Polyglott

Wie wir gelernt haben, kann GraalVM aber noch sehr viel mehr, als einfach nur JavaCode auszuführen. Dazu zählt das native-image building und der Support vieler anderer Programmiersprachen. Native-images schauen wir uns am Ende noch mal genauer an und fokussieren uns erstmal auf die anderen Programmiersprachen.

Um andere Programmiersprachen verwenden zu können müssen wir den entsprechenden Sprachenadapter für Truffle installieren. Dabei werden auch andere Tools, die mit einer normalen Installation der Programmiersprache benötigt werden mit installiert (z.B. npm im Falle von NodeJS).

Wir starten mit einem Beispiel in Python:

```
gu install python
which python
echo 'print("Hi :)"' > hello.py
graalpy hello.py
```

Jetzt haben wir Python Code über die GraalVM ausgeführt, wir können aber noch einen Schritt weiter gehen und die Sprachen miteinander mixen. Z.B. können wir Python Code aus Java aufrufen oder andersrum.

Probiert das ganze Mal selbst aus. Hier z. B. JavaScript bzw. NodeJS

```
gu install nodejs
which js
echo 'console.log("Hi :)"' > hello.js
js hello.js
```

Nun können wir noch einen Schritt weiter gehen; Graal erlaubt es uns, beliebig Sprachen zu mixen. Also können wir auch eine Sprache mit einer anderen kombinieren.

```
import org.graalvm.polyglot.*;
import org.graalvm.polyglot.proxy.*;

 public class HelloPolyglot {

     static String JS_CODE = "console.log('hello TestValue')";

     public static void main(String[] args) {
         System.out.println("Hello Java!");
         try (Context context = Context.create()) {
             context.eval("js", JS_CODE);
         }
     }
 }
```

Wir können auch ganze Berechnungen in eine andere Sprache auslagern. Ist vielleicht bei Data Science Aspekten hilfreich, die üblicherweise in Python besser aufgehoben sind, aber wir anderen Code in Java haben. Gerade wenn Drittanbieter-Software gebraucht wird, dann sind diese nicht immer in allen Sprache vorhanden.

```
import org.graalvm.polyglot.*;
import org.graalvm.polyglot.proxy.*;

public class HelloPolyglot {

    static String JS_CODE = "(function myFun(param){console.log('hello
'+param); return 'WOOP'})";

    public static void main(String[] args) {
        System.out.println("Hello Java!");
        try (Context context = Context.create()) {
            var value = context.eval("js", JS_CODE);
            var returnedValue = value.execute("TestValue");
            System.out.println(returnedValue);
        }
    }
}
```

Auch hier können wir noch einen Schritt weiter gehen und können sogar andere Abhängigkeiten z.B. von NPM verwenden.

npm i cowsayjs
npx cowsayjs Hello

```
import java.util.HashMap;

import org.graalvm.polyglot.*;
import org.graalvm.polyglot.proxy.*;

public class HelloWorld {

    public static void main(String[] args) {
        var options = new HashMap<String, String>();
        options.put("js.commonjs-require", "true");
        options.put("js.commonjs-require-cwd", ".");

        var cx = Context.newBuilder("js")
                .allowExperimentalOptions(true)
                .allowIO(true)
                .options(options)
                .build();

        cx.eval("js", """
            var cowsayjs = require("cowsayjs");

            console.log(cowsayjs.moo("can you see me?"));
                """);
    }
}
```