ipython2cwl

Release 0.0.4

Jul 11, 2020

Contents

1 HOW IT WORKS? 3

2 SUPPORTED TYPES 5
- 2.1 Basic Data Types . . . 5
- 2.2 Complex Dumpables Types . . . 5
- 2.3 THAT'S COOL! WHAT ABOUT LIST & OPTIONAL ARGUMENTS? . . . 8

3 SEEMS INTERESTING! WHAT ABOUT A DEMO? 9

4 WHAT IF I WANT TO VALIDATE THAT THE GENERATED SCRIPTS ARE CORRECT? 11

Python Module Index 13

Index 15

IPython2CWL is a tool for converting IPython Jupyter Notebooks to CWL Command Line Tools by simply providing typing annotation.

```
from ipython2cwl.iotypes import CWLFilePathInput, CWLFilePathOutput
import csv
input_filename: 'CWLFilePathInput' = 'data.csv'
with open(input_filename) as f:
  csv_reader = csv.reader(f)
  data = [line for line in csv_reader]
number_of_lines = len(data)
result_file: 'CWLFilePathOutput' = 'number_of_lines.txt'
with open(result_file, 'w') as f:
  f.write(str(number_of_lines))
```

IPython2CWL is based on repo2docker, the same tool used by mybinder. Now, by writing Jupyter Notebook and publishing them, including repo2docker configuration, the community can not only execute the notebooks remotely but can also use them as steps in scientific workflows.

- Install ipython2cwl: `pip install ipython2cwl`
- Ensure that you have docker running
- Create a directory to store the generated cwl files, for example cwlbuild
- Execute `jupyter repo2cwl https://github.com/giannisdoukas/cwl-annotated-jupyter-notebook.git -o cwlbuild`

CHAPTER 1

HOW IT WORKS?

IPython2CWL parses each IPython notebook and finds the variables with the typing annotations. For each input variable, the assigment of that variable will be generalised as a command line argument. Each output variable will be mapped in the cwl description as an output file.

CHAPTER 2

SUPPORTED TYPES

2.1 Basic Data Types

Each variable can be an input or an output. The basic data types are:

- Inputs:
  - CWLFilePathInput
  - CWLBooleanInput
  - CWLStringInput
  - CWLIntInput
- Outputs:
  - CWLFilePathOutput
  - CWLDumpableFile
  - CWLDumpableBinaryFile

2.2 Complex Dumpables Types

Dumpables are variables which are able to be written to a file, but the jupyter notebook developer does not want to write it, for example to avoid the IO overhead. To bypass that, you can use Dumpables annotation. See *dump()* for more details.

class ipython2cwl.iotypes.**CWLBooleanInput**

Use that hint to annotate that a variable is a boolean input. You can use the typing annotation as a string by importing it. At the generated script a command line argument with the name of the variable will be created and the assignment of value will be generalised.

```
>>> dataset1: CWLBooleanInput = True
>>> dataset2: 'CWLBooleanInput' = False
```

class ipython2cwl.iotypes.**CWLDumpable**

Use that class to define custom Dumpables variables.

classmethod dump(*dumper: Callable*, *filename*, **args*, ***kwargs*)

Set the function to be used to dump the variable to a file.

```
>>> import pandas
>>> d: CWLDumpable.dump(d.to_csv, "dumpable.csv", sep="\t", index=False) =␣
↪pandas.DataFrame(
...     [[1,2,3], [4,5,6], [7,8,9]]
... )
```

In that example the converter will add at the end of the script the following line: >>> d.to_csv("dumpable.csv", sep="t", index=False)

Parameters

- **dumper** – The function that has to be called to write the variable to a file.
- **filename** – The name of the generated file. That string must be the first argument in the dumper function. That file will also be mapped as an output in the CWL file.
- **args** – Any positional arguments you want to pass to dumper after the filename
- **kwargs** – Any keyword arguments you want to pass to dumper

class ipython2cwl.iotypes.**CWLDumpableBinaryFile**

Use that annotation to define that a variable should be dumped to a binary file. For example for the annotation:

```
>>> data: CWLDumpableBinaryFile = b"this is text data"
```

the converter will append at the end of the script the following lines:

```
>>> with open('data', 'wb') as f:
...     f.write(data)
```

and at the CWL, the data, will be mapped as a output.

class ipython2cwl.iotypes.**CWLDumpableFile**

Use that annotation to define that a variable should be dumped to a text file. For example for the annotation:

```
>>> data: CWLDumpableFile = "this is text data"
```

the converter will append at the end of the script the following lines:

```
>>> with open('data', 'w') as f:
...     f.write(data)
```

and at the CWL, the data, will be mapped as a output.

class ipython2cwl.iotypes.**CWLFilePathInput**

Use that hint to annotate that a variable is a string-path input. You can use the typing annotation as a string by importing it. At the generated script a command line argument with the name of the variable will be created and the assignment of value will be generalised.

```
>>> dataset1: CWLFilePathInput = './data/data.csv'
>>> dataset2: 'CWLFilePathInput' = './data/data.csv'
```

class ipython2cwl.iotypes.**CWLFilePathOutput**

Use that hint to annotate that a variable is a string-path to an output file. You can use the typing annotation as a string by importing it. The generated file will be mapped as a CWL output.

```
>>> filename: CWLBooleanInput = 'data.csv'
```

class ipython2cwl.iotypes.**CWLIntInput**

Use that hint to annotate that a variable is a integer input. You can use the typing annotation as a string by importing it. At the generated script a command line argument with the name of the variable will be created and the assignment of value will be generalised.

```
>>> dataset1: CWLBooleanInput = 1
>>> dataset2: 'CWLBooleanInput' = 2
```

class ipython2cwl.iotypes.**CWLPNGFigure**

The same with *CWLPNGPlot* but creates new figures before plotting. Use that annotation of you don't want to write multiple graphs in the same image

```
>>> import matplotlib.pyplot as plt
>>> data = [1,2,3]
>>> new_data: 'CWLPNGPlot' = plt.plot(data)
```

the converter will tranform these lines to

```
>>> import matplotlib.pyplot as plt
>>> data = [1,2,3]
>>> plt.figure()
>>> new_data: 'CWLPNGPlot' = plt.plot(data)
>>> plt.savefig('new_data.png')
```

class ipython2cwl.iotypes.**CWLPNGPlot**

Use that annotation to define that after the assigment of that variable the plt.savefig() should be called.

```
>>> import matplotlib.pyplot as plt
>>> data = [1,2,3]
>>> new_data: 'CWLPNGPlot' = plt.plot(data)
```

the converter will tranform these lines to

```
>>> import matplotlib.pyplot as plt
>>> data = [1,2,3]
>>> new_data: 'CWLPNGPlot' = plt.plot(data)
>>> plt.savefig('new_data.png')
```

Note that by default if you have multiple plot statements in the same notebook will be written in the same file. If you want to write them in separates you have to do it in separate figures. To do that in your notebook you have to create a new figure before the plot command or use the CWLPNGFigure.

```
>>> import matplotlib.pyplot as plt
>>> data = [1,2,3]
>>> plt.figure()
>>> new_data: 'CWLPNGPlot' = plt.plot(data)
```

class ipython2cwl.iotypes.**CWLStringInput**

Use that hint to annotate that a variable is a string input. You can use the typing annotation as a string by importing it. At the generated script a command line argument with the name of the variable will be created and the assignment of value will be generalised.

```
>>> dataset1: CWLBooleanInput = 'this is a message input'
>>> dataset2: 'CWLBooleanInput' = 'yet another message input'
```

2.3 THAT'S COOL! WHAT ABOUT LIST & OPTIONAL ARGUMENTS?

The basic input data types can be combined with the List and Optional annotations. For example, write the following annotation:

```
file_inputs: List[CWLFilePathInput] = ['data1.txt', 'data2.txt', 'data3.txt']
example: Optional[CWLStringInput] = None
```

CHAPTER 3

SEEMS INTERESTING! WHAT ABOUT A DEMO?

If you would like to see a demo before you want to start annotating your notebooks check here! github.com/giannisdoukas/ipython2cwl-demo

CHAPTER 4

WHAT IF I WANT TO VALIDATE THAT THE GENERATED SCRIPTS ARE CORRECT?

All the generated scripts are stored in the docker image under the directory `/app/cwl/bin`. You can see the list of the files by running `docker run [IMAGE_ID] find /app/cwl/bin/ -type f`.

Python Module Index

i

ipython2cwl.iotypes, 5

Index

C

CWLBooleanInput (*class in ipython2cwl.iotypes*), 5
CWLDumpable (*class in ipython2cwl.iotypes*), 6
CWLDumpableBinaryFile (*class in ipython2cwl.iotypes*), 6
CWLDumpableFile (*class in ipython2cwl.iotypes*), 6
CWLFilePathInput (*class in ipython2cwl.iotypes*), 6
CWLFilePathOutput (*class in ipython2cwl.iotypes*), 6
CWLIntInput (*class in ipython2cwl.iotypes*), 7
CWLPNGFigure (*class in ipython2cwl.iotypes*), 7
CWLPNGPlot (*class in ipython2cwl.iotypes*), 7
CWLStringInput (*class in ipython2cwl.iotypes*), 7

D

dump() (*ipython2cwl.iotypes.CWLDumpable class method*), 6

I

ipython2cwl.iotypes (*module*), 5