Python & C++ clang-repl integration

Ioana Ifrim

Progress Stages

- Integrate Python in clang-repl
- Expand Python utilities to include `ExecScriptPython` `ExecSimplePythonCommand`
- Manage Python’s global dict form clang-repl
- Update Python’s dictionary with C++ variables
- Find solutions to bring this development into Notebooks and have C++ and Python working together in the same notebook
- Next steps

```
./bin/clang-repl
clang-repl> int i = 12;
clang-repl> python
>>> import numpy as np
>>> a = np.asarray([1.1, 2.2, 3.3])
>>> a
array([ 1.1,  2.2,  3.3])
>>> quit()
clang-repl> int j = 10;
```

```
./bin/clang-repl
clang-repl> int i = 12;
clang-repl> python
>>> import numpy as np
>>> a = np.asarray([1.1, 2.2, 3.3])
>>> a
array([ 1.1,  2.2,  3.3])
>>> quit()
clang-repl> int j = 10;
```

Task

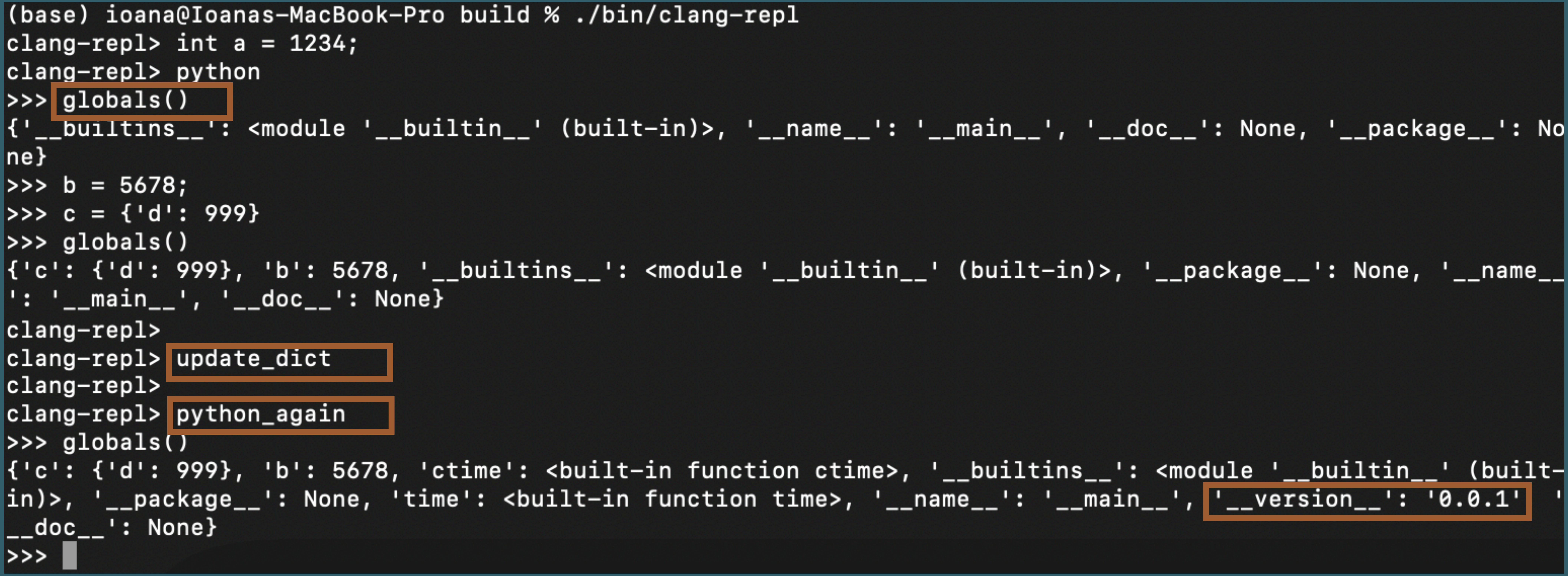
Integrate Python in clang-repl

```
./bin/clang-repl
clang-repl> int i = 12;
Calls a `test.py` script which prints the number of arguments
clang-repl> python_exec_script:

Runs a predefined Python command; normally the command would be passed
along with `simple_command`
clang-repl> simple_command:
('This is printed from Python: Today is', 'Tue May 24 19:47:05 2022')
```

Task

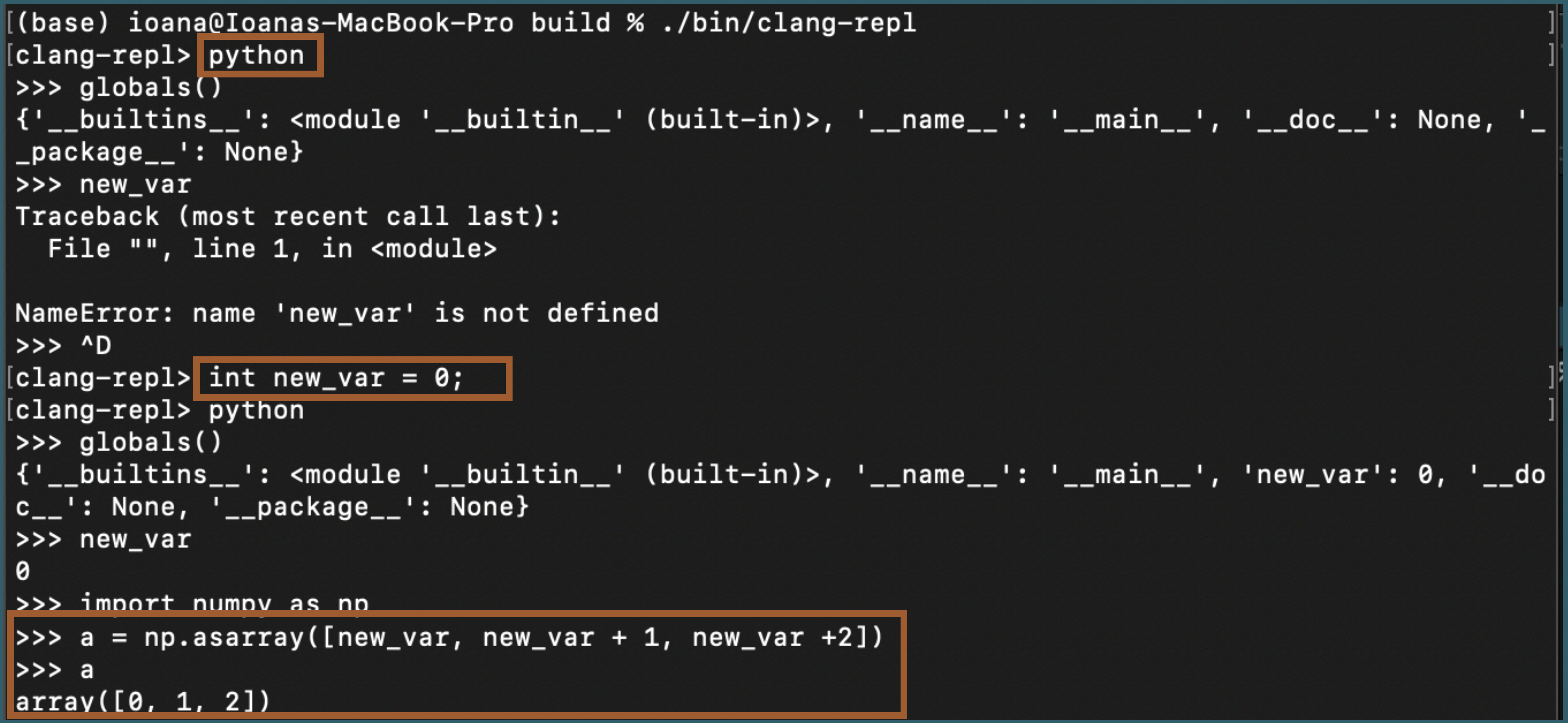
Expand Python utilities in clang-repl



```
(base) ioana@Ioanas-MacBook-Pro build % ./bin/clang-repl
clang-repl> int a = 1234;
clang-repl> python
>>> globals()
{'__builtins__': <module '__builtin__' (built-in)>, '__name__': '__main__', '__doc__': None, '__package__': No
ne}
>>> b = 5678;
>>> c = {'d': 999}
>>> globals()
{'c': {'d': 999}, 'b': 5678, '__builtins__': <module '__builtin__' (built-in)>, '__package__': None, '__name__
': '__main__', '__doc__': None}
clang-repl>
clang-repl> update_dict
clang-repl>
clang-repl> python_again
>>> globals()
{'c': {'d': 999}, 'b': 5678, 'ctime': <built-in function ctime>, '__builtins__': <module '__builtin__' (built-
in)>, '__package__': None, 'time': <built-in function time>, '__name__': '__main__', '__version__': '0.0.1', '
__doc__': None}
>>> 
```

Task

Manage Python's global dict form clang-repl



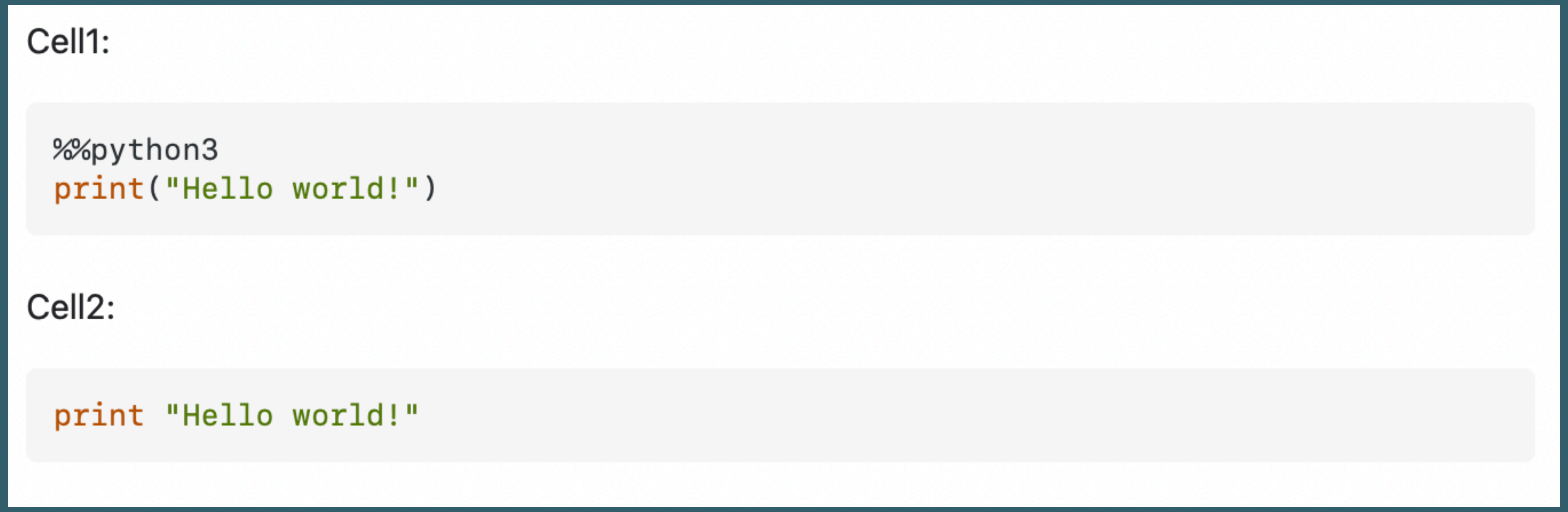
```
[(base) ioana@Ioanas-MacBook-Pro build % ./bin/clang-repl
[clang-repl> python
>>> globals()
{'__builtins__': <module '__builtin__' (built-in)>, '__name__': '__main__', '__doc__': None, '_
_package__': None}
>>> new_var
Traceback (most recent call last):
  File "", line 1, in <module>

NameError: name 'new_var' is not defined
>>> ^D
[clang-repl> int new_var = 0;
[clang-repl> python
>>> globals()
{'__builtins__': <module '__builtin__' (built-in)>, '__name__': '__main__', 'new_var': 0, '__do
c__': None, '__package__': None}
>>> new_var
0
>>> import numpy as np
>>> a = np.asarray([new_var, new_var + 1, new_var +2])
>>> a
array([0, 1, 2])
```

Task

Update Python's dictionary with C++ vars

Jupyter Notebook magic commands



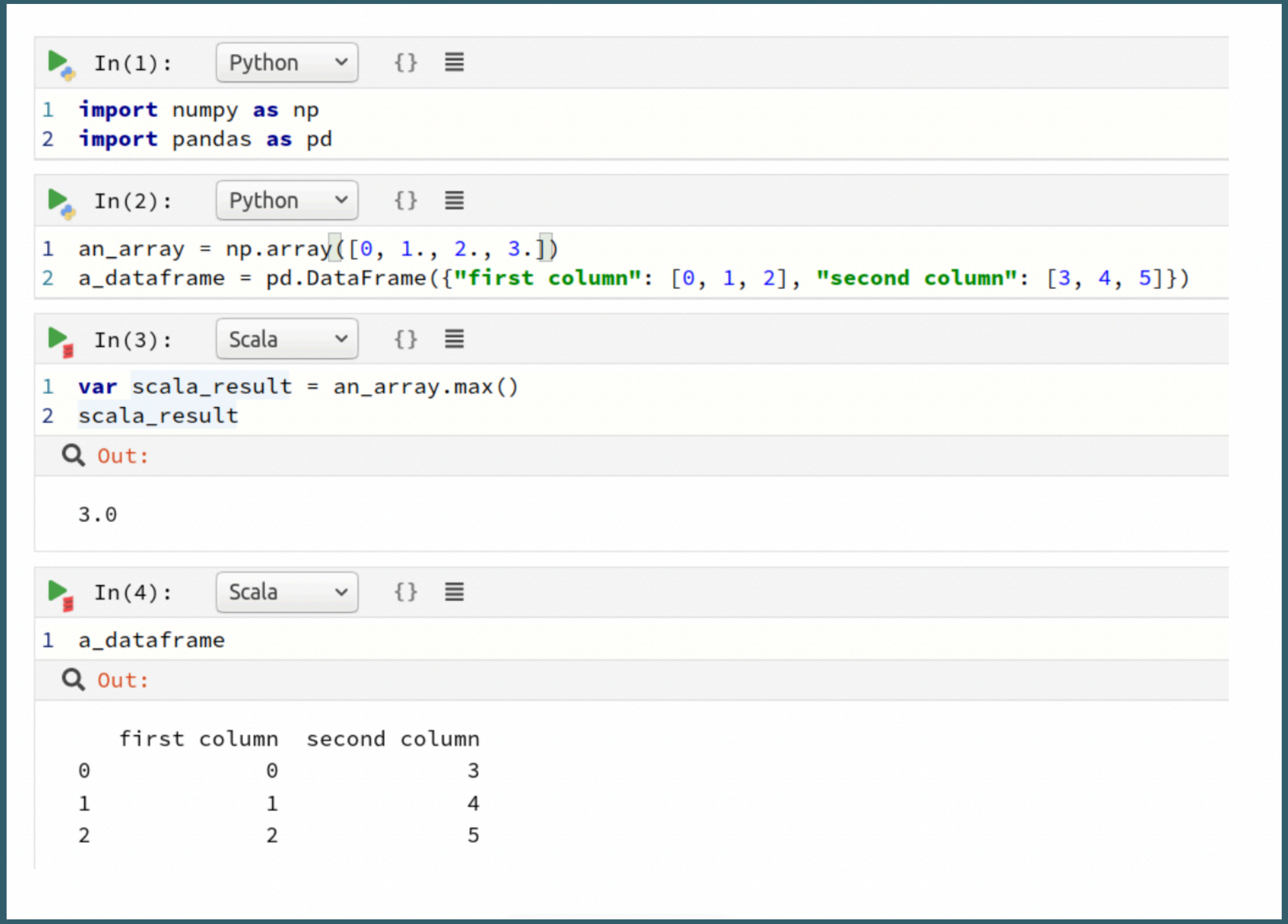
Cell1:

```
%%python3
print("Hello world!")
```

Cell2:

```
print "Hello world!"
```

Polynote



In(1): Python

```
import numpy as np
import pandas as pd
```

In(2): Python

```
an_array = np.array([0, 1., 2., 3.])
a_dataframe = pd.DataFrame({"first column": [0, 1, 2], "second column": [3, 4, 5]})
```

In(3): Scala

```
var scala_result = an_array.max()
scala_result
```

Out:

```
3.0
```

In(4): Scala

```
a_dataframe
```

Out:

```
   first column  second column
0             0              3
1             1              4
2             2              5
```

https://github.com/polynote/polynote

Task | Python and C++ in Notebooks options

Next Steps

- **WIP** Update the global dict with var addresses, not value => synchronise updates from both C++ and Python
- Extend beyond `int` type for dict updates (PyObject *s; s = PyInt_FromLong(value);
- Investigate both alternatives (magic commands / Polynote) for working with both Python and C++ in the same Notebook

Thank you!

- questions -