SBOL Workshop Workthrough

July 8, 2018

1 Introduction

Welcome to the SBOL developers tutorial. This tutorial will cover a few basic and advanced concepts when working with the core SBOL libraries. Primarily, we will learn how to read and write SBOL documents, create new devices, attach experiemental data to a device, and interact with external biological parts repositories such as SynBioHub. The high level goal is to create a new device from components drawn from different sources, attach experiement data to that device, and upload that device to a remote repository.

The tutorial will proceed through the following major steps: 1. Read in a simple device from an SBOL Compliant XML. 2. Extract a promoter from a characterized device from the iGEM interlab study. This data is housed in a remote repoitory. 3. Create a new device with the iGEM promoter swapped in for the original promoter. 4. Add an attachment to the new device representing experimental data. 5. Upload the new device, and its attachment, to SynBioHub

1.1 Prework

1. Create an account on SynBioHub
2. Download the data file they will need to attach
3. Download the SBOL document with the second part

1.2 Python Installation

```
pip install pysbol
```

2 Getting a Device from an SBOL Compliant XML

In this section, we will read in a new device from an SBOL compliant XML and explore its contents.

In [ ]:
```
import sbol

# Start a new SBOL Document to hold the device
cassette_doc = sbol.Document()

# Read in the XML and explore its contents. Notice it is composed of
# componentDefinitions and sequences
cassette_doc.read("gene_cassette.xml")
for obj in cassette_doc:
    print(obj)
```

3 Getting a Device from Synbiohub

In this section, we are going to download a device from SynBioHub. We want the medium strength promoter device from the iGEM interlab study. This device will contain a number of components, sequences, and other objects as well.

```
In [ ]: # Start an interface to the part shop
        part_shop = sbol.PartShop("https://synbiohub.org")

In [ ]: # Search for records from the interlab study
        records = part_shop.search("interlab")
        for record in records:
            print("{}: {}".format(record.displayId, record))

In [ ]: # Read the medium device into a new document
        inter_lab_doc = sbol.Document()
        medium_comp_uri = records[0].identity
        part_shop.pull(medium_comp_uri, inter_lab_doc)

In [ ]: # Explore the medium device document
        for obj in inter_lab_doc:
            print("{}: {}".format(sbol.parseClassName(obj.type), obj))
```

4 Extracting a ComponentDefinition from a Pre-existing Device

In this section, we will extract the medium strength promoter from the interlab study and add it to the cassette document

```
In [ ]: # Extract the medium strength promoter and its sequence
        medium_strength_promoter = inter_lab_doc.getComponentDefinition('http://examples.org/pu
        medium_strength_promoter_sequence = inter_lab_doc.getSequence('http://examples.org/publ

        # And add it to the cassette document
        cassette_doc.addComponentDefinition(medium_strength_promoter)
        cassette_doc.addSequence(medium_strength_promoter_sequence)

        # Show that the promoter is now in the cassette document
        for obj in cassette_doc:
            print("{}: {}".format(sbol.parseClassName(obj.type), obj))
```

5 Creating a New Device

In this section, we will create a new device by swapping in the promoter from the device in interlab study into the device from the XML. We will start by grabbing the necessary parts from cassette document, and then assembling them together into a new device.

```
In [ ]: # Get the dnaComponents from the cassette doc. Notice that we need to
        # use the promoter from the cassette document, not the original interlab
```

```
        # document
        medium_strength_promoter = cassette_doc.getComponentDefinition('http://examples.org/pul

        cds = cassette_doc.getComponentDefinition('http://www.examples.org/ComponentDefinition/
        rbs = cassette_doc.getComponentDefinition('http://www.examples.org/ComponentDefinition/
        terminator = cassette_doc.getComponentDefinition('http://www.examples.org/ComponentDefi
```

In [ ]:
```
        # Create a new gene representing the new device
        # and add it to the cassette document
        modified_gene = sbol.ComponentDefinition("modified_mfg")
        modified_gene.roles = sbol.SO_GENE
        cassette_doc.addComponentDefinition(modified_gene)
```

In [ ]:
```
        # Assemble a new gene
        modified_gene.assemblePrimaryStructure([ medium_strength_promoter, rbs, cds, terminato

        # Note, you need to also add a sequence to the gene.
        modified_gene_seq = sbol.Sequence("modified_mfg")
        cassette_doc.addSequence(modified_gene_seq)
        modified_gene.sequence = modified_gene_seq
```

In [ ]:
```
        # Explore the newly assembled gene
        for comp in modified_gene.components:
            print(comp.displayId)
```

In [ ]:
```
        # This causes a seg fault. In fact, any call to `modified_gene.sequence`
        # causes a seg fault. This is an issue because we cannot add the device
        # to synbiohub without the gene having an attached sequence.


        # modified_gene.assemble()
```

6 Adding Data to a Device via an Attachement

Now that we have a new device, we will add data to it as an attachment

In [ ]:
```
        # TODO I couldn't figure out how to do this :-\
```

7 Uploading the Device back to SynBioHub

Finally, we can create a new collection on SynBioHub with the new device and its attachement.

In [ ]:
```
        import getpass
        user_name = "<ENTER USERNAME HERE>"
        password = getpass.getpass()
```

In [ ]:
```
        part_shop.login(user_name, password)
```

```
In [ ]: cassette_doc.displayId = 'cassette_collection_1'
        cassette_doc.name = 'cassette collection 1'
        cassette_doc.version = '1'
        cassette_doc.description = 'a description of the cassette collection'

In [ ]: part_shop.submit(cassette_doc)
```