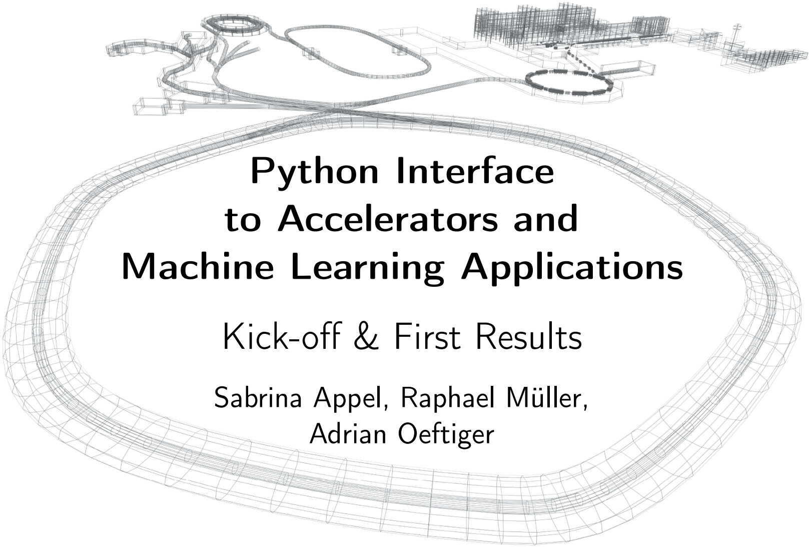


# Python Interface to Accelerators and Machine Learning Applications

Kick-off & First Results

Sabrina Appel, Raphael Müller, Adrian Oeftiger

## Context

**Small task force team from:**

- operations, accelerator control and beam dynamics departments

**Short-term goal:**

- obtain python access to machine control systems

  $\Longrightarrow$ proof-of-principle to get, set and subscribe to parameters

**Long-term goal:**

- make use of scripting for machine experiments
- involve python libraries to exploit machine learning algorithms

  $\Longrightarrow$ support optimisation of machine operation

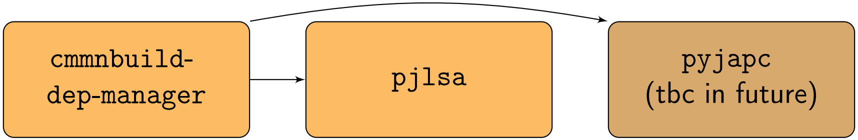
**Starting point:**

- CERN runs wealthy pool of python libraries accessing LSA/JAPC/...

  $\Longrightarrow$ can we make use of that?!

## Relevant Python Libraries

Repository forks on internal GSI gitlab in new group `scripting-tools`:



**cmmnbuild-dep-manager**

- manages Java dependencies (via `JPype`)
- downloads `jar` files via gradle (or CBNG for CERN)
- available fork on GSI gitlab ↗
- origin: CERN gitlab ↗

**pjlsa**

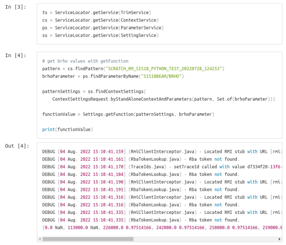
- low-level access to the full Java LSA API
- reads accelerator settings and sends trims
- available fork on GSI gitlab ↗
- origin: CERN gitlab ↗

**pyjapc**

- low-level access to "Java API for Parameter Control" (JAPC)
- GET / SET / SUBSCRIBE to parameters in FESA classes
- origin: CERN gitlab ↗

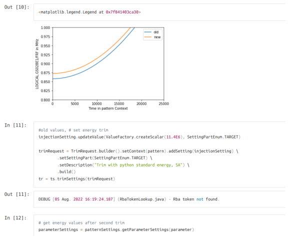
## First Achievements

Git repository with jupyter notebooks ↗ of first examples:



```
In [3]:  ts = ServiceLocator.getService(TrimService)
         cs = ServiceLocator.getService(ContextService)
         ps = ServiceLocator.getService(ParameterService)
         ss = ServiceLocator.getService(SettingService)

In [4]:  # get brho values with getFunction
         pattern = cs.findPattern("SCRATCH_RM_SIS18_PYTHON_TEST_20220728_124253")
         brhoParameter = ps.findParameterByName("SIS18BEAM/BRHO")

         patternSettings = ss.findContextSettings(
             ContextSettingsRequest.byStandAloneContextAndParameters(pattern, Set.of(brhoParameter)))

         functionValue = Settings.getFunction(patternSettings, brhoParameter)

         print(functionValue)

Out [4]: DEBUG [04 Aug. 2022 15:10:41,159] (RmiClientInterceptor.java) - Located RMI stub with URL [rmi:
         DEBUG [04 Aug. 2022 15:10:41,161] (RbaTokenLookup.java) - Rba token not found.
         DEBUG [04 Aug. 2022 15:10:41,170] (TraceIds.java) - setTraceId called with value d7334f20-13f6-
         DEBUG [04 Aug. 2022 15:10:41,184] (RbaTokenLookup.java) - Rba token not found.
         DEBUG [04 Aug. 2022 15:10:41,190] (RmiClientInterceptor.java) - Located RMI stub with URL [rmi:
         DEBUG [04 Aug. 2022 15:10:41,191] (RbaTokenLookup.java) - Rba token not found.
         DEBUG [04 Aug. 2022 15:10:41,316] (RmiClientInterceptor.java) - Located RMI stub with URL [rmi:
         DEBUG [04 Aug. 2022 15:10:41,316] (RbaTokenLookup.java) - Rba token not found.
         DEBUG [04 Aug. 2022 15:10:41,335] (RmiClientInterceptor.java) - Located RMI stub with URL [rmi:
         DEBUG [04 Aug. 2022 15:10:41,335] (RbaTokenLookup.java) - Rba token not found.
         [0.0 NaN, 113000.0 NaN, 226000.0 0.97514166, 242000.0 0.97514166, 258000.0 0.97514166, 259000.0
```

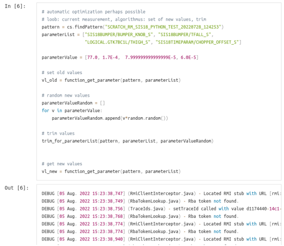
**Figure:** get BRHO ↗



```
Out [10]: <matplotlib.legend.Legend at 0x7f841403ca30>
```

```
In [11]:  #old values, # set energy trim
          injectionSetting.updateValue(ValueFactory.createScalar(11.4E6), SettingPartEnum.TARGET)

          trimRequest = TrimRequest.builder().setContext(pattern).addSetting(injectionSetting) \
                  .setSettingPart(SettingPartEnum.TARGET) \
                  .setDescription("Trim with python standard energy, SA") \
                  .build()
          tr = ts.trimSettings(trimRequest)

Out [11]: DEBUG [05 Aug. 2022 16:19:24,187] (RbaTokenLookup.java) - Rba token not found.

In [12]:  # get energy values after second trim
          parameterSettings = patternSettings.getParameterSettings(parameter)
```

**Figure:** set inj. energy ↗

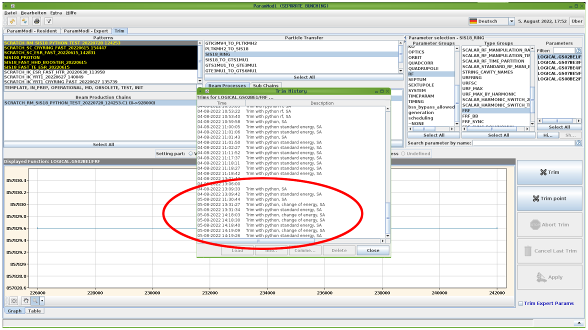


```
In [6]:  # automatic optimization perhaps possible
         # loob: current measurement, algorithmus: set of new values, trim
         pattern = cs.findPattern("SCRATCH_RM_SIS18_PYTHON_TEST_20220728_124253")
         parameterList = ["SIS18BUMPER/BUMPER_KNOB_S", "SIS18BUMPER/TFALL_S",
                          "LOGICAL.GTK7BC1L/THIGH_S", "SIS18TIMEPARAM/CHOPPER_OFFSET_S"]

         parameterValue = [77.0, 1.7E-4,  7.999999999999999E-5, 6.0E-5]

         # set old values
         vl_old = function_get_parameter(pattern, parameterList)

         # random new values
         parameterValueRandom = []
         for v in parameterValue:
             parameterValueRandom.append(v*random.random())

         # trim values
         trim_for_parameterList(pattern, parameterList, parameterValueRandom)



         # get new values
         vl_new = function_get_parameter(pattern, parameterList)

Out [6]: DEBUG [05 Aug. 2022 15:23:38,747] (RmiClientInterceptor.java) - Located RMI stub with URL [rmi:
         DEBUG [05 Aug. 2022 15:23:38,749] (RbaTokenLookup.java) - Rba token not found.
         DEBUG [05 Aug. 2022 15:23:38,756] (TraceIds.java) - setTraceId called with value d1174440-14c1-
         DEBUG [05 Aug. 2022 15:23:38,768] (RbaTokenLookup.java) - Rba token not found.
         DEBUG [05 Aug. 2022 15:23:38,774] (RmiClientInterceptor.java) - Located RMI stub with URL [rmi:
         DEBUG [05 Aug. 2022 15:23:38,774] (RbaTokenLookup.java) - Rba token not found.
         DEBUG [05 Aug. 2022 15:23:38,940] (RmiClientInterceptor.java) - Located RMI stub with URL [rmi:
```

**Figure:** set multi-turn inj. ↗

## Confirmation in TRIM Editor

Trims of new parameter settings are successfully applied:


Trim History

Trims for LOGICAL.GS02BE1/FRF ...

| Time | Description |
|---|---|
| 04-08-2022 10:53:22 | Trim with python rf, SA |
| 04-08-2022 10:53:40 | Trim with python rf, SA |
| 04-08-2022 10:59:58 | Trim with python, SA |
| 04-08-2022 11:00:05 | Trim with python standard energy, SA |
| 04-08-2022 11:01:06 | Trim with python standard energy, SA |
| 04-08-2022 11:01:43 | Trim with python, SA |
| 04-08-2022 11:01:50 | Trim with python standard energy, SA |
| 04-08-2022 11:02:27 | Trim with python standard energy, SA |
| 04-08-2022 11:11:52 | Trim with python standard energy, SA |
| 04-08-2022 11:17:37 | Trim with python standard energy, SA |
| 04-08-2022 11:18:11 | Trim with python standard energy, SA |
| 04-08-2022 11:18:27 | Trim with python standard energy, SA |
| 04-08-2022 11:18:42 | Trim with python standard energy, SA |
| 04-08-2022 13:01:43 | |
| 04-08-2022 13:06:00 | |
| 04-08-2022 13:09:33 | Trim with python, SA |
| 04-08-2022 13:09:42 | Trim with python standard energy, SA |
| 05-08-2022 11:30:44 | Trim with python, SA |
| 05-08-2022 13:31:27 | Trim with python, change of energy, SA |
| 05-08-2022 13:31:34 | Trim with python standard energy, SA |
| 05-08-2022 14:18:03 | Trim with python, change of energy, SA |
| 05-08-2022 14:18:30 | Trim with python, change of energy, SA |
| 05-08-2022 14:18:40 | Trim with python standard energy, SA |
| 05-08-2022 14:19:09 | Trim with python, change of energy, SA |
| 05-08-2022 14:19:26 | Trim with python standard energy, SA |

⟹ YES, we CAN make use of CERN python libraries, python access to accelerator control system with full LSA interface is possible!

still manual hacks necessary, require smooth integration for future!

## Next Steps

Next steps:

- integrate proper set of GSI jars into `cmmnbuild-dep-manager` (currently need to manually copy them into the python package directory, replacing the CERN counterparts)
- potentially provide auto-download of GSI jars via gradle
- demonstrate simple example with python optimisation algorithms: injection energy adjustment via Schottky spectrum evaluation
- address `pyjapc`