

Day 2:  
YALE PATHWAYS TO  
SCIENCE SUMMER  
WORKSHOP 2021

Aakash Kumar  
[aakash.kumar@yale.edu](mailto:aakash.kumar@yale.edu)

# Introduction to Parallel Computing: Hands-on

DESIGN OF NEW MATERIALS USING SUPERCOMPUTERS

# LEARNING GOALS

- Run basic python scripts to sum numbers and arrays
- Run parallel script to access difference cores
- Apply the ideas of broadcasting, scattering and gathering from parallel computing lecture earlier

ak2589@c16n06.grace:~ x Launcher x

OPEN TABS Close All

ak2589@c16n06.grace:~  
Launcher

KERNELS Shut Down All

TERMINALS Shut Down All

terminals/1

scratch60

Notebook

Python 3 Python 3

Console

Python 3

Other

Terminal Text File Markdown File Show Contextual Help

Open the terminal then type day2 and hit Enter

# PYTHON

- Scripting language
- Object-oriented
- Easy to read, friendly design

[Terminal] python

```
>>> a = 2
```

```
>>> b = 3
```

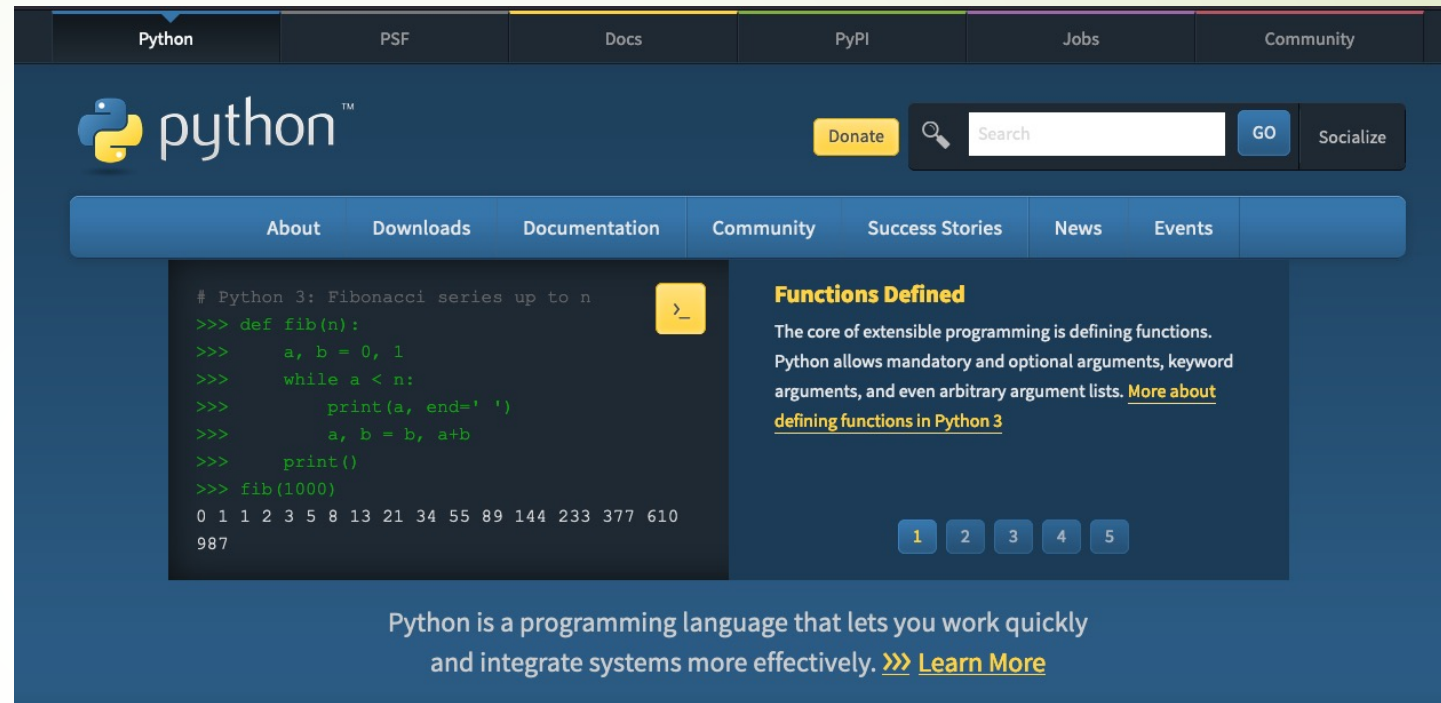
```
>>> c = a + b
```

```
>>> c
```

```
5
```

- Exit the python environment (>>>) by pressing Ctrl+D
- We can also write the above code in a file `sum.py`
- then run it as

[Terminal] python sum.py



The screenshot shows the Python.org website. At the top, there are navigation tabs for Python, PSF, Docs, PyPI, Jobs, and Community. The Python logo and name are prominently displayed. A search bar and a 'Donate' button are visible. Below the navigation, there are links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a terminal window on the left with the following code:

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
987
```

On the right, there is a section titled 'Functions Defined' with the text: 'The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)'.

At the bottom of the page, there is a tagline: 'Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)'.

# EXERCISE 1a, 1b, 1c

► run the code in file 1a\_sum.py

```
a = 2
```

```
b = 3
```

```
c = a + b
```

```
print(c)
```

```
print("The sum is", c)
```

► repeat the above with floating point numbers, so 2 is 2.0 (1b\_sum\_float.py)

► Use numpy module of python to define a and b as `np.a`, `np.b` and sum (1c\_sum\_numpy.py)

```
import numpy as np
```

```
np.a = 2
```

```
np.b = 3
```

```
np.c = np.a + np.b
```

```
print(np.c)
```

# EXERCISE 1d, 1e

- Array is a list of data, represented in [], so `a = [1, 2, 3]` is an array of size 3
- Its elements are `a[0]`, `a[1]`, `a[2]`
- numpy module can be used to create an array

```
np.a = np.array([1, 2, 3])
```

```
np.b = np.array([4, 5, 6])
```

- Run the following code in a file `1d_sum_array.py`

```
np.c = np.a + np.b
```

```
print(np.c)
```

```
print("The sum is", np.c)
```

# EXERCISE 2

- Copy the below code snippet to a file called hello.py

```
from mpi4py import MPI
```

```
comm = MPI.COMM_WORLD
```

```
size = comm.Get_size()
```

```
rank = comm.Get_rank()
```

```
# comm is the MPI object we will use
```

```
# how many processes (1 on each core)
```

```
# rank of processor (ID) 0 is head processor
```

```
print("Hello world from rank", str(rank), "of", str(size))
```

```
mpirun -n 6 python greetings.py
```

- Run the python program `greetings.py` on 6 cores (format is slightly different than what is shown here, notice the `comm.send` and `comm.recv`)
- Each core will print its rank and total number of cores

# EXERCISE 3: BROADCAST

- broadcast data to all cores
- Code snippet:

```
if rank == 0:  
    data = np.arange(4.0)  
  
else:  
    data = None
```

```
data = comm.bcast(data, root=0)
```

```
if rank == 0:  
    print('Process {} broadcast data:'.format(rank), data)  
else:  
    print('Process {} received data:'.format(rank), data)
```

```
mpirun -n 4 python bcast.py
```

```
Process 0 broadcast data: [0. 1. 2. 3.]  
Process 2 received data: [0. 1. 2. 3.]  
Process 1 received data: [0. 1. 2. 3.]  
Process 3 received data: [0. 1. 2. 3.]
```



# EXERCISE 4: SCATTER

- ▶ scatter different data to various cores
- ▶ Code snippet:

```
if rank == 0:  
    data = np.arange(4.0)
```

```
else:  
    data = None
```

```
data = comm.scatter(data, root=0)
```

```
if rank == 0:  
    print('Process {} broadcast data:'.format(rank), data)
```

```
else:  
    print('Process {} received data:'.format(rank), data)
```

- ▶ 4b\_scatter.py scatters arrays of data to all cores.

```
mpirun -n 4 python 4a_scatter.py
```

```
Process 0 has data: 0.0  
Process 1 has data: 1.0  
Process 2 has data: 2.0  
Process 3 has data: 3.0
```

# EXERCISE 5: GATHER

➤ Gather data from all processes/cores and Do something (SUM it up, etc.)

➤ Code snippet:

```
mpirun -n 4 python 5_gather.py
```

```
do something
```

```
.....
```

```
.....
```

```
.....
```

```
.....
```

```
partial_sum = comm.gather(partial_sum, root=0)
```

```
if rank == 0:
```

```
    print('Sum is {} from all data:'.format(sum(partial_sum)))
```

# EXSERICSE 5:REDUCE

➤ reduce data from all processes/cores and sum it while collecting

➤ Code snippet:

```
mpirun -n 4 python 5_reduce.py
```

```
do something
```

```
.....
```

```
.....
```

```
.....
```

```
.....
```

```
total_sum = comm.reduce(partial_sum, op=MPI.SUM, root=0)
```

```
if rank == 0:
```

```
    print('Sum is {} from all data:'.format(total_sum))
```

# RESOURCES

- ▶ <https://hpc.llnl.gov/training/tutorials/introduction-parallel-computing-tutorial>
- ▶ <https://towardsdatascience.com/parallel-programming-in-python-with-message-passing-interface-mpi4py-551e3f198053>
- ▶ <https://www.kth.se/blogs/pdc/2019/08/parallel-programming-in-python-mpi4py-part-1/>