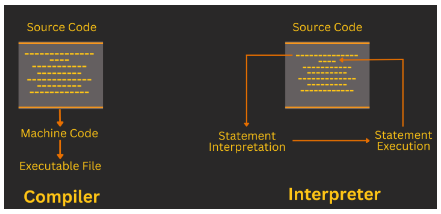


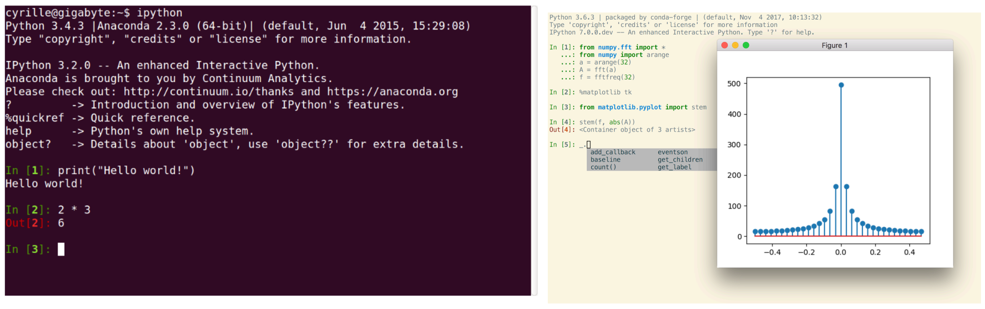
Abbildung 1.2: Example Compiler and Interpreter in Java

1.1.2 ipykernel (Jupyter Notebook)

Reference: Difference Kernel and Interpreter, IPython vs Python in Python, reddit: difference between a Python Kernel (as in Jupyter notebooks) and a Python interpreter (like in PyCharm)?, jupyter kernel is an interface to the Python interpreter.

IPython (Python3) To understand the Kernel (IPython/ Jupyter) it is helpful to understand *IPython (Notebook)*. The Python Interpreter passes command to it. This only allows commands for Python.

IPyhton creates an interactive command line terminal for Python.



```
cyrille@gigabyte:~$ ipython
Python 3.4.3 |Anaconda 2.3.0 (64-bit)| (default, Jun  4 2015, 15:29:08)
Type "copyright", "credits" or "license" for more information.

IPython 3.2.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print("Hello world!")
Hello world!

In [2]: 2 * 3
Out[2]: 6

In [3]: 
```

```
Python 3.6.3 | packaged by conda-forge | (default, Nov  4 2017, 10:13:32)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.0.0.dev -- An enhanced Interactive Python. Type '?' for help.

In [1]: from numpy.fft import *
   ...: from numpy import arange
   ...: a = arange(32)
   ...: A = fft(a)
   ...: f = fftfreq(32)

In [2]: %matplotlib tk

In [3]: from matplotlib.pyplot import stem

In [4]: stem(f, abs(A))
Out[4]: <Container object of 3 artists>

In [5]: _.
          add_callback      eventson
          baseline          get_children
          count()           get_label
```

Abbildung 1.3: IPython interactive command-line terminal

Jupyter Notebook With the reorganization of *IPython*, the new tool **Notebook** has been created. Under the project name **Jupyter**. This Jupyter Notebook (Web Interface) is a web interface for Python. It has the same interactive interface kept. Being a web-interface, it can integrate with many of the existing web libraries for data visualization.

The concept of a *kernel* comes into play as the engine behind the web interface. The *IPython* is now the backend with the Kernel (IPython/ Jupyter) for Python. The Kernel (IPython/ Jupyter) with the advent of Jupyter Notebook (Web Interface) is able to handel *markdown* and LaTeXtext input.

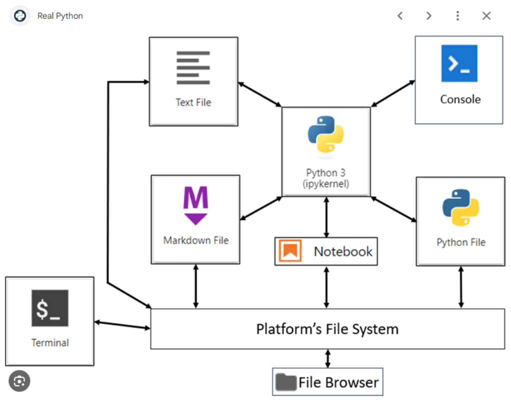


Abbildung 1.4: JupyterLab for an Enhanced Notebook, Jupyter Lab

Note: The interaction with the Kernel (IPython/ Jupyter) is done through the Jupyter Notebook (Web Interface). With the latest tool: **Jupyter Lab**, interaction with separate files is possible.

"Inside" the Kernel (IPython/ Jupyter) lives the Python Interpreter. Another way of saying this is that the Kernel (IPython/ Jupyter) is the interface to the Python Interpreter.

Note: *Jupyter Lab* allows for a more interactive and simultaneous way to code.

1.1.3 Interacting with different (language) Kernels

Theoretical Multi languages Kernel As said before the Kernel (IPython/ Jupyter) is for interacting with the Python Interpreter and the other functionality of a Jupyter Notebook (Web Interface).

The community around, *Juypter Notebook*, the application, developed more Kernel (General Jupyter) for the Jupyter Notebook (Web Interface). Those Kernel (General Jupyter) allows to interact with different languages like *Ruby, Scale, R*.

It would be possible (Unclear how) to create a Kernel (General Jupyter) which allows to interact with all those languages in one Jupyter Notebook (Web Interface). The current research for this chapter could not found one. For example the Kernel (IPython/ Jupyter) allows us to interact for example with Python, Markdown, Shell Script. This does not conclude for example Structured Query Language (SQL).

VSCode Compute Cells Confusion If a Jupyter Notebook (Web Interface) is open, for each cell it is possible to select the language for this cell. This those two things. First it changes the *IntelSence* syntax highlights. Secondly, it provides the Kernel (General Jupyter) information about the lanuage which is used in this cell.

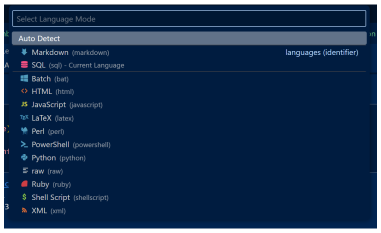


Abbildung 1.5: Cell Language Mode selection.

However, this <u>does not mean</u> that all languages are supported by the selected Kernel (General Jupyter).

Azure Data Studio In Azure Data Studio you can connect to different Kernel (General Jupyter), Azure Data Studio:

- SQL Kernel,
- PySpark3 and PySpark Kernel,
- Spark Kernel,
- Python Kernel (for local development)

1.1.4 VEN and ipykernel

Spinning Up Jupyter Notebook with Anaconda

- First a conda ven gets created.
- The Jupyter Notebook "application" gets installed

This in turn will install the package for the web interface, *IPython* and the default Kernel (IPython/ Jupyter) for Python.

```
pip install jupyter
```

Listing 1.1: pip commands to get ready for Jupyter Notebok

This command will import all dependence to work with a Kernel (IPython/ Jupyter). An example of package environment see below.

```
PS C:\Users\PaulJulitz\iCloudDrive\TexMaker\GitHub_Notizen_DSci\
    Notizen_DSci> conda list
# packages in environment at C:\Users\PaulJulitz\anaconda3\envs\VSCode:
#
# Name                    Version                   Build  Channel
...
azure-common              1.1.28                   pypi_0    pypi
azure-core                1.27.1           py38haa95532_0
azure-identity            1.12.0                   pypi_0    pypi
azure-keyvault-secrets    4.6.0                    pypi_0    pypi
azureml                   0.2.7                    pypi_0    pypi
...
ipykernel                 5.3.4            py38h5ca1d4c_0
ipython                   7.27.0           py38hd4e2768_0
ipython_genutils          0.2.0              pyhd3eb1b0_1
...
jupyter_client            7.0.1              pyhd3eb1b0_0
jupyter_core              4.8.1            py38haa95532_0
jupyter_server            1.4.1            py38haa95532_0
jupyterlab                3.2.1              pyhd3eb1b0_1
jupyterlab_pygments       0.1.2                      py_0
jupyterlab_server         2.8.2              pyhd3eb1b0_0
...
```

Listing 1.2: Example Jupyter Notebook conda ven

Multiple Kernel (Python) If the setup is done through the Anaconda interface (conda). The complete installation is done by Anaconda if the IDE Jupyter Notebook (Web Interface) is installed. By default *IPython* is installed with the Kernel (IPython/ Jupyter).

The command for the installation of the Kernel (IPython/ Jupyter) is

```
pip install ipykernel
```

Listing 1.3: pip to install ipykernel

For example to use Scala, Link:

```
# Install the package
pip install spylon-kernel
# To select the kernel in the notebook, create a kernel spec
python -m spylon_kernel install
# Start Jupyter Notebook
ipython notebook
```

Listing 1.4: Using Scala for a Notebook

In the notebook the kernel can be selected. The command is to list the available kernels is

```
$ jupyter kernelspec list
>>>>Available kernels:
>>>>python2.7        /Users/jakevdp/.ipython/kernels/python2.7
>>>>python3.3        /Users/jakevdp/.ipython/kernels/python3.3
>>>>python3.4        /Users/jakevdp/.ipython/kernels/python3.4
>>>>python3.5        /Users/jakevdp/.ipython/kernels/python3.5
>>>>python2          /Users/jakevdp/Library/Jupyter/kernels/python2
>>>>python3          /Users/jakevdp/Library/Jupyter/kernels/python3
```

Listing 1.5: pip to install ipykernel

See, Running Multiple Kernel for understand how to select a Kernel (General Jupyter) throught the command line.

To see which Kernel (General Jupyter) is running, run the following code

```python
import sys
print(sys.executable)
print(sys.version)
print(sys.version_info)
```

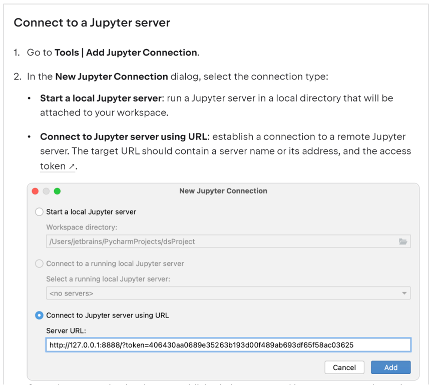
1.1.5 Connecting a Notebook to a Compute Ressources

To use a Jupyter Notebook (Web Interface), the web interface creates either a

- local server,
- or provides the possibility to connect to a Jupyter hosted server.

Note: The local server is still your own machine.

The packages for connecting to a server are installed in the ven. The ßpinningüp is basically the configuration to either a local port on your machine or to a hosted server. The former is used for computing.



Connect to a Jupyter server

1. Go to **Tools | Add Jupyter Connection**.
2. In the **New Jupyter Connection** dialog, select the connection type:
   - **Start a local Jupyter server**: run a Jupyter server in a local directory that will be attached to your workspace.
   - **Connect to Jupyter server using URL**: establish a connection to a remote Jupyter server. The target URL should contain a server name or its address, and the access token ↗.

New Jupyter Connection

Start a local Jupyter server

Workspace directory: /Users/jetbrains/PycharmProjects/dsProject

Connect to a running local Jupyter server

Select a running local Jupyter server: <no servers>

[✓ Connect to Jupyter server using URL]

Server URL: http://127.0.0.1:8888/?token=406430aa0689e35263b193d00f489ab693df65f58ac03625

Abbildung 1.6: Mac Setup of Jupter Notebook Server

If the notebook should be connected to the compute cluster, this is done by a SHH Tunnel.

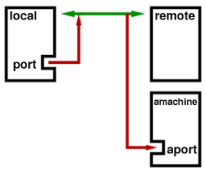


Abbildung 1.7: Tunnel Trafic, SHH Tunnel

1.2 Basics Azure ML SDK

2 azureml-core (Python SDK)

2.1 Connect to Workspace

A workspace is a resource, which is tied (child) to a subscription and a resource group. A workspace links the different object to run a Machine Learning (ML) model: Experiment, Training, Deployment. A workspace comes with a Software Development Kit (SDK) to interactive with.

To connect to the workspace, the constructor requires different parameters

```
Workspace(
subscription_id, resource_group, workspace_name,
auth=None,
_location=None,
_disable_service_check=False,
_workspace_id=None,
sku='basic',tags=None, _cloud='AzureCloud'
)
```

Kapitel 2

Application: Person Course

1 Creating a Experiment

Or simpler: Just import jupter notebook

2 Hyperparameter Tuning

2.1 Batch vs. Mini Batch Size gradient descent

The training set get's split into smaller *batches*.

3 Excurse

3.1 Gradient Descent

3.1.1 Optimization Using Gradient Descent

We now consider the problem of solving for the *minimum* of a real-values function

$$\min_x f(x). \tag{3.1}$$

We assume

- $f : \mathbb{R}^4 \to R$ captures the machine learning problem at hand,
- f is differentiable,
- we are unable to analytically find the solution in closed form.

The *gradient descent* takes steps a proportionaly (γ) to the negative of the gradient of the function ∇f at the current point x_0:

$$\nabla x_1 = x_0 - \gamma((\nabla f)(x_0))^T. \tag{3.2}$$

At the point x_0 the function $f(x_0)$ degresses fastes if on moves from x_0 in the direction of the negative gradient $-\gamma((\nabla f)(x_0))^T$ of f at x_0. From this follows: For a small *step-size*[1] $\gamma \geq 0$ $f(x_1) \geq f(x_0)$.

[1]In Azure ML is called *learning-rate*

The Simple Gradient Descent Algorithm To find the local optimum f_{x_*} we start with a inial guess x_0 of the parameter we wish to optimize. Then we iterate according to

$$\nabla x_{i+1} = x_i - \gamma_i((\nabla f)(x_i))^T. \tag{3.3}$$

For a suitable step-size γ_i, the sequence $f(x_0) \leq f(x_1) \leq ...$ converges to a *local* minimum.

Graphical Gradient Descent Only if the step-size γ is a big, that the $f(x_{i+1}$ overshots the local minimum, the functional value $y = f(x_i)$ converges to the local minimum.

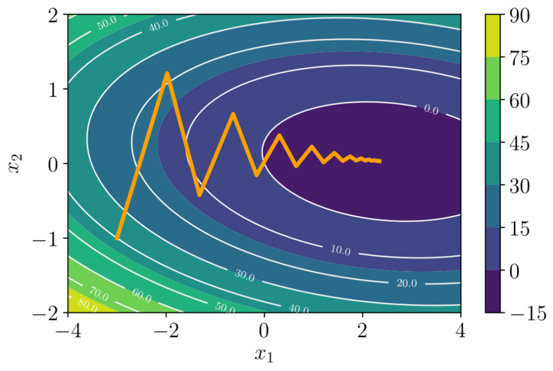


Abbildung 2.1: Graph shows the gradient descent on a two dimensional quadratic surface.

The zig-zag of the graph

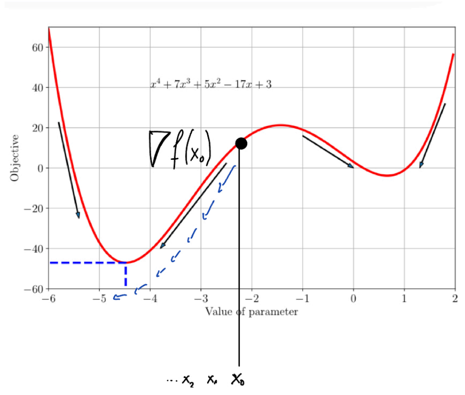


Abbildung 2.2: Example of a one dimentional function. Negative gradient descent is indicated by the arrows.

3.1.2 Solving a Linear Equation System

In Form Ax=b

Teil II

Anhang

Anhang A

Abkürzungsverzeichnis

Symbole

.ipynb Jupyter Notebook file format which in interoperable accross many platforms. The name also referes to the user-facing web interface called Jupyter Notebook.. *Glossar:* Jupyter Notebook (Web Interface)
.json JavaScript Object Notation. *Glossar:* JSON

C

cmd command-line interpreter. 4

E

EC2 Amazon Elastic Computing Cloud. *Glossar:* Amazon Elastic Computing Cloud

H

HTTP Hypertext Tranfer Protokoll. *Glossar:* HTTP

I

IDE Integrated development environment. 5, 8

J

JSON JavaScript Object Notation. *Glossar:* JSON

M

ML Machine Learning. 10

N

Na Not available. *Glossar:* Na (R)
NaN Not a Number. *Glossar:* NaN

O

O(OKR) Objective form the OKR logic. *Glossar:* Objective (OKR)
ODBC Open Database Connectivity (Connection). *Glossar:*

S

SDK Software Development Kit. 10
SQL Structured Query Language. 7, *Glossar:* SQL

U

URL Uniform Resource Locator. *Glossar:* URL

V

ven Virtuel Environment first. 7, 9