Table of Contents

GEOL 6670 Geophysical Inverse Theory

Tutorial on Python

Shane Zhang*

University of Colorado Boulder

Aug 31st, 2021

[*shane.zhang@colorado.edu (mailto:shane.zhang@colorado.edu)]

Why Python?

- Versatile
  - A personal story: My old workflow is to use `C/Fortran` for computing, then use `Shell/Perl` for I/O, and `Matlab/GMT` for plotting.
- Easy to read

```python
def add(a, b):
  # Return the sum of two numbers.
  return a + b
```

- "Efficient"
  - I believe human time is more precious than machine time, and thus appreciate the fast feedback from Python in the development stage.
- Open-source
- Popular

Fig: Trends of programming languages in [StackOverflow](https://insights.stackoverflow.com/trends?tags=java%2Cc%2Cc%2B%2B%2Cpython%2Cc%23%2Cvb.net%2Cjavascript%2Casse c).

A google search of `python tutorial` yields ~1 billion results, thus I would not try to provide a list of useful resources. In addition, the tutorial aims to be useful for a diverse audience but will become more biased in the later part.

Some examples are provided only as counterparts to Menke's introdution to Matlab.

Python syntax

Variables and formats

In [1]:

```python
# Single line comment
"""
Multiple
line
comments.
"""

# Integer division
5 // 3
-5 // 3

# Exponentiation
2**3

# Boolean
True
not True
True or False
True and False

# Comparison
1 == 1
1 != 2
1 < 2
2 <= 2
1 < 2 < 3



x = 0

print('temperature data from {} to {}'.format(1.1, 2))

print('{0} and {1}'.format('spam', 'eggs'))
print('{1} and {0}'.format('spam', 'eggs'))

print('x = ' + str(x))
print(f'x = {x}')
print(f'x = {x:.3f}')
```

```
temperature data from 1.1 to 2
spam and eggs
eggs and spam
x = 0
x = 0
x = 0.000
```

Data structures

In [ ]:

```
# List
li = []
li.append(1)
li.append(2)
li.append(3)
li[1]
li[1:2]
len(li)
a, b, c = li
2 in li

# Tuple
tup = (1, 2, 3)
# tup[0] = 2    # Tuples are immutable!

# Dictionary
di = {'one': 1, 'two': 2, 'three': 3}
di['one']
list(di.keys())
list(di.values())
'one' in di
di.get('one')
di.update({'four': 4})

# Set
se = set({1, 1, 2, 2, 3})
se.add(4)
other_set = {3, 4}
se & other_set    # intersection
se | other_set    # union
se - other_set    # difference
```

Control flows

In [ ]:

```python
x = 5
if x > 1:
    print('x > 1')
elif x < 1:
    print('x < 1')
else:
    print('x == 1')


s = 0
for i in range(1, 10, 1):
    s += i
s


x = 0
while x < 2:
    print(x)
    x += 1


# try:
#     open('a/b')
# except FileNotFoundError as e:
#     raise e
# else:
#     pass
# finally:
#     pass
```

I/O (cf. using numpy, pandas)

In [ ]:
```
fpath = '/Volumes/GoogleDrive/My Drive/6670/gda/data/global_temp.txt'
with open(fpath, 'r') as f:
#     d = f.read()
#     d = f.readline()

#     d = f.readlines()
    d = list(f)

#     for i, line in enumerate(f):
#         print(i, line)

d



year, temperature = [], []
with open(fpath, 'r') as f:
    for line in f:
        y, t = line.split()
        year.append(int(y))
        temperature.append(float(t))

# year, temperature
```

Functions

In [ ]:
```
def function_name(arg1, *args, **kwargs):
    print(arg1)
    print(args)
    print(kwargs)

    return

arg1 = 1
args = (1, 2)
kwargs = {'a': 1}
# function_name(arg1)
function_name(arg1, *args, **kwargs)
```

Modules

In [ ]:
```
import math
```

In [ ]:
```
a = 6
b = 8
c = math.sqrt(a**2 + b**2)
c
```

In [ ]:
```
n = 3
x = 4
x0 = 1
L = 6
c = math.sin(n * math.pi * (x-x0) / L)
c
```

In [ ]:
```
from math import ceil

ceil(3.2)
```

Classes

In [ ]:
```python
class Human(object):
    def __init__(self, name, skills=[]):
        self.name = name
        self.skills = skills

    def solve_problems(self):
        return True


class Geologist(Human):
    def __init__(self, name, skills=['geology']):
        super().__init__(name=name, skills=skills)
#         super(Human, self).__init__(name=name, skills=skills)

    def solve_geological_problems(self):
        return True


class Physicist(Human):
    def __init__(self, name, skills=['physics']):
        super().__init__(name=name, skills=skills)

    def solve_physcis_problems(self):
        return True


class Geophysicist(Human):
    def __init__(self, name, skills=['geophysics']):
        super().__init__(name=name, skills=skills)

    def solve_geophysical_problems(self):
        return True


class GeoPhysicist(Geologist, Physicist):
    def __init__(self, name, skills=['geology, physics']):
        Geologist.__init__(self, name=name, skills=skills)
        Physicist.__init__(self, name=name, skills=skills)


p1 = GeoPhysicist(name='anonymous')
p1.name
p1.skills
p1.solve_problems()
# p1.solve_geological_problems()
# p1.solve_geophysical_problems()
```

Command line

In [ ]:
```
pwd
```

In [ ]:
```
cd /Volumes/GoogleDrive/My\ Drive/6670
```

In [ ]:
```
# dir()
```

In [ ]:
```
ls -lh
```

Essential packages

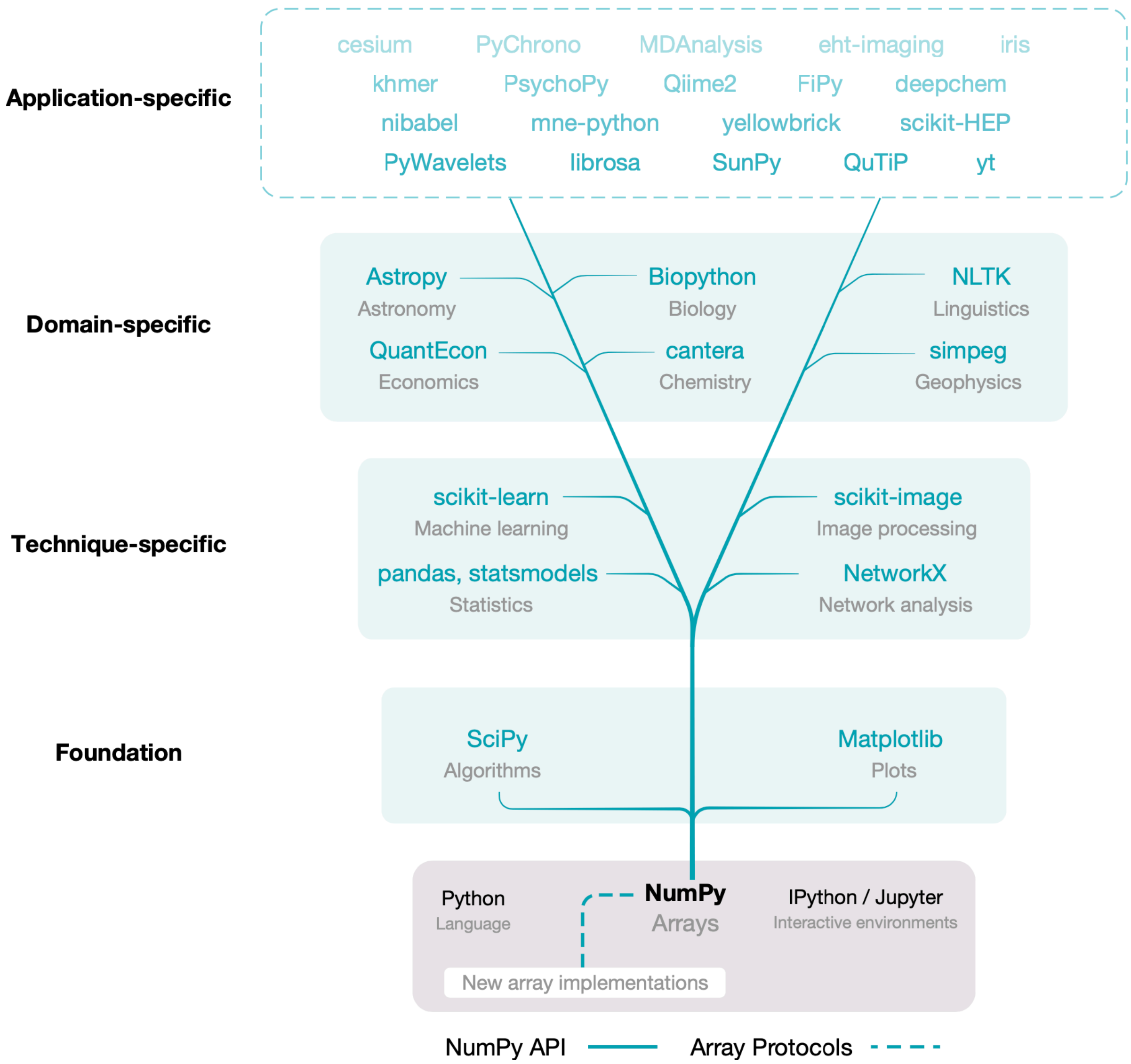

Fig: [Harris et al., 2020](https://doi.org/10.1038/s41586-020-2649-2).

Array programming with `NumPy`

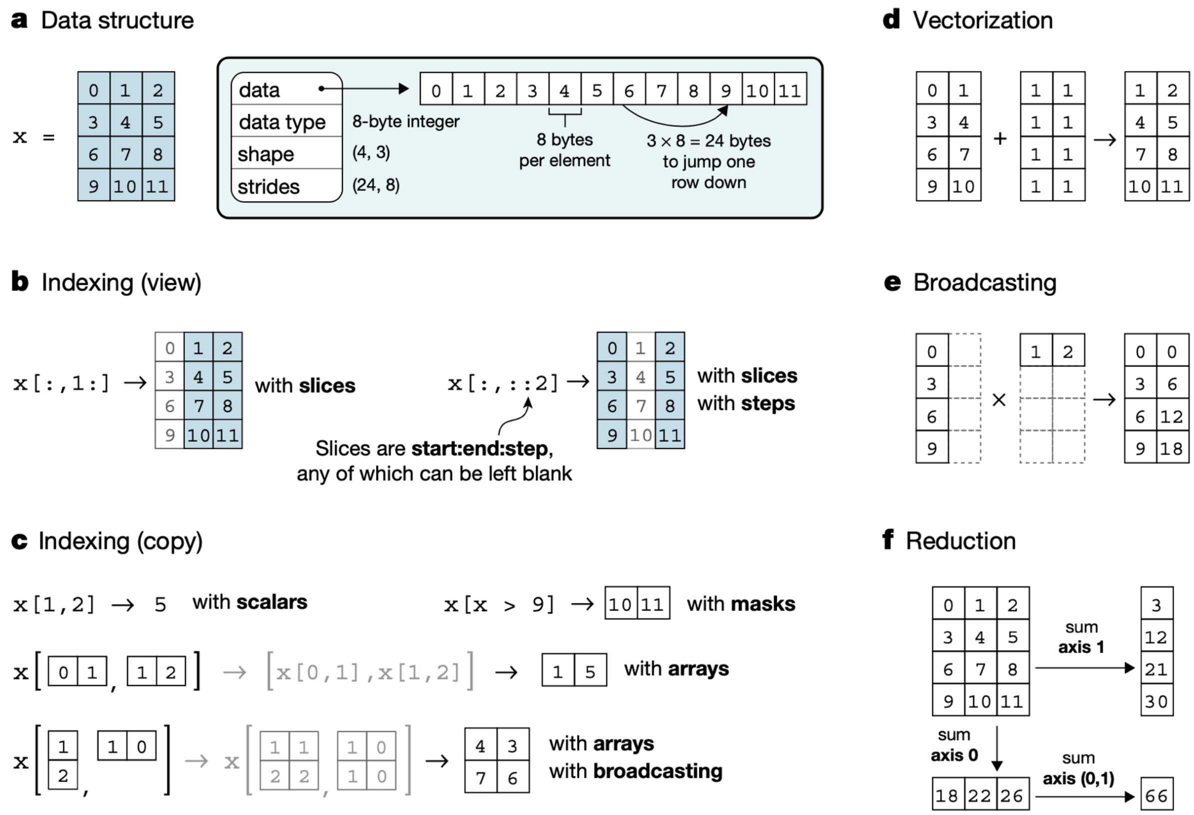

Fig: [Harris et al., 2020](https://doi.org/10.1038/s41586-020-2649-2).

In [ ]:
```
import numpy as np
```

In [ ]:
```
# Data structure
x = np.arange(12).reshape(4, 3)
x.dtype
x.shape
x.strides

# Indexing (view)
x[:, 1:]
x[:, ::2]

# Indexing (copy)
x[1, 2]
x[x > 9]
x[[0, 1], [1, 2]]
x[[[1], [2]], [1, 0]]

# Vectorization
a = np.column_stack([np.linspace(0, 9, 4), np.linspace(1, 10, 4)])
b = np.ones((4, 2))
a + b

# Broadcasting
a[:, 0].reshape(4, 1) * np.array([1, 2]).reshape(1, 2)

# Reduction
x.sum(axis=0)
x.sum(axis=1)
x.sum()
```

In [ ]:
```
r = np.array([2, 4, 6]).reshape(1, 3)
c = np.array([1, 3, 5]).reshape(3, 1)

r, c
```

In [ ]:
```
a = r[0, 1]
b = c[2, 0]
a, b
```

In [ ]:
```
M = np.arange(1, 10, 1).reshape(3, 3)
M
```

In [ ]:
```
c = M[0, 2]
c
```

In [ ]:
```python
M = np.array([
    1, 0, 2,
    0, 1, 0,
    2, 0, 1,
]).reshape(3, 3)
N = np.array([
    1, 0, -1,
    0, 2, 0,
    -1, 0, 3,
]).reshape(3, 3)
S = M + N
S
```

In [ ]:
```python
D = M - N
D
```

In [ ]:
```python
a = np.array([1, 3, 5]).reshape(3, 1)
b = np.array([2, 4, 6]).reshape(3, 1)
print(a.T @ b)
print(np.dot(a.T, b))
print(a.T.dot(b))
print(a @ b.T)
print(M @ a)
print(M @ N)
print(a * b)
```

- Loading data from a file

In [ ]:
```python
d = np.loadtxt(
    fpath,
    dtype={
        'names': ('year', 'temperature'),
        'formats': ('i4', 'f4')
    },
)
d
d['year']
```

Scientific computing with `SciPy`

In [ ]:
```python
import scipy as sp
from scipy import linalg        # Linear algebra
```

In [ ]:

```python
# Matrix inverse
# A = np.array([
#     [1, 5, 13],
#     [2, 7, 17],
#     [3, 11, 19],
# ])
# b = np.array([1, 2, 3]).reshape(3, 1)
# print(A)

# B = linalg.inv(A)
# print(B)
# print(A @ B)
# print(B @ A)

# c, error, rank, s = linalg.lstsq(A, b)
# print(c, error)


# B = np.array([
#     [1, 3, 4],
#     [2, 3, 2],
#     [0, 0, 4],
# ])
# D = B @ linalg.inv(A)
# print(D)

# Eigenvalues and eigenvectors
M = np.array([
    [1, 2, 0],
    [2, 2, 0],
    [0, 0, 4],
])
print(M)
LAMBDA, V = linalg.eig(M)
print(LAMBDA, '\n', V)
print(V.T @ V)
```

- SVD (Singular Value Decomposition)

$$M = USV^H$$

where $UU^H = U^H U = I, VV^H = V^H V = I$, and S is rectangular diagonal.

In [ ]:
```
M = np.array([
    [1, 0, 0, 0, 2],
    [0, 0, 3, 0, 0],
    [0, 0, 0, 0, 0],
    [0, 2, 0, 0, 0],
])
U, s, Vh = linalg.svd(M)
print(U)
print(s)
print(Vh)
```

Visualization with `Matplotlib`

In [ ]:
```
import matplotlib.pyplot as plt

plt.rcParams.update({
    'figure.dpi': 300,
})
```

In [ ]:
```
fig, ax = plt.subplots()
ax.plot(
    d['year'], d['temperature'], 'r-', linewidth=2,
    marker='o', markerfacecolor='none', markeredgecolor='k',
)
ax.set_xlim(1965, 2020)
ax.set_ylim(-0.5, 1)
ax.set_xlabel(r'Calendar year, $t_i$ (years)')
ax.set_ylabel(r'Temperature anomaly, $T_i$ ($^\circ$C)')
ax.set_title('Global temperature data', fontsize=14)
# fig.savefig('test.pdf')
```

Specific packages

Analysis of tabular data with `Pandas`

In [ ]:
```
import pandas as pd
```

- Load data

In [ ]:
```
df = pd.read_csv(
    fpath,
    sep='\s+',
    header=None,
    names=['year', 'temperature'],
#     parse_dates=['year'],
)
# df
```

- Plot data

In [ ]:
```
ax = df.plot('year', 'temperature', kind='line', c='r')
df.plot.scatter('year', 'temperature', ax=ax, c='k')
ax.set_ylim(-.5, 1)
```

- Data analysis

In [ ]:
```
df = pd.DataFrame(
    {
        "A": ["foo", "bar", "foo", "bar", "foo", "bar", "foo", "foo"],
        "B": ["one", "one", "two", "three", "two", "two", "one", "thre
        "C": np.random.randn(8),
        "D": np.random.randn(8),
    }
)
df
df.groupby('A').sum()
```

Machine learning with `scikit-learn`

Lasso (l_1) regression

$$\arg\min_m ||d - Gm||_2^2 + \alpha||m||_1$$

In [ ]:
```
from sklearn.linear_model import Lasso
```

In [ ]:

```
np.random.seed(42)
n_samples, n_features = 50, 100
X = np.random.randn(n_samples, n_features)

idx = np.arange(n_features)
coef = -1**idx * np.exp(-idx/10)
coef[10:] = 0
y = np.dot(X, coef)

y += 0.01 * np.random.normal(size=n_samples)

X_train, y_train = X[:n_samples//2], y[:n_samples//2]
X_test, y_test = X[n_samples//2:], y[n_samples//2:]

alpha = 0.1
lasso = Lasso(alpha=alpha)
model = lasso.fit(X_train, y_train)
y_pred = model.predict(X_test)


fig, axes = plt.subplots(1, 2, figsize=(10, 4))
ax = axes[0]
ax.stem(
    np.where(coef)[0], coef[coef != 0], 'b',
    linefmt='--',
    label='True coefficients',
    markerfmt='bx',
)
ax.stem(
    np.where(lasso.coef_)[0], lasso.coef_[lasso.coef_ != 0],
    'k', markerfmt='kx', label='Lasso coefficients',
)
ax.legend(loc='best')

ax = axes[1]
ax.scatter(y_train, model.predict(X_train), label='Train', alpha=.5)
ax.scatter(y_test, y_pred, label='Test', alpha=.5)
ax.plot([-6, 6], [-6, 6], c='k', zorder=-1)
ax.set_xlabel('True')
ax.set_ylabel('Prediction')
ax.set_aspect(1)
ax.set_xlim(-6, 6)
ax.set_ylim(ax.get_xlim())
ax.legend(loc='best')
```

~~Cartography with cartopy~~

In [ ]:

```
import cartopy.crs as ccrs
import cartopy.feature as cfeature
```

In [ ]:

```
fig, ax = plt.subplots(subplot_kw={'projection': ccrs.Mercator()})

ax.stock_img()

ax.gridlines(draw_labels=True, alpha=.1)
ax.add_feature(cfeature.COASTLINE, lw=1)
ax.add_feature(cfeature.BORDERS, lw=1)
ax.add_feature(cfeature.STATES, lw=1)
ax.set_extent([245, 260, 35, 43])

# x, y = np.meshgrid(
#     np.linspace(252, 256),
#     np.linspace(37, 41)
# )
# z = x + y
# im = ax.pcolormesh(
#     x, y, z,
#     transform=ccrs.PlateCarree(),    # required for plotting on maps
# )
# cb = fig.colorbar(im, orientation='horizontal')
# cb.set_label('Z (a.u.)')
```