Adv. Geosci., 45, 295–303, 2018
https://doi.org/10.5194/adgeo-45-295-2018

Advances in Geosciences
Open Access
EGU

Programming as a soft skill for project managers: How to have a computer take over some of your work

Nikolay V. Koldunov[1,2] **and Luisa Cristini**[2]

[1]MARUM – Center for Marine Environmental Sciences, Bremen, Germany
[2]AWI – Alfred Wegener Institute for Polar and Marine Research, Bremerhaven, Germany

Correspondence: Nikolay Koldunov (nikolay.koldunov@awi.de)

Received: 1 July 2018 – Revised: 20 September 2018 – Accepted: 9 October 2018 – Published: 18 October 2018

Abstract. Large part of the project manager's work can be described in terms of retrieving, processing, analysing and synthesizing various types of data from different sources. The types of information become more and more diverse (including participants, task and financial details, and dates) and data volumes continue to increase, especially for large international collaborations. In this paper we explore the possibility of using the python programming language as a tool for retrieving and processing data for some project management tasks. python is a general-purpose programming language with a very rich set of libraries. In recent years python experienced explosive growth leading to development of several libraries that help to efficiently solve many data related tasks without very deep knowledge of programming in general and python in particular. In this paper we present some of the core python libraries that can be used to solve some typical project management tasks and demonstrate several real-world applications using a HORIZON 2020 type European project and as example.

1 Introduction

Project management deals with many information flows that should be found, processed, reorganised and visualized. This includes extracting information from documents (e.g., person-months, task details, team composition), compiling data (e.g., from spreadsheet tables), sorting data (e.g., abstract titles and authors from large tables) and more. In the following we are going to call all information that a project manager deals with data including information received via email, lists of participants (e.g., of a conference or meeting), financial information (e.g., cost reports, team efforts), statistics (e.g., gender, international composition), roles and responsibilities of the partners (e.g., in task teams).

A way to cope with these flows of information is to use a general-purpose office software. The most popular are Microsoft Office products, and in particular MS Excel is a software of choice when it comes to processing moderately large amount of data (not exceeding MS Excel limit of 1 048 576 rows and 16 384 columns per spreadsheet) that can be presented in a table form.

One of the biggest challenges, however, is to actually gather the data in the table (or any other desirable) form. This process is called data wrangling or data extraction. The input data for the tables is received by the project manager usually in a variety of forms that includes emails, MS Word documents, .pdf files and increasingly web pages and postings in social media. The first task of the project manager is to parse this information, and organise it in an appropriate way, often an MS Excel spreadsheet. The information is then analysed and exported as graphics, a Word document or one of the other many formats the project manager has to work with.

Especially difficult and time consuming is the step of data extraction and parsing as data preparation can take large portion of the work required to complete a data-related task. When it comes to data extraction and parsing many project managers rely on automatic pre-existing tools (e.g. MS Excel import) that, however, often are source of systematic errors and also fail to import the right data. They also do not allow to create an automatic process for data export from any existing format and, more importantly, for parsing and organisation of the exported data in the form one requires. When pre-existing automatic tools fail, the common practice is to turn to the manual extraction of data by simply copying information from the source document and inserting it in to

the MS Excel spreadsheet. This is however a tedious, time consuming and error-prone process. This data extraction-organisation-synthesis process is comparable among different fields that use data beyond project management and often the amount of data is so large that the manual extraction is practically impossible, so the automatization of this process is the only feasible option. The amount of data needed to be processed by project managers grows constantly so the science project management community could benefit greatly from leveraging experience in automatic data extraction and processing already gained in other communities.

The best way to gain full flexibility for data extraction, parsing and processing is using a programming language. There are many popular programming languages that are used for data analysis such as *R* or Java. However in this paper we explore the possibility of using python. python is considered an easy to learn programming language and often used as a first language to teach programming fundamentals (e.g. Massachusetts Institute of Technology, Caltech, UC Berkeley and many others, e.g., Guo, 2014). In recent years, python gained popularity also in scientific data analysis and data processing (Lin, 2014) resulting in the development of a considerable amount of quality code packages (libraries) and documentation available. Another advantage of python that we are going to actively explore in this paper is the well established platform for writing code and data analysis in the browser, Jupyter Notebook, which further improves python learning experience (Jacobs et al., 2016).

Recognising the growing need for a tool to process data for project management tasks and the capabilities of python, the main objectives of this paper are:

1. Identify some project management tasks related to data extraction and processing that can be automatized through programming.

2. Introduce some of the core python libraries that can be used to solve the identified tasks.

3. Demonstrate several real-world applications using the documentation of a European HORIZON 2020 project and a national-funded project as sources of data.

The structure of the paper is as follows: We identify basic project management tasks that can be automatized in Sect. 2. In Sect. 3 we provide description of the case studies performed on the basis of EU Horizon 2020 and national project. Finally, Sect. 4 contains discussion and conclusions.

2 Project management activities and python libraries to automatize them

In this section, we identify several key project management data related activities that can be automatized by using programming and for each activity we provide a short description of the way it can be atomatized using python. In particular we mention several python libraries that will be used in the case studies of the next section. The python libraries are collections of programs that expand its basic functionality. Most of the python libraries are free and open source. Code examples from this section can be found in supporting materials and online at https://github.com/koldunovn/Programming_for_project_managers (Koldunov and Cristini, 2018, last access: 12 October 2018) in the python_examples.ipynb notebook. The README file in the repository describes how to get started without installation of python, just by using online interactive computational environment.

2.1 Text reformatting

Text reformatting is a quite general task that project managers have to deal with. Automatization can help when large amounts of similar text entries (e.g. names, phone numbers, addresses, publications) have to be available in an harmonised format. One of the main features of python is simple text processing. The basic text processing functionality that allows to work with text data (strings, in python terminology) is part of the language core and does not require installation of any additional libraries. This allows users to, for example, split in parts, reorder and concatenated text. The code snippet below shows a python code that demonstrates an example of work with strings in python. The code converts an initial string that contains the name that should be reformatted (all lowercase, family name, given name name order) to the desired format (capitalize first letters, given name, family name name order). The output of the program is indicated by ">>" sign.

Code snippet 1

```
name = 'lastname firstname'
name_split = name.split()
name_result = (name_split[1]+'
'+name_split[0]).title()
print("Processed name:" + name_result)

>> Processed name: Firstname Lastname
```

We omit to describe the language syntax, as this is out of the scope of this paper. A good introduction to python strings for interested readers can be found at https://developers.google.com/edu/python/strings (last access: 12 October 2018). Here we demonstrate that the code that needs to be written to perform such an operation is short and does not require any additional libraries to be imported. Automatisation of this process is very useful especially when this operation has to be repeated several times. The working version of the code is part of the Supplement. With the code above one can simply replace the "lastname firstname" string and run the code to get the result. Another way is to have a long list of text entries that should be processed and

run the code on them. Below is the code that demonstrates processing of several text entries at once:

Code snippet 2

```
names=['rhodes marie', 'key sarah',
'christensen joe']
for name in names:
  name_split=name.split()
  name_result=(name_split[1]+'
  '+name_split[0]).title()
  print("Processed name:"+name_result)
```

```
>> Processed name: Marie Rhodes
>> Processed name: Sarah Key
>> Processed name: Joe Christensen
```

In this example code we use the "for loop", one of the standard control structures in most programming languages that allows to repeat a set of operations on a sequence of input values. When the number of values one would like to process grows, at some point putting the list values directly in to the text of the computer program become unpractical. A more convenient option is to read them from another source in to the program. We are not going to describe all the ways text data can be imported in python, but in the next section we will mention several ways that can be useful for project managers.

2.2 Work with table data

Data tables are at the core of many data processing and analysis tasks. The most popular software to work with such a data is MS Excel. MS Excel works well for most tasks when the data is already organised and harmonised, but is inflexible and problematic to use for data preparation. In python the most popular solution to work with table data is a library called pandas (McKinney, 2010, 2011). This library provides tools to work with structured datasets (tables in particular) and is widely used in finance, statistical sciences, Earth sciences and many other fields. pandas provides capabilities to work with MS Excel files, so that information can be both imported from and exported back to MS Excel spreadsheets.

Below we show a code snippet that converts Table 1 to Table 2. The operations performed include opening the MS Excel spreadsheet with Table 1, converting last name to uppercase and adding the DR title, capitalizing the first name, reordering of columns and saving data back to MS Excel spreadsheet:

Code snippet 3

```
import pandas
names=pandas.read_excel('./names.xlsx')
names['Last name']='DR '+names['Last
name'].str.upper()
names['First name']=names['First
name'].str.capitalize()
names=names[['First name', 'Last
name']]
# saving
writer=pd.ExcelWriter
('names_processed.xlsx')
names.to_excel(writer, 'Sheet1')
writer.close()
```

Table 1. Input for Example 3.

Last name	First name
rhodes	marie
key	sarah
christensen	joe

Table 2. Output of Example 3.

First name	Last name
Marie	DR RHODES
Sarah	DR KEY
Joe	DR CHRISTENSEN

In this example, we use capabilities that are not included in the core python distribution but provided by the pandas library. Before using it, the library has to be installed and then imported inside the program with an import statement at the beginning of the code. The example above is used as a demonstration that the pandas library makes it possible to apply changes to the data in a whole column at once. In other words, you can work with all strings in the column in the same way as you would work with a single string, without the need for looping over every string in the column.

Beside MS Excel spreadsheets, another popular format for table data exchange is CSV (comma separated values). This is a simple text document with columns separated by commas or another standard delimiter. This format is one of the main import/export formats for pandas and there is a rich functionality that allows for data homogenisation already while importing.

2.3 Data extraction from different sources

Data does not always come in form of the MS Excel spreadsheet or CVS document, and often has to be extracted from different sources including text files or MS Word documents. In this section we will provide a short overview of ways to extract data from those sources using python.

Table 3. Input for Example 4.

LN rhodes
FN marie
LN key
FN sarah
LN christensen
FN joe

2.4 Text files

Text files can be easily opened with python and simple data extraction can be performed without using additional libraries. The code snippet below shows a simple program that opens a text file and reads it line by line (see Table 3 for file content of the file). Each line is checked for the starting sequence ("LN" or "FN") and its content is added ether to "lastnames" of "firstnames" variables.

Code snippet 4

```
f=open('./names.txt')
firstnames=[]
lastnames=[]
for line in f.readlines():
  if line.startswith('LN'):
    lastnames.append(line.split()[1])
  elif line.startswith('FN'):
    firstnames.append(line.split()[1])
  else:
    pass
f.close()
```

The content of the "firstnames" and "lastnames" variables can then be processed as shown in the previous examples or exported to Excel spreadsheet. Reading the file line by line, getting information from each line and putting it to the variable is a very common pattern in data extraction and python allows to do it relatively easily.

When working with numerical information only, an even easier way to open a set of numbers than reading values one by one is using the numpy library, a fundamental package for doing numerical computations in python. One of the features of numpy is its ability to open text files containing numbers and create data structure called n-dimensional array. The code in the example below reads a text file that contain the lines shown in Table 4, reshapes the data into 3×3 array and calculates the average over the third column (counting in python starts from zero).

Code snippet 5

```
import numpy
numbers=numpy.loadtxt('./numpers.txt')
reshaped=numbers.reshape(3,3)
reshaped[:,2].mean()
```

Table 4. Input for Example 5.

1980
12
0.37
1981
1
0.8
1982
2
0.22

```
>> 0.46333333333333332
```

The data in Table 4 represent year, month and a given value, all appearing in one column. Using only four lines of code (including importing the numpy library) it is possible to create the mean of the data values.

2.4.1 MS Word documents

The document file format that is used by the MS Word (.docx) is very different from the plain text files or CSV documents. It contains complicated data structures since all the information about text formatting, images, tables and other objects is also included. The library that can help to extract text and table information from MS Word documents and that is used in the case study below is called docx. Docx converts MS Word documents into the data structures that can be easily handled by python and allows data extraction (mainly text). The examples below demonstrate how one can extract tables from MS Word documents using docx. The library can also help to create MS Word documents, which can be useful also to create MS Word documents automatically. A simple example of such a task is the creation of name badges for a conference participants. An advanced exercise could be to automatically create more complicated reports in MS Word. This however would require good proficiency in python.

3 Case studies

In this section we will briefly cover several cases of using python for extracting processing and exporting information. The complete code with more extensive explanations in the form of Jupyter Notebooks is located in the Supplement and at https://github.com/koldunovn/Programming_for_project_managers (last access: 12 October 2018). Jupyter Notebooks (Shen, 2014; Kluyver et al., 2016) allow to combine code execution and narrative with explanations of what the code does in the system that just works in the internet browser. A Jupyter Notebook is is splitted in to the text and code cells where the result of the code execution is outputted under the code cells . In this section we are going to describe the

(a)

1.3.2. WT2 list of deliverables

Deliverable Number[14]	Deliverable Title	WP number[9]	Lead beneficiary	Type[15]	Dissemination level[16]	Due Date (in months)[17]
D1.1	Ocean deliverable 1	WP1	1 - PARTNER 1	Report	Public	4
D1.2	Ocean deliverable 2	WP1	1 - PARTNER 1	Other	Public	12
D1.3	Ocean deliverable 3	WP1	8 - PARTNER 8	Other	Public	24
D1.4	Ocean deliverable 4	WP1	9 - PARTNER 9	Report	Public	30
D1.5	Ocean deliverable 5	WP1	7 - PARTNER 7	Report	Public	48

(b)

	A	B	C	D	E	F	G	H	I
		Deliverable number	Deliverable title	WP number	Lead beneficiary	Type	Emination	Date (in mo	Due date
0		D1.1	Ocean deliverable 1	WP1	1 - PARTNER 1	Report	Public	4	2017-02-15
1		D1.2	Ocean deliverable 2	WP1	1 - PARTNER 1	Other	Public	12	2017-10-15
2		D1.3	Ocean deliverable 3	WP1	8 - PARTNER 8	Other	Public	24	2018-10-15
3		D1.4	Ocean deliverable 4	WP1	9 - PARTNER 9	Report	Public	30	2019-04-15
4		D1.5	Ocean deliverable 5	WP1	7 - PARTNER 7	Report	Public	48	2020-10-15

Figure 1. The "list of deliverables" table from standard EU Horizon 2020 DoW document **(a)**, and data automatically extracted from this document and saved to MS Excel spreadsheet **(b)**.

workflow and obtained results using real world data sources, whereas the detailed explanation of the python code is given in the Supplement Jupyter Notebooks.

3.1 Extract information from EU Horizon 2020 DoW

The Description of Work is one of the main documents of an EU H2020 project because it explains the project work plan in detail, including the description of each work package and task. It includes many tables (e.g., for deliverables, milestones, resources) that are used as reference by the project manager to keep track of the project progress. However organisation of this document is not optimal, and getting structured information from it can be challenging, especially when many partners are involved. In the example below we extract several tables from the project's DoW, perform some simple data sorting and output the result. The detailed notebook with the code and detailed explinations can be found online at https://github.com/koldunovn/Programming_for_project_managers/blob/master/EU_HORIZON2020.ipynb (last access: 12 October 2018) and in the Supplement.

First, we open the file "EU-PYTHON.docx" in python using the docx library. One can directly access tables in this document, so we select a table with the list of work packages, extract the information, put it into the pandas data structure (DataFrame) and output to MS Excel spreadsheet. Similar steps are repeated with the "list of Beneficiaries" table. The "list of deliverables" is more complicated as it spans through several pages. In the Supplement we show how to extract information from such multi page tables and then combine it into pandas DataFrame. The "list of deliverables" contains only "due dates" for project deliverables in months starting from the beginning of the project. If the project start date is known, it is easy to convert this information to real time. The parts of the source "list of deliverables" and corresponding processed parts exported to Excel are shown on Fig. 1.

In the second part of the notebook we show how to perform some analysis of the table directly in python including selection of all deliverables for one partner, sorting deliverables by the date, statistics on how many deliverables every lead beneficiary has and so on. It is worth noticing that most of the analysis can be done with just one line of code.

In the standard EU Horizon 2020 DoW deliverables and milestones are separated. However in practice it is easier to have them together to keep track on the timing for project partners. In another notebook (https://github.com/koldunovn/Programming_for_project_managers/blob/master/EU_HORIZON202_merge_tables.ipynb, last access: 12 October 2018) we demonstrate how to merge the table of deliverables with the table of milestones. This can be used as an example for merging two tables that have partly different columns.

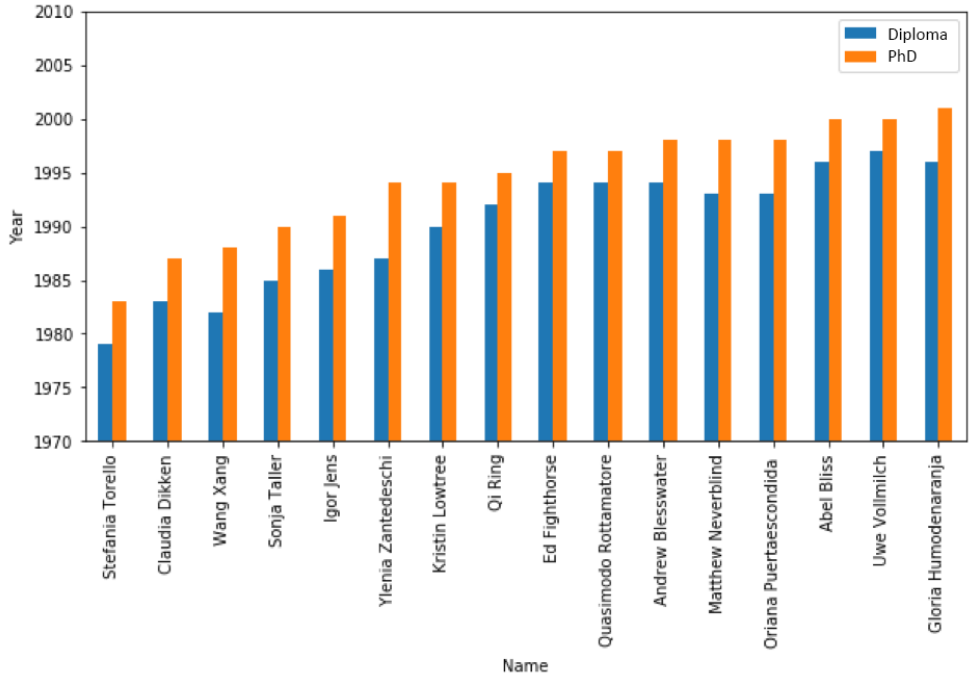

Figure 2. Bar chart of the years when each of the PIs get their Diploma and PhD sorted by the year of the PhD defence.

3.2 Extract information from national project description

For this example we use a MS Word document of a national project's proposal that is structured differently from the EU H2020 DoW used in the previous section. In particular not all relevant information is available in the table from, so some of it has to be extracted from the plain text. The detailed notebook with the code is located online at https://github.com/koldunovn/Programming_for_project_managers/blob/master/National_project.ipynb (last access: 12 October 2018)

We again begin by opening the document with the docx package and then get the information from the tables. The tables contain information on project principal investigators (PIs), but their cells are not regular and fields often contain large portions of text (e.g., "Areas of expertise and key competences", "Four most relevant publications"). We show how to reformat such tables into a more convenient format for data extraction and how to export only some parts of the table (e.g. only PI's name and affiliation) into MS Excel. Two possible ways to extract numerical information (e.g., year) from long strings (e.g., "Diploma in physics (1996), PhD in physics (2000)") are demonstrated. We also extract information on the PI's h-index and perform some simple analysis. Figure 2 shows an example of such analysis, i.e., bar chart of the years when each of the PIs get their Diploma degree and PhD degree sorted by the year of the PhD defence.

4 Discussion and conclusions

Some tasks within science project management that involve data processing (in particular data retrieval, compilation, homogenisation and synthesis) are extremely time consuming for project managers, especially when the projects involve large international teams and complex financing schemes (such as European projects). However many of these tasks are repetitive and can therefore be automatized through programming. This saves precious time that the project manager can use in more fruitful ways (e.g., for activities that require thinking and creativity).

In this paper we explored the possibility of using python as a tool for solving typical scientific project management tasks that involve data. We identified some example tasks related to data extraction and processing that can be automatized through programming and we provided a preliminary review of some of the core python libraries that can be used to solve these tasks. We also demonstrated several real-world applications using a typical European and national project's documentation as source of data.

python is a very useful and versatile language for organising and visualizing data related to the management of scientific projects, especially some libraries such as pandas and numpy. There are several programming languages that can be used for the same purpose (such as *R* or Java to name two examples), however we focused on python as it is flexible, relatively easy to learn and has become popular in several fields including geosciences. python's unique strength is the

wide coverage of different topics of its tool suite (libraries) and the possibility to transfer innovations between different communities and disciplines (Lin, 2012).

The main disadvantage of using a programming language such as python to cope with data-related project management tasks is that the automatization process requires time to be set up (in fact the time needed to write the code itself is longer than time required for manually performing each task taken individually) so it might not seem attractive initially for project managers who are always working with tight timelines. There is therefore a considerable learning effort that needs to be taken into account to start using python in an efficient way. However this investment in time for learning and code writing has a return on the long term, i.e., when a code written for one task can be utilised several times and therefore reduce the total time spent on performing similar tasks again and again. Furthermore the skills acquired by the project manager in the course of learning the programming language are beneficial also for other aspects of their job. For example it can help when applying project management processes where one have to have a workflow that consider inputs, tools and procedures, and outputs.

In conclusion, we found that python is a flexible and powerful tool that can help ease the daily work of science project managers especially for data and information-related tasks. Through our overview of tools and case studies we demonstrated python utility for science project management and we advocate for more programming training to be available for project managers to take full advantage of coding and thus work in a more time-efficient way.

Code availability. The code is available at https://github.com/koldunovn/Programming_for_project_managers (last access: 12 October 2018) (Koldunov and Cristini, 2018).

Appendix A: How to start learning python

In our own experience, we find that the best way to learn a computer language is to cover the very basics of the language syntax and then begin a small project (or set of projects) that is directly related to our work (or hobby) so progress is much faster than it would be by focusing on abstract exercises. It is also very helpful to find someone in the organisation with python knowledge that could help at the beginning of the learning process.

Below we list some of the resources that in our opinion are useful for beginners with no prior knowledge of python:

- "Learn Python 3 the Hard Way" (https://learnpythonthehardway.org/, last access: 12 October 2018) – an eBook, not free.
- Code Academy (https://www.codecademy.com/learn/learn-python, last access: 12 October 2018) – free online tutorials, require no python installation, all done in the browser.
- LearnPython.org (http://learnpython.org/en/, last access: 12 October 2018) – free online tutorials.
- Programming with Python from Software Carpentry (https://swcarpentry.github.io/python-novice-inflammation/, last access: 12 October 2018) – materials for one day introductionary course.
- Dive into Python 3 (http://www.diveintopython3.net/your-first-python-program.html, last access: 12 October 2018) – free online book.

Supplement. The supplement related to this article is available online at: https://doi.org/10.5194/adgeo-45-295-2018-supplement.

Competing interests. The authors declare that they have no conflict of interest.

Special issue statement. This article is part of the special issue "Project management in geosciences: systems and practices for high-impact research". It is a result of the EGU General Assembly 2018, Vienna, Austria, 8–13 April 2018.

Acknowledgements. We thank Lars Kaleschke and anonymous reviewer for their very helpful comments. Nikolay Koldunov is supported by project S1 (Climate Models as Metrics) of the Collaborative Research Centre TRR 181 Energy Transfer in Atmosphere and Ocean program funded by the German Research Foundation. Luisa Cristini is funded through the European Commission H2020 project APPLICATE (Grant number 727862) and the project "Advanced Earth System Modelling Capacity" supported by the Initiative and Networking Fund of the Helmholtz Association.

The article processing charges for this open-access publication were covered by the University of Bremen.

Edited by: Sylvia Walter
Reviewed by: Lars Kaleschke and one anonymous referee

References

Guo, P.: Python Is Now the Most Popular Introductory Teaching Language at Top U.S. Universities, Communications of the ACM, available at: https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext (last access: 12 October 2018), 7 July, 2014.

Jacobs, C. T., Gorman, G. J., Rees, H. E., and Craig, L. E.: Experiences with efficient methodologies for teaching computer programming to geoscientists, J. Geosci. Ed., 64, 183–198, 2016.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., and Willing, C.: Jupyter Notebooks – A publishing format for reproducible computational workflows, in: Positioning and Power in Academic Publishing: Players, Agents and Agendas, IOS Press, Amsterdam, the Netherlands, 87–90, https://doi.org/10.3233/978-1-61499-649-1-87, 2016.

Koldunov, N. V. and Cristini, L.: Programming for project managers, https://doi.org/10.5281/zenodo.1452254, 2018.

Lin, J. W. B.: Why Python is the next wave in earth sciences computing, B. Am. Meteorol. Soc., 93, 1823–1824, 2012.

McKinney, W.: Data structures for statistical computing in python, in: Proceedings of the 9th Python in Science Conference, 445, 51–56, 2010.

McKinney,W.: Pandas: a foundational Python library for data analysis and statistics, Python for High Performance and Scientific Computing (PyHPC2011), available at: http://www.dlr.de/sc/Portaldata/15/Resources/dokumente/pyhpc2011/submissions/pyhpc2011_submission_9.pdf (last access: 12 October 2018), 1–9, 2011.

Shen, H.: Interactive notebooks: Sharing the code, Nature News, 515, 151–152, 2014.