Building blocks for a numerical simulation and output processing construction kit

- Prof. Raffaele Montella
- Dr. Diana Di Luccio

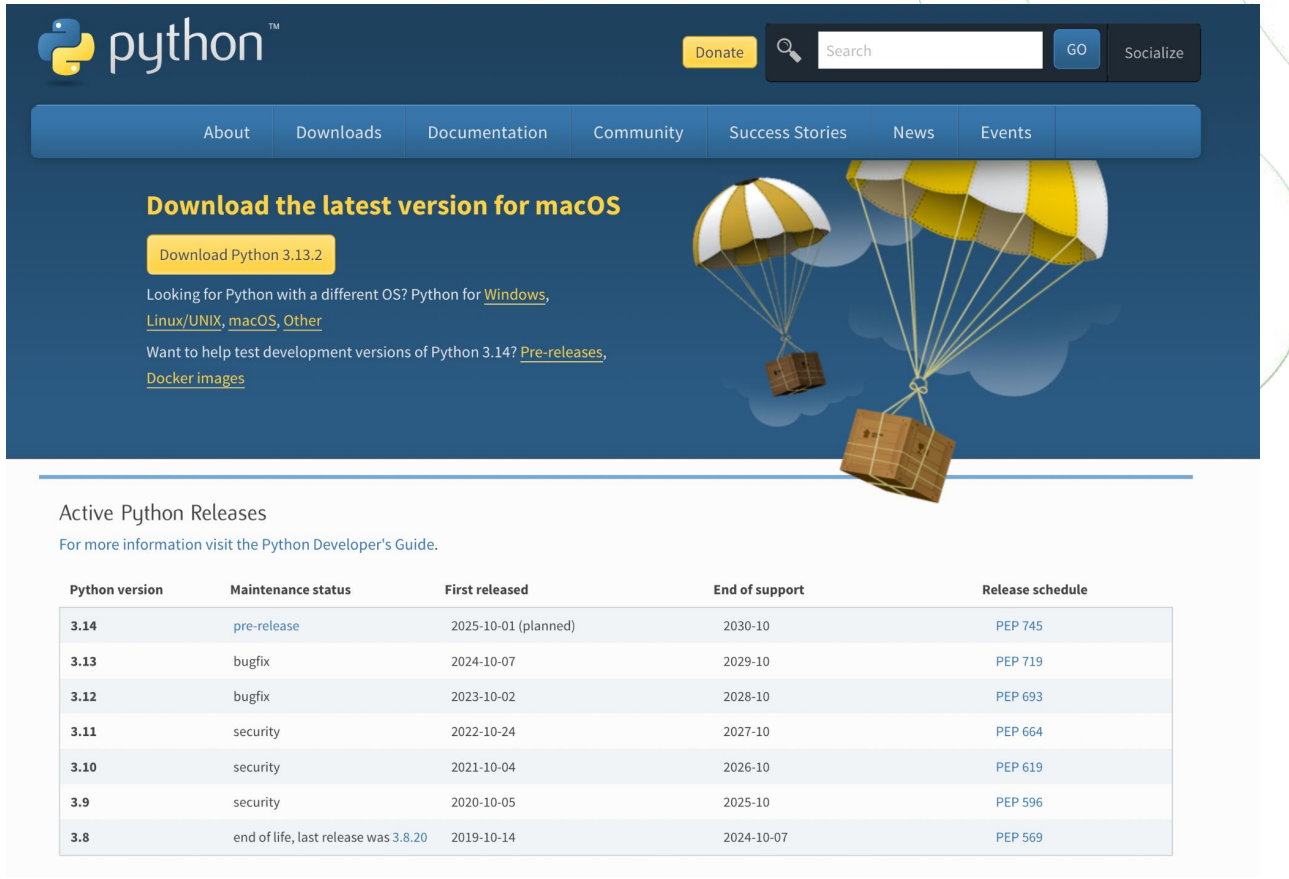
IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Python download

https://www.python.org/downloads/

Python 3.12.0



PyCharm (IDE)

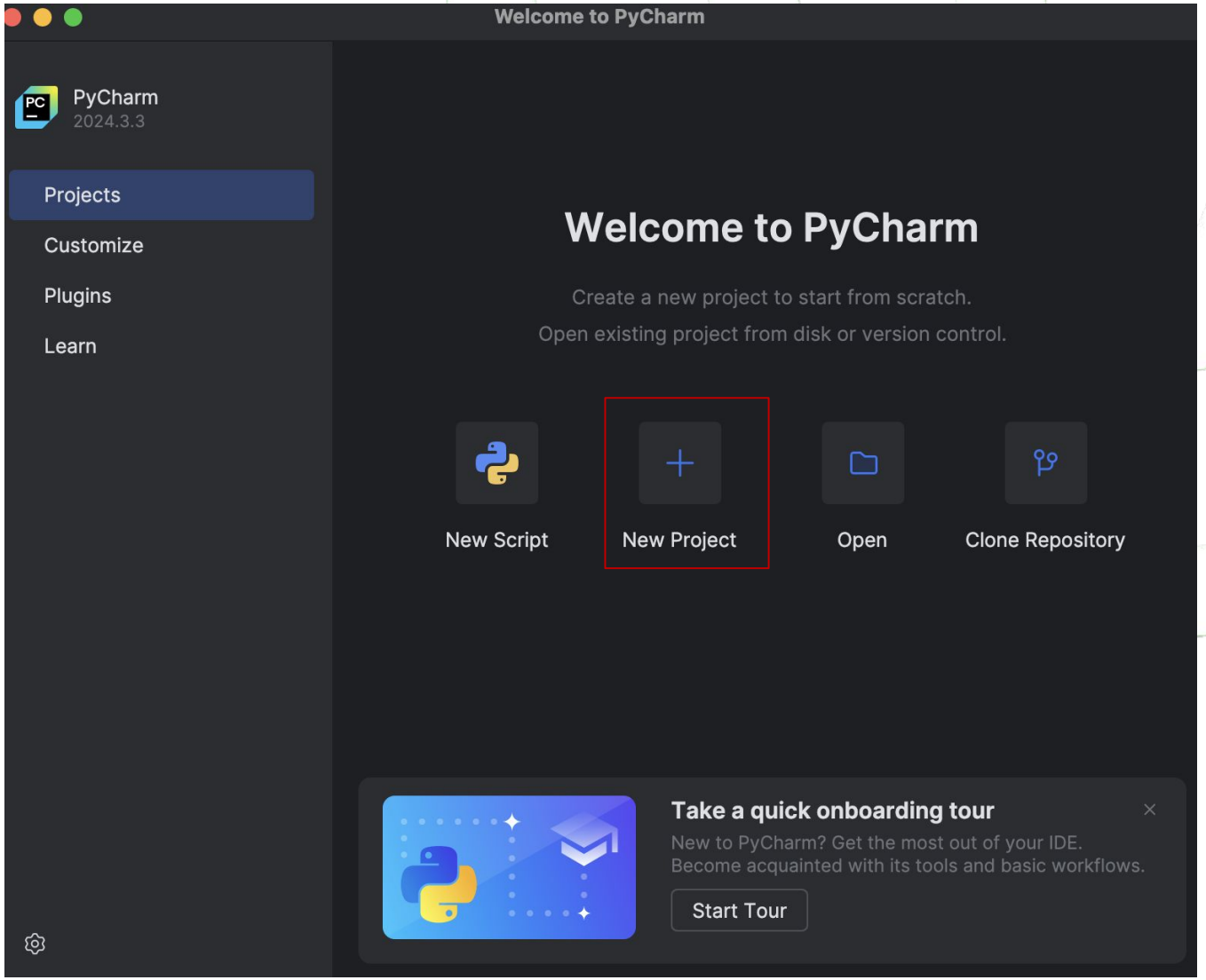
PyCharm is a development environment that includes an editor and various tools for analyzing, testing, and debugging code.
PyCharm is not free software.

However, there is both a paid professional version and a free version for personal use. The free version for Windows is called **PyCharm Community Edition**.

https://www.jetbrains.com/pycharm/download/

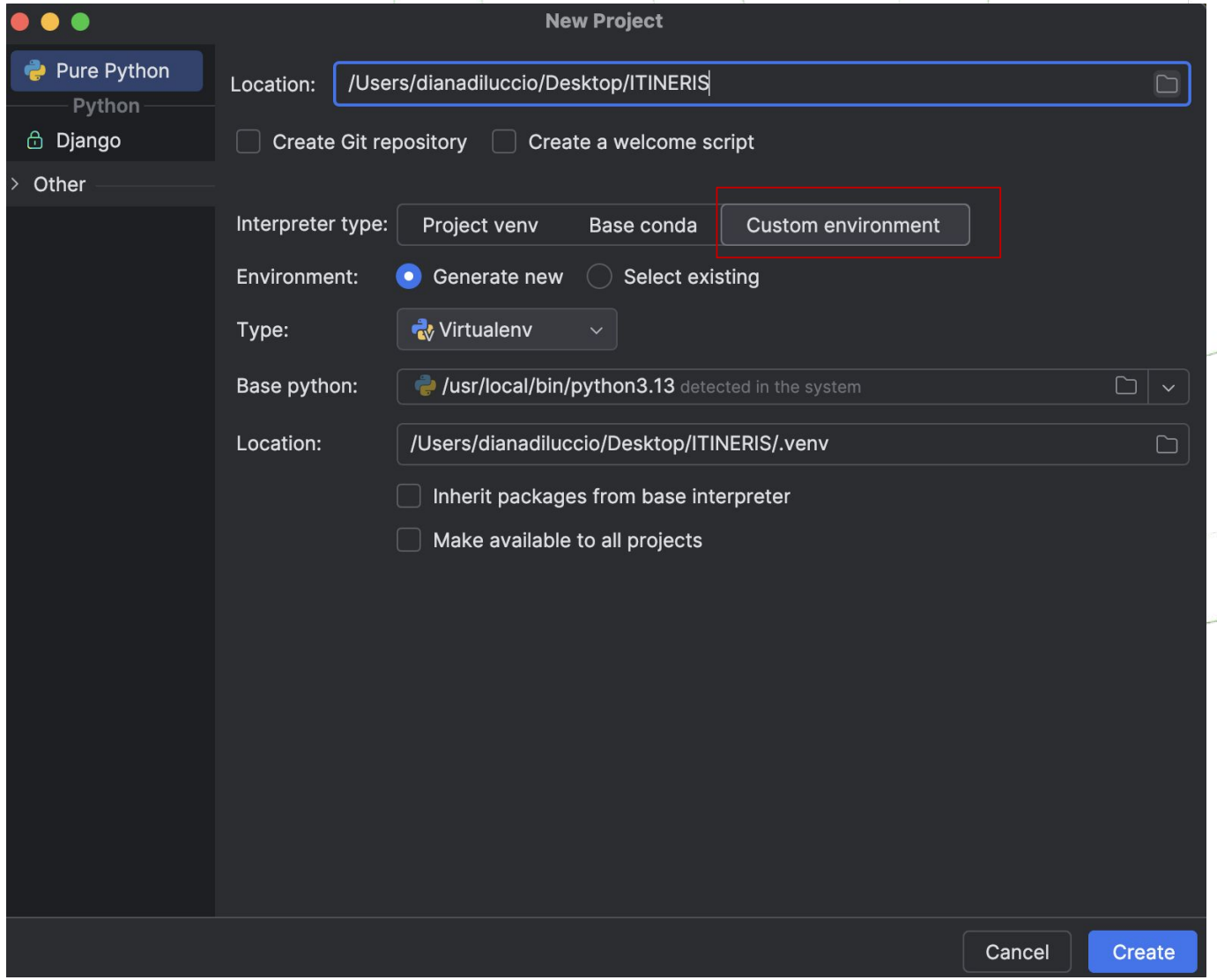
PyCharm (IDE)

- Open **PyCharm**
- Select **New Project**



PyCharm (IDE)

- Choose the project location.
- Select **Custom environment**
- Select **Virtualenv** as the development environment.
- Click Create

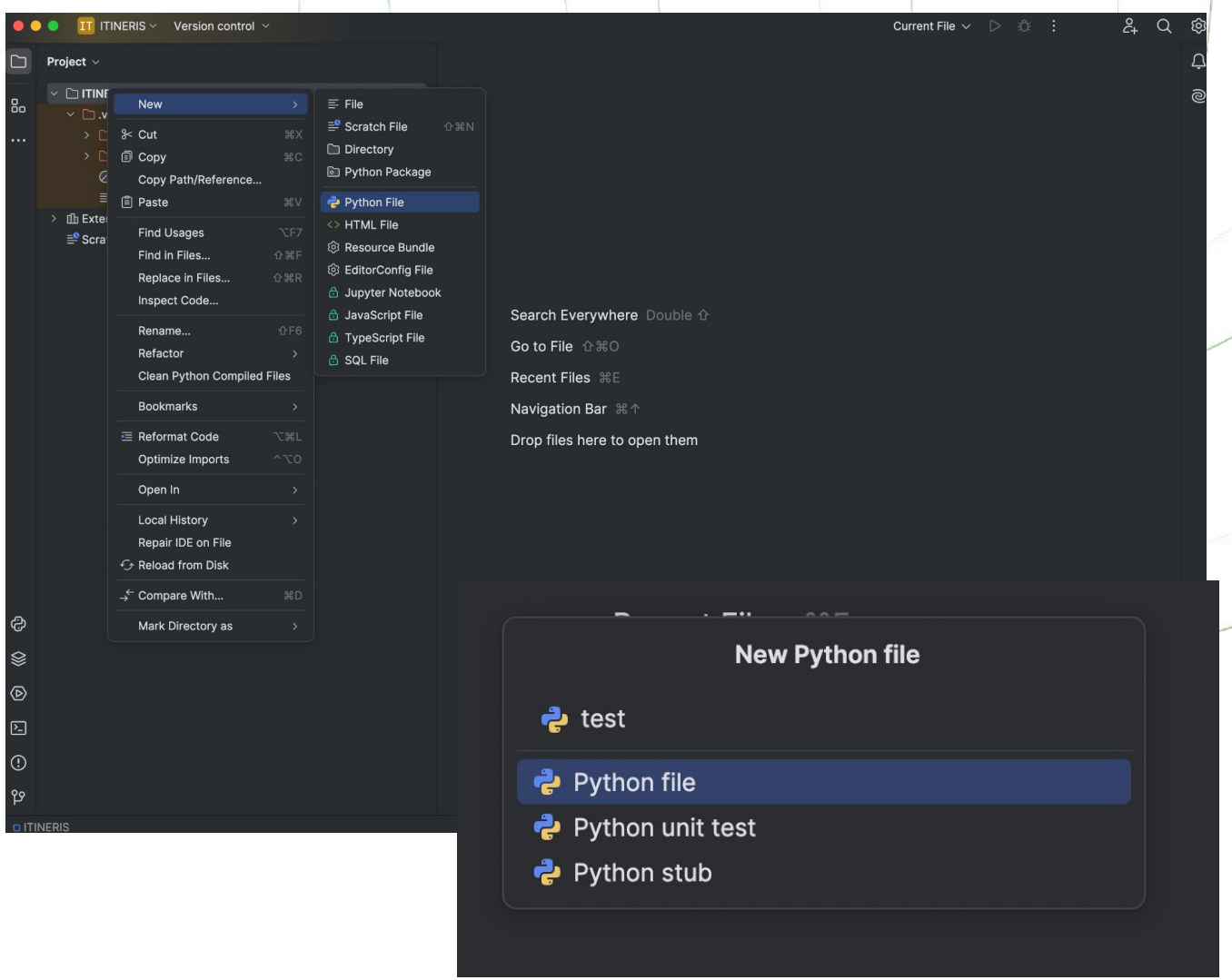


PyCharm (IDE)

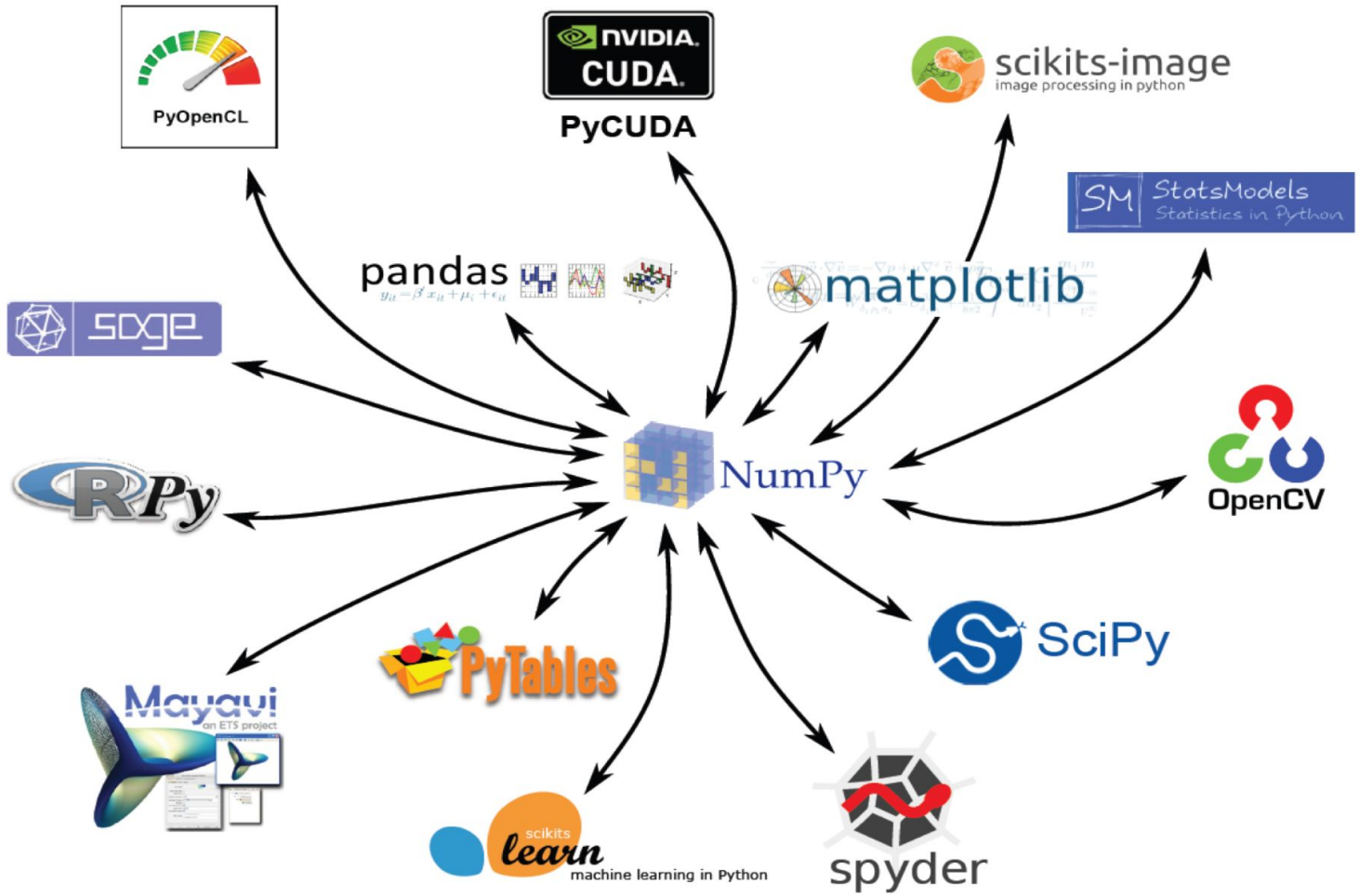
- A new project will be created
- right click on the project name and select New → **Python File**
- choose the file name and a .py file will be created
- copy and past

print("Hello!")

- Click **Run** to execute the code.



Ecosistema Python per il calcolo scientifico



NumPy

NumPy (Numerical Python) is a module that extends Python with data structures and methods useful for technical scientific calculations.

In pure Python we have:
– high-level numeric objects: integers, floating point
– containers: lists (insert and append at low cost) and dictionaries (fast lookup operations)

NumPy
– Introduces a natural and efficient way to use multi-dimensional arrays
– Adds a series of useful basic mathematical functions (linear algebra, FFT, random number,...)

For two arrays A and B of the same size, if we wanted to do a vector multiplication in Python:

```
c = []
for i in range(len(a)):
 c.append(a[i]*b[i])
```

In numpy, this can simply be done with the following line of code:

```
c = a*b
```

NumPy

Importiamo il modulo come di consueto

```
>>> import numpy
```

```
>>> from numpy import *
```

```
>>> import numpy as np #default in molti codici e nella documentazione numpy
```

Numpy fornisce un nuovo tipo di dato: un array N-dimensionale (**ndarray**).

Matplotlib

Matplotlib is a graphing library for the Python programming language and the NumPy mathematics library. It provides object-oriented APIs that allow you to embed graphs into applications using general-purpose GUI toolkits.

https://matplotlib.org/

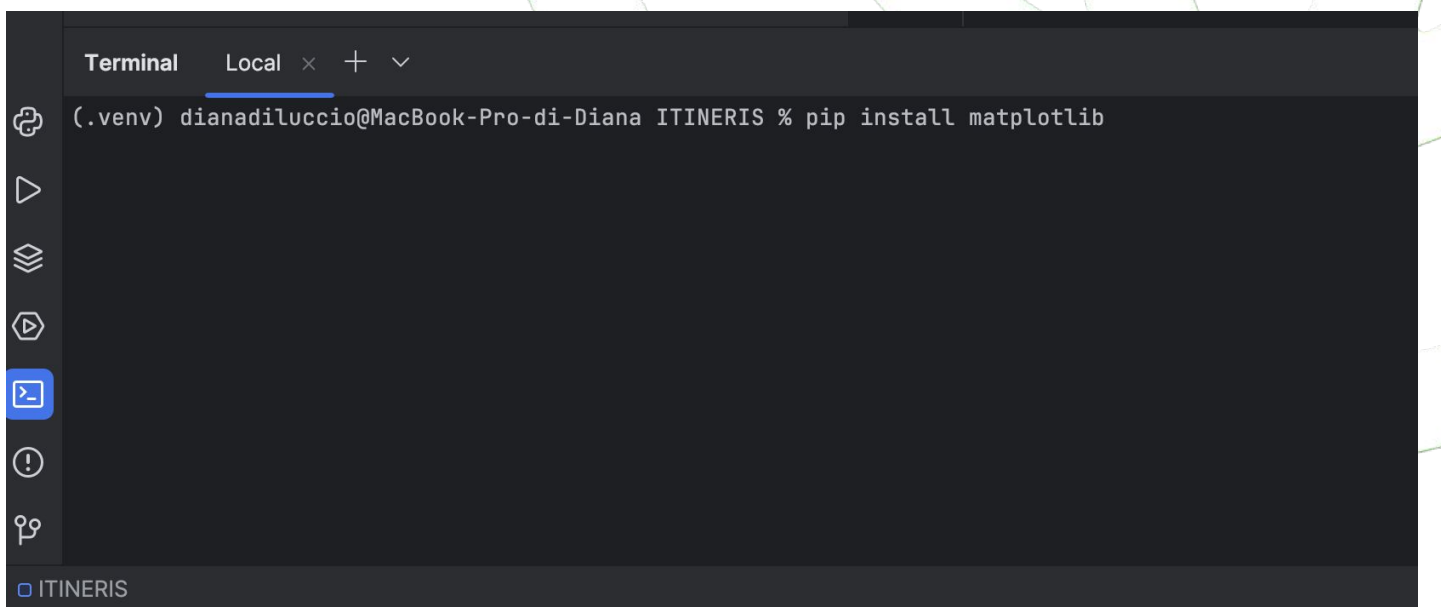
Matplotlib

To install matplotlib:

pip install matplotlib

To import matplotlib:

import matplotlib.pyplot as plt



plot() and show()

One of the first methods encountered with Matplotlib is plot, typically dedicated to the visualization of lines. Often the data that will be represented will belong to datasets managed by NumPy data structures.

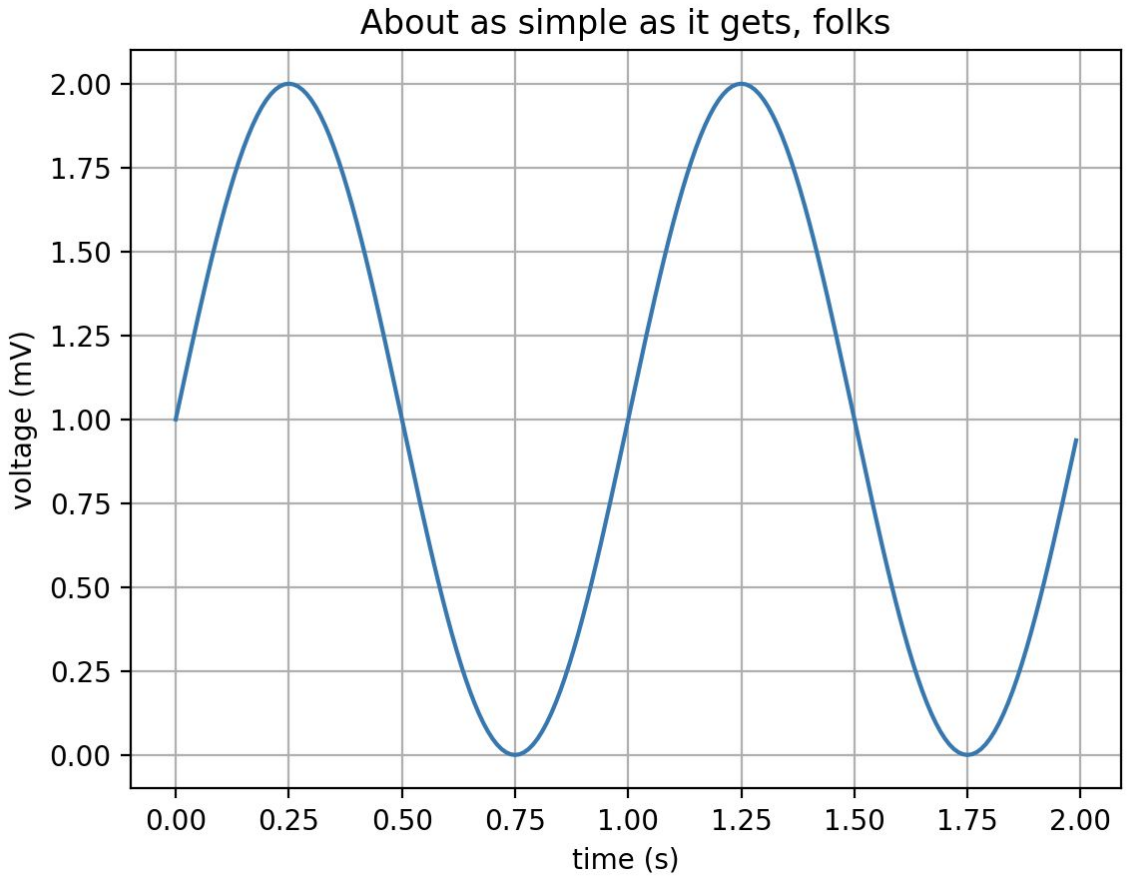
plt.plot(x,y)

To display the result in the form of a graph, at the end of the code the **show** method is used:

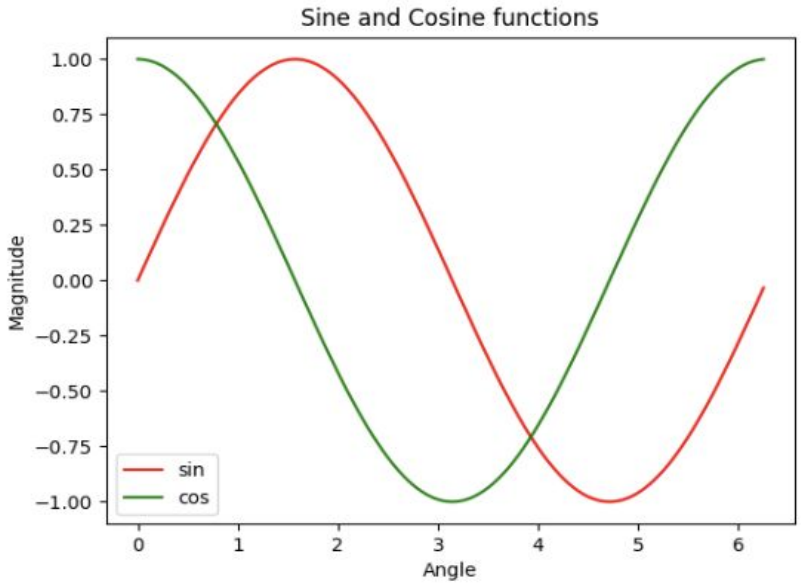
plt.show()

Example

```
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets, folks')
ax.grid()

fig.savefig("test.png")
plt.show()
```

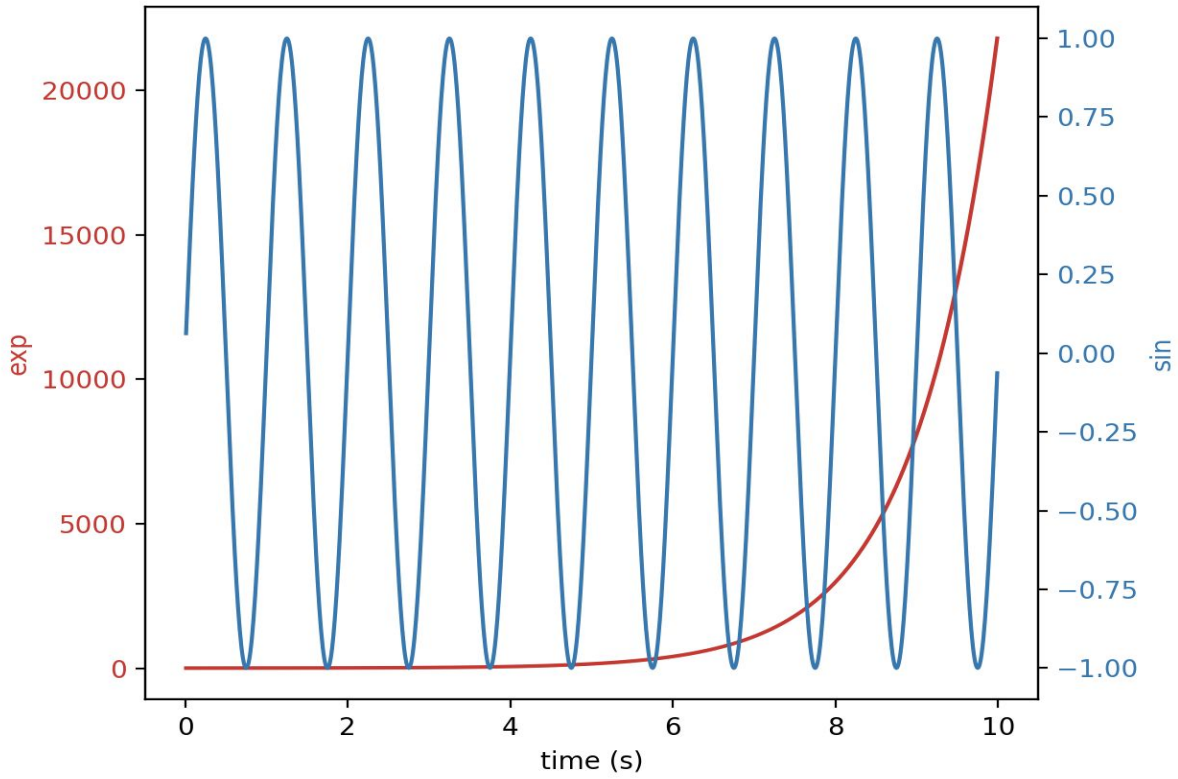


Example



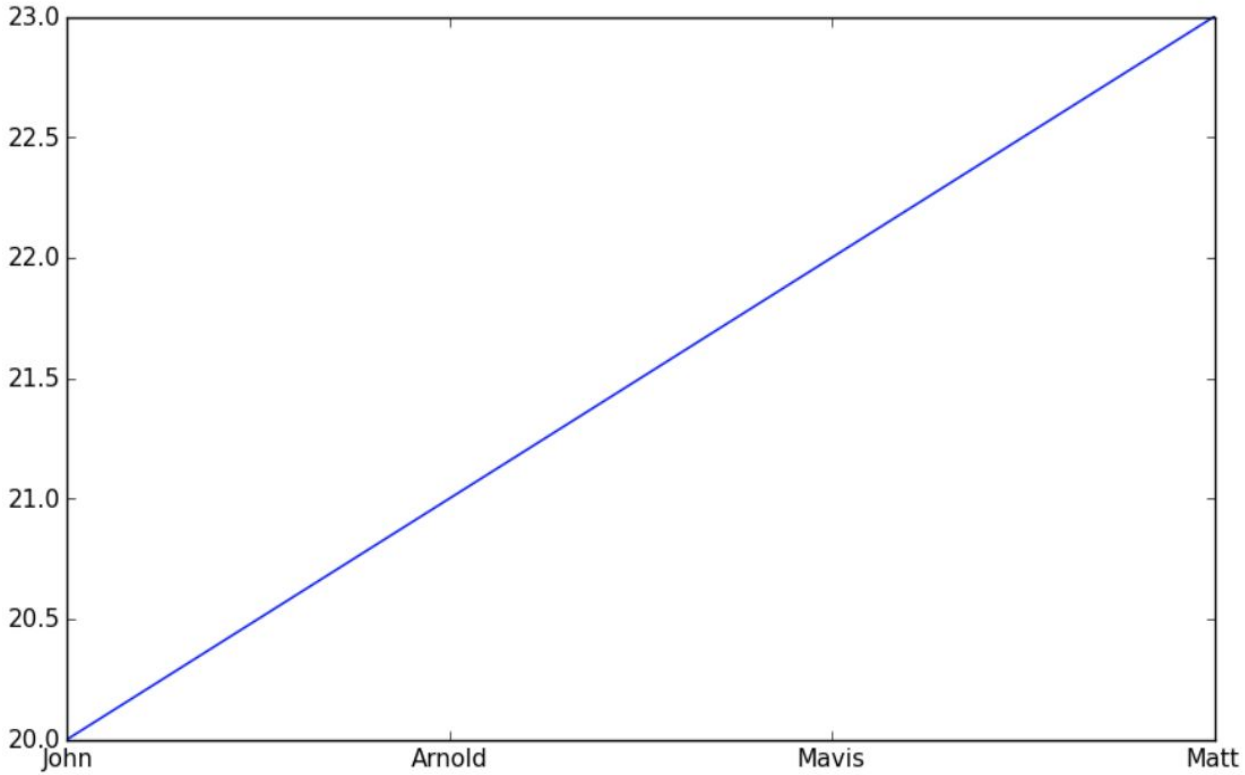

```python
# Importing libraries
import matplotlib.pyplot as plt
import numpy as np
import math
# Using Numpy to create an array X
X = np.arange(0, math.pi*2, 0.05)
# Assign variables to the y axis part of the curve
y = np.sin(X)
z = np.cos(X)
# Plotting both the curves simultaneously
plt.plot(X, y, color='r', label='sin')
plt.plot(X, z, color='g', label='cos')
# Naming the x-axis, y-axis and the whole graph
plt.xlabel("Angle")
plt.ylabel("Magnitude")
plt.title("Sine and Cosine functions")
# Adding legend, which helps us recognize the curve according to it's
color
plt.legend()
# To load the display window
plt.show()
```

Example




```python
import numpy as np
import matplotlib.pyplot as plt

# Create some mock data
t = np.arange(0.01, 10.0, 0.01)
data1 = np.exp(t)
data2 = np.sin(2 * np.pi * t)

fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('time (s)')
ax1.set_ylabel('exp', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('sin', color=color)  # we already handled the x-label with ax1
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```

Example

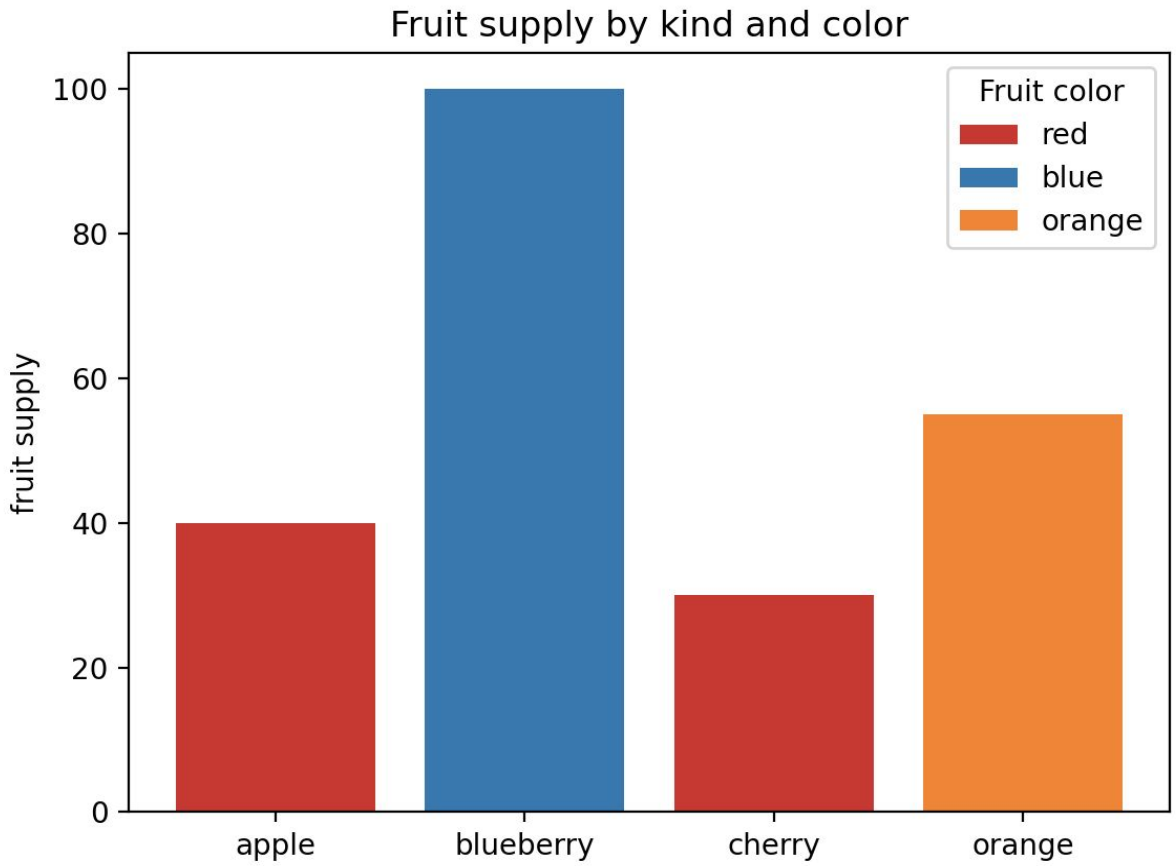



```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([0,1,2,3])
y = np.array([20,21,22,23])
my_xticks = ['John','Arnold','Mavis','Matt']
plt.xticks(x, my_xticks)
plt.plot(x, y)
plt.show()
```

```
plt.xticks(range(5), ["some", "words", "as", "x", "ticks"], rotation=45)
```

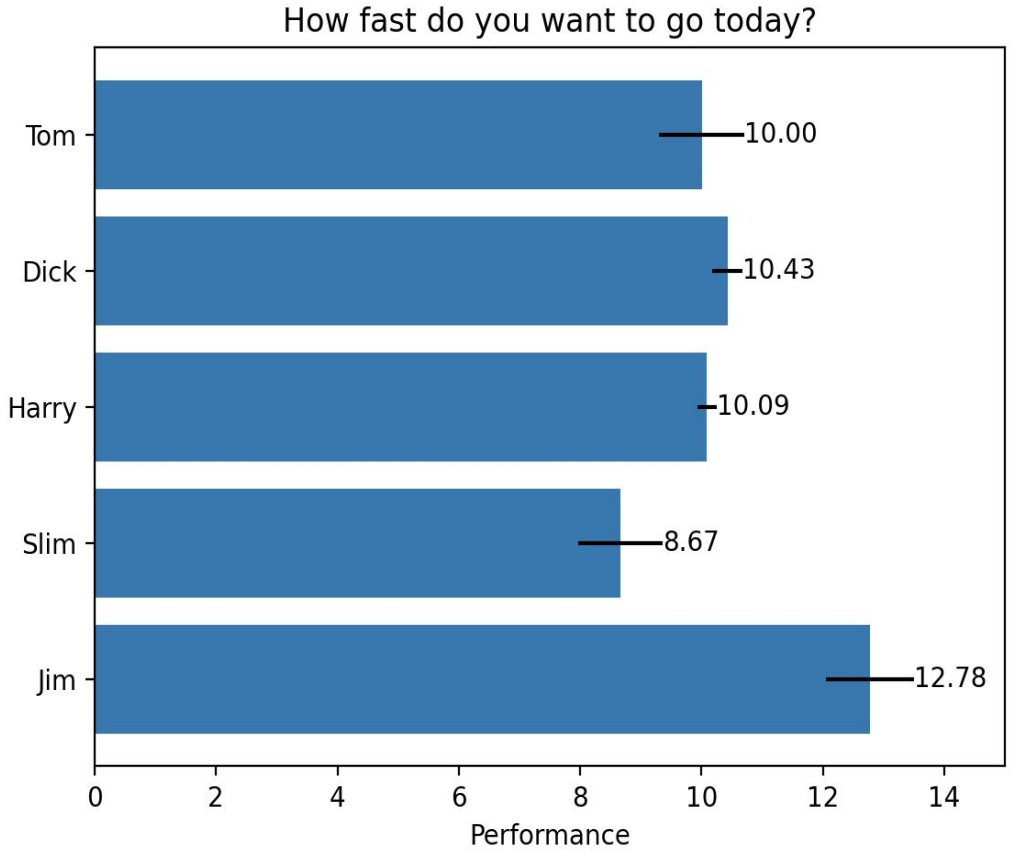
Example

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

fruits = ['apple', 'blueberry', 'cherry', 'orange']
counts = [40, 100, 30, 55]
bar_labels = ['red', 'blue', '_red', 'orange']
bar_colors = ['tab:red', 'tab:blue', 'tab:red', 'tab:orange']

ax.bar(fruits, counts, label=bar_labels, color=bar_colors)

ax.set_ylabel('fruit supply')
ax.set_title('Fruit supply by kind and color')
ax.legend(title='Fruit color')

plt.show()
```



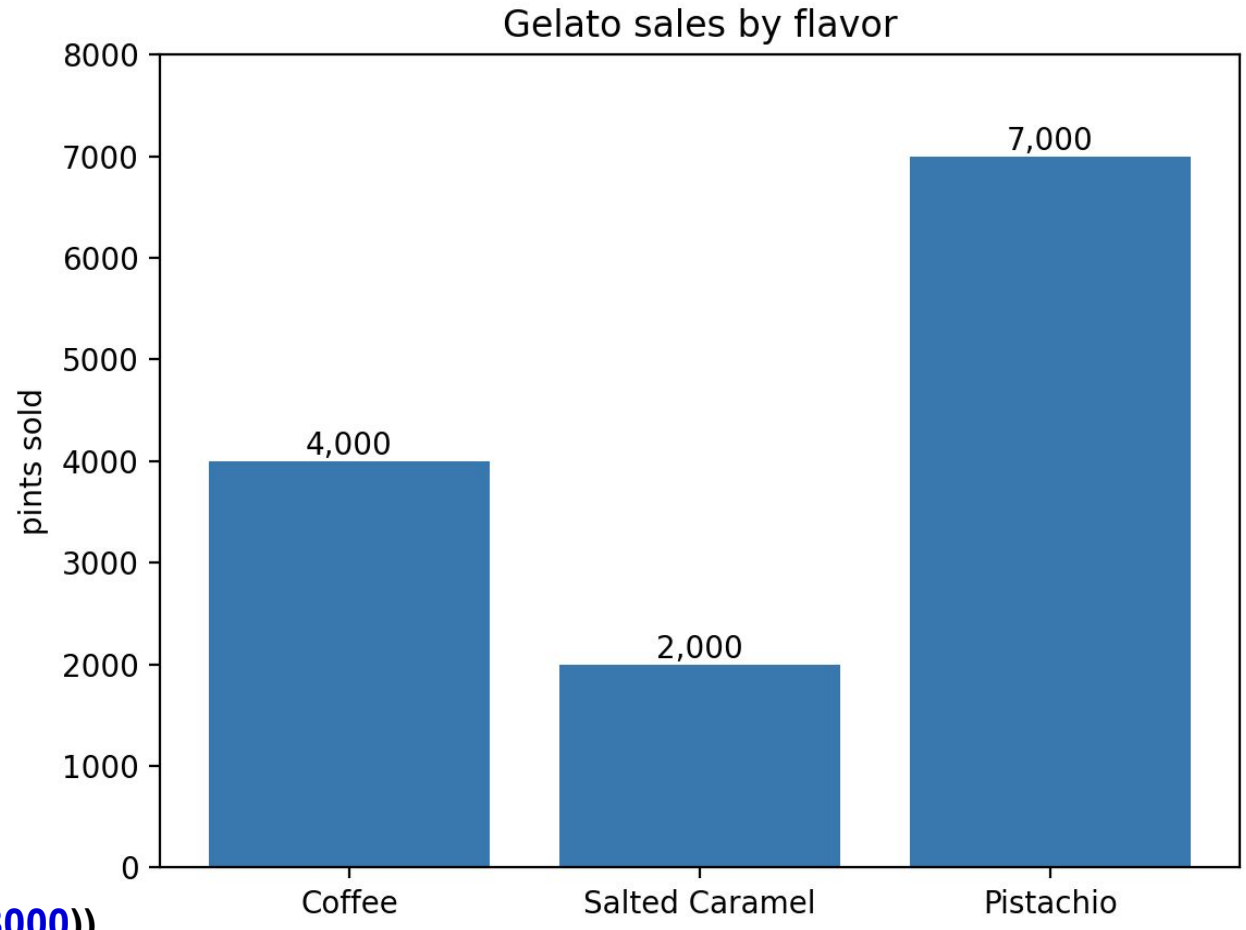
Example

```
# Fixing random state for reproducibility
np.random.seed(19680801)

# Example data
people = ('Tom', 'Dick', 'Harry', 'Slim', 'Jim')
y_pos = np.arange(len(people))
performance = 3 + 10 * np.random.rand(len(people))
error = np.random.rand(len(people))

fig, ax = plt.subplots()

hbars = ax.barh(y_pos, performance, xerr=error, align='center')
ax.set_yticks(y_pos, labels=people)
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('Performance')
ax.set_title('How fast do you want to go today?')

# Label with specially formatted floats
ax.bar_label(hbars, fmt='%.2f')
ax.set_xlim(right=15)  # adjust xlim to fit labels

plt.show()
```

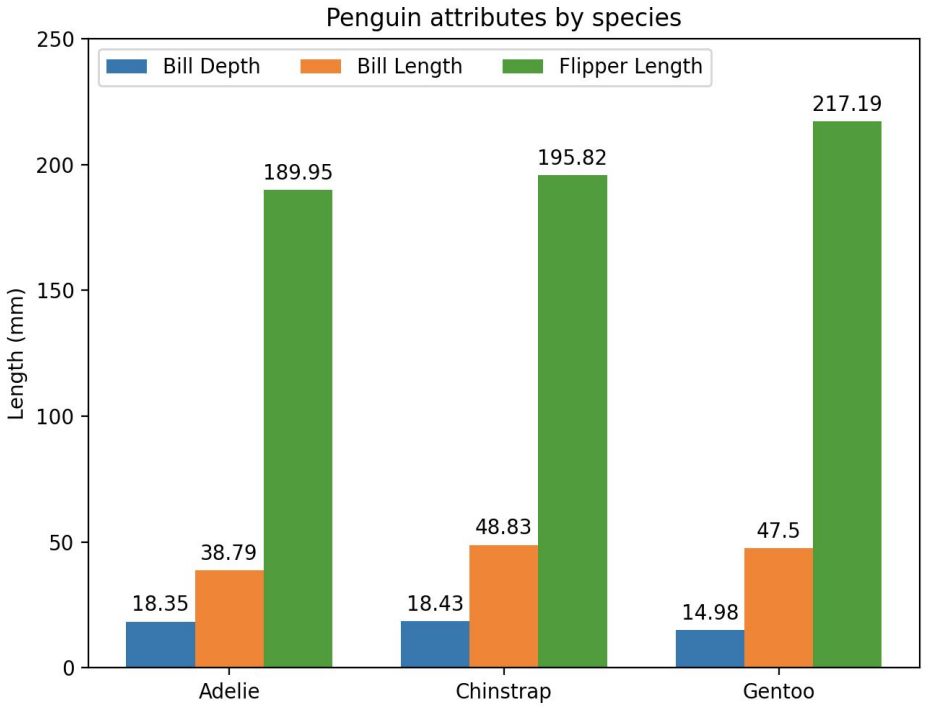


Example

```
fruit_names = ['Coffee', 'Salted Caramel', 'Pistachio']
fruit_counts = [4000, 2000, 7000]

fig, ax = plt.subplots()
bar_container = ax.bar(fruit_names, fruit_counts)
ax.set(ylabel='pints sold', title='Gelato sales by flavor', ylim=(0, 8000))
ax.bar_label(bar_container, fmt='{:,.0f}')
```

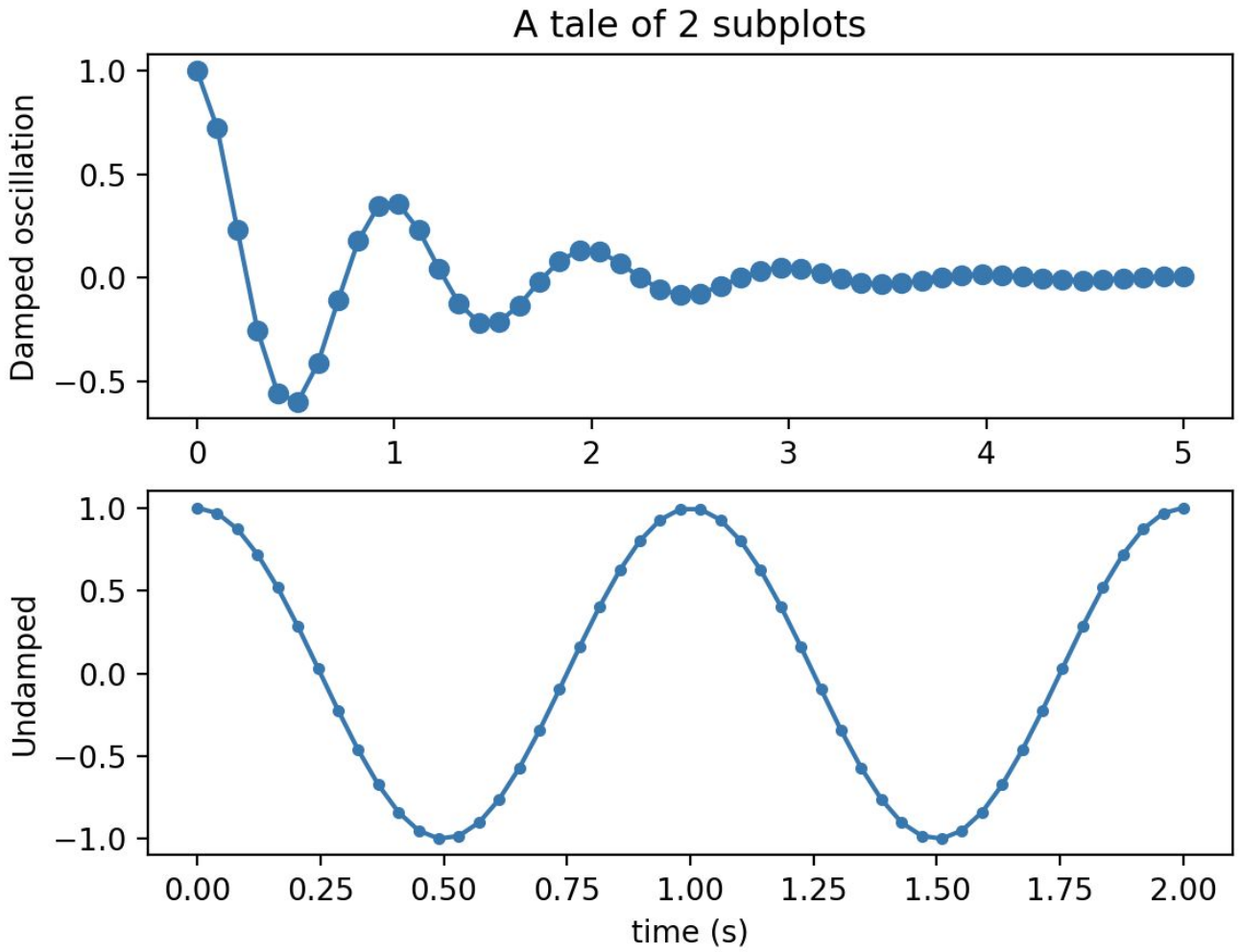


Example



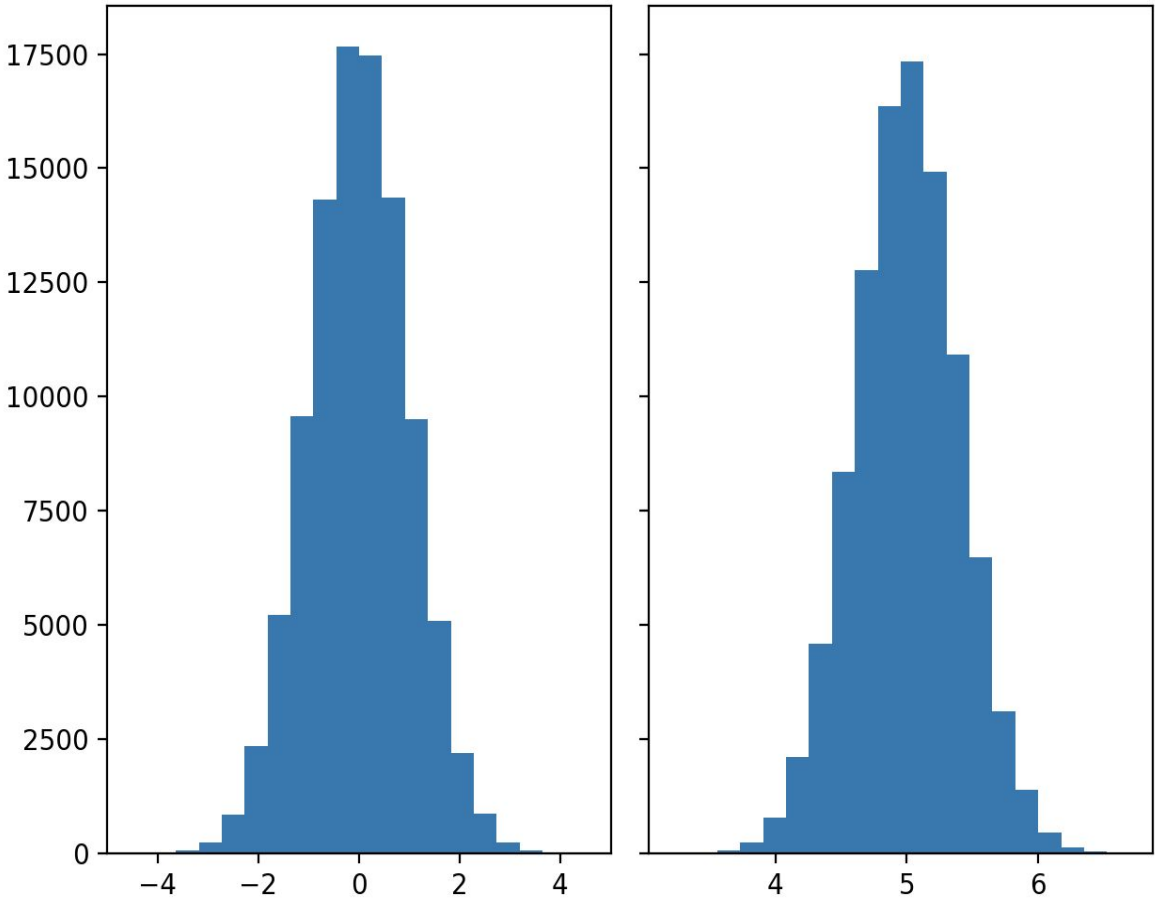

```python
import matplotlib.pyplot as plt
import numpy as np

species = ("Adelie", "Chinstrap", "Gentoo")
penguin_means = {
    'Bill Depth': (18.35, 18.43, 14.98),
    'Bill Length': (38.79, 48.83, 47.50),
    'Flipper Length': (189.95, 195.82, 217.19),
}

x = np.arange(len(species))  # the label locations
width = 0.25  # the width of the bars
multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for attribute, measurement in penguin_means.items():
    offset = width * multiplier
    rects = ax.bar(x + offset, measurement, width, label=attribute)
    ax.bar_label(rects, padding=3)
    multiplier += 1

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Length (mm)')
ax.set_title('Penguin attributes by species')
ax.set_xticks(x + width, species)
ax.legend(loc='upper left', ncols=3)
ax.set_ylim(0, 250)

plt.show()
```

Example



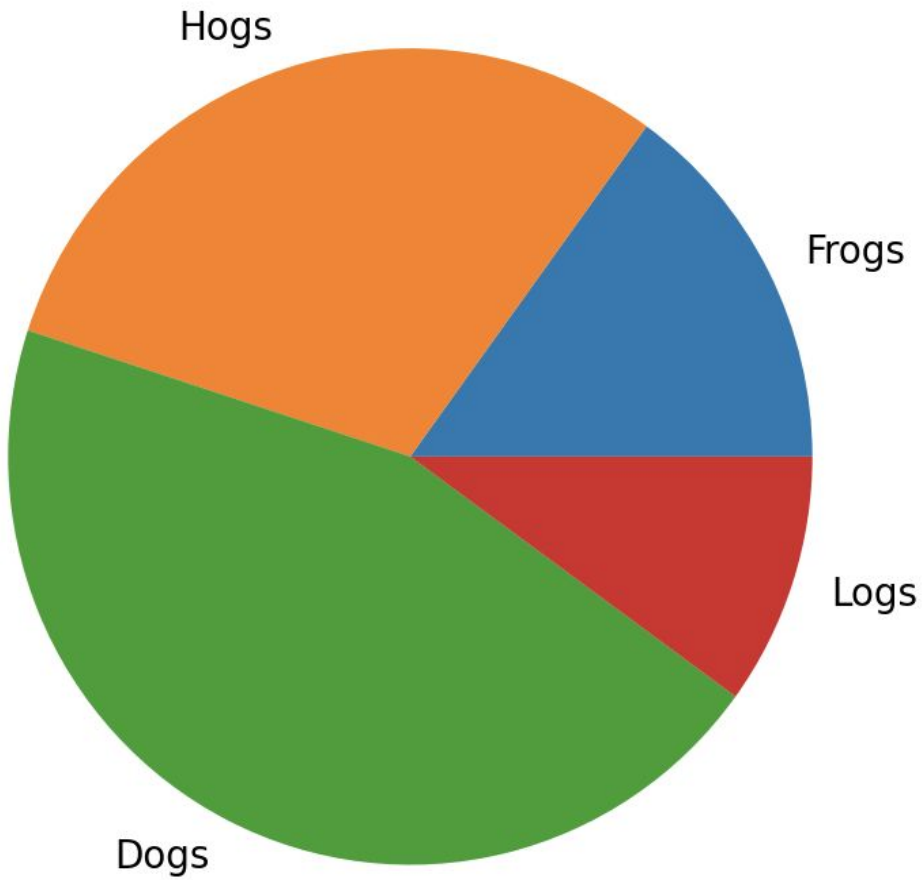

```
plt.subplot(2, 1, 1)
plt.plot(x1, y1, 'o-')
plt.title('A tale of 2 subplots')
plt.ylabel('Damped oscillation')

plt.subplot(2, 1, 2)
plt.plot(x2, y2, '.-')
plt.xlabel('time (s)')
plt.ylabel('Undamped')

plt.show()
```

Example

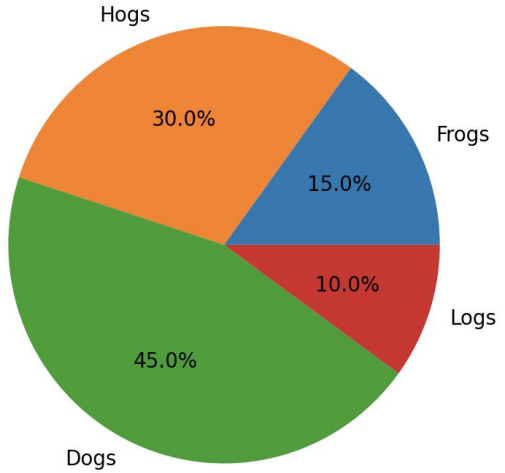



```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import colors
from matplotlib.ticker import PercentFormatter

# Create a random number generator with a fixed seed for reproducibility
rng = np.random.default_rng(19680801)
N_points = 100000
n_bins = 20

# Generate two normal distributions
dist1 = rng.standard_normal(N_points)
dist2 = 0.4 * rng.standard_normal(N_points) + 5

fig, axs = plt.subplots(1, 2, sharey=True, tight_layout=True)

# We can set the number of bins with the *bins* keyword argument.
axs[0].hist(dist1, bins=n_bins)
axs[1].hist(dist2, bins=n_bins)
```

Example




```
import matplotlib.pyplot as plt
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]

fig, ax = plt.subplots()
ax.pie(sizes, labels=labels)

fig, ax = plt.subplots()
ax.pie(sizes, labels=labels, autopct='%1.1f%%')
```



Data Sources - Meteo@Uniparthenope

http://193.205.230.6/files/ww33/d03/archive/

Index of /files/

../		
aiq3/	15-Mar-2023 22:56	-
gaiola/	11-Dec-2023 17:54	-
rdr1/	22-Sep-2020 14:10	-
rdr2/	22-Sep-2020 14:10	-
rms3/	22-Sep-2020 14:06	-
sam3/	25-Mar-2022 13:36	-
wcm3/	15-Mar-2023 14:56	-
wrf5/	16-Mar-2023 05:26	-
ww33/	15-Mar-2023 23:00	-
wrf_output.nc	21-Feb-2024 15:41	4725494622

Data Sources - Meteo@Uniparthenope

http://193.205.230.6/files/
http://193.205.230.6/files/ww33/d02/archive/

Index of /files/

../		
aiq3/	15-Mar-2023 22:56	-
gaiola/	11-Dec-2023 17:54	-
rdr1/	22-Sep-2020 14:10	-
rdr2/	22-Sep-2020 14:10	-
rms3/	22-Sep-2020 14:06	-
sam3/	25-Mar-2022 13:36	-
wcm3/	15-Mar-2023 14:56	-
wrf5/	16-Mar-2023 05:26	-
ww33/	15-Mar-2023 23:00	-
wrf_output.nc	21-Feb-2024 15:41	4725494622

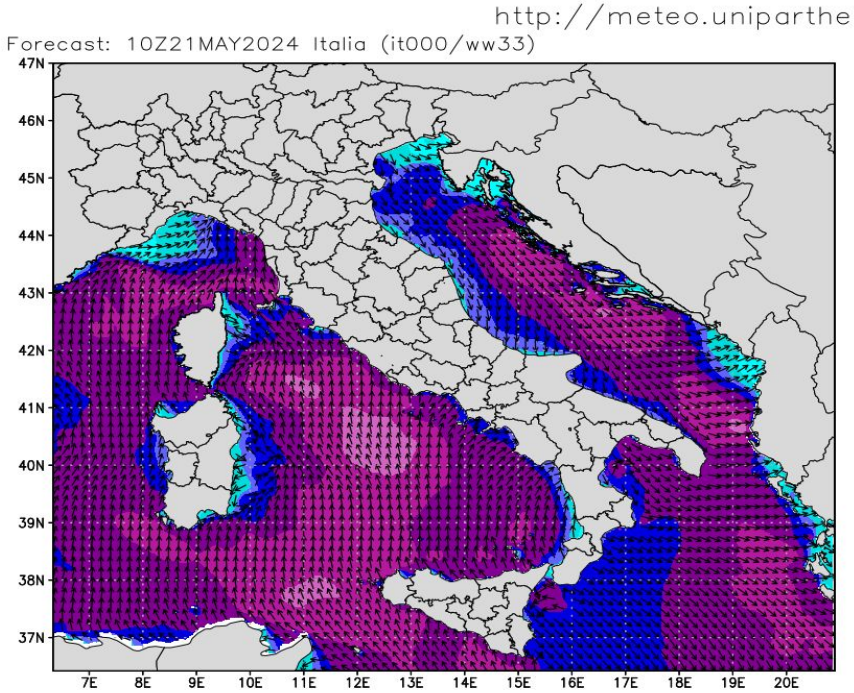
Data Sources - Meteo@Uniparthenope

http://193.205.230.6/files/ww33/d02/archive/

http://193.205.230.6/files/ww33/d02/archive/2024/05/25/

Index of /files/ww33/d02/archive/2024/05/25/

../		
ww33_d02_20240525z0000.nc	21-May-2024 07:27	24448506
ww33_d02_20240525z0100.nc	21-May-2024 07:27	24448506
ww33_d02_20240525z0200.nc	21-May-2024 07:27	24448506
ww33_d02_20240525z0300.nc	21-May-2024 07:27	24448506
ww33_d02_20240525z0400.nc	21-May-2024 07:27	24448506
ww33_d02_20240525z0500.nc	21-May-2024 07:28	24448506
ww33_d02_20240525z0600.nc	21-May-2024 07:28	24448506
ww33_d02_20240525z0700.nc	21-May-2024 07:28	24448506
ww33_d02_20240525z0800.nc	21-May-2024 07:28	24448506
ww33_d02_20240525z0900.nc	21-May-2024 07:28	24448506
ww33_d02_20240525z1000.nc	21-May-2024 07:28	24448506
ww33_d02_20240525z1100.nc	21-May-2024 07:29	24448506
ww33_d02_20240525z1200.nc	21-May-2024 07:29	24448506
ww33_d02_20240525z1300.nc	21-May-2024 07:29	24448506
ww33_d02_20240525z1400.nc	21-May-2024 07:29	24448506
ww33_d02_20240525z1500.nc	21-May-2024 07:29	24448506
ww33_d02_20240525z1600.nc	21-May-2024 07:29	24448506
ww33_d02_20240525z1700.nc	21-May-2024 07:30	24448506
ww33_d02_20240525z1800.nc	21-May-2024 07:30	24448506
ww33_d02_20240525z1900.nc	21-May-2024 07:30	24448506
ww33_d02_20240525z2000.nc	21-May-2024 07:30	24448506
ww33_d02_20240525z2100.nc	21-May-2024 07:30	24448506
ww33_d02_20240525z2200.nc	21-May-2024 07:30	24448506
ww33_d02_20240525z2300.nc	21-May-2024 07:31	24448506



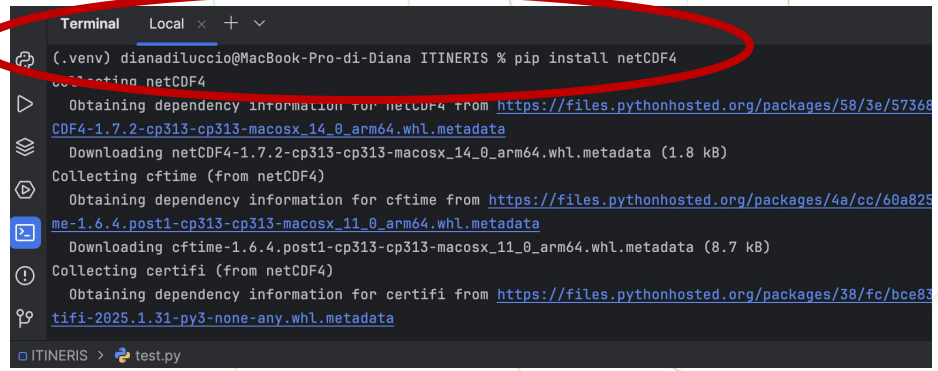
https://www.earthinversion.com/utilities/reading-NetCDF4-data-in-python/

https://matplotlib.org/basemap/stable/

Explore a NetCDF file

```
pip install netCDF4
```

```
import netCDF4
f = netCDF4.Dataset('ww33_d02_20240525Z0000.nc')
print(f)
```



root group (NETCDF4 data model, file format HDF5):
dimensions(sizes): time(1), depth(11), latitude(941), longitude(1297)
variables(dimensions): float32 time(time), float32 **longitude**(longitude), float32 **latitude**(latitude), float32 **hs**(time, latitude, longitude), float32 lm(time, latitude, longitude), float32 fp(time, latitude, longitude), float32 **dir**(time, latitude, longitude), float32 **period**(time, latitude, longitude)
groups:

```
print(f.variables.keys()) # get all variable names
```

dict_keys(['time', 'longitude', 'latitude', 'hs', 'lm', 'fp', 'dir', 'period'])

Explore a NetCDF file

```
hs = f.variables['hs'] # temperature variable
print(hs)
```

```
<class 'netCDF4._netCDF4.Variable'>
float32 hs(time, latitude, longitude)
    _FillValue: 1e+37
    description: Significant wave height
    long_name: significant height of wind and swell waves
    standard_name: sea_surface_wave_significant_height
    globwave_name: significant_wave_height
    units: m
    scale_factor: 1.0
    add_offset: 0.0
    valid_min: 0.0
    valid_max: 100.0
unlimited dimensions:
current shape = (1, 941, 1297)
filling on
```

Explore a NetCDF file

```
for d in f.dimensions.items():
  print(d)
```

```
('time', <class 'netCDF4._netCDF4.Dimension'>: name = 'time', size = 1)
('depth', <class 'netCDF4._netCDF4.Dimension'>: name = 'depth', size = 11)
('latitude', <class 'netCDF4._netCDF4.Dimension'>: name = 'latitude', size = 941)
('longitude', <class 'netCDF4._netCDF4.Dimension'>: name = 'longitude', size = 1297)
```

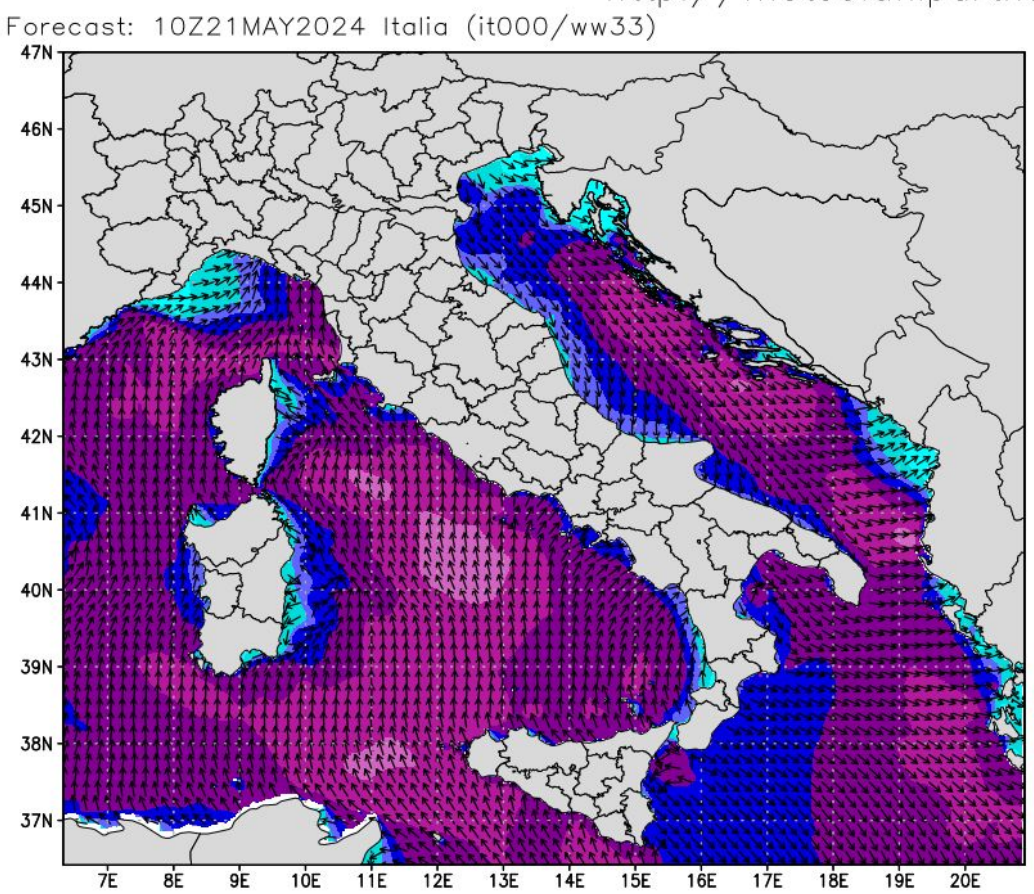
```
dim=hs.dimensions
print(dim)

forma=hs.shape
print(forma)
```

```
('time', 'latitude', 'longitude')
(1, 941, 1297)
```

Explore a NetCDF file

```
hs = hs[:]
print(hs)
```

```
[[[    nan     nan     nan ... 0.31213549 0.31755057 0.3228755 ]
  [    nan     nan     nan ... 0.31520805 0.3224794  0.3242774 ]
  [    nan     nan     nan ... 0.31520805 0.3224794  0.3242774 ]
  ...
  [    nan     nan     nan ...     nan      nan      nan]
  [    nan     nan     nan ...     nan      nan      nan]
  [    nan     nan     nan ...     nan      nan      nan]]]
```



```
from netCDF4 import Dataset as NetCDFFile
import matplotlib.pyplot as plt
import numpy as np
import os
from mpl_toolkits.basemap import Basemap

shapefile_path = "shapefiles/europe"
data_file = "ww33_d02_20240528Z1900.nc"
var_name = "hs"
countour = True
dir_var_name = "dir"
type = "angle"

point_lat = 40.8
point_lon = 14.2

nc = NetCDFFile(data_file)
lat = nc.variables['latitude'][:]
lon = nc.variables['longitude'][:]
minLat = lat[0]
maxLat = lat[-1]
minLon = lon[0]
maxLon = lon[-1]
lons, lats = np.meshgrid(lon, lat)
var = nc.variables[var_name][:][0]

var_long_name = nc.variables[var_name].getncattr('long_name')
if 'long_name' in nc.variables[var_name].ncattrs()
else var_name
var_units = nc.variables[var_name].getncattr('units')
if 'units' in nc.variables[var_name].ncattrs()
else ''
```

```
basemap = Basemap(projection='merc',
                  llcrnrlon=8, #minLon
                  llcrnrlat=37, #minLat
                  urcrnrlon=15, #maxLon
                  urcrnrlat=45, #maxLat
                  resolution="l")

shapefile_name = os.path.basename(shapefile_path)
basemap.readshapefile(shapefile_path, shapefile_name, default_encoding='iso-8859-15')

point_x, point_y = basemap(point_lon, point_lat)
basemap.plot(point_x, point_y, 'rx', markersize=15)

# plot lines of longitudes
basemap.drawparallels(np.arange(-90.,+90.,2),labels=[1,0,0,1])
# plot latitudes
basemap.drawmeridians(np.arange(-180.,180,2),labels=[1,0,0,1])

filled_contours = basemap.contourf(lons, lats, var, cmap='jet', latlon=True)
if countour:
        contours = basemap.contour(lons, lats, var, colors='k', latlon=True)
        plt.clabel(contours, inline=True, fontsize=8, fmt='%1.1f', colors='white')

scale = 10
skip = 16
skip2 = (slice(None, None, skip), slice(None, None, skip))
```

```
if type == "versor":
        u = nc.variables[u_var_name][:][0]
        v = nc.variables[v_var_name][:][0]

        uv = np.hypot(u[skip2], v[skip2])
        basemap.quiver(
                lons[skip2], lats[skip2],
                u[skip2] / uv, v[skip2] / uv,
                latlon=True, scale=scale, scale_units="inches", pivot='middle', linewidths=.01,  edgecolors='gray')
if type == "angle":
        dir = nc.variables[dir_var_name][:][0]
        dir = 90-dir[skip2]
        dir[dir<0]+=360
        dir[dir>360]-=360
        dir[dir==360]=0
        dir = dir * 0.0174533
        dir1 = np.cos(dir)
        dir2 = np.sin(dir)
        basemap.quiver(lons[skip2], lats[skip2],dir1, dir2,latlon=True, scale=scale, scale_units="inches", pivot='middle', linewidths=.01,  edgecolors='gray')

colorbar = plt.colorbar(filled_contours)
colorbar.set_label(f'{var_long_name} ({var_units})')

plt.show()
```

THANKS!

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani