Exercises

Exercises may be the most important part of this module container. We suggest that you do them actively and in small groups. Really, the only way to learn Python is to do it.

0 Exercises 0: Getting ready

0.1 Installation and development environment

Python is an open-source environment with significant input from the user community. Many of the developments are packaged in libraries designed for specific tasks. These need to be installed prior to usage which at times can be a bit tricky because of dependencies between different libraries. In order to alleviate you from installation difficulties that we have experienced in the past, we provide a fully installed python environment in a virtual environment that can be run on any laptop. If you use your own Laptop for the course this can be useful. Follow these steps to download and run the environment on your own computer:

XX Willi fill in details with one tiny Hello World Example in Visual Studio Code so that they know everything works. XX

It is best if you do that BEFORE the first class, then we can hit the road running. If you prefer to run your own installation (let's say with Anaconda), feel free to do it: XX Required packages are matplotlib, numpy, pandas, ... XX

Solutions

test

```python
import numpy as np

a = 5.0
b = 10.0
c = 'Hallo'
print('Hello World.')
print('Hello World.')
print('Hello World.')
for k in np.arange(0,10):
    print(k)
  # print(;asd)
```

Src/Ex1/HelloWorld.py

1 Data types and Visualization

After the first subtle introduction, you should be able to fill out this table (< 5 mins).

1.1 Data Types

Data Type	How to write in Python?	Geo- Env. Context
Integer		
Float		
Boolean		
List		
Tuple		
Dictionary		
String		
Numpy Array [Vector]		c
Numpy Array [Matrix]		c

What is the difference between an element of an array, and the index of an array?

Solutions

Data Type	How to write in Python?	Geo- Env. Context
Integer	a = 1	Sample number
Float	t = 20.9	Temperature value
Boolean	TRUE, FALSE	Above or Below threshold?
List	[20.9,23.3,23.1]	Temperature time series
Tuple	MLoc = (”Location”,-71.1,8.4)	Measurement Location
Dictionary	s = ”Name:”,”samp1”,”Year:”,2022	Data structure
String	”Error!”	Error message
Numpy Array [Vec.]	np.array[(43.1,2.09,1)]	Time series
Numpy Array [Mat.]	np.array[(43.1,2],[.09,1)]	Image

The element of an array is the value for a given index (indices are only integers). The first element of an array is evaluated with the index 0. The last one with the index -1.

1.2 Loading and visualization of ASCII-txt input data

Load the file `monthly_in_situ_co2_mlo_ready4loading.txt` which is the Keeling curve. If you don't know what this is please inform yourself. The data are decimal years in the first column and CO2 in ppm in the second column. A lack of data is marked with negative numbers.

- Load the data (e.g., using numpy.loadtxt()).
- How many datapoints are in the time series? What are the dimensions of the data array?
- Visualise the data (e.g., using matplotlib) with meaningful x- and y- limits and axis labels. Find out if the data are better visualised as points, curves or both. Change the color and symbols of the points.
- Using python functions calculate basic quantities such as the mean, minimum and maximum of your time series. Are those values meaningful?
- Visualize only the first ten elements, the last ten elements, and elements from 20 to 30.

1.3 Visualization of mathematical functions

Make use of some numpy functions (arange) and get a feel for some vector manipulation.

- Visualize $f(t) = \sin(\omega t + \phi_0)$ for a given frequency $\omega = 1.2$Hz and phase shift ϕ_0 over the time interval $t = 0...10$ at discrete time intervals dt=0.001 s.
- Visualize $\mathrm{f}^2(\mathrm{t})$. What happens if you reduced dt to 1 s ?
- Visualize a Gaussian peak:
$$f(x) = Ae^{-\frac{(x-x_0)^2}{\sigma_x}}$$
where x is the independent variable and the factors x_0, A, and σ_x determine the shape of the function. Make sure that you understand what each parameter pertains to.

1.4 Basic filtering and data manipulation

Load the Keeling curve from 1.2 again. We have understood that no data values are marked with negative numbers. Let's remove those with a for loop (there are better ways at a later stage). Take your time, if this is your first time of writing a loop this can take some time.

- Print all CO_2 values on the screen using a for loop. (How could this be done much easier?)
- Print only the time intervals where no data are available
- Create a new vector where the no-data values are removed.
- Visualize the new vector. Anything different compared to 1.2?

Start a new for loop block and calculate the time derivative of the time series. Visualize it.

1.5 Loops continued.

Visualize a 2D Gaussian peak:

$$f(x,y) = Ae^{-\frac{(x-x_0)^2}{\sigma_x} - \frac{(y-y_0)^2}{\sigma_y}}$$

using a nested for loop. This is harder. In essence you will need to fill a 2D array and visualize it with colors using plt.pcolormesh(). Then do the same but replace the nested for loops using numpys "meshgrid". What is easier?

. This is harder. In essence you will need to fill a 2D array and visualize it with colors using plt.pcolormesh(). Then do the same but replace the nested for loops using numpys "meshgrid". What is easier?

2 Functions and loops

2.1 Linear regression

Load the Keeling curve from 1.4. Fit a first and second order polynomial to the dataset. We understand that this type of fitting must have been solved by somebody else already and therefore use functions to do the task, specifically numpy's polyfit and polyval combination. Visualize the seasonality by subtracting the fit from the observations. Which models (i.e. first or second order polynomial) fits better and what does that mean?

2.2 Writing your own functions

Calculating the time (or spatial) derivative of a 1D dataset is something that may occur quite frequently. Occuring tasks are best written in functions for many reasons (name at least two). Write a function called *ForwardDifferencingXY* that takes a 2 x n array as input where the independent variable (e.g., time) is stored in the first column, and the dependent variable (e.g., CO_2) in the second column. The output should also be a 2 x n array with the independent variable in the first column and the derivative of the dependent variable in the second colum.
Here you should do this for the Keeling curve from 1.4 and for the Gausspeak from 1.3.

- Use your function *ForwardDifferencingXY* to calculate rates of change for the Keeling curve 1.4, the sinsoid and the 1D Gauss Peak 1.3
- Make your function more robust so that it catches wrong use interactions (e.g., passing on a n x 2 array instead of 2 x n array)
- Make your user function more user friendly by providing and optional figure with sub-panels showing the original data on top, and the time derivative at the bottom.
- Add noise to your input data using numpys randn function, what does that do to your derivatives. Why?

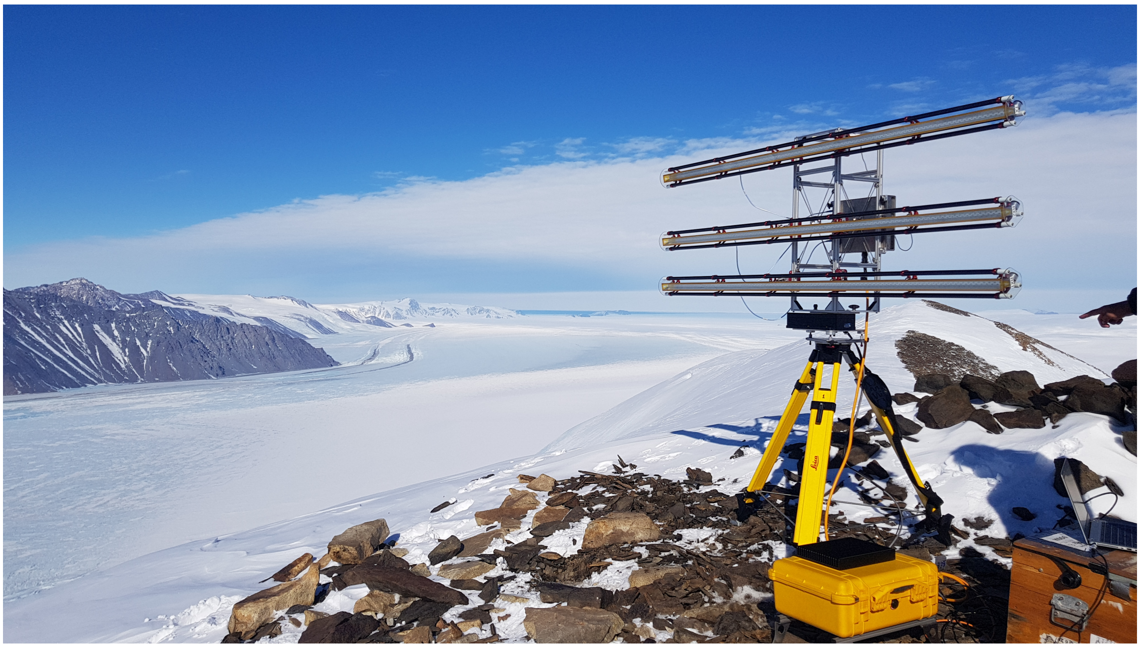

Figure 1: View downstream of Priestley Glacier. At the far end you can see the ocean. The ice starts to float on ocean water approximately at the end of the mountain chain located at the glaciers true left side.

3 More advanced exercises

3.1 GPS Data analysis

Today's topic is focused on Antarctica. The data file *XYShirase_GPS_Small.txt* is the output of GPS measurements located at Priestley Glacier feeding into the Nansen Ice Shelf. Ice shelves are the floating extensions of the Antarctic Ice Sheet. Beneath them is only water and consequently the ice lifts up and down with ocean tides. The data file contains 6 columns (1: Position in polar stereographic x, 2: Position in polar stereographic y, 3: Longitude, 4: Latitude, 5: Elevation, 6: Time). The polar stereographic projection (EPSG:3031) is a rectangular coordinate system with units meters (sort of comparable to UTM coordinates). The time is given in modified Julian days which are days since November 17, 1858 (just for information, not needed in exercise). Please complete the following tasks. Some are easier, some are harder.

- Visualize the movement of the GPS station in a x-y scatter plot (easy)
- Visualize the vertical GPS position as a time series (easy). Which tidal regime is visible here?
- Calculate the mean horizontal GPS velocity in the observational period Pythagoras is your friend. Always. (medium).
- The vertical position has a fair amount of scatter. This is normal, as GPS measurements are about three times worse in the vertical compared to the horizontal direction. Smooth the data with your own moving average filter with variable window size. This filter should average the vertical position at time t as a mean value for a given number of positions recorded close to t. Assume that the data

are regularly spaced in time. Cut-off the smoothed vector at the beginning and the end. It can be smaller than the data file (medium, involves for loops).

- Find functions in python that do a similar job and compare your results. (easy)
- Discuss the pitfalls of filtering and suggest improvements. Memorize that a moving average filter is usually not the best way to go for noise reduction. Better options are weighted moving average or bandpass filters.
- Design your own FIR low-pass filter, e.g., following `https://tomroelandts.com/articles/how-to-creat` This is advanced and outside the scope of this course. Only do this if everything else is easy for you. We will not discuss the theory here.

3.2 Cellular Automata: Abelian Sandpile Model

Self-organized criticality is a fascinating subject that occurs in a number of geo- and environmental processes. Your job is to simulate an Abelian Sandpile Model which has in fact some quite realistic insights into the nature of landslides. The rules of this system are (Wikipedia entry has a more formal version of it):

- Choose a random location on your grid and drop a grain of sand (i.e. add +1)
- Continue unless one of your grid points as more than 3 grains.
- If a grid point as more than 3 grains then empty that cell, and distribute all grains to gridpoints in the surrounding (this is a landslide). Then continue.
- Grains fall of the boundaries are lost.

Implement this set of rules on a grid and visualize its evolution.

3.3 Cellular Automata: Langton's Ant as a 2D example

Imagine you are in a 2D universe that is intially completely white on consists out of 100 x 100 cells that you can walk on. Your life depends on two rules only:

- At a white square, turn 90 right, flip the color of the square, move forward one unit
- At a black square, turn 90 left, flip the color of the square, move forward one unit

Will you ever leave your world? The rules are simple enough, but an answer to this question is very difficult if not impossible. However, you can find the answer in Python. Go for it (with our help)! Cellular automata such as this have many applications in Geosciences. Another important model is, e.g., the sandpile model.