

# Introduction to Parallel Processing

Lecture 6 : MPI for Python

Professor Amanda Bierenz

# MPI API

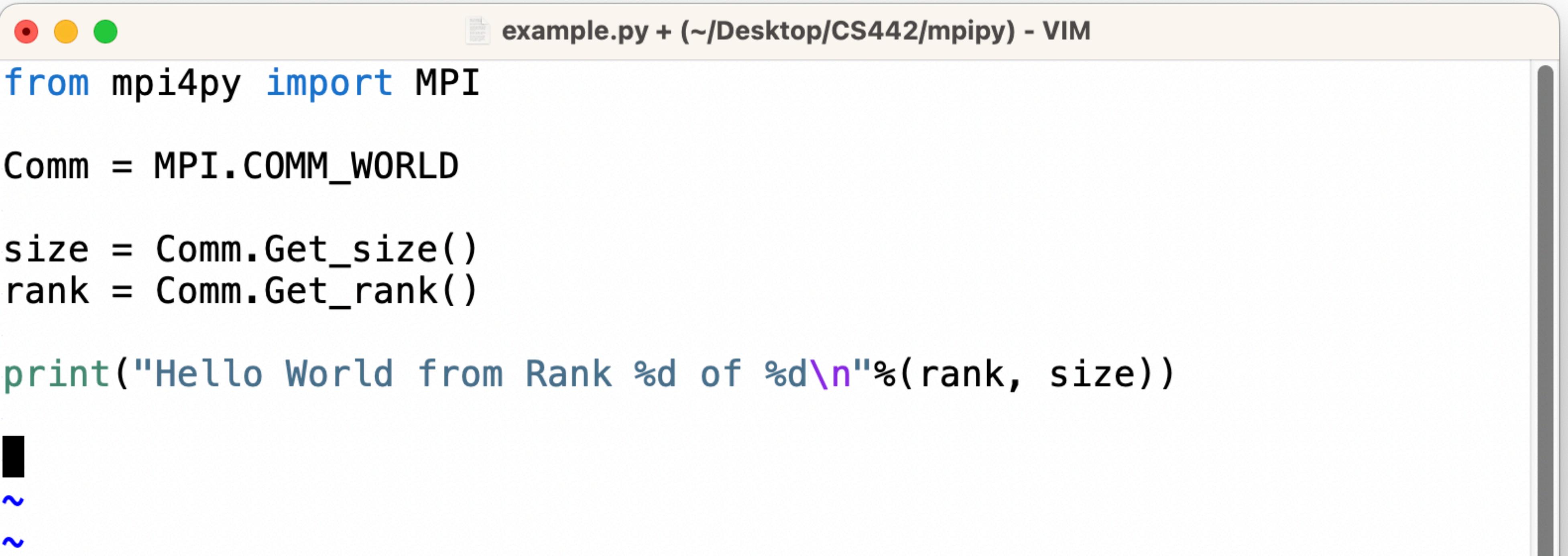
- Technically, exists only for C, C++, and Fortran
- Python interfaces to the MPI API
  - Not an implementation of the interface, calls an existing implementation under the hood
  - Does not include every part of the MPI API

# Installing MPI4PY

- First, you need to install an implementation of MPI (e.g. MPICH, OpenMPI, MVAPICH, etc)
- Then, you can install MPI4PY to wrap that installed implementation
  - pip install mpi4py
  - <https://mpi4py.readthedocs.io/en/stable/install.html>
  - **CARC** : module load miniconda3  
conda create -n mpi\_numpy mpi mpi4py numpy  
[https://www.youtube.com/watch?v=JrCMI3\\_yCA0](https://www.youtube.com/watch?v=JrCMI3_yCA0)

# Hello World

- No MPI\_Init/MPI\_Finalize needed in Python
- Simply import MPI from mpi4py



A screenshot of a VIM editor window titled "example.py + (~/Desktop/CS442/mpipy) - VIM". The code in the buffer is:

```
from mpi4py import MPI

Comm = MPI.COMM_WORLD

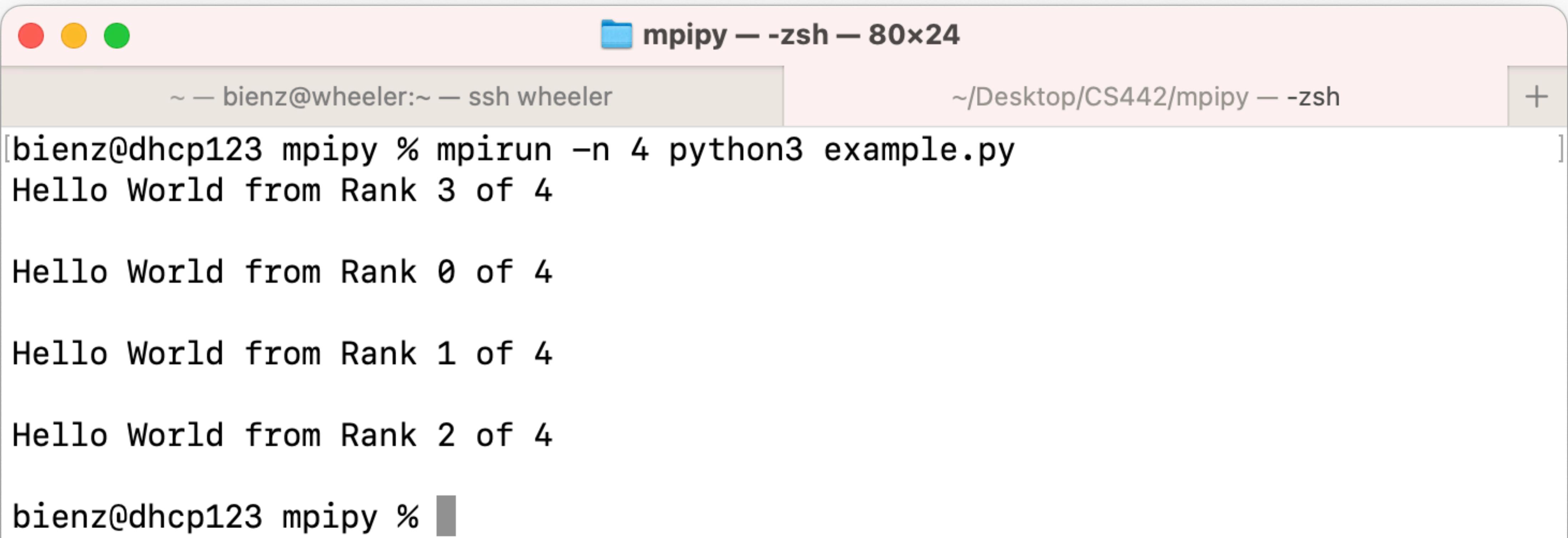
size = Comm.Get_size()
rank = Comm.Get_rank()

print("Hello World from Rank %d of %d\n"%(rank, size))
```

The code uses the mpi4py library to print "Hello World" from each MPI rank. The VIM status bar at the bottom shows the file name and path.

# Running Hello World

- No compilation as python is scripted language
- Still need to launch with mpirun



The screenshot shows a terminal window with the title bar 'mpipy -- zsh -- 80x24'. The window has three colored window control buttons (red, yellow, green) on the top-left. The terminal interface includes a menu bar with 'File', 'Edit', 'View', 'Shell', 'Help', and a tab bar with two tabs: '~ — bienz@wheeler:~ — ssh wheeler' and '~/Desktop/CS442/mpipy — zsh'. A '+' button is located on the right side of the tab bar. The main pane displays the command-line output:

```
[bienz@dhcp123 mpipy % mpirun -n 4 python3 example.py
Hello World from Rank 3 of 4
Hello World from Rank 0 of 4
Hello World from Rank 1 of 4
Hello World from Rank 2 of 4
bienz@dhcp123 mpipy %
```

# Point to Point Communication



example.py (~/Desktop/CS442/mpipy) - VIM

```
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

size = Comm.Get_size()
rank = Comm.Get_rank()

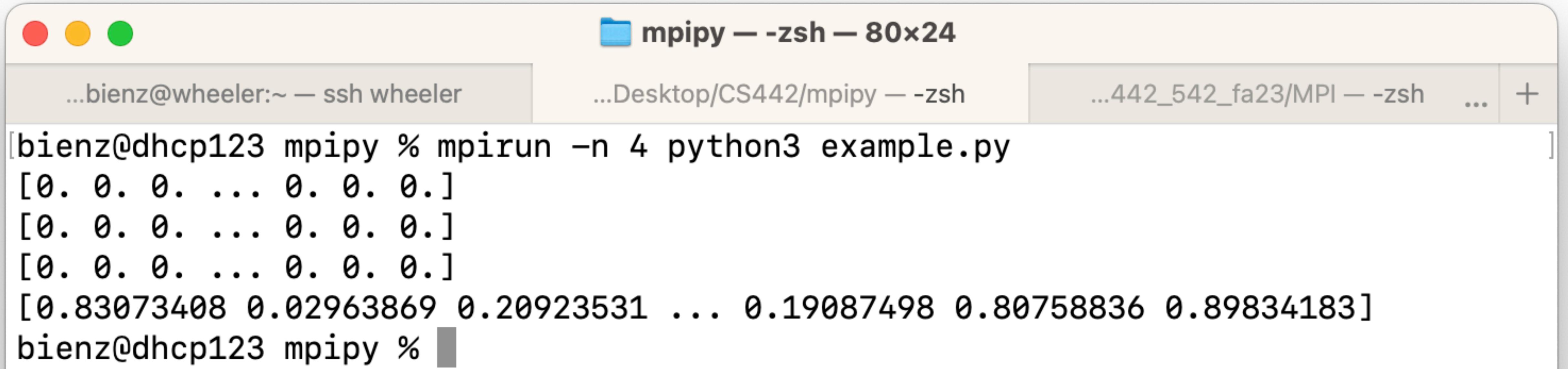
proc = rank - 1
if (rank % 2 == 0):
    proc = rank + 1

size = 10000
array = np.random.rand(size);
array_recv = np.zeros(size);

if (rank == 0):
    Comm.Send(array, 1, tag=1234)
elif (rank == 1):
    Comm.Recv(array_recv, 0, tag=1234)

print(array_recv)
```

# Point to Point Communication



The screenshot shows a macOS terminal window with three tabs:

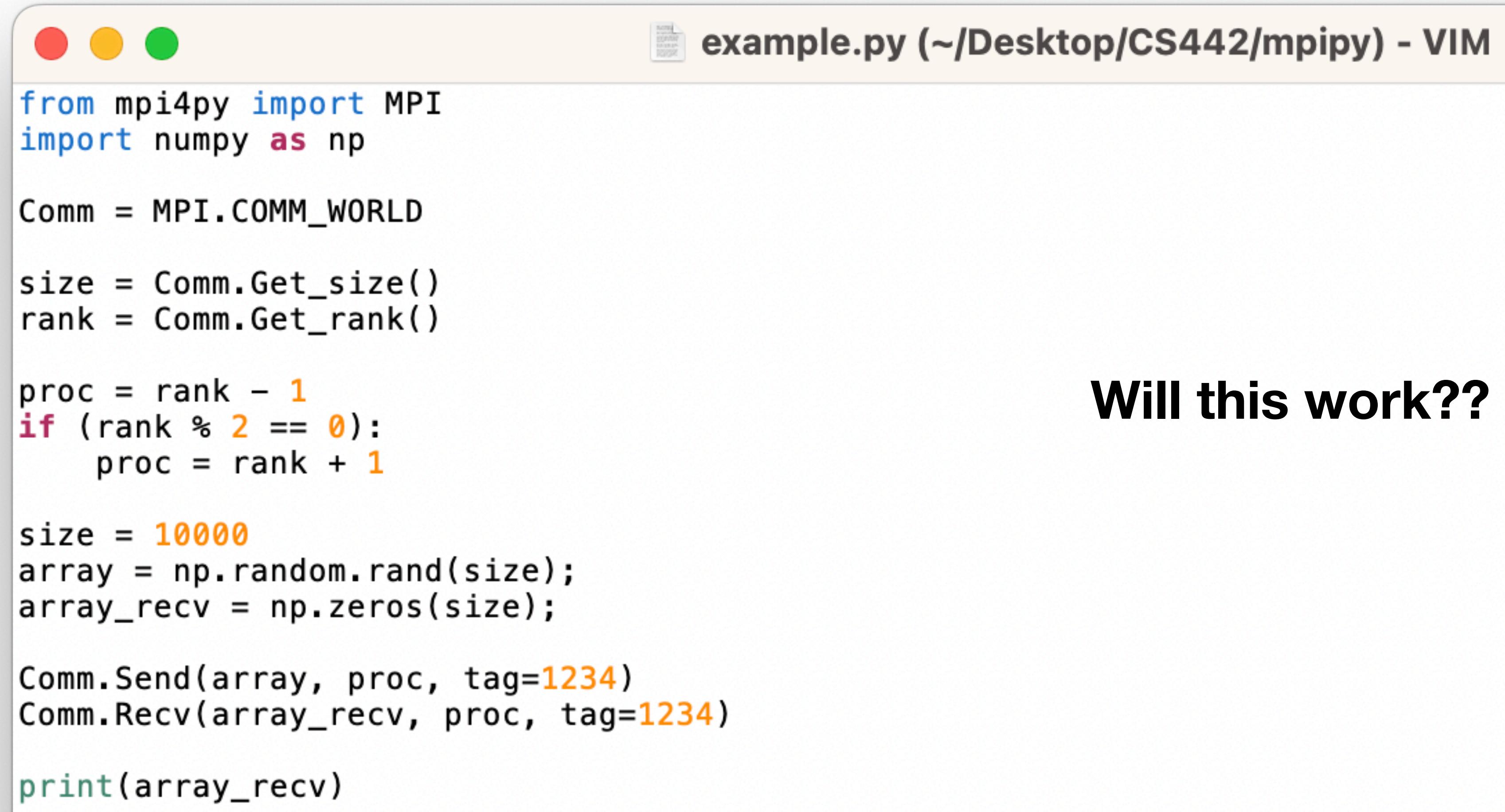
- ...bienz@wheeler:~ — ssh wheeler
- ...Desktop/CS442/mpipy — zsh
- ...442\_542\_fa23/MPI — zsh ... +

The active tab displays the following MPI output:

```
[bienz@dhcp123 mpipy % mpirun -n 4 python3 example.py
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0.83073408 0.02963869 0.20923531 ... 0.19087498 0.80758836 0.89834183]
bienz@dhcp123 mpipy %
```

# Point to Point Communication

- Want every process to exchange with neighboring (e.g. rank 0 exchanges with rank 1, rank 2 with rank 3, etc)



The image shows a terminal window titled "example.py (~/Desktop/CS442/mpipy) - VIM". The code inside the terminal is as follows:

```
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

size = Comm.Get_size()
rank = Comm.Get_rank()

proc = rank - 1
if (rank % 2 == 0):
    proc = rank + 1

size = 10000
array = np.random.rand(size);
array_recv = np.zeros(size);

Comm.Send(array, proc, tag=1234)
Comm.Recv(array_recv, proc, tag=1234)

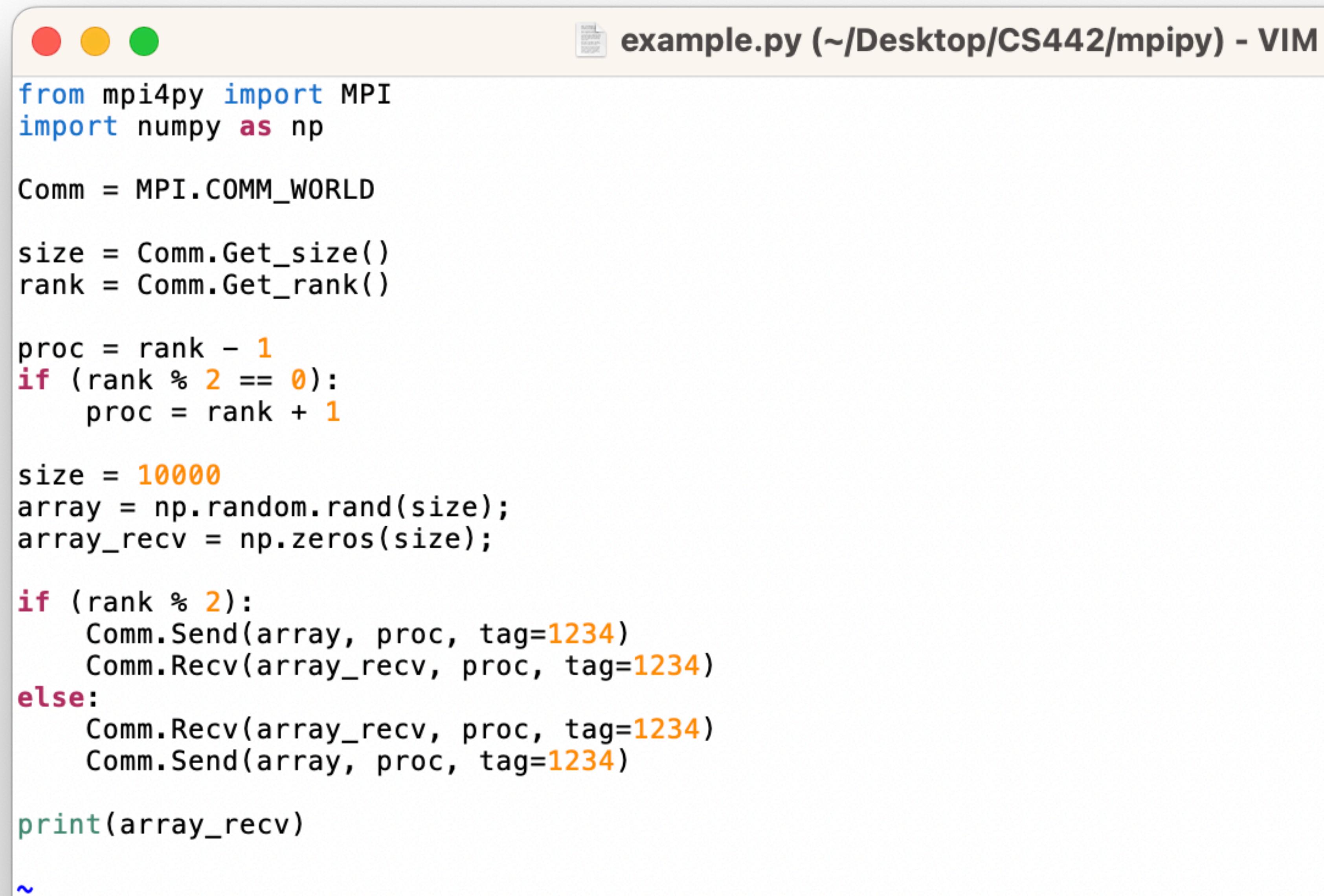
print(array_recv)
```

**Will this work??**

# Point to Point Communication

- Python is wrapping standard MPI implementation, meaning if there is an issue in MPI it will also be an issue in MPI4PY
- All processes are sending at same time, so all may hang waiting for matching receive!

# Point to Point Communication



The screenshot shows a terminal window titled "example.py (~/Desktop/CS442/mpipy) - VIM". The code is written in Python and uses the mpi4py library to demonstrate point-to-point communication between MPI processes.

```
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

size = Comm.Get_size()
rank = Comm.Get_rank()

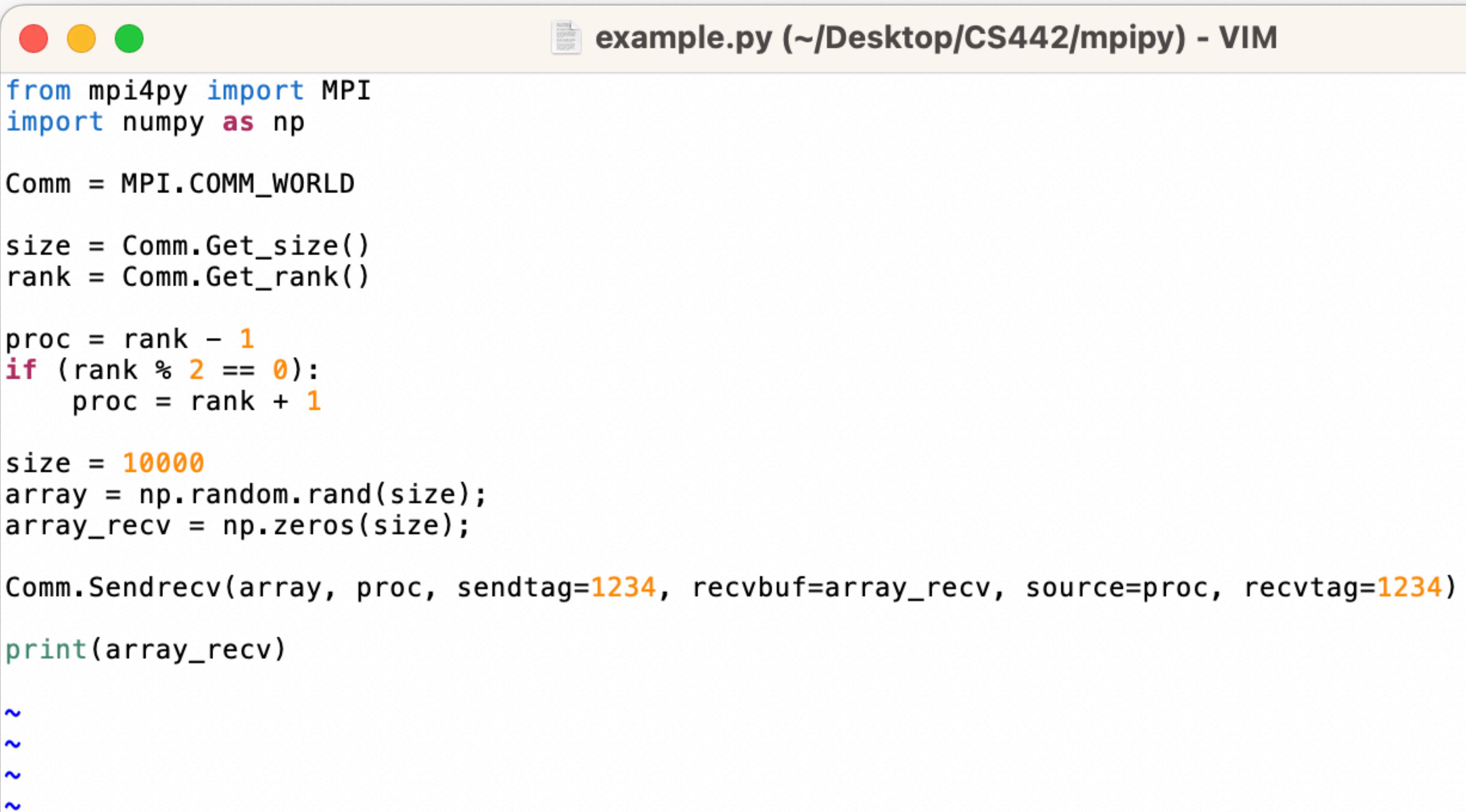
proc = rank - 1
if (rank % 2 == 0):
    proc = rank + 1

size = 10000
array = np.random.rand(size);
array_recv = np.zeros(size);

if (rank % 2):
    Comm.Send(array, proc, tag=1234)
    Comm.Recv(array_recv, proc, tag=1234)
else:
    Comm.Recv(array_recv, proc, tag=1234)
    Comm.Send(array, proc, tag=1234)

print(array_recv)
```

# MPI Sendrecv Still Exists



The screenshot shows a VIM editor window with three colored status bars at the top (red, yellow, green). The title bar reads "example.py (~/Desktop/CS442/mpipy) - VIM". The code in the editor is as follows:

```
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

size = Comm.Get_size()
rank = Comm.Get_rank()

proc = rank - 1
if (rank % 2 == 0):
    proc = rank + 1

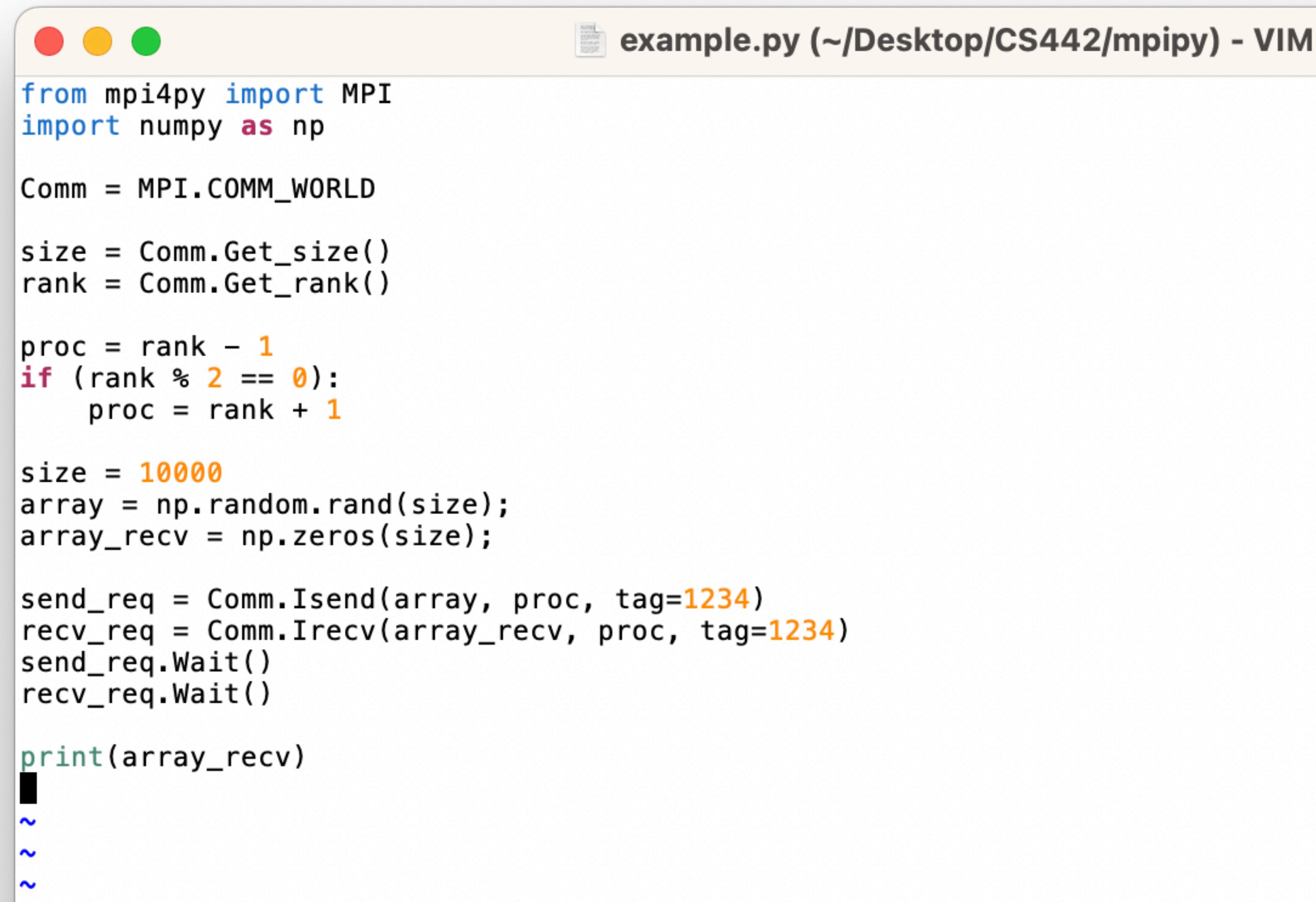
size = 10000
array = np.random.rand(size);
array_recv = np.zeros(size);

Comm.Sendrecv(array, proc, sendtag=1234, recvbuf=array_recv, source=proc, recvtag=1234)

print(array_recv)

~
```

# As Does Non-Blocking Communication



The image shows a screenshot of a VIM editor window titled "example.py (~/Desktop/CS442/mpipy) - VIM". The code in the buffer is as follows:

```
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

size = Comm.Get_size()
rank = Comm.Get_rank()

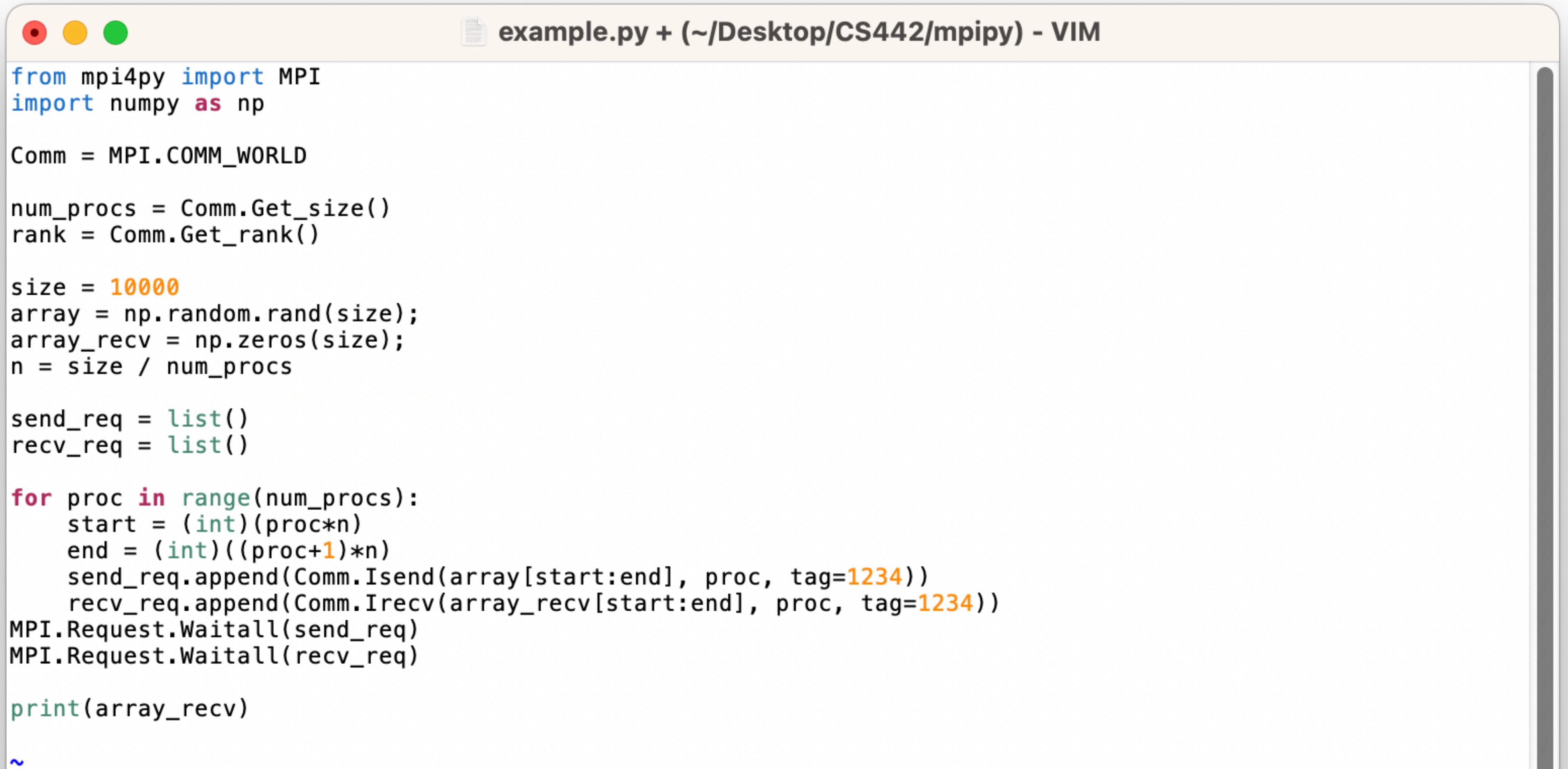
proc = rank - 1
if (rank % 2 == 0):
    proc = rank + 1

size = 10000
array = np.random.rand(size);
array_recv = np.zeros(size);

send_req = Comm.Isend(array, proc, tag=1234)
recv_req = Comm.Irecv(array_recv, proc, tag=1234)
send_req.Wait()
recv_req.Wait()

print(array_recv)
~
```

# Can Send to Everyone



The image shows a screenshot of a VIM editor window. The title bar reads "example.py + (~/Desktop/CS442/mpipy) - VIM". The code in the editor is as follows:

```
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

num_procs = Comm.Get_size()
rank = Comm.Get_rank()

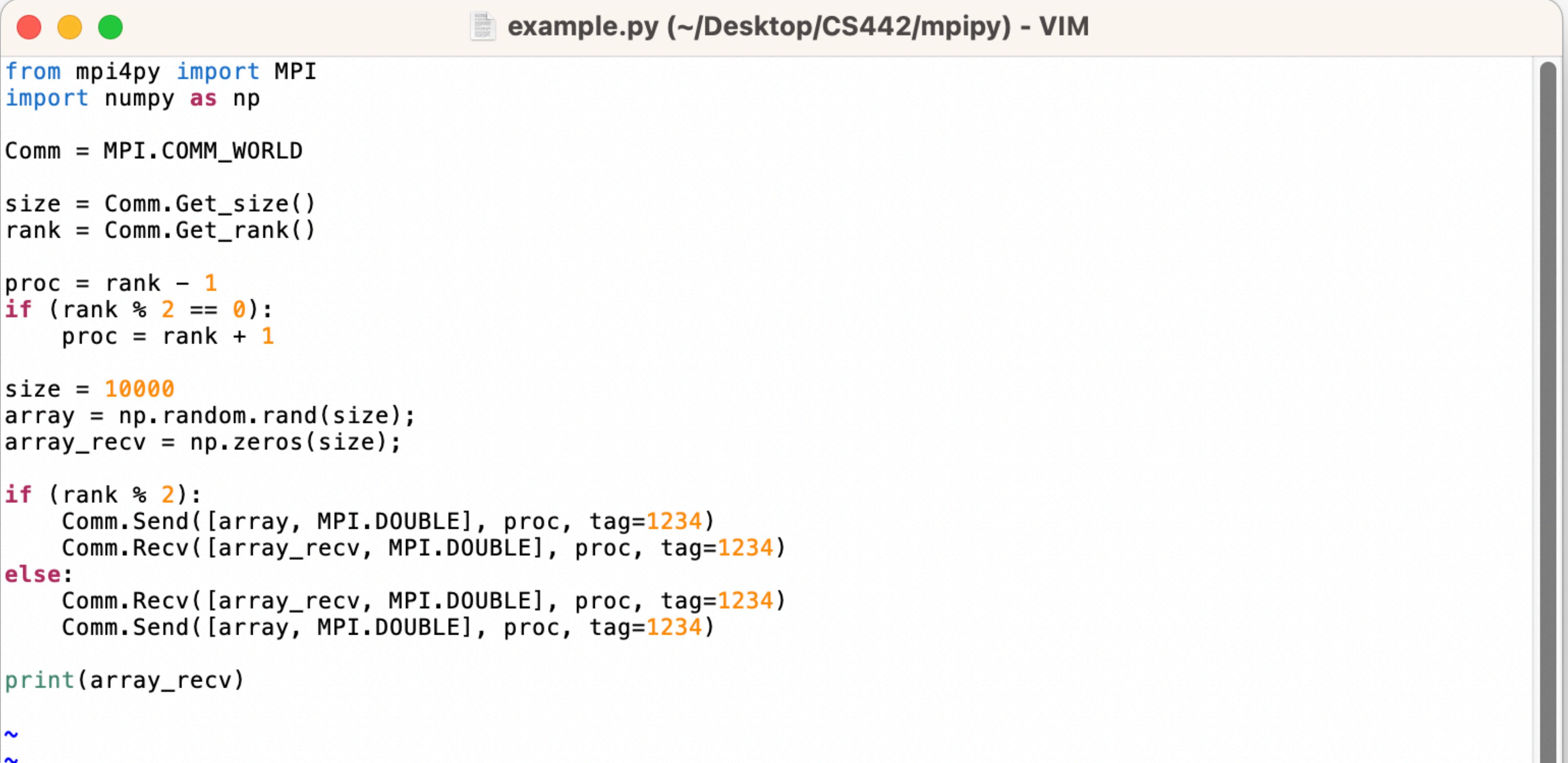
size = 10000
array = np.random.rand(size);
array_recv = np.zeros(size);
n = size / num_procs

send_req = list()
recv_req = list()

for proc in range(num_procs):
    start = (int)(proc*n)
    end = (int)((proc+1)*n)
    send_req.append(Comm.Isend(array[start:end], proc, tag=1234))
    recv_req.append(Comm.Irecv(array_recv[start:end], proc, tag=1234))
MPI.Request.Waitall(send_req)
MPI.Request.Waitall(recv_req)

print(array_recv)
~
```

# Can Specify Data Type



The image shows a screenshot of a VIM editor window titled "example.py (~/Desktop/CS442/mpipy) - VIM". The code in the buffer is as follows:

```
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

size = Comm.Get_size()
rank = Comm.Get_rank()

proc = rank - 1
if (rank % 2 == 0):
    proc = rank + 1

size = 10000
array = np.random.rand(size);
array_recv = np.zeros(size);

if (rank % 2):
    Comm.Send([array, MPI.DOUBLE], proc, tag=1234)
    Comm.Recv([array_recv, MPI.DOUBLE], proc, tag=1234)
else:
    Comm.Recv([array_recv, MPI.DOUBLE], proc, tag=1234)
    Comm.Send([array, MPI.DOUBLE], proc, tag=1234)

print(array_recv)

~
```

# MPI Collectives : Bcast



example.py (~/Desktop/CS442/mpipy) - VIM

```
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

size = Comm.Get_size()
rank = Comm.Get_rank()

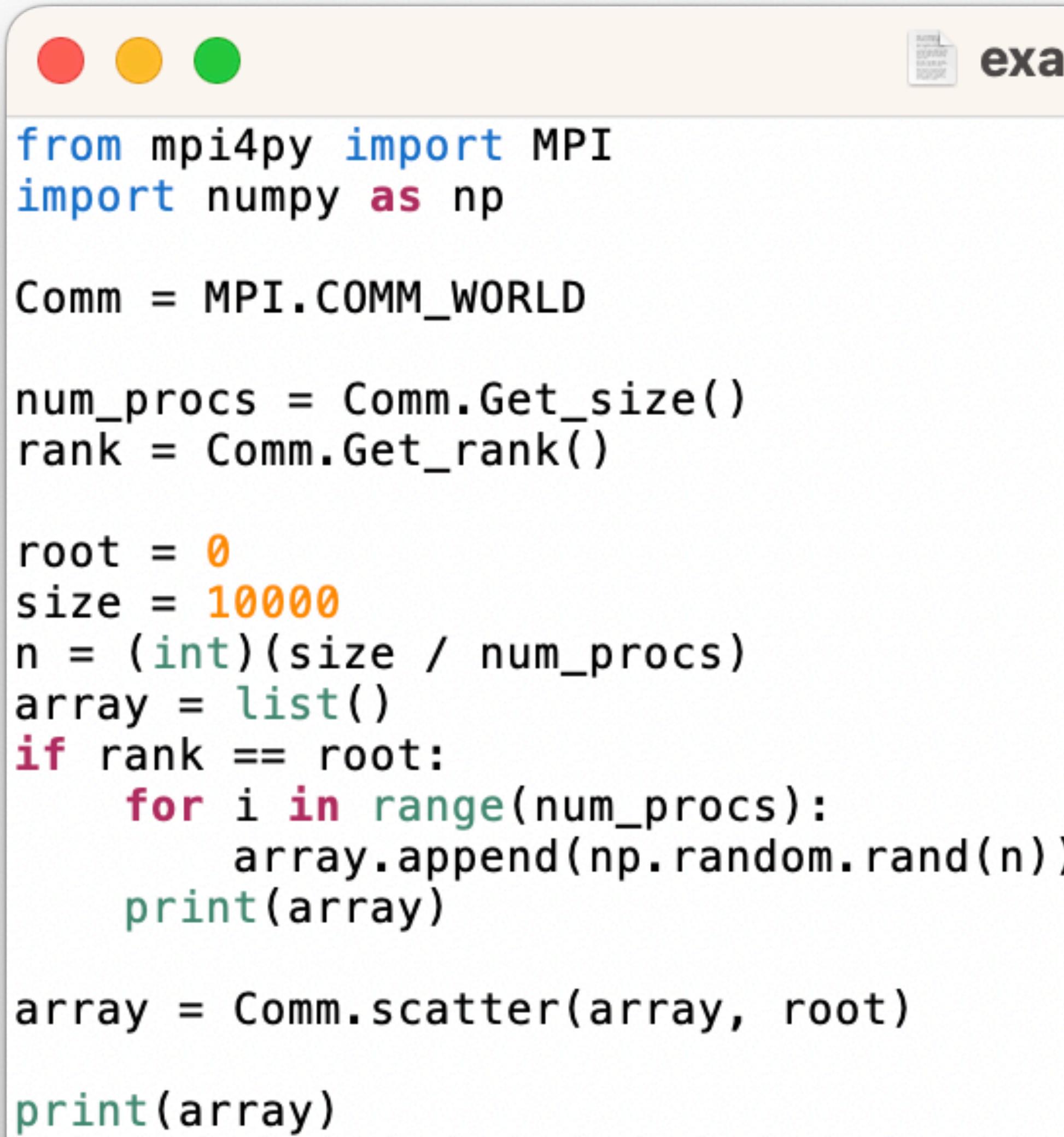
root = 0
size = 10000
array = ""
if rank == root:
    array = np.random.rand(size)

array = Comm.bcast(array, root)

print(array)
```

```
...bienz@wheeler:~ — ssh wheeler ...Desktop/CS442/mpipy — zsh ...442_542_fa23/MPI — zsh ...
[ bienz@dhcp123 mpipy % mpirun -n 4 python3 example.py
[ 0.80592772 0.11930063 0.38635772 ... 0.56694023 0.81398224 0.1095385 ]
[ 0.80592772 0.11930063 0.38635772 ... 0.56694023 0.81398224 0.1095385 ]
[ 0.80592772 0.11930063 0.38635772 ... 0.56694023 0.81398224 0.1095385 ]
[ 0.80592772 0.11930063 0.38635772 ... 0.56694023 0.81398224 0.1095385 ]
bienz@dhcp123 mpipy % ]
```

# MPI Collectives : Scatter



A screenshot of a VIM editor window titled "example.py (~/Desktop/CS442/mpipy) - VIM". The code in the buffer demonstrates MPI scatter operations:

```
from mpi4py import MPI
import numpy as np

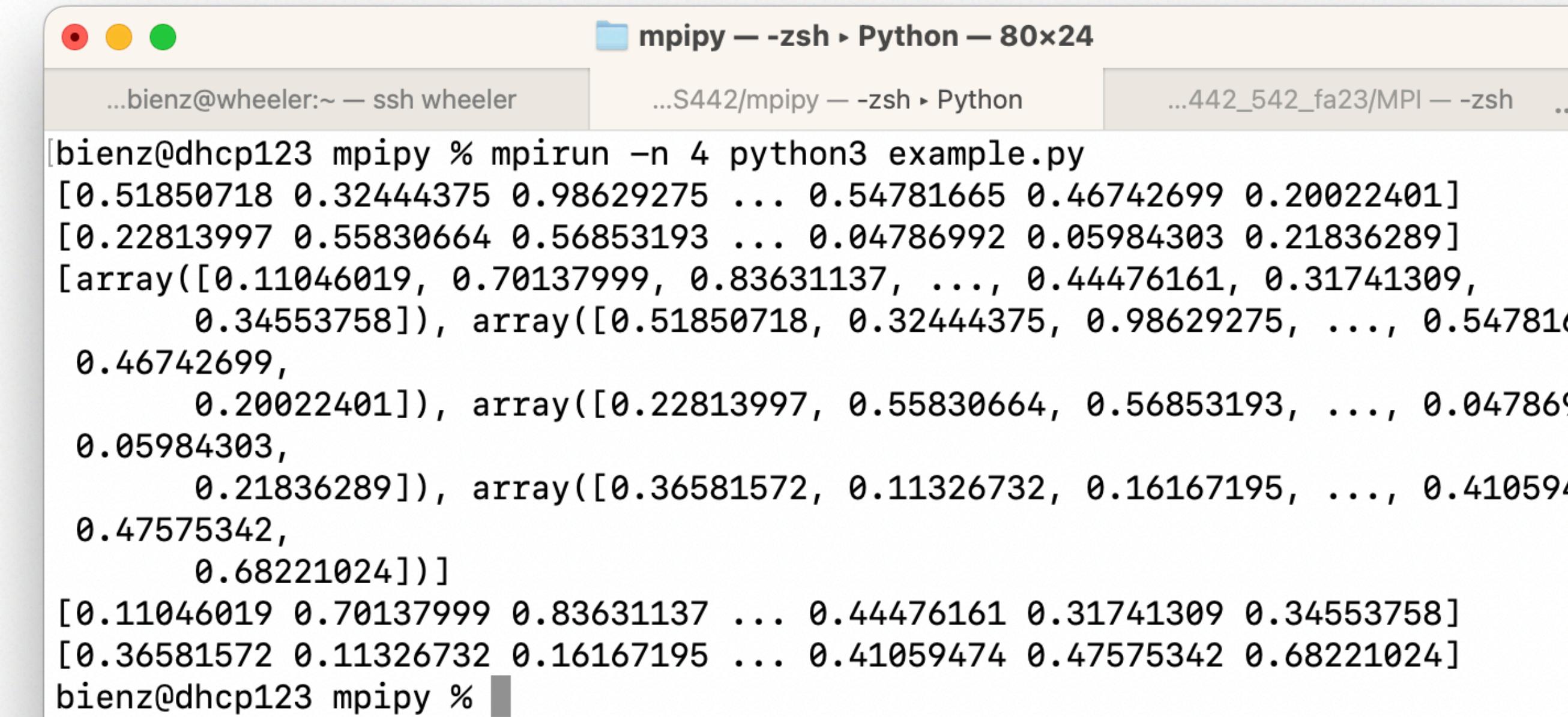
Comm = MPI.COMM_WORLD

num_procs = Comm.Get_size()
rank = Comm.Get_rank()

root = 0
size = 10000
n = (int)(size / num_procs)
array = list()
if rank == root:
    for i in range(num_procs):
        array.append(np.random.rand(n))
    print(array)

array = Comm.scatter(array, root)

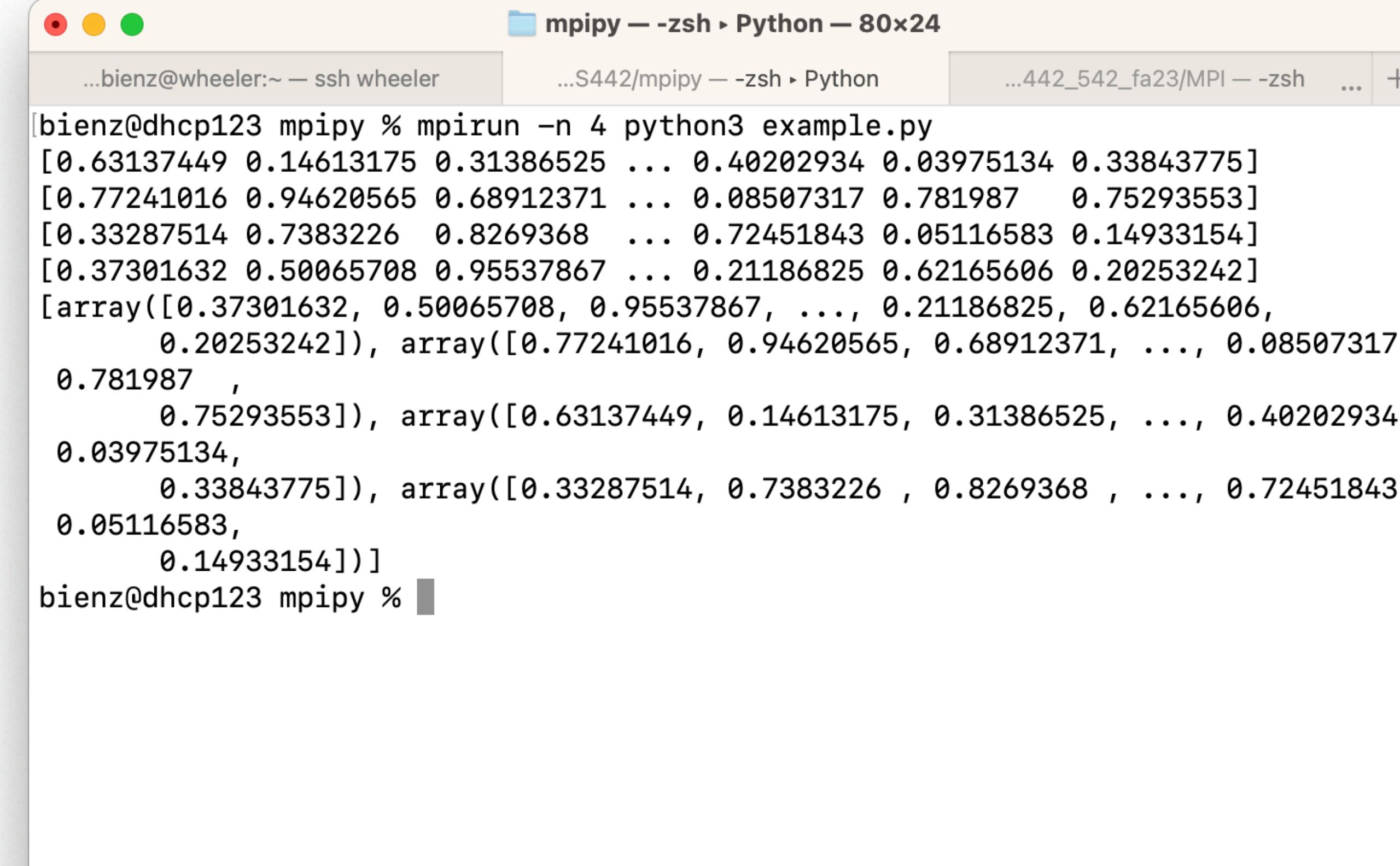
print(array)
```



A screenshot of a terminal window titled "mpipy --zsh > Python -- 80x24". The terminal shows the execution of the MPI code and the resulting scattered arrays:

```
[bienz@dhcp123 mpipy % mpirun -n 4 python3 example.py
[0.51850718 0.32444375 0.98629275 ... 0.54781665 0.46742699 0.20022401]
[0.22813997 0.55830664 0.56853193 ... 0.04786992 0.05984303 0.21836289]
[array([0.11046019, 0.70137999, 0.83631137, ..., 0.44476161, 0.31741309,
       0.34553758]), array([0.51850718, 0.32444375, 0.98629275, ..., 0.54781665,
       0.46742699,
       0.20022401]), array([0.22813997, 0.55830664, 0.56853193, ..., 0.04786992,
       0.05984303,
       0.21836289]), array([0.36581572, 0.11326732, 0.16167195, ..., 0.41059474,
       0.47575342,
       0.68221024])]
[0.11046019 0.70137999 0.83631137 ... 0.44476161 0.31741309 0.34553758]
[0.36581572 0.11326732 0.16167195 ... 0.41059474 0.47575342 0.68221024]
bienz@dhcp123 mpipy %
```

# MPI Collectives : Gather



The terminal window shows the command `mpirun -n 4 python3 example.py` running on a system with 4 MPI processes. The output displays four arrays of floating-point numbers, each 1000 elements long, gathered from the four processes.

```
example.py (~/Desktop/CS442/mpipy) - VIM
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

num_procs = Comm.Get_size()
rank = Comm.Get_rank()

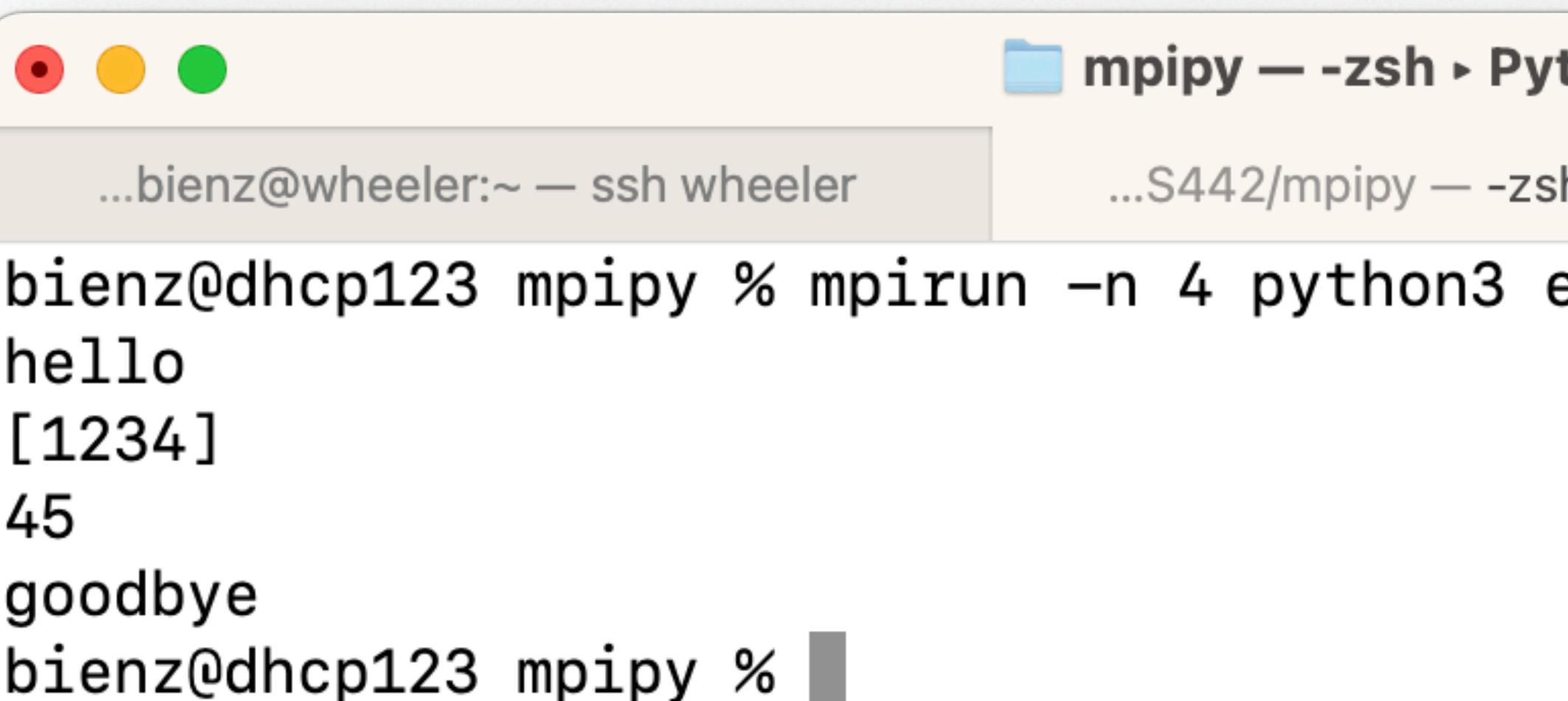
root = 0
size = 1000
n = (int)(size / num_procs)
array = np.random.rand(n)
print(array)

array = Comm.gather(array, root)

if rank == root:
    print(array)
~
```

```
bienz@dhcp123 mpipy % mpirun -n 4 python3 example.py
[0.63137449 0.14613175 0.31386525 ... 0.40202934 0.03975134 0.33843775]
[0.77241016 0.94620565 0.68912371 ... 0.08507317 0.781987 0.75293553]
[0.33287514 0.7383226 0.8269368 ... 0.72451843 0.05116583 0.14933154]
[0.37301632 0.50065708 0.95537867 ... 0.21186825 0.62165606 0.20253242]
[array([0.37301632, 0.50065708, 0.95537867, ..., 0.21186825, 0.62165606,
       0.20253242]), array([0.77241016, 0.94620565, 0.68912371, ..., 0.08507317
       0.781987,
       0.75293553]), array([0.63137449, 0.14613175, 0.31386525, ..., 0.40202934
       0.03975134,
       0.33843775]), array([0.33287514, 0.7383226, 0.8269368, ..., 0.72451843
       0.05116583,
       0.14933154])]
bienz@dhcp123 mpipy %
```

# Different Datatypes, Different Sizes



```
example.py (~/Desktop/CS442/mpipy) - VIM
```

```
from mpi4py import MPI
import numpy as np

Comm = MPI.COMM_WORLD

num_procs = Comm.Get_size()
rank = Comm.Get_rank()

root = 0
array = ""
if rank == root:
    array = ["hello", 45, [1234], "goodbye"]

array = Comm.scatter(array, root)

print(array)
```

```
...bienz@wheeler:~ — ssh wheeler ...S442/mpipy — zsh
[bienz@dhcp123 mpipy % mpirun -n 4 python3 ex]
hello
[1234]
45
goodbye
bienz@dhcp123 mpipy %
```