Replacing providers

Overview

It is a common need to be able to replace a provider, either with another provider or with a specific value.

Lets look at a situation where we have some "raw" data files and the workflow consists of three steps

- loading the raw data
- cleaning the raw data
- computing a sum of the cleaned data.

In [ ]:
```
from typing import NewType

Filename = NewType('Filename', str)
RawData = NewType('RawData', list)
CleanData = NewType('CleanData', list)
Result = NewType('Result', list)

filesystem = {'raw.txt': list(map(str, range(10)))}

def load(filename: Filename) -> RawData:
    """Load the data from the filename."""
    data = filesystem[filename]
    return RawData(data)

def clean(raw_data: RawData) -> CleanData:
    """Clean the data, convert from str."""
    return CleanData(list(map(float, raw_data)))

def process(clean_data: CleanData) -> Result:
    """Compute the sum of the clean data."""
    return Result(sum(clean_data))
```

In [ ]:
```
import sciline

pipeline = sciline.Pipeline(
    [load, clean, process,],
    params={ Filename: 'raw.txt', })
pipeline
```

Replacing a provider with a value

Select `Result`, the task graph will use the `Filename` input because it needs to read the data from the file system:

In [ ]:
```
pipeline.get(Result)
```

But if the cleaned data has already been produced it is unnecessary to "re-clean" it, in

that case we can proceed directly from the clean data to the compute sum step. To do this we replace the `CleanData` provider with the data that was loaded and cleaned:

In [ ]:
```
data = pipeline.compute(CleanData)
pipeline[CleanData] = data
pipeline
```

Then if we select `Result` the task graph will no longer use the `Filename` input and instead it will proceed directly from the `CleanData` as input:

In [ ]:
```
pipeline.get(Result)
```

In [ ]:
```
pipeline.compute(Result)
```

Replacing a provider with another provider

If the current provider doesn't do what we want it to do we can replace it with another provider.

In [ ]:
```
import sciline

pipeline = sciline.Pipeline(
    [load, clean, process,],
    params={ Filename: 'raw.txt', })
pipeline
```

Let's say the `clean` provider doesn't do all the preprocessing that we want it to do, we also want to remove either the odd or even numbers before processing:

In [ ]:
```
from typing import Literal, Union

Target = NewType('Target', str)

def clean_and_remove_some(raw_data: RawData, target: Target) -> CleanData
    if target == 'odd':
        return [n for n in map(float, raw_data) if n % 2 == 1]
    if target == 'even':
        return [n for n in map(float, raw_data) if n % 2 == 0]
    raise ValueError
```

To replace the old `CleanData` provider we need to use `Pipeline.insert`:

In [ ]:
```
pipeline.insert(clean_and_remove_some)
pipeline[Target] = 'odd'
```

In [ ]:
```
pipeline
```

Now if we select the `Result` we see that the new provider will be used in the computation:

In [ ]:
```
pipeline.get(Result)
```

In [ ]:
```
pipeline.compute(Result)
```