Geo-Python: An open online introduction to programming in Python for geoscientists

David Whipp, **Henrikki Tenkanen**, and **Vuokko Heikinheimo**
david.whipp@helsinki.fi
Department of Geosciences and Geography, University of Helsinki, Finland

General course information

Introduction and motivation

- Geology and geography are becoming more quantitative
- Basic programming skills are an increasingly important asset for geoscientists
- The Geo-Python course is designed to provide students with these essential skills using Python[1]

Course design and philosophy

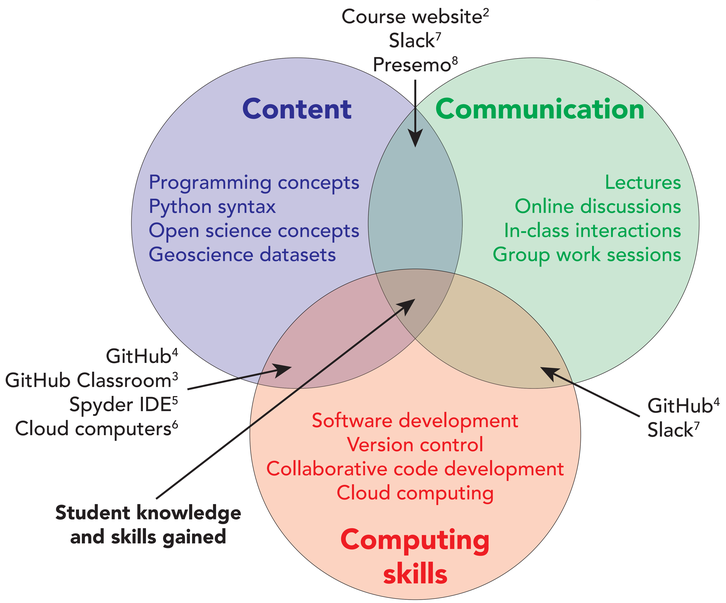


Figure 1. Overview of the Geo-Python course components.

Geoscience students often want to learn to program to solve geoscientific problems. The **blended learning** environment for the Geo-Python course (inspired by Software Carpentry workshops[9]) is designed to provide students with programming experience and essential computing skills. We also provide hands-on experience with real-world tools[1,4-7] (Fig. 1) used by professionals.

Lessons have 3-4 learning goals with exercises that allow instructors to assess student performance related to those goals (**constructive alignment**).

Experience and lessons learned

The need for familiar concepts

Geoscience students often struggle to understand fundamental programming concepts such as lists or arrays, loops, and conditional statements. Teaching using everyday experience and familiar concepts helps students learn these ideas.

Concept	Everyday example
Lists and index values	Button to push on a vending machine and the item you select (Fig. 2)
Loops	Daily morning activities (wake up, brush teeth, eat breakfast, etc.)
Conditional statements	Deciding what to wear based on the weather

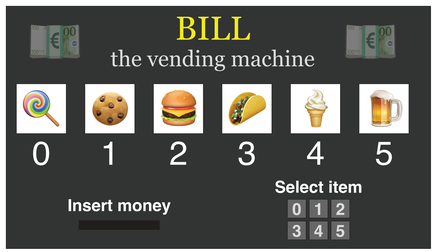


Figure 2. Bill the vending machine, used to illustrate the difference between list indices and list values.

Effective learning methods and tools

We have found some teaching and content delivery methods are more effective than others.

Cloud computers vs. personal computers
The cloud computer software is easy to manage. Students tend to prefer using their personal computers.
Winner: Personal computers

GitHub issues vs. Slack
Course-related questions can be posted in GitHub keeping everything in one place. Slack requires visiting another website, but everyone sees the questions/responses.
Winner: Slack

Basic GitHub documentation vs. Sphinx
Creating course lessons in GitHub is simple. Sphinx[10] requires more effort, but produces a more navigable course website (Fig. 3).
Winner: Sphinx

More material vs. more time
More material is tempting, but students seem to need time to learn Python fundamentals.
Winner: More time

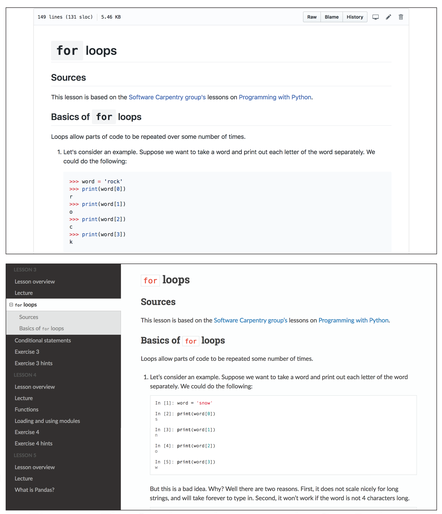


Figure 3. The `for` loops lesson on the Geo-Python course pages from 2016 (upper) and 2017 (lower).

Outcomes and future work

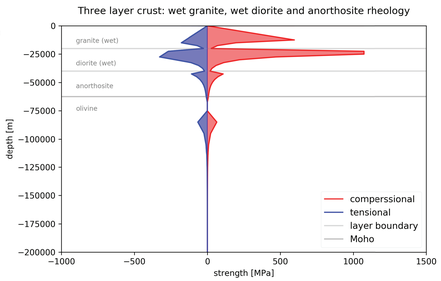
How have we fared?

The primary goal of the Geo-Python course is to teach students how to write and use simple Python programs. How well have we done?

Do students understand key concepts?
Signs point to yes, but we need more data. Students score highly on assignments that focus on key programming concepts.

Do students continue using Python?
Many do. Students are increasingly using Python to complete their assignments in other courses (80% in one recent course).

What helps their learning?
- Having an easily navigable course website
- Posting videos of course lectures online
- Providing ample time to complete course exercises




```
def brittle_shear_strength(cohesion, coeff_friction, normal_stress, fluid_P_factor):
    tau=cohesion+coeff_friction*(normal_stress-(fluid_P_factor*normal_stress))
    tau=tau/10**6
    return tau


# Plastic rock failure under compression and tension
compressional_failure_wetgranite=[]                          #wet granite
tensional_failure_wetgranite=[]
for i in range(len(depth)):
    failure_wetgranite=fls.plastic_failure(c_o,u,stress_normal_wetgranite[i],lambd)
    compressional_failure_wetgranite.append(failure_wetgranite[0])
    tensional_failure_wetgranite.append(failure_wetgranite[1])
```

Figure 4. Example plot and Python code snippet from an assignment in a course taken after completing the Geo-Python course.

Future work and course development

We have several plans to further develop the Geo-Python course.

1. Collection of detailed student survey data
We are currently designing surveys for students from the past 2 years to gain a better assess how well they understand Python and whether they have continued to use it at work or in their studies.

2. Course changes to ensure more advanced students stay engaged
Some students already have programming experience when they take the Geo-Python course. We are working to make sure the course design has methods to challenge more experienced students while not overwhelming new programmers.

3. Integration of code and documentation using JupyterLab
We are taking steps to explore ways in which code and its documentation (answers to exercise questions) can be integrated. JupyterLab[11], for example, could provide a means to teach introductory Python concepts in a Python console and later transition to Jupyter Notebooks where code and documentation could coexist.

Check out our course online
geo-python.github.io

Use/modify our course materials
github.com/Geo-Python

Watch course lecture videos
bit.ly/geo-python

References
1. Python Software Foundation, https://www.python.org
2. H. Tenkanen and D. Whipp, https://geo-python.github.io
3. GitHub Classroom, https://classroom.github.com
4. GitHub development platform, https://github.com
5. Spyder development environment, https://github.com/spyder-ide
6. cPouta cloud computing environment, https://research.csc.fi/cpouta
7. Slack communication platform, https://slack.com
8. Presemo live participation system, https://screen.io/en
9. Software carpentry, https://software-carpentry.org
10. Sphinx documentation generator, http://www.sphinx-doc.org
11. JupyterLab environment, https://github.com/jupyterlab/jupyterlab