

sciris-lightning-slides

July 26, 2024



Sciris: Simplifying Scientific Python

Cliff Kerr



BILL & MELINDA
GATES *foundation*

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

SCIRIS







		HOW OFTEN YOU DO THE TASK					
		50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
HOW MUCH TIME YOU SHAVE OFF	1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
	5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
	30 SECONDS	 4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
	1 MINUTE	 8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
	5 MINUTES	9 MONTHS	 4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
	30 MINUTES		6 MONTHS	 5 WEEKS	5 DAYS	1 DAY	2 HOURS
	1 HOUR		10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
	6 HOURS				2 MONTHS	 2 WEEKS	1 DAY
	1 DAY					 8 WEEKS	5 DAYS

Image credit: [xkcd](#)

This example illustrates how the same block of fairly typical “scientific Python” code – which performs tasks like collecting data from a function running in parallel, saving and loading files, and 3D plotting – looks like when written in “vanilla Python” compared to using Sciris. See docs.sciris.org for more information.

```
[2]: %% Shared code: define the random wave generator

import numpy as np

def randwave(std, xmin=0, xmax=10, npts=50):
    np.random.seed(int(100*std)) # Ensure differences between runs
    a = np.cos(np.linspace(xmin, xmax, npts))
    b = np.random.randn(npts)
    return a + b*std
```

```
[3]: ### Vanilla Python code: 29 lines

# Other imports
import time
import concurrent.futures
import pickle
import gzip
import matplotlib.pyplot as plt

# Start timing
start = time.time()

# Calculate output in parallel
with concurrent.futures.ProcessPoolExecutor() as executor:
    waves = list(executor.map(randwave, np.linspace(0, 1, 11)))

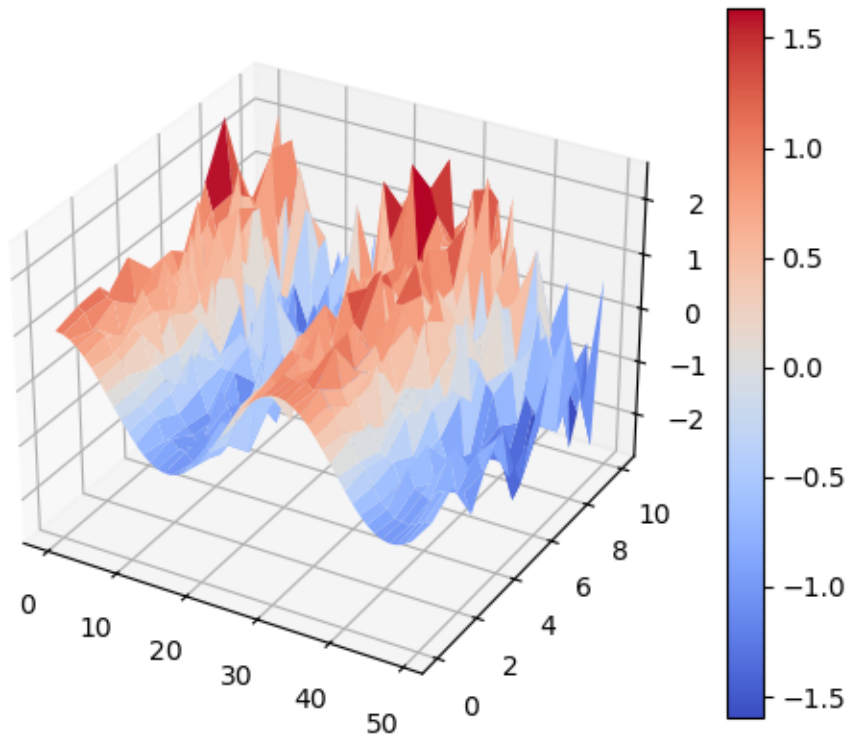
# Save to files
filenames = []
for i, wave in enumerate(waves):
    filename = f'wave{i}.obj'
    with gzip.GzipFile(filename, 'wb') as fileobj:
        fileobj.write(pickle.dumps(wave))
    filenames.append(filename)

# Create dict from files
data_dict = {}
for fname in filenames:
    with gzip.GzipFile(fname) as fileobj:
        filestring = fileobj.read()
        data_dict[fname] = pickle.loads(filestring)
data = np.array([data_dict[fname] for fname in filenames])

# Create 3D plot
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
ny, nx = np.array(data).shape
x = np.arange(nx)
y = np.arange(ny)
X, Y = np.meshgrid(x, y)
surf = ax.plot_surface(X, Y, data, cmap='coolwarm')
fig.colorbar(surf)

# Print elapsed time
elapsed = time.time() - start
print(f'Elapsed time: {elapsed} s')
```

Elapsed time: 0.14980697631835938 s



```
[4]: ### Sciris equivalent -- 7 lines (4x shorter)

import sciris as sc
T = sc.timer()
waves = sc.parallelize(randwave, np.linspace(0, 1, 11))
filenames = [sc.save(f'wave{i}.obj', wave) for i, wave in enumerate(waves)]
data = sc.odict({fname:sc.load(fname) for fname in filenames})
sc.surf3d(data[:], cmap='orangeblue')
T.toc()
```

Elapsed time: 0.143 s

