

Testing

(Ziti tools Vorlesung, 2026)

URLs Dokument:

<https://einc.eu/f/t26>

Übersicht

- Jupyter Notebook
- Test driven development (Beispiel)
- Gitlab + Test-setup in gitlab
- Übung mit gitlab, git (mit CI in gitlab)
- (EBRAINS account setup)
- End-to-end testing mit selenium an einem Beispiel in der EBRAINS RI

Vorbereitung: Python -> Jupyter notebook

- Mit Jupyter notebook Python interaktiv ausführen
- Gut geeignet für kleine Auswertungen, Plots für die Arbeit oder zum Ausprobieren
- Nicht wirklich geeignet für große Programme (dafür besser mit einer IDE arbeiten (disclaimer: ich selber arbeite normalerweise in Java))

Starting Jupyter notebook environment (1)

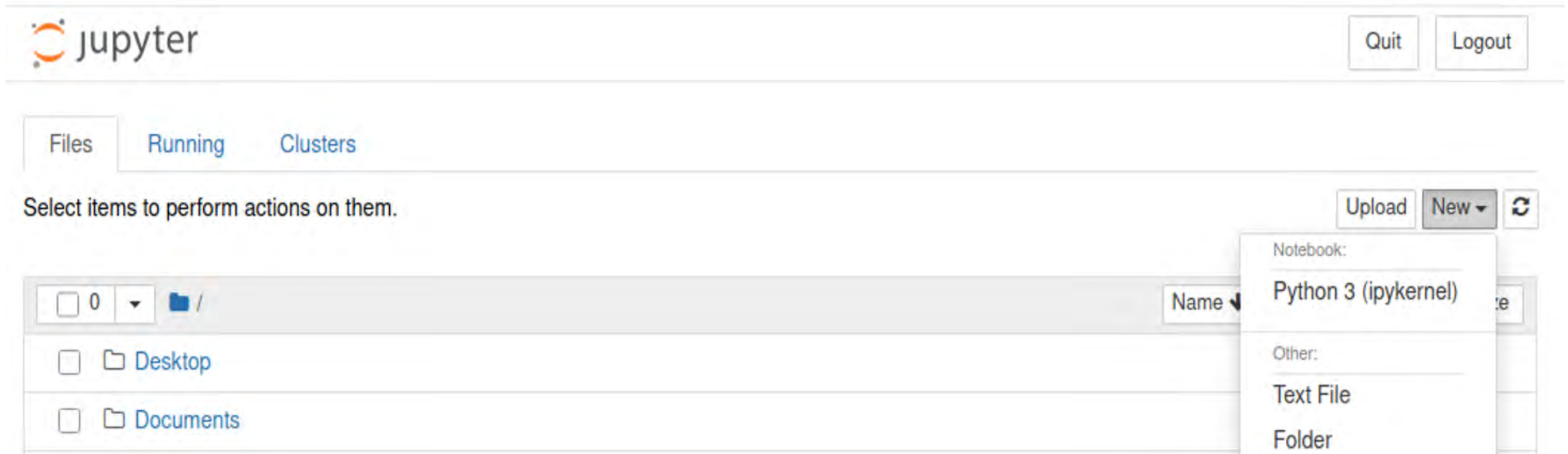
- Loggen Sie sich auf einem der zitipool Rechner ein und starten Sie eine shell (“terminal”)
- Gehen Sie einmal in das Verzeichnis /shares/ziti-opt/ (damit es gemounted wird und Sie Pfad-Vervollständigung in der Shell nutzen können)
- Dann zurück in Ihr Home-Directory
- Starten Sie eine shell in einem apptainer image:
apptainer shell /shares/ziti-opt/software/nca/jul25.sif
- Überprüfen Sie, ob Sie sich in der apptainer shell in Ihrem home-Directory befinden: pwd

Jupyter notebook environment (2)

- Starten Sie einen Jupyter Prozess in Ihrer apptainer session:
jupyter notebook

Es wird eine URL angezeigt, unter der Sie Ihren Server erreichen können (und es sollte automatisch ein Browser für diese URL gestartet werden)

Im Browser sehen Sie jetzt eine Liste der Dateien in Ihrem home directory



Mit New > Python 3 (ipykernel) können Sie ein neues “Notebook” anlegen

jupyter Untitled1 Last Checkpoint: 6 minutes ago (autosaved)



Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted



Python 3 (ipykernel) ○

Save + Copy Paste Undo Redo Run Stop Restart Code

In []:

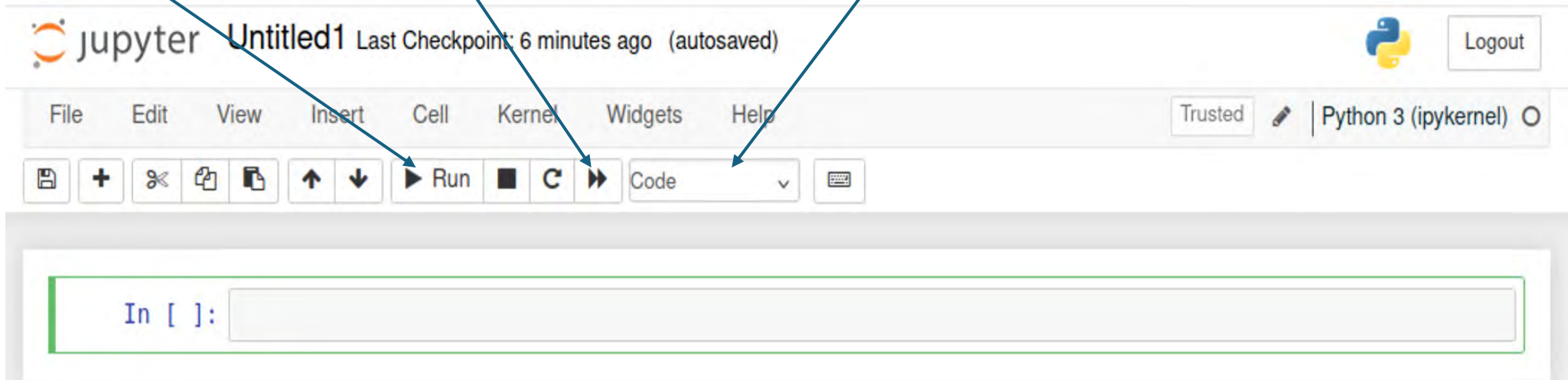
“Kernel” info

Zelle

Kernel neu starten und alle Zellen ausführen

Zelle ausführen

Typ der Zelle (Code, Markdown, ...)



“Kernel” info

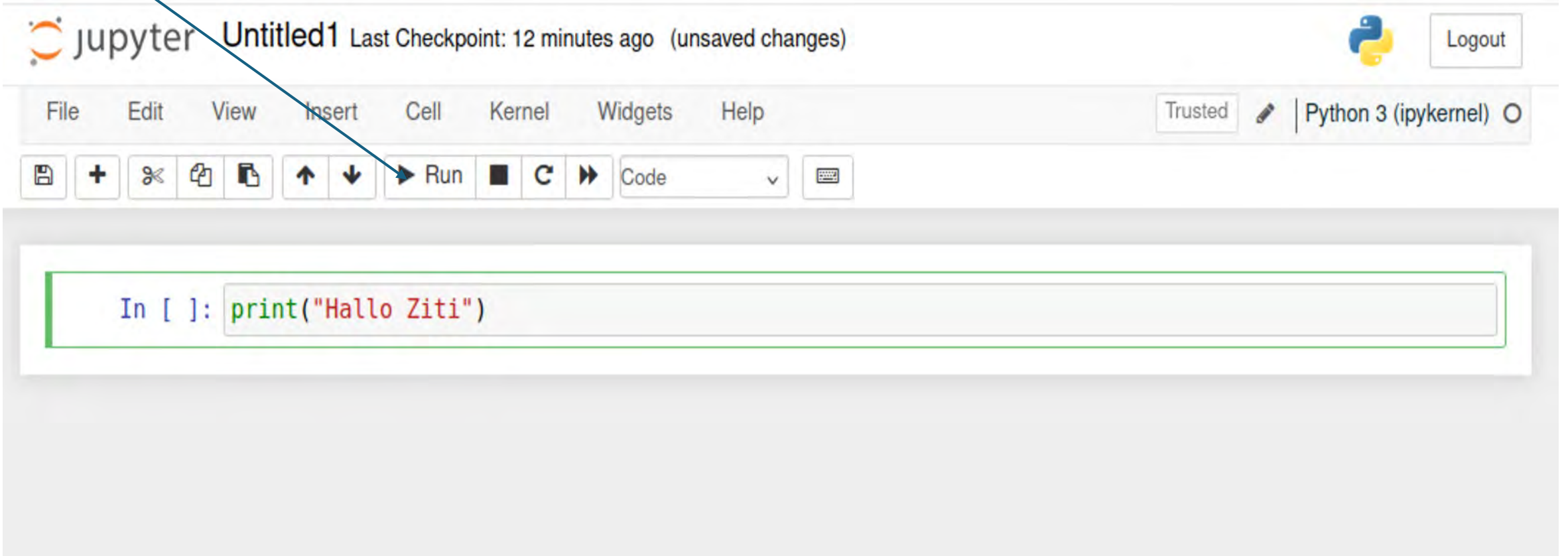
Zelle

Ausführungs-Indikator:

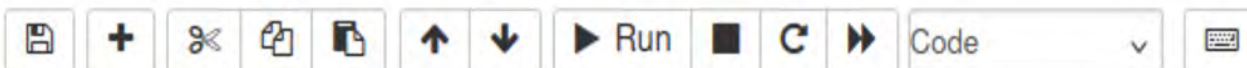
* = wird ausgeführt bzw. wird gestartet, wenn die vorherige Zelle fertig ausgeführt wurde

Zahl = fortlaufender Zähler der Ausführungen

Zelle ausführen klicken (oder SHIFT ENTER)



The screenshot displays the Jupyter Notebook interface. At the top, the header shows the Jupyter logo, the file name 'Untitled1', and the status 'Last Checkpoint: 12 minutes ago (unsaved changes)'. On the right, there is a Python logo and a 'Logout' button. Below the header is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. To the right of the menu bar, it indicates 'Trusted' status and the active kernel 'Python 3 (ipykernel)'. A toolbar below the menu bar contains icons for saving, adding, deleting, copying, pasting, moving up/down, running (highlighted by a blue arrow), interrupting, resetting, and stepping through code. The main area shows a code cell with the prompt 'In []:' followed by the code `print("Hallo Ziti")`.

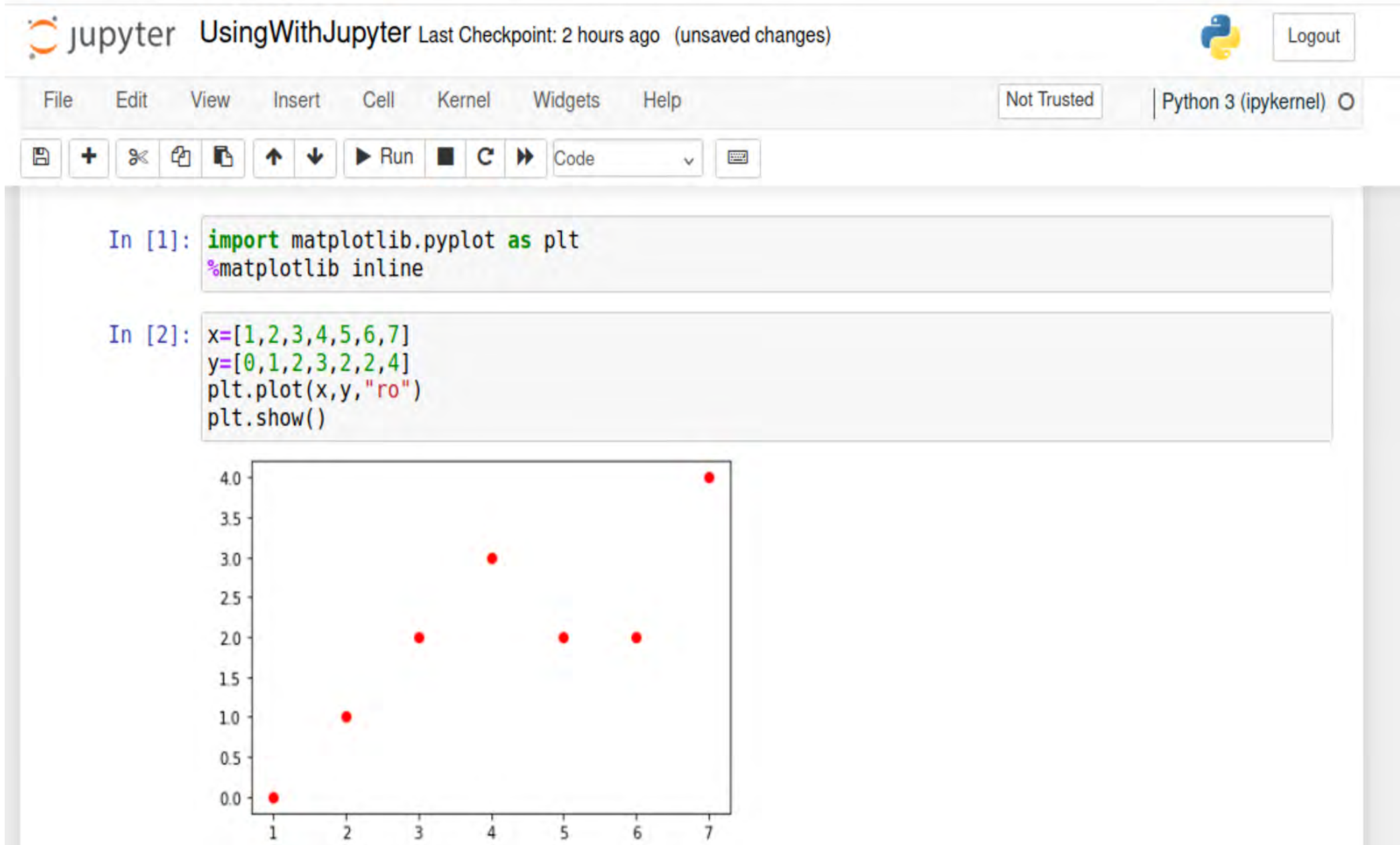


```
In [1]: print("Hallo Ziti")
```

```
Hallo Ziti
```

```
In [ ]:
```

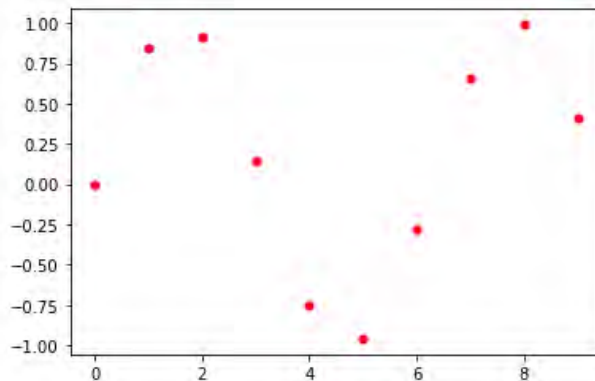
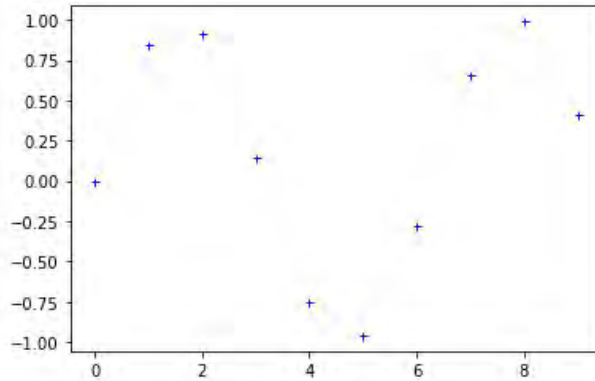
Jupyter notebook und matplotlib



Matplotlib – array per Hand

```
In [1]: import matplotlib.pyplot as plt  
import math  
import numpy as np  
%matplotlib inline
```

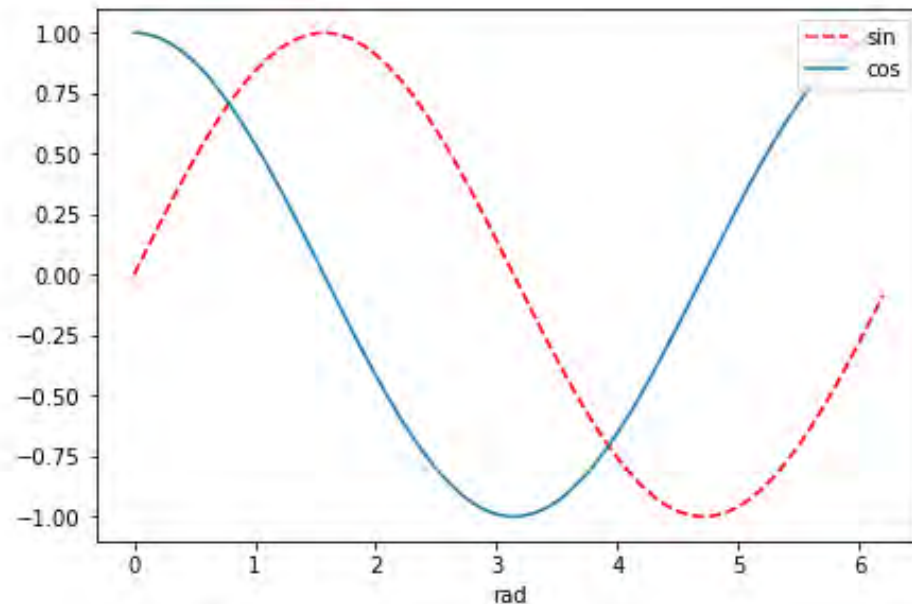
```
In [2]: x=range(0,10)  
y=[]  
for i in x:  
    y.append(math.sin(i))  
plt.plot(x,y,"b+-",linewidth=0.1, markersize=5)  
plt.show()  
plt.plot(x,y,"ro-",linewidth=0.1, markersize=5)  
plt.show()
```



Matplotlib – array per numpy

```
In [1]: import matplotlib.pyplot as plt  
import math  
import numpy as np  
%matplotlib inline
```

```
In [4]: x=np.arange(0,2*math.pi,0.1)  
y=np.sin(x)  
y2=np.cos(x)  
fig, ax0 = plt.subplots(1, 1, sharex=True, constrained_layout=True)  
  
ax0.plot(x, y, "r--", label='sin')  
ax0.plot(x, y2, label='cos')  
ax0.legend(loc='upper right')  
ax0.set_xlabel('rad')  
plt.show()
```

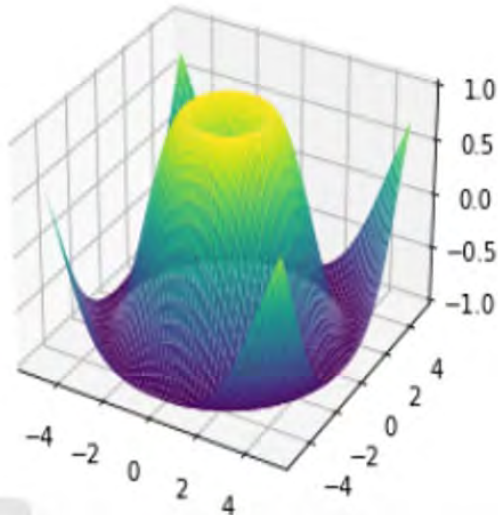


Kann das was?

```
In [1]: import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

```
In [2]: fig = plt.figure()
ax = fig.add_subplot(projection='3d')
stepsize=0.01
X = np.arange(-5, 5, stepsize)
Y = np.arange(-5, 5, stepsize)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
ax.plot_surface(X, Y, Z, cmap='viridis')
```

```
Out[2]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0xffff8c5c7a30>
```





File

Edit

View

Insert

Cell

Kernel

Widgets

Help

Not Trusted



Python 3 (ipykernel)



Run



Markdown



Funktionen in Python

- * kann man mit Parametersn implementieren (auch mit optionalen Paramtern, die default-Werte haben)
- * können dann im weiteren Programverlauf aufgerufen werden

```
In [5]: def printMeTwice(text):  
        print(text)  
        print(text)
```

```
In [ ]: printMeTwice("Hallo")
```

```
In [ ]:
```

```
In [ ]:
```



Run



Code



Funktionen in Python

- kann man mit Parametersn implementieren (auch mit optionalen Paramtern, die default-Werte haben)
- können dann im weiteren Programverlauf aufgerufen werden

```
In [7]: def printMeTwice(text):  
        print(text)  
        print(text)
```

```
In [8]: printMeTwice("Hallo")
```

```
Hallo  
Hallo
```

```
In [ ]:
```

Klassen in Python

```
In [1]: class EineKlasse:
        info="Wert aus der Klasse"

        def __init__(self,wert):
            self.info=wert

        def eineFunktion(self,text):
            print(text)
            print(self.info)
```

```
In [2]: a=EineKlasse("mit Wert")
        a.eineFunktion("ein Text")
```

```
ein Text
mit Wert
```

Testing: test driven development

Grundgedanke:

- Zuerst den Test schreiben (der dann natürlich erst mal fehl schlägt, denn die Funktion gibt es ja noch nicht)
- Erst dann die Funktion implementieren – und zwar nur so weit, so daß der Test erfüllt wird.

Erstellen einer minimalen Klasse “ClassToTest” und Erstellen einer Instanz der Klasse (hier: ctt)

```
In [1]: class ClassToTest:  
        def one(self):  
            return 1
```

Klasse definiert + eine
Methode (one) definiert

```
In [2]: ctt=ClassToTest()
```

Instanz der Klasse erstellt

```
In [3]: ctt.one()
```

Methode der Instanz
aufgerufen

```
Out[3]: 1
```

Jetzt eine Klasse (mit unittest.TestCase als parent class) erstellen. Methoden, die mit test_ beginnen werden darin vom unittest framework automatisch ausgeführt

Da es die Methode minus in der ClassToTest noch gar nicht gibt schlägt der Test fehl.

```
In [14]: class ClassToTest:
         def one(self):
             return 1
```

```
In [15]: ctt=ClassToTest()
```

```
In [16]: import unittest

         class TestClass(unittest.TestCase):

             def test_minus(self):
                 self.assertEqual(ctt.minus(1,0.1,1),0.9)

         unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
test_minus (__main__.TestClass) ... ERROR
```

```
=====
ERROR: test_minus (__main__.TestClass)
-----
```

```
Traceback (most recent call last):
```

```
  File "/tmp/ipykernel_26535/3800936997.py", line 6, in test_minus
    self.assertEqual(ctt.minus(1,0.1,1),0.9)
```

```
AttributeError: 'ClassToTest' object has no attribute 'minus'
```

```
-----
Ran 1 test in 0.003s
```

```
FAILED (errors=1)
```

Minimalste Implementation (nicht sinnvoll, aber minimal...)

```
In [23]: class ClassToTest:
         def minus(self, start, step, count):
             return 0.9
```

```
In [24]: ctt=ClassToTest()
```

```
In [25]: import unittest

         class TestClass(unittest.TestCase):

             def test_minus(self):
                 self.assertEqual(ctt.minus(1,0.1,1),0.9)

         unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
test_minus (__main__.TestClass) ... ok
```

```
-----
Ran 1 test in 0.002s
```

```
OK
```

```
Out[25]: <unittest.main.TestProgram at 0xfffffa3e88e20>
```

Die minimalste Methode lässt den Test gelingen. Den Fehler in der Methode konnte dieser Test nicht finden. Also muß ein weiterer Test her....hier mit assertTrue(...)

```
In [23]: class ClassToTest:
         def minus(self, start, step, count):
             return 0.9
```

```
In [24]: ctt=ClassToTest()
```

```
In [29]: import unittest

         class TestClass(unittest.TestCase):

             def test_minus(self):
                 self.assertEqual(ctt.minus(1,0.1,1),0.9)
                 self.assertTrue(ctt.minus(1,0.1,2)==0.8)

         unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
test_minus (__main__.TestClass) ... FAIL
```

```
=====
FAIL: test_minus (__main__.TestClass)
-----
```

```
Traceback (most recent call last):
```

```
  File "/tmp/ipykernel_26535/40440789.py", line 7, in test_minus
```

```
    self.assertTrue(ctt.minus(1,0.1,2)==0.8)
```

```
AssertionError: False is not true
```

```
-----
Ran 1 test in 0.003s
```

```
FAILED (failures=1)
```

Jetzt eine Methode minus(), die auch wirklich rechnet -- Test gelingt!

```
In [30]: class ClassToTest:
        def minus(self, start, step, count):
            for i in range(0, count):
                start -= step
            return start
```

```
In [31]: ctt=ClassToTest()
```

```
In [34]: import unittest

        class TestClass(unittest.TestCase):

            def test_minus(self):
                self.assertEqual(ctt.minus(1,0.1,1),0.9)
                self.assertTrue(ctt.minus(1,0.1,2)==0.8)

        unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
test_minus (__main__.TestClass) ... ok
```

```
-----
Ran 1 test in 0.001s
```

```
OK
```

Was wird jetzt passieren – mit dem 3. Test?

```
In [ ]: class ClassToTest:
        def minus(self, start, step, count):
            for i in range(0, count):
                start -= step
            return start
```

```
In [ ]: ctt=ClassToTest()
```

```
In [ ]: import unittest

        class TestClass(unittest.TestCase):

            def test_minus(self):
                self.assertEqual(ctt.minus(1, 0.1, 1), 0.9)
                self.assertTrue(ctt.minus(1, 0.1, 2) == 0.8)
                self.assertTrue(ctt.minus(1, 0.1, 10) == 0)

unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
In [1]: class ClassToTest:
        def minus(self, start, step, count):
            for i in range(0, count):
                start -= step
            return start
```

```
In [2]: ctt=ClassToTest()
```

```
In [3]: import unittest

        class TestClass(unittest.TestCase):

            def test_minus(self):
                self.assertEqual(ctt.minus(1,0.1,1),0.9)
                self.assertTrue(ctt.minus(1,0.1,2)==0.8)
                self.assertTrue(ctt.minus(1,0.1,10)==0)

unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
test_minus (__main__.TestClass) ... FAIL
```

```
=====
FAIL: test_minus (__main__.TestClass)
```

```
-----
Traceback (most recent call last):
```

```
  File "/tmp/ipykernel_28826/1344471396.py", line 8, in test_minus
    self.assertTrue(ctt.minus(1,0.1,10)==0)
AssertionError: False is not true
```

```
-----
Ran 1 test in 0.003s
```

```
FAILED (failures=1)
```

Bessere Fehlermeldung mit assertEquals()

```
In [1]: class ClassToTest:
        def minus(self, start, step, count):
            for i in range(0, count):
                start -= step
            return start
```

```
In [2]: ctt = ClassToTest()
```

```
In [5]: import unittest

        class TestClass(unittest.TestCase):

            def test_minus(self):
                self.assertEqual(ctt.minus(1, 0.1, 1), 0.9)
                self.assertTrue(ctt.minus(1, 0.1, 2) == 0.8)
                self.assertEqual(ctt.minus(1, 0.1, 10), 0)

        unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
test_minus (__main__.TestClass) ... FAIL
```

```
=====
FAIL: test_minus (__main__.TestClass)
```

```
-----
Traceback (most recent call last):
```

```
  File "/tmp/ipykernel_28826/3956267745.py", line 8, in test_minus
    self.assertEqual(ctt.minus(1, 0.1, 10), 0)
AssertionError: 1.3877787807814457e-16 != 0
```

```
-----
Ran 1 test in 0.002s
```

```
FAILED (failures=1)
```

Und nun? Einfach runden?

```
In [1]: class ClassToTest:
        def minus(self, start, step, count):
            for i in range(0, count):
                start -= step
            return round(start)
```



```
In [2]: ctt=ClassToTest()
```

```
In [3]: import unittest

        class TestClass(unittest.TestCase):

            def test_minus(self):
                self.assertEqual(ctt.minus(1, 0, 1), 1)
                #self.assertTrue(ctt.minus(1, 0.1, 2) == 0.8)
                self.assertEqual(ctt.minus(1, 0.1, 10), 0)

unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
test_minus (__main__.TestClass) ... ok
```

```
-----
Ran 1 test in 0.002s
```

```
OK
```

Aber mit einem Test, der keine Ganzzahl erwartet, fällt der round-Trick auf (und damit aus):

```
In [1]: class ClassToTest:
        def minus(self, start, step, count):
            for i in range(0, count):
                start -= step
            return round(start)
```

```
In [2]: ctt=ClassToTest()
```

```
In [5]: import unittest

        class TestClass(unittest.TestCase):

            def test_minus(self):
                self.assertEqual(ctt.minus(1,0,1),1)
                self.assertEqual(ctt.minus(1,0.1,2),0.8)
                self.assertEqual(ctt.minus(1,0.1,10),0)

        unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
test_minus (__main__.TestClass) ... FAIL
```

```
=====
FAIL: test_minus (__main__.TestClass)
```

```
-----
Traceback (most recent call last):
```

```
  File "/tmp/ipykernel_29490/1454781752.py", line 7, in test_minus
    self.assertEqual(ctt.minus(1,0.1,2),0.8)
AssertionError: 1 != 0.8
```

```
-----
Ran 1 test in 0.001s
```

```
FAILED (failures=1)
```

Was dann?

Es kommt auf die Anforderung an:

- Exakt rechnen mit `decimal.Decimal`
- Statt mit `==` zu vergleichen prüfen, ob die Abweichung “klein genug” ist

Exact rechnen (mit Decimal)

```
In [90]: from decimal import Decimal
a1=1
a2=Decimal(1)
for i in range(0,10):
    print(a1, " ", a2)
    a1=a1-0.1
    a2=a2-Decimal("0.1")
print(a1, " ", a2, " is a2 0 ?", a2==Decimal(0))
```

```
1    1
0.9   0.9
0.8   0.8
0.700000000000000001  0.7
0.600000000000000001  0.6
0.500000000000000001  0.5
0.4000000000000000013  0.4
0.3000000000000000016  0.3
0.2000000000000000015  0.2
0.1000000000000000014  0.1
1.3877787807814457e-16  0.0  is a2 0 ? True
```

```
In [1]: class ClassToTest:
        def minus(self, start, step, count):
            for i in range(0, count):
                start -= step
            return start
```

```
In [2]: ctt=ClassToTest()
```

```
In [3]: import unittest
import decimal

class TestClass(unittest.TestCase):

    def test_minus(self):
        self.assertEqual(ctt.minus(1,0,1),1)
        self.assertEqual(ctt.minus(1,0.1,2),0.8)
        self.assertEqual(ctt.minus(1,decimal.Decimal("0.1"),10),0)

unittest.main(argv=["v"], verbosity=3, exit=False)
```

```
test_minus (__main__.TestClass) ... ok
```

```
-----
Ran 1 test in 0.003s
```

```
OK
```

Test-Pyramide

- **Unit testing**

- Unit testing involves isolating your application into units and then testing the behavior of each as a function independent from external parties, databases, or configurations. Unit testing often occurs during the build period and is considered the first layer of testing.

- **Integration testing**

- Integration testing evaluates how several units are logically integrated, and how this affects the system functionality without unintended errors in the integration process. The main purpose of integration testing is to test the compliance of a system by verifying how disparate modules work together.

- **Regression testing**

- Regression testing ensures that bug fixes or other changes have not adversely affected existing functionality. Automating regression tests allows developers to quickly and efficiently identify and fix any issues that may have been introduced by code changes, ensuring that the software remains reliable and bug-free.

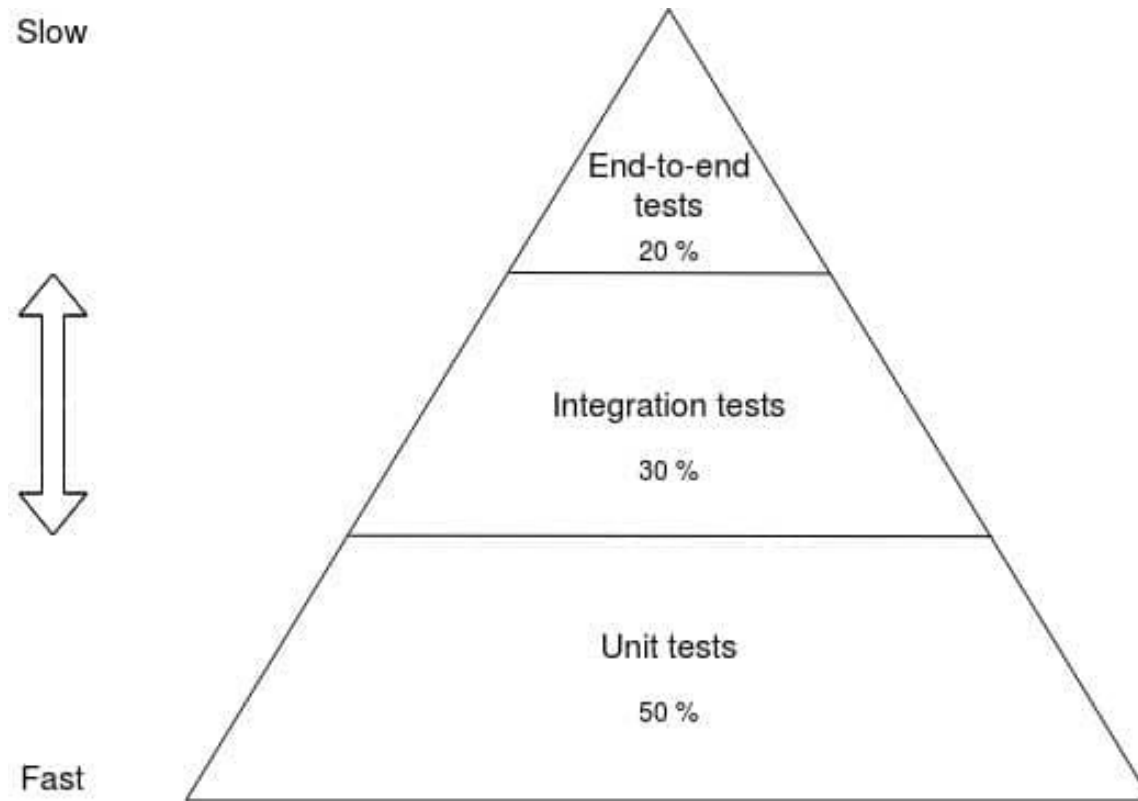
- **End-to-end testing**

- An end-to-end testing framework tests the functionality and performance of the application by simulating the user's expectations and needs from start to finish. The end goal isn't just to ensure the application validates and checks all the user's needs, but to ensure it operates and behaves at least as well as expected.

- **Exploratory testing**

- Exploratory testing is a more sophisticated software testing strategy that involves parallel learning, testing, and reviewing various functional and visual components from the user's perspective.

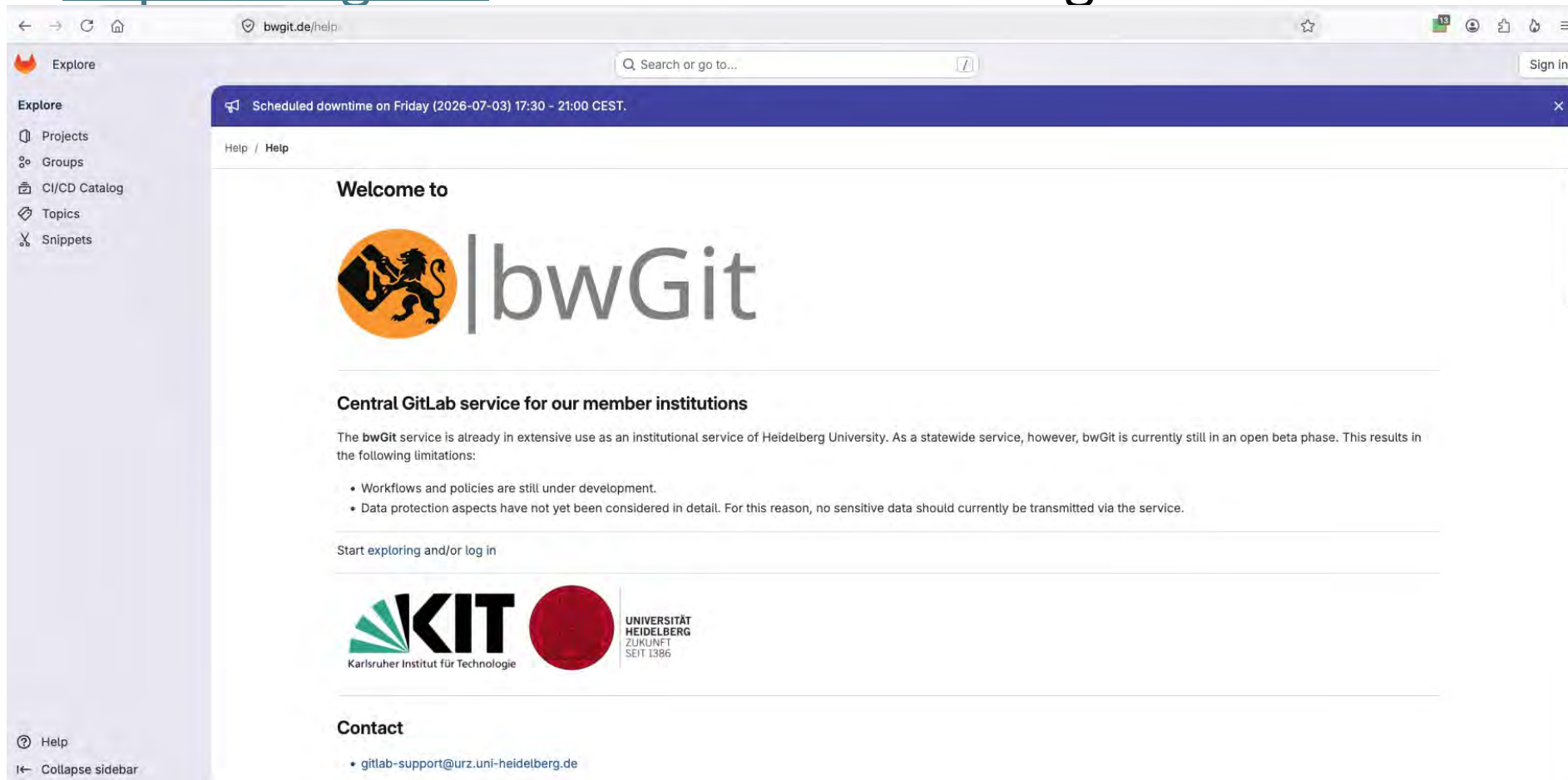
Testing: Test-Pyramide



From
<https://testdriven.io/blog/modern-tdd/>

Gitlab – Plattform für Codeentwicklung im Team

- <https://bwgit.de> – direkt mit URZ-ID login nutzbar



bwgit – setup access

The screenshot displays the bwGit web application interface. On the left is a sidebar titled "Your work" containing navigation links: Home, Projects, Groups, Work items, Merge requests, To-Do List, Milestones, Snippets, Activity, Import history, Workspaces, Environments, Operations, and Security. The main content area shows the user's home page for Björn Kindler, featuring a large heading "Big things await" and four summary cards: "Merge requests Waiting for your review Just now", "Merge requests Assigned to you Just now", "Work items Assigned to you Just now", and "Work items Authored by you Just now". On the right, a user profile dropdown menu is open for Björn Kindler (@hd-ca120), listing options: Set status, Edit profile, Preferences, and Sign out. Below the menu is a "Quick access" section with tabs for "Recently viewed" and "Projects", showing recent actions like "Merge from token upload" and "removed the failing test line. Use bra...".

bwGit

Search or go to...

Your work

- Home
- Projects
- Groups
- Work items
- Merge requests
- To-Do List
- Milestones
- Snippets
- Activity
- Import history
- Workspaces
- Environments
- Operations
- Security

Your work / Home

Björn Kindler
@hd-ca120

Big things await

Merge requests

0
Waiting for your review
Just now

Merge requests

0
Assigned to you
Just now

Work items

0
Assigned to you
Just now

Work items

0
Authored by you
Just now

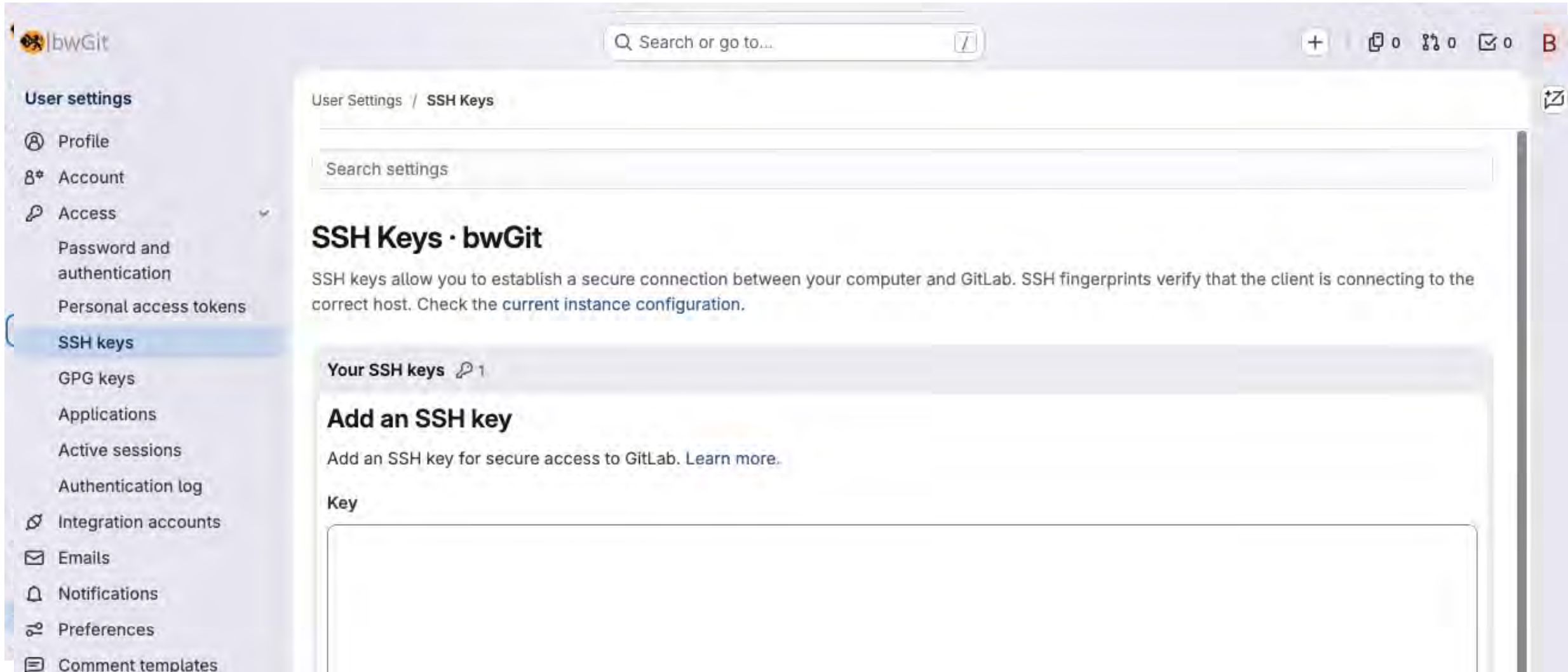
Quick access

Recently viewed Projects

Merge from token upload

removed the failing test line. Use bra...

Methode SSH Key



The screenshot shows the GitLab web interface. On the left is a sidebar with 'User settings' expanded, showing options like Profile, Account, Access, Password and authentication, Personal access tokens, SSH keys (highlighted), GPG keys, Applications, Active sessions, Authentication log, Integration accounts, Emails, Notifications, Preferences, and Comment templates. The main content area is titled 'User Settings / SSH Keys'. It includes a search bar for settings. Below this, the heading 'SSH Keys · bwGit' is followed by an explanatory paragraph: 'SSH keys allow you to establish a secure connection between your computer and GitLab. SSH fingerprints verify that the client is connecting to the correct host. Check the [current instance configuration](#).' A section titled 'Your SSH keys' with a link icon and the number '1' contains a card for 'Add an SSH key'. This card has a sub-heading 'Add an SSH key' and a description 'Add an SSH key for secure access to GitLab. [Learn more.](#)'. Below the description is a label 'Key' and a large, empty text input field for pasting the key.

bwGit

Search or go to...

User settings

- Profile
- Account
- Access
- Password and authentication
- Personal access tokens
- SSH keys**
- GPG keys
- Applications
- Active sessions
- Authentication log
- Integration accounts
- Emails
- Notifications
- Preferences
- Comment templates

User Settings / SSH Keys

Search settings

SSH Keys · bwGit

SSH keys allow you to establish a secure connection between your computer and GitLab. SSH fingerprints verify that the client is connecting to the correct host. Check the [current instance configuration](#).

Your SSH keys 1

Add an SSH key

Add an SSH key for secure access to GitLab. [Learn more.](#)

Key

Oder mit einem personal token

User settings

- Profile
- Account
- Access
 - Personal access tokens
 - SSH keys
 - GPG keys
 - Applications
 - Active sessions
 - Authentication log
- Integration accounts
- Emails
- Notifications
- Preferences
- Comment templates
- Usage quotas

Personal access tokens

You can generate a personal access token for each application you use that needs access to the GitLab API. You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Active tokens: 1 [Filter list](#)

Tokens expiring in 2 weeks: 0 [Filter list](#)

Revoked tokens: 0 [Filter list](#)

Expired tokens: 0 [Filter list](#)

State = Active X Q Expiration date v t

Name	Status	Scopes	Usage ?	Lifetime
personal test token BK	Active	read_repository, read_api, write_repository	Last used: 2 minutes ago IPs: 147.142.158.18, 129.206.224.72	in 3 weeks Jul 03, 2026

Feed token

Your feed token authenticates you when your RSS reader loads a personalized RSS feed or when your calendar application loads a personalized calendar. It is visible in those feed URLs. It cannot be used to access any other data.

Access to the project

- Nennen Sie mir Ihre URZ-ID, dann kann ich Sie zum Projekt für diese Vorlesung einladen (funktioniert eventuell erst nach dem ersten Login auf bwgit.de)

bwGit

Search or go to...

/

+

0

0

0

Your work

Home

Projects

Groups

Work items

Merge requests

To-Do List

Milestones

Snippets

Activity

Import history

Workspaces

Environments

Operations

Security

What's new

Help

Collapse sidebar

Your work / Projects

Projects

Explore projects

New project

Contributed 1

Starred 0

Personal 1

Member 1

Inactive 0

Filter or search (3 character minimum)

Q

Name

↑

B

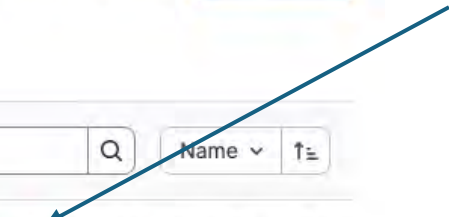
Björn Kindler / Tools2026-Testing

Owner

Testing for the tools 2026 lecture

Created 4 weeks ago

Letzes test-outcome



bwGit

Search or go to...

Project

- Tools2026-Testing
- Pinned
- Work items
- Merge requests
- Manage
- Plan
- Code
- Merge requests
- Repository**
- Branches
- Commits
- Tags
- Repository graph
- Compare revisions
- Snippets
- Locked files
- Build
- Secure
- Deploy
- Operate
- Monitor
- What's new
- Help
- Collapse sidebar

Björn Kindler / Tools2026-Testing / Repository

Files

testing tools2026-testing

Search files (*.vue, *.rb...)

tools2026-testing

Find file Code

A notebook with some matplotlib and some t...
Bjoern Kindler authored 2 days ago

6d632fda History

Name	Last commit	Last update
jupyterNotebooks	A notebook with some m...	2 days ago
public	Initialized from 'Pages/Pl...	2 years ago
python	fixed indentation and test	3 days ago
.gitlab-ci.yml	Unittest class test_main a...	3 days ago
README.md	Edit README.md for testi...	4 days ago

README.md

Example plain HTML site using GitLab Pages.

Learn more about GitLab Pages at <https://pages.gitlab.io> and the official documentation <https://docs.gitlab.com/ce/user/project/pages/>.

Table of Contents generated with DocToc

- GitLab CI
- GitLab User or Group Pages
- Did you fork this project?

Provide feedback

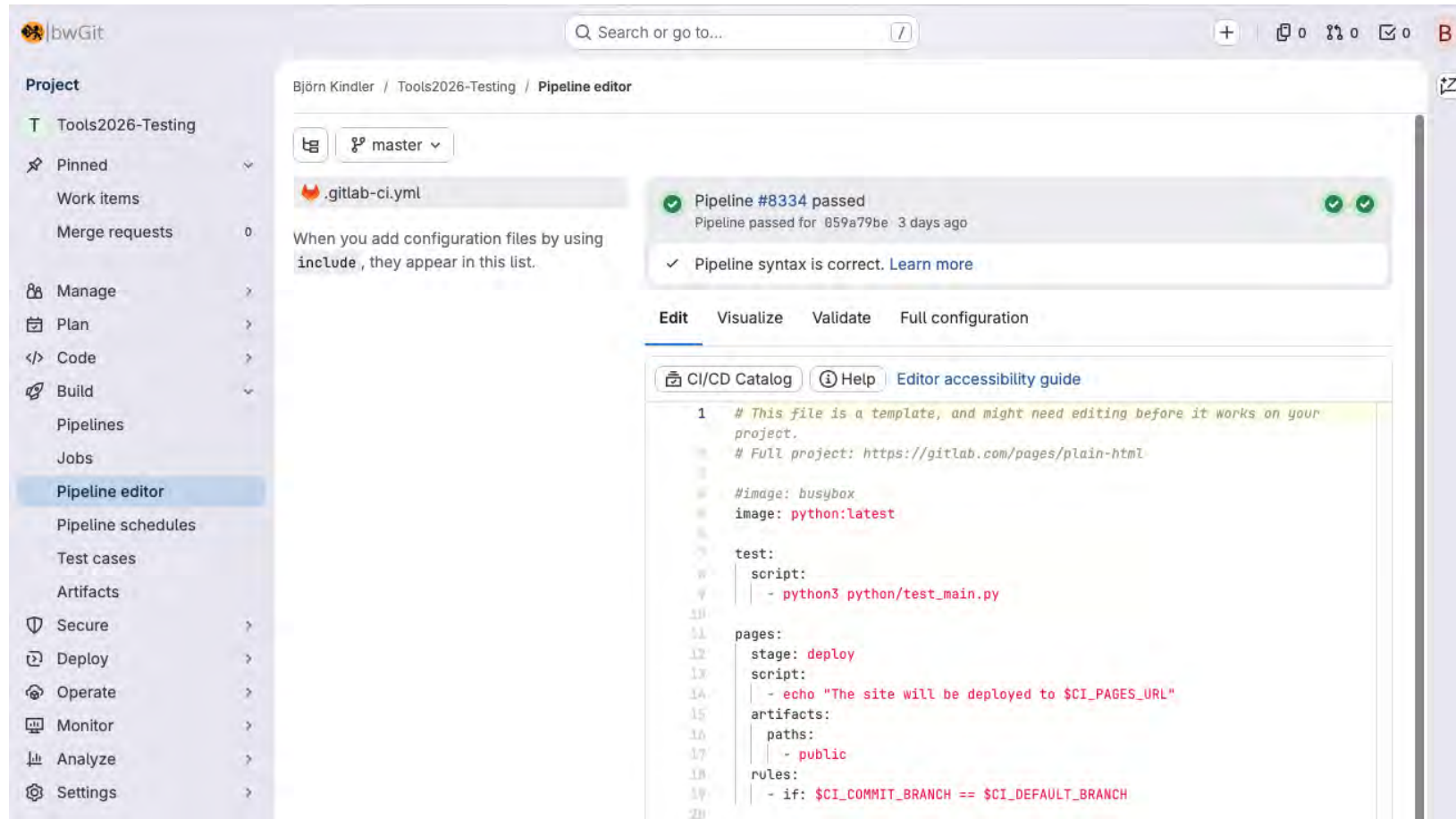
Branch

CI / CD -- Pipelines

- Pipelines definieren Aktionen. Z.B. build oder test-Aktivitäten
- Für die Ausführung der definierten Aktionen gibt es “Runner” – die können für ein Projekt, eine Gruppe und die ganze Installation zur Verfügung stehen und können auch auf eigenen Ressourcen zur Verfügung gestellt werden
- Für die Vorlesung verwenden wir von der Instanz bereit gestellte Runner
- Definition der Aktionen in einer Konfigurations-Datei

.gitlab-ci.yml

- Hier “test”, das bei jedem commit aufgerufen wird (keine “rules” Zeile) und hier einfach nur die test_main.py Script ausführt



Status	Pipeline	Created by	Stages	Actions
✔ Passed 🕒 00:00:18 📅 2 days ago	#8337 A notebook with some mat... 🔗 testing 🔗 6d632fda 👤 latest branch		✔	📄 ▼
✔ Passed 🕒 00:00:35 📅 3 days ago	#8334 Merge branch 'testing' into... 🔗 master 🔗 059a79be 👤 latest branch		✔✔	📄 ▼
✔ Passed 🕒 00:00:14 📅 3 days ago	#8332 fixed indentation and test 🔗 testing 🔗 75769590 👤 branch		✔	📄 ▼
✖ Failed 🕒 00:00:31 📅 3 days ago	#8331 Added a test for 5 and rem... 🔗 testing 🔗 adcd9393 👤 branch		✖	🔄 📄 ▼
✖ Failed 🕒 00:00:18 📅 3 days ago	#8328 Added again a failing test 🔗 testing 🔗 92dd1da8 👤		✖	🔄 📄 ▼

Pipelines – Übersicht
mit Anzeige der Status der
Ausführung (passed oder
failed) und Anzeige, welche
Stages ausgeführt wurden
incl. Ergebnis

Nach einem

The screenshot shows the GitLab CI/CD interface for the project 'Tools2026-Testing'. The pipeline #8407 is shown as failed. The job 'test' is highlighted in red, indicating it failed. The interface includes a sidebar with navigation options like 'Pipelines', 'Jobs', 'Pipeline editor', 'Pipeline schedules', 'Test cases', 'Artifacts', 'Secure', 'Deploy', 'Operate', 'Monitor', 'Analyze', and 'Settings'. The main content area shows the pipeline details, including the commit hash 'bb466edf' and the job status 'Failed'.

The screenshot shows the 'Jobs' section of the GitLab CI/CD interface. The sidebar on the left lists various options: 'Project', 'Tools2026-Testing', 'Pinned', 'Work items', 'Merge requests', 'Manage', 'Plan', 'Code', 'Build', 'Pipelines', 'Jobs', 'Pipeline editor', 'Pipeline schedules', 'Test cases', 'Artifacts', 'Secure', 'Deploy', 'Operate', 'Monitor', 'Analyze', 'Settings', 'What's new', and 'Help'. The main content area shows the 'Jobs' section, which is currently empty.

The screenshot shows the log output of a failed job. The log is displayed in a dark theme with a search bar at the top. The log content includes timestamps and job details, such as 'Running on GitLab Runner on gpp-runner-1.git.uni-heidelberg.de', 'Preparing the "docker" executor', 'Using Docker executor with image python:latest', 'Using effective pull policy of [always] for container python:latest', 'Pulling docker image python:latest', 'Using docker image sha256:0711502b4632bc1db9e89a70c4c3d2437fc6b0c9d2699188c4d5f41e05ecae7b for python:latest', 'Preparing environment', 'Using effective pull policy of [always] for container sha256:ed025aed354703775a6b235beb5570593f3cb7440e3ed6586a0f25df69b4b911', 'Running on runner-ldtosf9p2-project-1145-concurrent-0 via gpp-runner-1...', 'Getting source from Git repository', 'Gitally correlation ID: 01KWM89V8P6PKQZ6B69CG1YV8Q', 'Fetching changes with git depth set to 20...', 'Initialized empty Git repository in /builds/hd-ca120/tools2026-testing/.git/', 'Created fresh repository.', 'Checking out bb466edf as detached HEAD (ref is testing)...', 'Skipping Git submodules setup', 'Executing "step_script" stage of the job script', 'Using effective pull policy of [always] for container python:latest', 'Using docker image sha256:0711502b4632bc1db9e89a70c4c3d2437fc6b0c9d2699188c4d5f41e05ecae7b for python:latest', 'atest with digest python@sha256:09b29c360b84742bf98eba40b214f7f6b4b53286bb2c8a8b5b1afa188a8d9c0e ...', '\$ python3 python/test_main.py', 'test_atAll (__main__.TestClass.test_atAll) ... ok', 'test_ca120_plusBK (__main__.TestClass.test_ca120_plusBK) ... ERROR', 'test_minus (__main__.TestClass.test_minus) ... ok', 'ERROR: test_ca120_plusBK (__main__.TestClass.test_ca120_plusBK)', 'Traceback (most recent call last):', 'File "/builds/hd-ca120/tools2026-testing/python/test_main.py", line 16, in test_ca120_plusBK', 'self.assertEqual(ctt.plusBK(1,1,1),2)', 'AttributeError: \'ClassToTest\' object has no attribute \'plusBK\'', 'Ran 3 tests in 0.001s', 'FAILED (errors=1)', 'Cleaning up project directory and file based variables', and 'ERROR: Job failed: exit code 1'.

Nach dem

bwGit

Project

- Tools2026-Testing
- Pinned
- Work items
- Merge requests

Manage

- Plan
- Code
- Build
- Pipelines**
- Jobs
- Pipeline editor
- Pipeline schedules
- Test cases
- Artifacts
- Secure
- Deploy
- Operate
- Monitor
- Analyze
- Settings

Björn Kindler / Tools2026-Testing / Pipelines

View analytics Clear runner caches New pipeline

Alt 25 Finished Branches Tags

Filter pipelines

Status	Pipeline
Running	#8408 added the missing method P testing → 89ea2665 latest branch
Failed	#8407 added a ca120 test 00:00:41 13 minutes ago P testing → bb466edf branch
Passed	#8337 A notebook with some matplot... 00:00:18 2 days ago P testing → 6d632fda branch
Passed	#8334 Merge branch 'testing' into 'm... 00:00:35 3 days ago P master → 859a79be latest branch
Passed	#8332 fixed indentation and test 00:00:14 3 days ago P testing → 75769598 branch
Failed	#8331 Added a test for 5 and remove... 00:00:31 3 days ago P testing → addc9393 branch

What's new 10

Help

Collapse sidebar

Help | Impressum | Contact | For your c

bwGit

Project

- Tools2026-Testing
- Pinned
- Work items
- Merge requests

Manage

- Plan
- Code
- Build
- Pipelines**
- Jobs
- Pipeline editor
- Pipeline schedules
- Test cases
- Artifacts
- Secure
- Deploy
- Operate
- Monitor
- Analyze
- Settings

Björn Kindler / Tools2026-Testing / Jobs

View analytics Clear runner caches New pipeline

Alt 25 Finished Branches Tags

Filter pipelines

Status	Pipeline
Running	#8408 added the missing method P testing → 89ea2665 latest branch
Failed	#8407 added a ca120 test 00:00:41 13 minutes ago P testing → bb466edf branch
Passed	#8337 A notebook with some matplot... 00:00:18 2 days ago P testing → 6d632fda branch
Passed	#8334 Merge branch 'testing' into 'm... 00:00:35 3 days ago P master → 859a79be latest branch
Passed	#8332 fixed indentation and test 00:00:14 3 days ago P testing → 75769598 branch
Failed	#8331 Added a test for 5 and remove... 00:00:31 3 days ago P testing → addc9393 branch

What's new 10

Help

Collapse sidebar

Help | Impressum | Contact | For your c

bwGit

Project

- Tools2026-Testing
- Pinned
- Work items
- Merge requests

Manage

- Plan
- Code
- Build
- Pipelines**
- Jobs
- Pipeline editor
- Pipeline schedules
- Test cases
- Artifacts
- Secure
- Deploy
- Operate
- Monitor
- Analyze
- Settings

Björn Kindler / Tools2026-Testing / Jobs / #24844

test

Passed Started 1 minute ago by Björn Kindler

Log timestamps in UTC. Search visible log output

```
1 15:21:25 Running with gitlab-runner 18.11.3 (ad1797b3)
2 15:21:25 on GitLab Runner on gpp-runner-2.git.uni-heidelberg.de QrD-d3Zi0, system ID: s_c4573749ee78
3 15:21:25 Resolving secrets
4 15:21:25 Preparing the "docker" executor 00:08
5 15:21:25 Using Docker executor with image python:latest ...
6 15:21:29 Using effective pull policy of [always] for container python:latest
7 15:21:29 Pulling docker image python:latest ...
8 15:21:33 Using docker image sha256:0711502b4632bc1db9e89a70c4c3d2437fc6b0c9d2699188c4d5f41e05ecae7b for python:l
atest with digest python@sha256:09b29c360b84742bf98eba40b214f7f6b4b53286bb2c8a8b5b1afa188a8d9c0e ...
9 15:21:33 Preparing environment 00:08
10 15:21:33 Using effective pull policy of [always] for container sha256:48c652389f0af6f0b41e7408243d96deb0baab12ae
d1915e2a2a927424ad7506
11 15:21:41 Running on runner-qrd-d3zi0-project-1145-concurrent-0 via gpp-runner-2...
12 15:21:41 Getting source from Git repository 00:04
13 15:21:42 Gitaly correlation ID: 01KWM93WHDB02KS3YEE4P335QE
14 15:21:42 Fetching changes with git depth set to 20...
15 15:21:42 Initialized empty Git repository in /builds/hd-ca120/tools2026-testing/.git/
16 15:21:42 Created fresh repository.
17 15:21:43 Checking out 89ea2665 as detached HEAD (ref is testing)...
18 15:21:44 Skipping Git submodules setup
19 15:21:45 Executing "step_script" stage of the job script 00:03
20 15:21:45 Using effective pull policy of [always] for container python:latest
21 15:21:45 Using docker image sha256:0711502b4632bc1db9e89a70c4c3d2437fc6b0c9d2699188c4d5f41e05ecae7b for python:l
atest with digest python@sha256:09b29c360b84742bf98eba40b214f7f6b4b53286bb2c8a8b5b1afa188a8d9c0e ...
22 15:21:46 $ python3 python/test_main.py
23 15:21:47 test_atAll (__main__.TestClass.test_atAll) ... ok
24 15:21:47 test_ca120_plusBK (__main__.TestClass.test_ca120_plusBK) ... ok
25 15:21:47 test_minus (__main__.TestClass.test_minus) ... ok
26 15:21:47 -----
27 15:21:47 Ran 3 tests in 0.000s
28 15:21:47 OK
29 15:21:48 Cleaning up project directory and file based variables 00:01
30 15:21:49 Job succeeded
```

Help | Impressum | Contact | For your own non-personal groups; please apply by email

Wenn es beim push einen Konflikt gibt...

- Beispiel: in test_mail.py wurde etwas im Repository geändert nach Ihrem checkout

```
kindler@BKs2022 python % git status
On branch testing
Your branch is up to date with 'origin/testing'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test_main.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

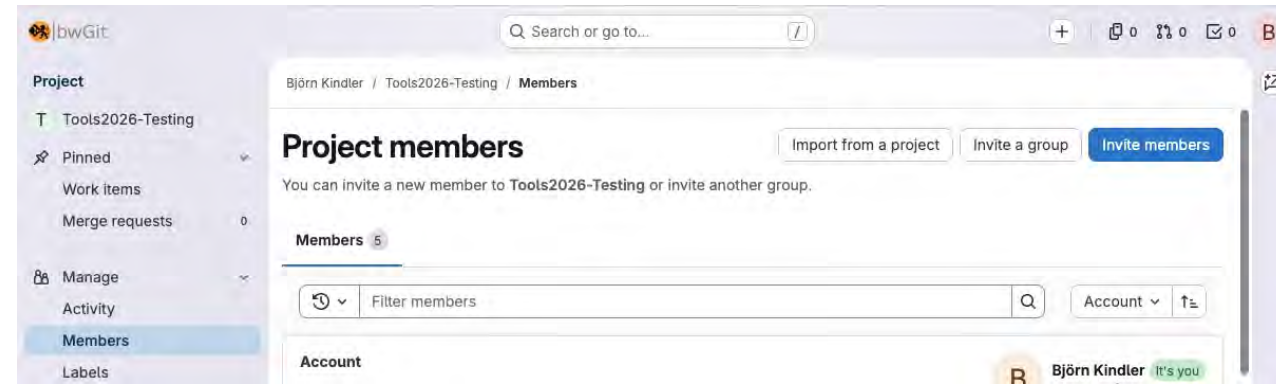
        TestClass.py

no changes added to commit (use "git add" and/or "git commit -a")
kindler@BKs2022 python % git add test_main.py
kindler@BKs2022 python % git commit -m "added a test_another "
[testing ce28101] added a test_another
1 file changed, 4 insertions(+), 2 deletions(-)
kindler@BKs2022 python % git push
To bwgit.de:hd-ca120/tools2026-testing.git
 ! [rejected]        testing -> testing (fetch first)
error: failed to push some refs to 'git@bwgit.de:hd-ca120/tools2026-testing.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
kindler@BKs2022 python %
```

git pull ... then fix ... then push again

Übung mit gitlab

- Login auf <https://bwgit.de>
- (dann füge ich Sie zum “tools2026-testing” Projekt hinzu)
- Clonen Sie das Projekt
- und wechseln Sie zum “testing” branch
- fügen Sie einen Test zum python/test_main.py hinzu (test_URZID)
- Committen Sie Ihre Änderung und laden Sie sie hoch
- Dann passen Sie ClassToTest.py so an, daß der Test wieder geht...
- Committen Sie Ihre Änderungen und laden Sie sie hoch



Testing für Software im team

- Lokal entwickeln (mit git für die Verfolgung der Änderungen)
- Änderungen in einem changeset zur Verfügung stellen
- Automatischer Test der Software mit allen Test-Suites (hier nur ein minimales Test-Script gezeigt)
- Manueller code review on top (hier nicht gezeigt)
- Deployment

Testing: end-to-end testing of a web service

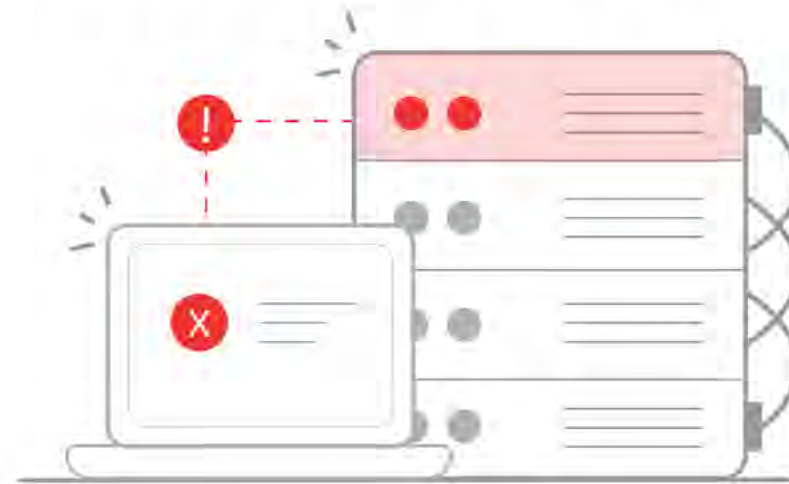
- Nötig t.B. für Services, die Forschungsgruppen im Web für die Forschungsgemeinschaft anbieten um sicher zu stellen, daß Dinge “weiterhin / immer noch funktionieren”
- Beispiel: Neuromorphic Computing via EBRAINS
<https://ebrains.eu/nmc> -> BrainScaleS

Intended end-us



- Um ein Beispiel Notebook ausführen zu können so Webbrowser mit Zugang
- URL: <https://ebrains.eu> the quick try link

Scheduled Maintenance



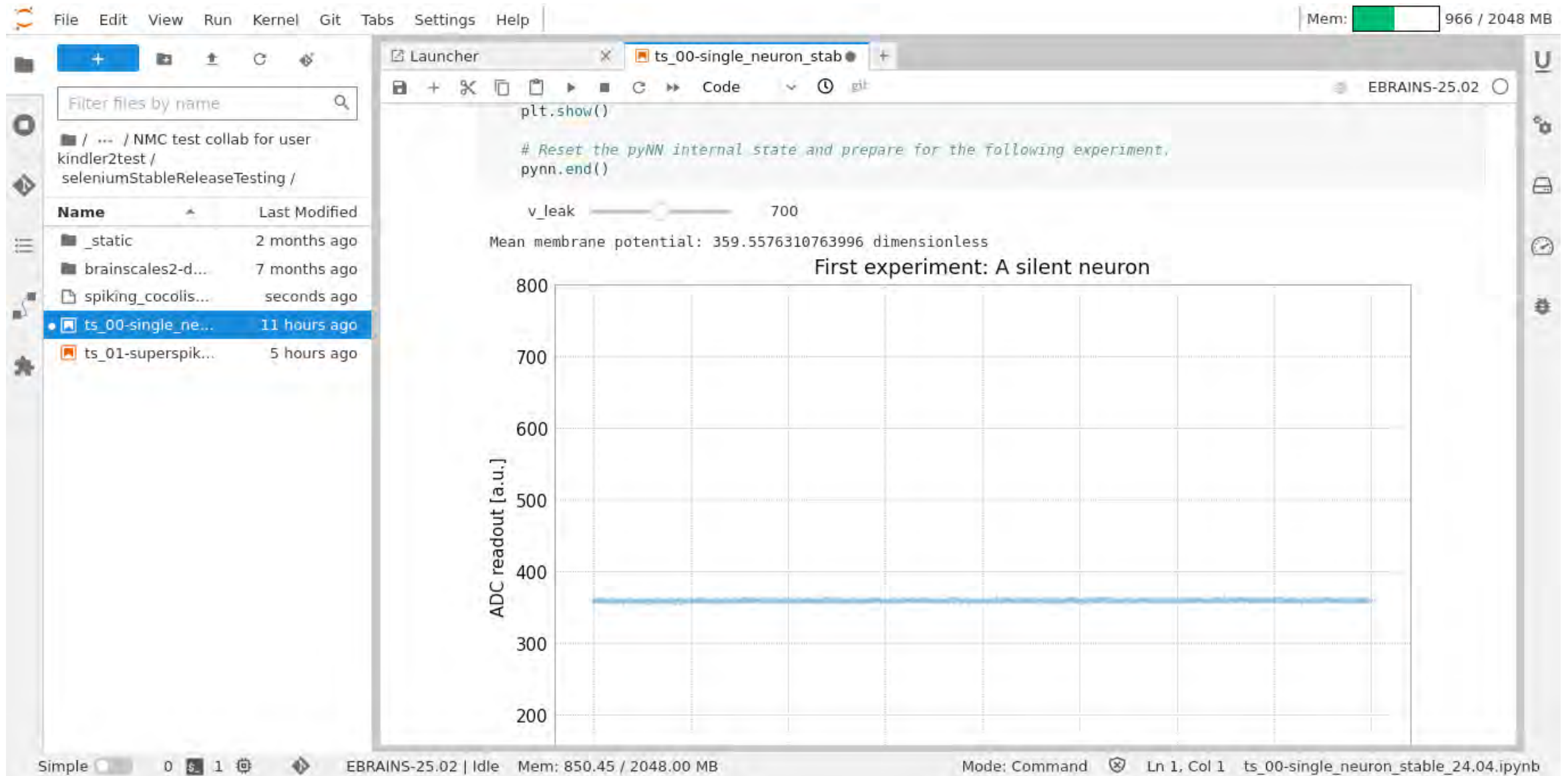
The service you tried to access and other EBRAINS services that might depend on it are down for scheduled maintenance.

If you have further questions or need assistance, please reach out to [support](#) ↗

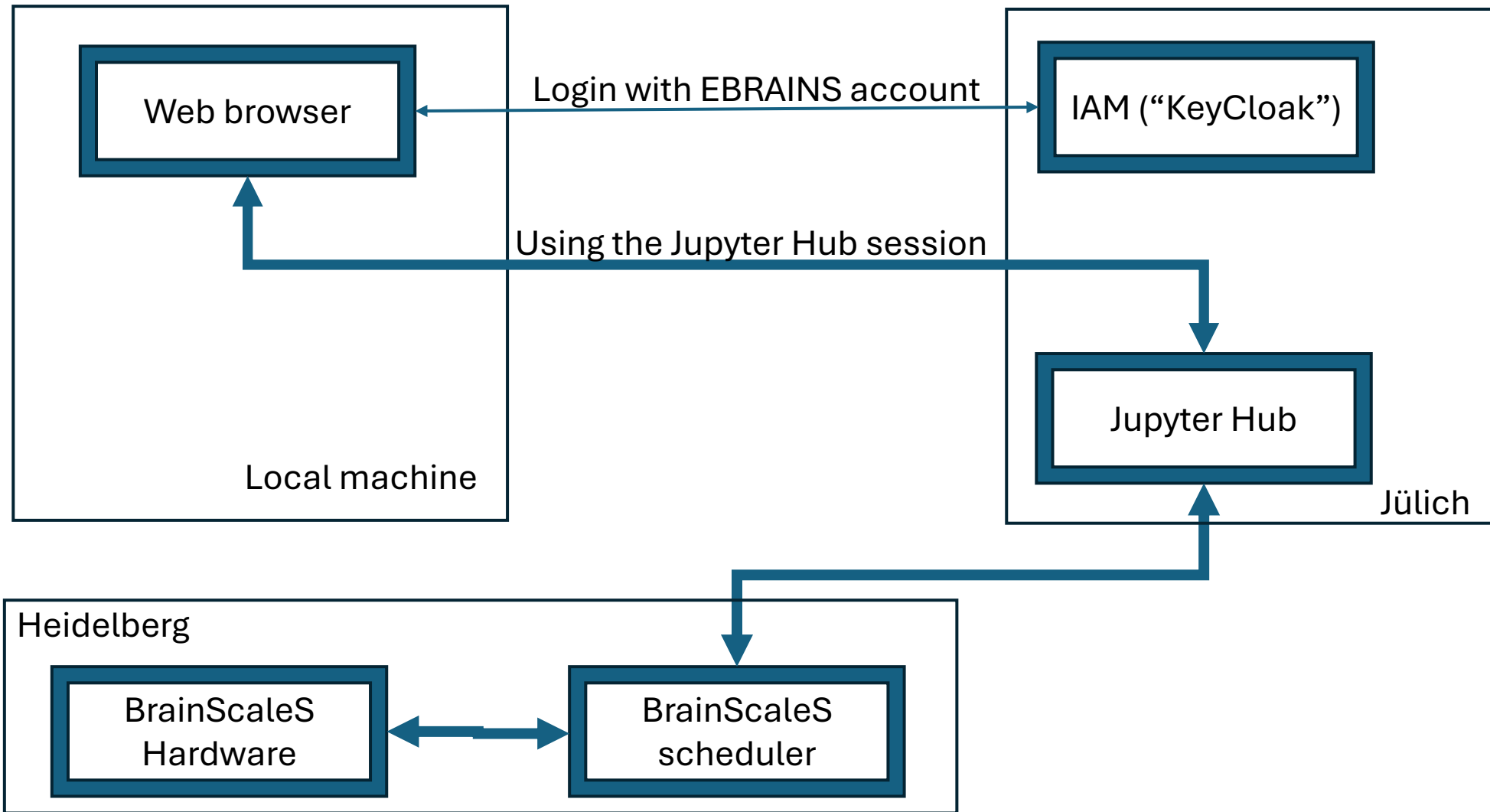
You can also check our [service health](#) ↗ page for a summary of the availability of our main services.

We apologize for the inconvenience and thank you for your understanding.

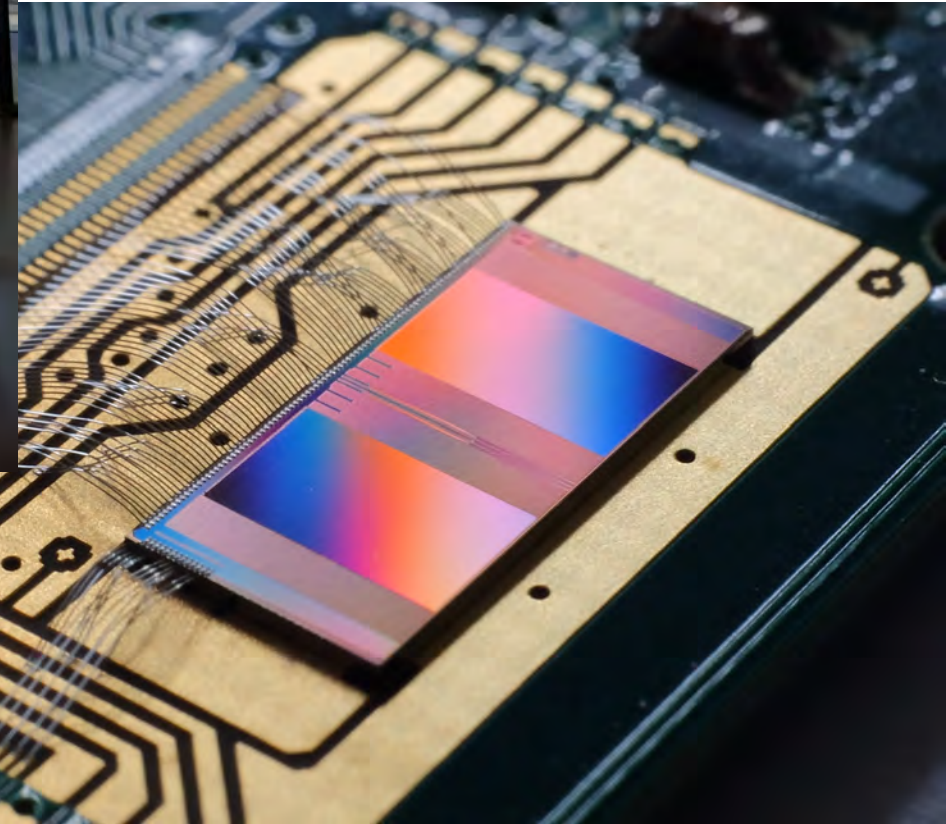
Just execute a notebook cell = easy (?)



“What could possibly go wrong”

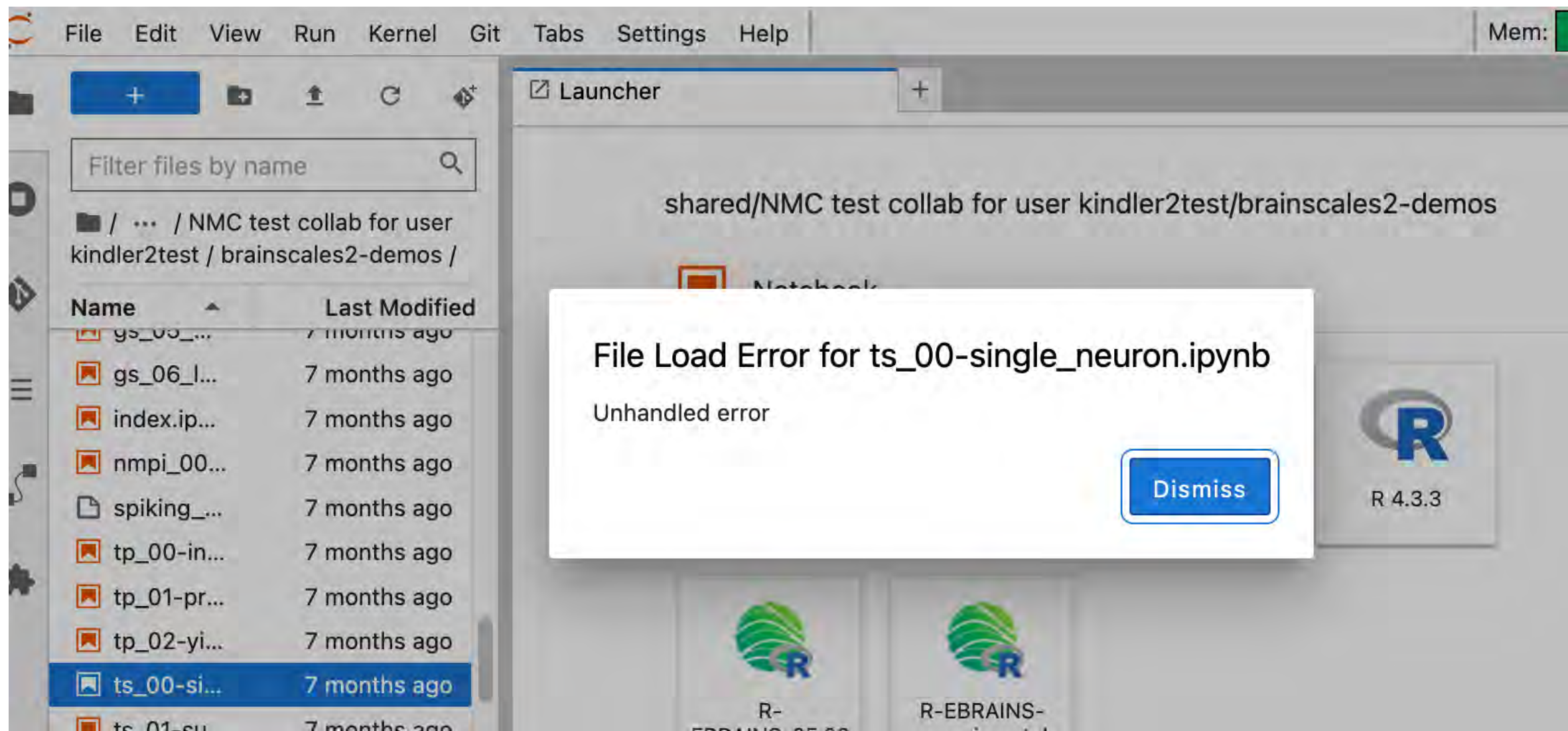


BrainScaleS systems at EINC,
accessible via the EBRAINS RI

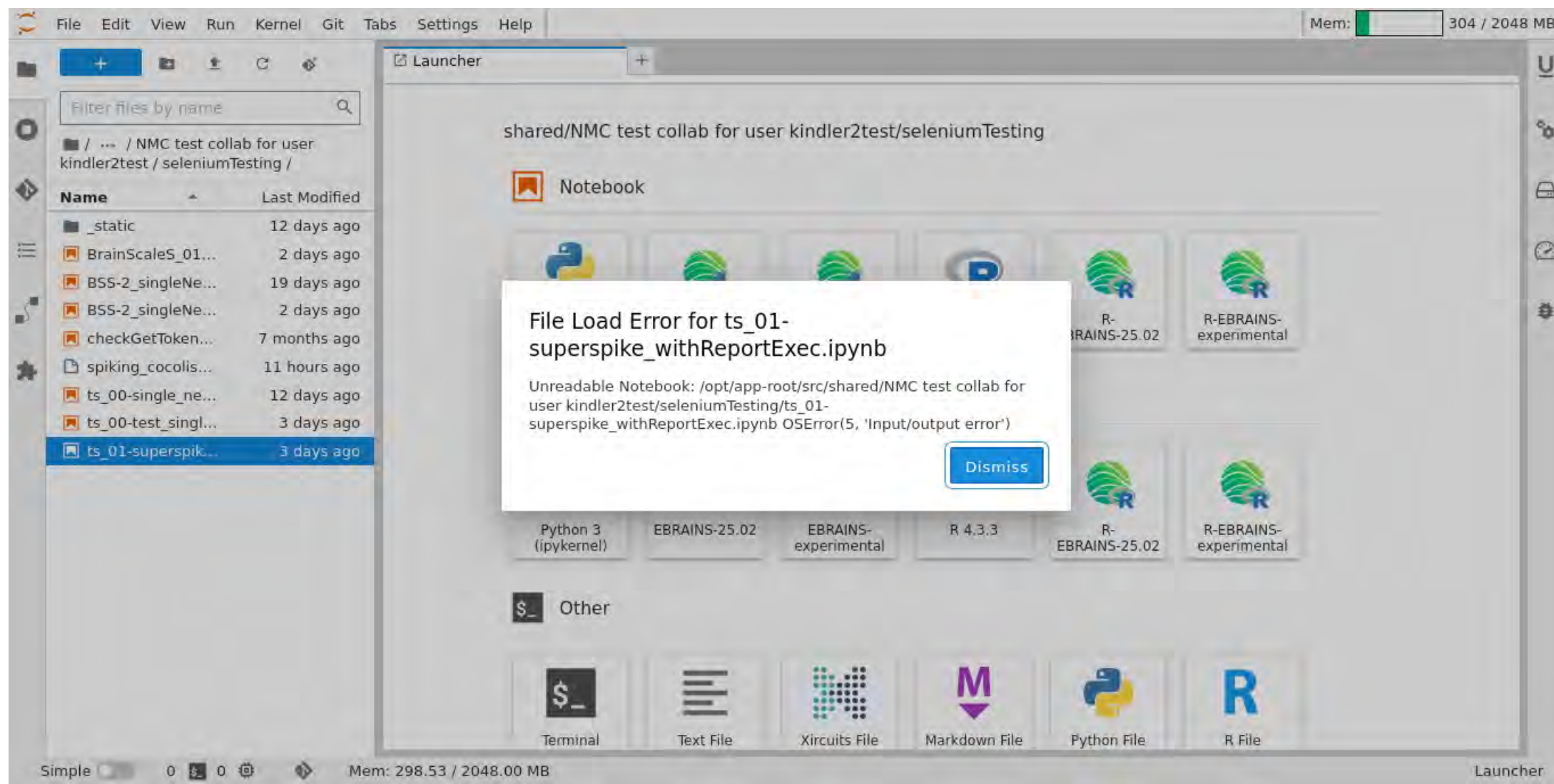


A single BrainScaleS-2 chip

“What could possibly go wrong”



“What could possibly go wrong”



“What could possibly go wrong”

jupyterhub Home Token Services ▾

400 : Bad Request
kindler2test is pending spawn

“What could possibly go wrong”

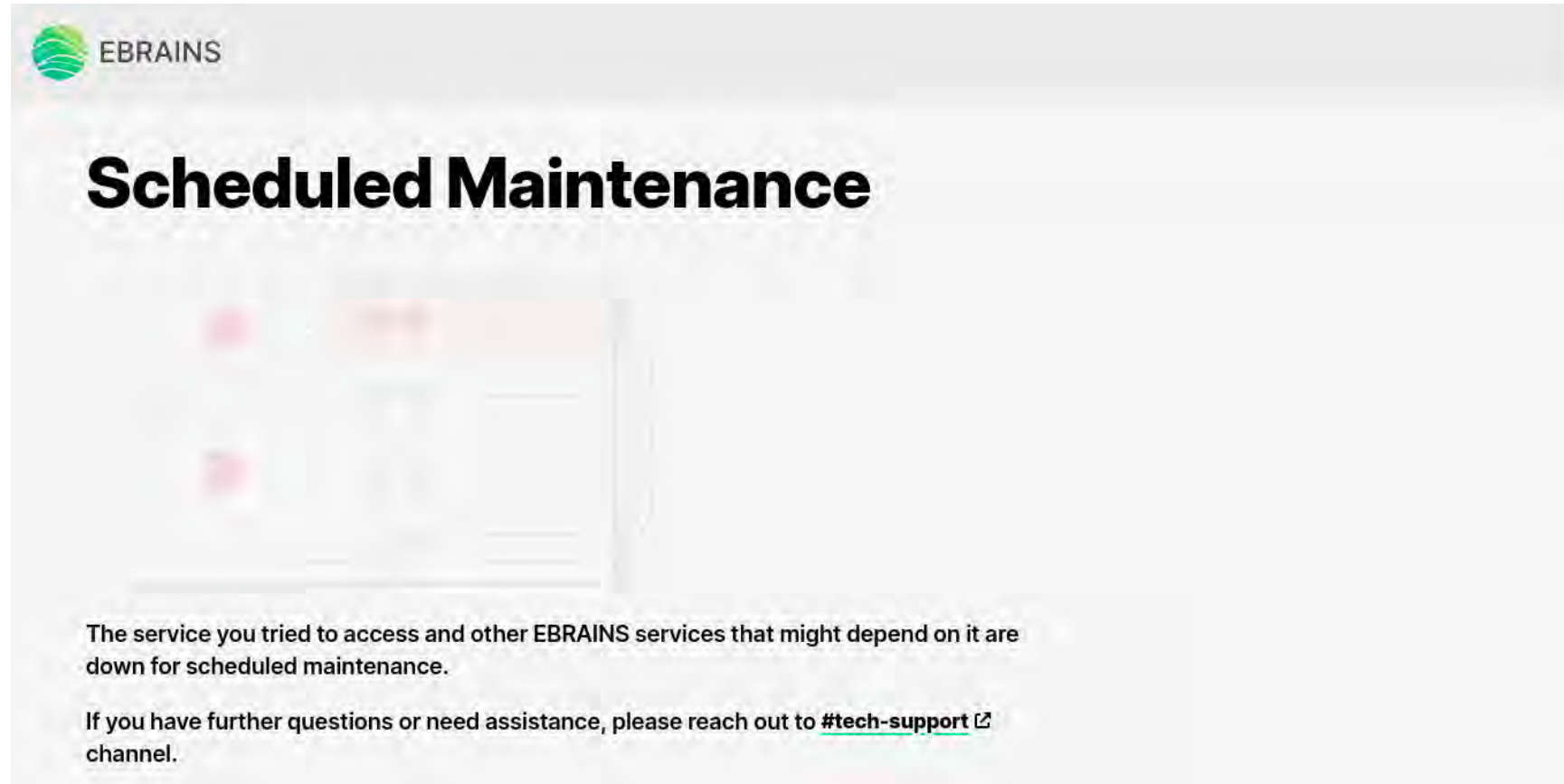
Service Unavailable

The server is temporarily unable to service your request due to maintenance downtime or capacity problems. Please try again later.

Additionally, a 503 Service Unavailable error was encountered while trying to use an ErrorDocument to handle the request.

Apache/2.4.52 (Ubuntu) Server at iam.ebrains.eu Port 443

“What could possibly go wrong”



“What could possibly go wrong”

The screenshot shows a JupyterLab environment. On the left is a file explorer with a search bar and a list of files. The file 'ts_00-test_single_neuron...' is selected. On the right is a code editor with a title bar 'ts_00-test_single_neuron...'. The code in the editor is as follows:

```
BrainScaleS-2 single neuron experiments -- only the first
part (silent neuron) for testing

In order to use the microscheduler we have to set some environment variables first:

!date
from _static.common.helpers import setup_hardware_client
import time
setup_hardware_client()
!date

!env | grep LAB_ | sort
!env | grep Q | sort

%matplotlib inline
import numpy as np
from ipywidgets import Interact, IntSlider
from functools import partial
IntSlider = partial(IntSlider, continuous_update=False)
import matplotlib.pyplot as plt
plt.style.use("_static/matplotlibrc")

import pynn_brainscales.brainscales2 as pynn
from pynn_brainscales.brainscales2 import Population
from pynn_brainscales.brainscales2.standardmodels.cells import SpikeSourceArray
from pynn_brainscales.brainscales2.standardmodels.synapses import StaticSynapse
```

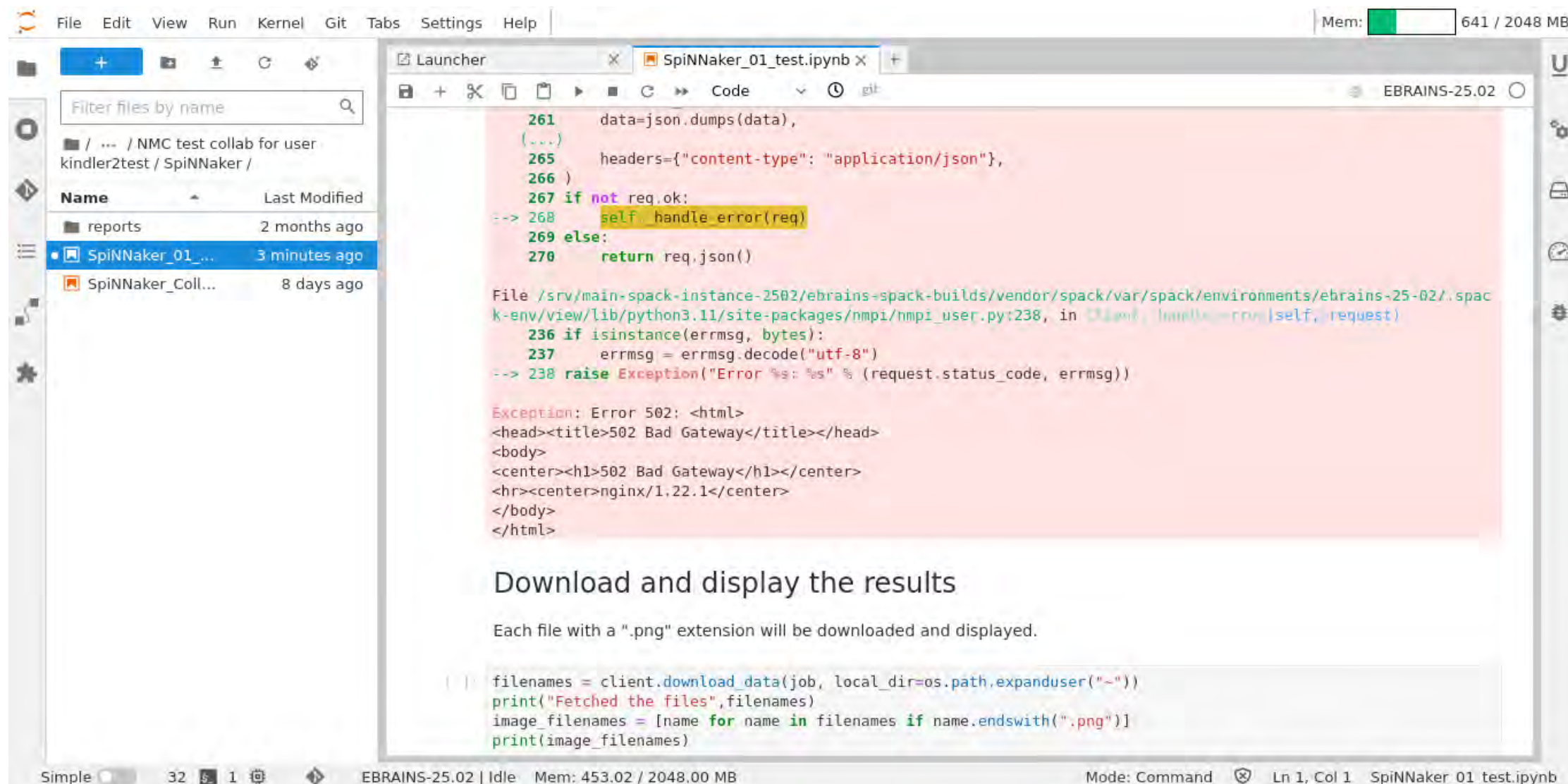
A blue arrow points from the file 'ts_00-test_single_neuron...' in the file explorer to the code editor.

Mem: 343 / 2048 MB

Simple 0 1 EBRAINS-experimental | Unknown Mem: 417.10 / 2048.00 MB Mode: Command Ln 1, Col 1 ts_00-test_single_neuron_firstExpOnly.ipynb

“What could possibly go wrong”

Sometimes it is also our system in Heidelberg....



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like 'reports', 'SpiNNaker_01_...', and 'SpiNNaker_Coll...'. The code editor shows a Python script with a 502 Bad Gateway error. The error message is displayed in a red box, indicating a failure in the HTTP request. The script is attempting to download data from a server, but the server is returning a 502 status code, which typically indicates a bad gateway or a server-side error. The error message includes HTML content: <html><head><title>502 Bad Gateway</title></head><body><center><h1>502 Bad Gateway</h1></center><hr><center>nginx/1.22.1</center></body></html>.

```
261 data=json.dumps(data),
(... )
265 headers={"content-type": "application/json"},
266 )
267 if not req.ok:
--> 268     self.handle_error(req)
269 else:
270     return req.json()

File /srv/main+spack-instance-2502/ebbrains=spack-builds/vendor/spack/var/spack/environments/ebbrains-25-02/.spack
k-env/view/lib/python3.11/site-packages/nmpi/nmpi_user.py:238, in Client.handle_error(self, request)
236 if isinstance(errmsg, bytes):
237     errmsg = errmsg.decode("utf-8")
--> 238 raise Exception("Error %s: %s" % (request.status_code, errmsg))

Exception: Error 502: <html>
<head><title>502 Bad Gateway</title></head>
<body>
<center><h1>502 Bad Gateway</h1></center>
<hr><center>nginx/1.22.1</center>
</body>
</html>
```

Download and display the results

Each file with a ".png" extension will be downloaded and displayed.

```
( | ) filenames = client.download_data(job, local_dir=os.path.expanduser("~"))
print("Fetched the files", filenames)
image_filenames = [name for name in filenames if name.endswith(".png")]
print(image_filenames)
```

Simple 32 1 EBRAINS-25.02 | Idle Mem: 453.02 / 2048.00 MB Mode: Command Ln 1, Col 1 SpiNNaker_01_test.ipynb

EBRAINS RI als Beispiel (NMC BrainScaleS)

- Account unter <https://ebrains.eu/register> erstellen (mit @uni-heidelberg.de Adresse, z.B. der URZID@uni-heidelberg.de) (Zugang zu den URZID@... Mails: z.B. über <https://sogo.uni-heidelberg.de>)
- Mit dem neuen EBRAINS account <https://einc.eu/bss> >> “Getting Access” >> “access this URL”

(Info: diese Stelle ist non-persistent = alle Änderungen gehen beim Ausloggen verloren! Alternative: Collab erstellen und das “Drive” nutzen)

Selenium

Ist eine Library, mit der man einen browser (Chrome oder Firefox) programmatisch steuern kann.

Damit ist es möglich, einen Web-Service “end-to-end” zu testen, also so, wie ein Nutzer den Service erlebt.

(Ein end-to-end Test sollte natürlich nicht der einzige Test sein. Direkte Service Tests erlauben normalerweise ein einfacheres Debugging, können den end-to-end Test aber nicht ersetzen)

Selenium setup

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
import time
from selenium.webdriver.firefox.service import Service as geckoService
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver import Keys

gecko_service=geckoService("/gecko/geckodriver")
browser_options = webdriver.FirefoxOptions()
if False:
    browser_options.add_argument("-headless")
driver = webdriver.Firefox(service=gecko_service,options=browser_options)
```

Open page

- `driver.get("URL_to_open")`

The URL for the example:

`https://lab.jsc.ebrains.eu/hub/user-redirect/git-pull?repo=https%3A%2F%2Fgithub.com%2Felectronicvisions%2Fbrainscales2-demos.git&urlpath=lab%2Ftree%2Fbrainscales2-demos.git%2Fts_00-single_neuron.ipynb&branch=jupyter-notebooks`

A small example

```
In [87]: 1 driver.get("https://lab.jsc.ebrains.eu/hub/user-redirect/git-pull"\n2             + "?repo=https%3A%2F%2Fgithub.com%2Felectronicvisions%2Fbrainscales2-demos.git"\n3             + "&urlpath=lab%2Ftree%2Fbrainscales2-demos.git%2Fts_00-single_neuron.ipynb"\n4             + "&branch=jupyter-notebooks-experimental")
```

```
In [88]: 1 for e in driver.find_elements(By.CLASS_NAME, "p-MenuBar-itemLabel"):\n2     print(e.text)
```

File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help

Elemente auf der Seite finden (Beispiele)

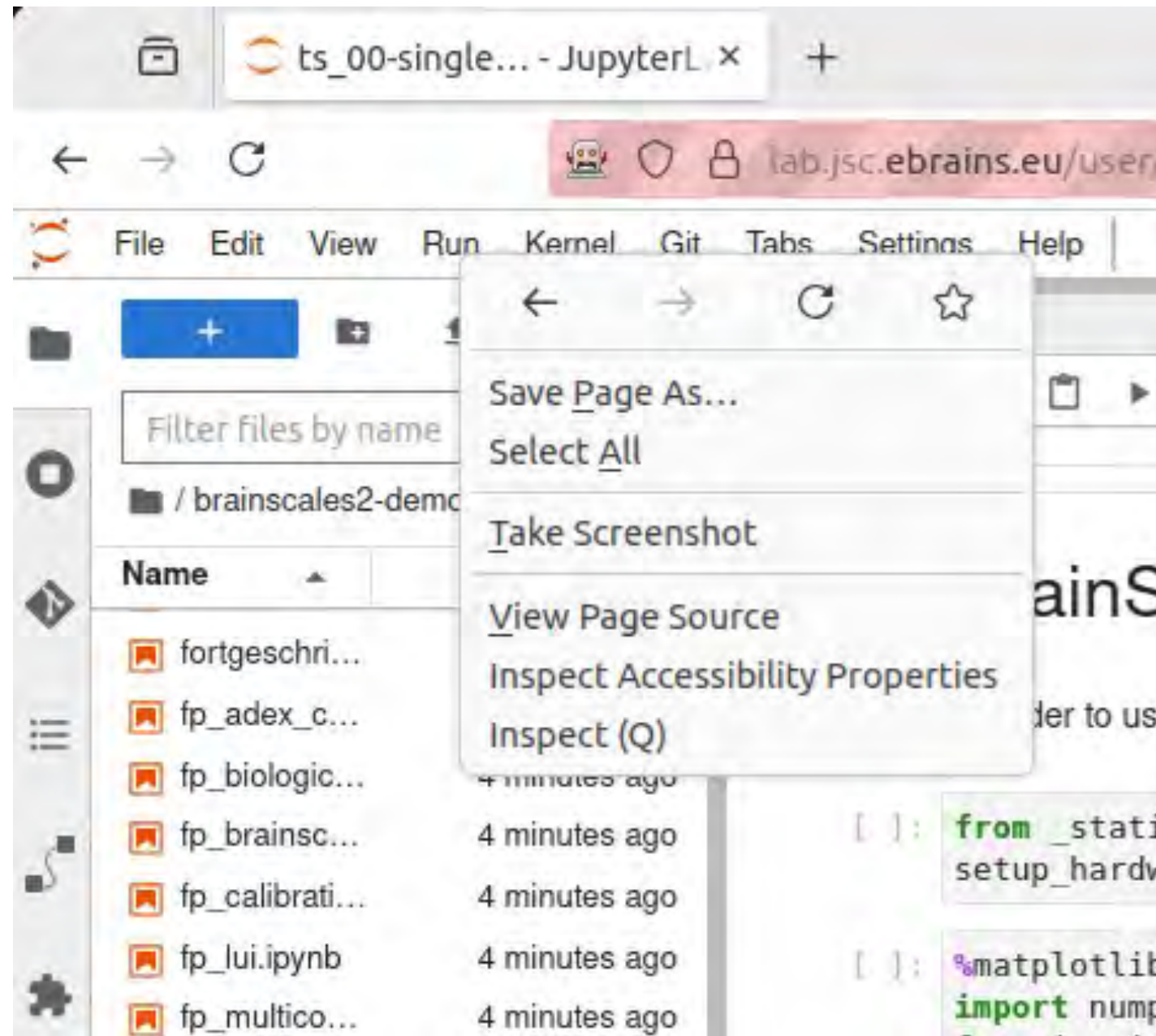
- `elems=driver.find_elements(By.TAG_NAME,"h1")`
- `elems=driver.find_elements(By.CLASS_NAME,"control")`
- `elems = driver.find_elements(By.ID,"username")`
- Oder Elemente unterhalb eines Elementes:
`elems=driver.find_elements(By.CLASS_NAME,"jp-FileBrowser-crumbs")`
`if len(elems) > 0:`
 `el=elems[0].find_elements(By.CSS_SELECTOR,"*")`
 for e in el:
 (do something with them)

A small example

```
In [87]: 1 driver.get("https://lab.jsc.ebrains.eu/hub/user-redirect/git-pull"\n2             + "?repo=https%3A%2F%2Fgithub.com%2Felectronicvisions%2Fbrainscales2-demos.git"\n3             + "&urlpath=lab%2Ftree%2Fbrainscales2-demos.git%2Fts_00-single_neuron.ipynb"\n4             + "&branch=jupyter-notebooks-experimental")
```

```
In [88]: 1 for e in driver.find_elements(By.CLASS_NAME, "p-MenuBar-itemLabel"):\n2     print(e.text)
```

File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help



Mit “rechter Maustaste” und SHIFT auf ein Element klicken und “Inspect (Q)” auswählen

Dies zeigt im "Inspector" in Firefox das Element im html an - und zum Glück verwendet Jupyter an ganz vielen Stellen class Namen:



The screenshot shows the Firefox DevTools Inspector with the 'Inspector' tab selected. The HTML tree on the left shows a list of menu items. The second item is expanded, showing a 'div' with the class 'lm-MenuBar-itemIcon' and another 'div' with the class 'lm-MenuBar-itemLabel' containing the text 'Run'. The 'Run' button is highlighted in blue. The breadcrumb at the bottom shows the path: 'li.lm-MenuBar-item.p-MenuBar-item > div.lm-MenuBar-itemLabel.p-MenuBar-itemL...'.

```
Inspector Console Debugger Network Style Editor Performance
rich HTML
  <li class="lm-MenuBar-item p-MenuBar-item" tabindex="0" role="menuitem"
    aria-haspopup="true"> ... </li> event
  <li class="lm-MenuBar-item p-MenuBar-item" tabindex="0" role="menuitem"
    aria-haspopup="true"> event
    <div class="lm-MenuBar-itemIcon p-MenuBar-itemIcon"> ... </div>
    <div class="lm-MenuBar-itemLabel p-MenuBar-itemLabel">Run</div>
  </li>
  <li class="lm-MenuBar-item p-MenuBar-item" tabindex="0" role="menuitem"
    aria-haspopup="true"> ... </li> event
  <li class="lm-MenuBar-item p-MenuBar-item" tabindex="0" role="menuitem"
    aria-haspopup="true"> ... </li> event
Bar-content > li.lm-MenuBar-item.p-MenuBar-item > div.lm-MenuBar-itemLabel.p-MenuBar-itemL...
```

A small example

```
In [87]: 1 driver.get("https://lab.jsc.ebrains.eu/hub/user-redirect/git-pull"\n2             + "?repo=https%3A%2F%2Fgithub.com%2Felectronicvisions%2Fbrainscales2-demos.git"\n3             + "&urlpath=lab%2Ftree%2Fbrainscales2-demos.git%2Fts_00-single_neuron.ipynb"\n4             + "&branch=jupyter-notebooks-experimental")
```

```
In [88]: 1 for e in driver.find_elements(By.CLASS_NAME, "p-MenuBar-itemLabel"):\n2     print(e.text)
```

File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help

```
In [97]: 1 for e in driver.find_elements(By.CLASS_NAME, "p-MenuBar-itemLabel"):
          2     print(e.text)
          3     if "Run"==e.text:
          4         e.click()
          5         for f in driver.find_elements(By.CLASS_NAME, "p-Menu-itemLabel"):
          6             print(f.text)
          7             if f.text=="Run All Cells":
          8                 f.click()
          9                 break
         10         break
         11
```

(Warum die “break” Anweisung? Was passiert, wenn die nicht da ist?)

Screenshots

- Einen screenshot der aktuellen Seite im Browser machen:
`driver.get_screenshot_as_file('/PFAD/test.png')`

- Screenshots dann im notebook anzeigen:

```
from IPython.core.display import display, HTML
import os
a=os.listdir("/PFAD")
s=""
for x in a:
    l="screenshots/"+x
    s+='<a href="'+l+'" target=_blank></a><br>\n'
    print(x)
print(s)
display(HTML(s))
```

```
In [105]: 1 driver.get_screenshot_as_file("test.png")
```

```
Out[105]: True
```

```
In [111]: 1 from IPython.core.display import display, HTML
2 s("<img src=test.png>")
3 display(HTML(s))
```

The screenshot shows a Jupyter Notebook interface. On the left is a file explorer for the directory `/brainscales2-demos.git/`. It lists various files including `fp_supers...`, `fp_synapli...`, `fp_syntfire...`, `girlsday.py...`, `gs_01_ne...`, `gs_02_pyn...`, `gs_03_sin...`, `gs_04_1_...`, `gs_04-2_...`, `gs_05_ma...`, `gs_06_jogi...`, `index.ipynb`, `nmpi_00-n...`, `spiking_co...`, `tp_00-intro...`, `tp_01-prop...`, `tp_02-yin...`, and `ts_00-singl...` (highlighted). The main area shows a notebook cell with the title "BrainScaleS-2 single neuron experiments". The cell contains two code blocks. The first block imports `setup_hardware_client` from `static.common.helpers` and prints an info message: `INFO 14:49:25,419 demo_helpers Connection to hxcube7fpga0chip57_1 established`. The second block imports `numpy`, `Interact`, `IntSlider`, `matplotlib`, and `pyNN` related modules. It defines a function `plot_membrane_dynamics` which takes a `Population` object and a `segment_id`. The status bar at the bottom indicates the notebook is busy, with memory usage at 1.47 / 2.00 GB.

File Edit View Run Kernel Git Tabs Settings Help Mem: 1.63 / 2.00 GB

Launcher ts_00-single_neuron.ipynb

Filter files by name

/ brainscales2-demos.git /

Name	Last Modified
fp_supers...	15 minutes ago
fp_synapli...	15 minutes ago
fp_syntfire...	15 minutes ago
girlsday.py...	15 minutes ago
gs_01_ne...	15 minutes ago
gs_02_pyn...	15 minutes ago
gs_03_sin...	15 minutes ago
gs_04_1_...	15 minutes ago
gs_04-2_...	15 minutes ago
gs_05_ma...	15 minutes ago
gs_06_jogi...	15 minutes ago
index.ipynb	15 minutes ago
nmpi_00-n...	15 minutes ago
spiking_co...	3 minutes ago
tp_00-intro...	15 minutes ago
tp_01-prop...	15 minutes ago
tp_02-yin...	15 minutes ago
ts_00-singl...	1 minute ago

BrainScaleS-2 single neuron experiments

In order to use the microscheduler we have to set some environment variables first:

```
[111]: from static.common.helpers import setup_hardware_client
setup_hardware_client()

INFO 14:49:25,419 demo_helpers Connection to hxcube7fpga0chip57_1 established
```

```
[112]: %matplotlib inline
import numpy as np
from ipywidgets import interact, IntSlider
from functools import partial
IntSlider = partial(IntSlider, continuous_update=False)
import matplotlib.pyplot as plt
plt.style.use("static/matplotlib")

import pyNN_brainscales.brainscales2 as pyNN
from pyNN_brainscales.brainscales2 import Population
from pyNN_brainscales.brainscales2.standardmodels.cells import SpikeSourceArray
from pyNN_brainscales.brainscales2.standardmodels.synapses import StaticSynapse

def plot_membrane_dynamics(population: Population, segment_id=1, ylim=None):
    """Plot the membrane potential of the neuron in a given population using PyNN"""
```

Simple 2 0 jupyter-notebooks-experimental EBRAINS-experimental | Busy Mem: 1.47 / 2.00 GB Mode: Command Ln 1, Col 1 ts_00-single_neuron.ipynb

Übungsaufgabe: log-in hinzu fügen

```
In [112]: 1 import getpass  
          2 username="guestbktest2july2"  
          3 password = getpass.getpass()
```

.....

Einfügen von text: mit e.send_keys(...)

And where are the slides?????

- Falls Sie die Folien des heutigen Tages brauchen: sie stehen als .pdf Export im Branch “Slides” im Repository.