




# Teaching introductory GIS programming to geographers using an open source Python approach

Thomas R. Etherington


**To cite this article:** Thomas R. Etherington (2016) Teaching introductory GIS programming to geographers using an open source Python approach, Journal of Geography in Higher Education, 40:1, 117-130, DOI: [10.1080/03098265.2015.1086981](https://doi.org/10.1080/03098265.2015.1086981)


**To link to this article:** <https://doi.org/10.1080/03098265.2015.1086981>

 View supplementary material 

 Published online: 19 Oct 2015.

 Submit your article to this journal 

 Article views: 1707

 View related articles 

 View Crossmark data 

 Citing articles: 6 View citing articles 

## Teaching introductory GIS programming to geographers using an open source Python approach

Thomas R. Etherington\*

*Institute for Applied Ecology New Zealand, School of Applied Sciences, Auckland University of Technology, Auckland, New Zealand*

*(Received 14 December 2014; final version received 13 July 2015)*

Computer programming is not commonly taught to geographers as a part of geographic information system (GIS) courses, but the advent of NeoGeography, big data and open GIS means that programming skills are becoming more important. To encourage the teaching of programming to geographers, this paper outlines a course based around a series of laboratories that aims to provide an introduction to programming. Methods for teaching and assessment are recommended. These laboratories use core spatial concepts that are relevant for all areas of geographic study, and use an open source Python approach that has wider logistical and pedagogical benefits.

**Keywords:** constructive alignment; core spatial concepts; peer assessment; reflective writing; self-assessment; structured and open-ended laboratories

### 1. Introduction

Computer programming is becoming an increasingly important scientific skill (Merali, 2010) as it enables scientists to: save time by automating repetitive tasks, accurately document and quickly repeat or modify analytical processes, and ask questions that could not be asked using pre-built software. Geographic information system (GIS) software is an important tool for geographers (Matthews & Herbert, 2008), and GIS programming is becoming increasingly important with the advent of NeoGeography, big data and open GIS. NeoGeography recognises that anyone can now use application programming interfaces, such as Google Maps, to create online maps that blend together geographic information from a variety of sources (Batty, Hudson-Smith, Milton, & Crooks, 2010). Big data from sources such as social media are useful for geographical research, but big data cannot be readily used with a conventional desktop GIS approach due to the volume, velocity, variety and veracity of the data (Gorman, 2013). Open GIS seeks to make GIS data, software, research and education more accessible and transparent (Sui, 2014), but implementing an open GIS approach often requires programming skills.

Currently programming is rarely taught in GIS courses, and usually only at the post-graduate level (Wikle & Fagin, 2014), as it is not considered to be as important as other GIS skills (Wikle & Fagin, 2015). This situation is both worrying, as scientific findings may become erroneous when code is not well tested or documented (Merali, 2010), and unfortunate as some simple formal training in basic programming best practices relating to design and documentation could help to avoid many issues (Wilson et al., 2014).

---

\*Email: [thomas.etherington@aut.ac.nz](mailto:thomas.etherington@aut.ac.nz)

Therefore, it is time GIS programming became a more common component of a geographer's higher education.

There are many computer languages, and some of these are more appropriate for different geographical disciplines. For example, FORTRAN has particular relevance for climatology and meteorology (Muller & Kidd, 2014), and R is often used for spatial analysis (Bivand, Pebesma, & Gómez-Rubio, 2008). For GIS programming, Python ([www.python.org](http://www.python.org)) is the most commonly used computer language. Python became one of the scripting language options with version 9 of ArcGIS (Zandbergen, 2013), and it was this event that helped Python become the prominent GIS programming language. However, Python is more than a scripting language for ArcGIS. Python is open source, free, cross-platform and can be used as the sole software for scientific computing (Oliphant, 2007). This benefits teaching for logistical and pedagogical reasons. Logistically, using Python means there are no financial or hardware obstacles to teaching, such as not being able to afford proprietary software or having to use a specific operating system. Teachers can develop a tailored, stable and adaptable course, as what is taught is not affected by changes or limitations in proprietary software. Students will always be able to use their Python programming skills, as the software will always be available to them in the future.

Given these benefits, more geography programs should be encouraged to include introductory GIS programming courses using an open source Python approach. I acknowledge that some geographers will want to learn specific ArcGIS Python programming skills. However, having learnt how to program Python in an open source context, it is easy to then program with ArcGIS – but the reverse is certainly not true. Therefore, this paper outlines a course based around a set of open source Python computer laboratories used for teaching introductory GIS programming to geographers that aims to interweave the teaching of: programming as a general skill, Python as a language and GIS as a specific application. Student feedback derived from standard course reviews provides a means of assessing the effectiveness of this approach to teaching GIS programming to geographers.

## **2. Course content**

### ***2.1. Core spatial concepts***

When designing a course it is important that what is being taught is perceived as relevant by the students. This is challenging as a class of geography students may have a varied set of geographical interests that could range from geomorphology to social networks. Therefore, to maximise the relevancy of a programming course to all geography students, the laboratories need to focus on geographic concepts that are as widely applicable as possible.

There have been a variety of attempts to identify core concepts of geographical information and thinking (Golledge, 2002; Janelle & Goodchild, 2011; Kuhn, 2012), that could provide a framework for teaching GIS programming that has a broad relevance. Kuhn's (2012) six core spatial concepts of location, event, neighbourhood, field, object and network (Figure 1a) are a succinct list of spatial concepts used as the basis for six laboratory exercises reported here that are sufficiently generic to be relevant to all geographical disciplines (Figure 1b). This approach does ignore Kuhn's (2012) information concepts of granularity, accuracy, meaning and value, but this is acceptable

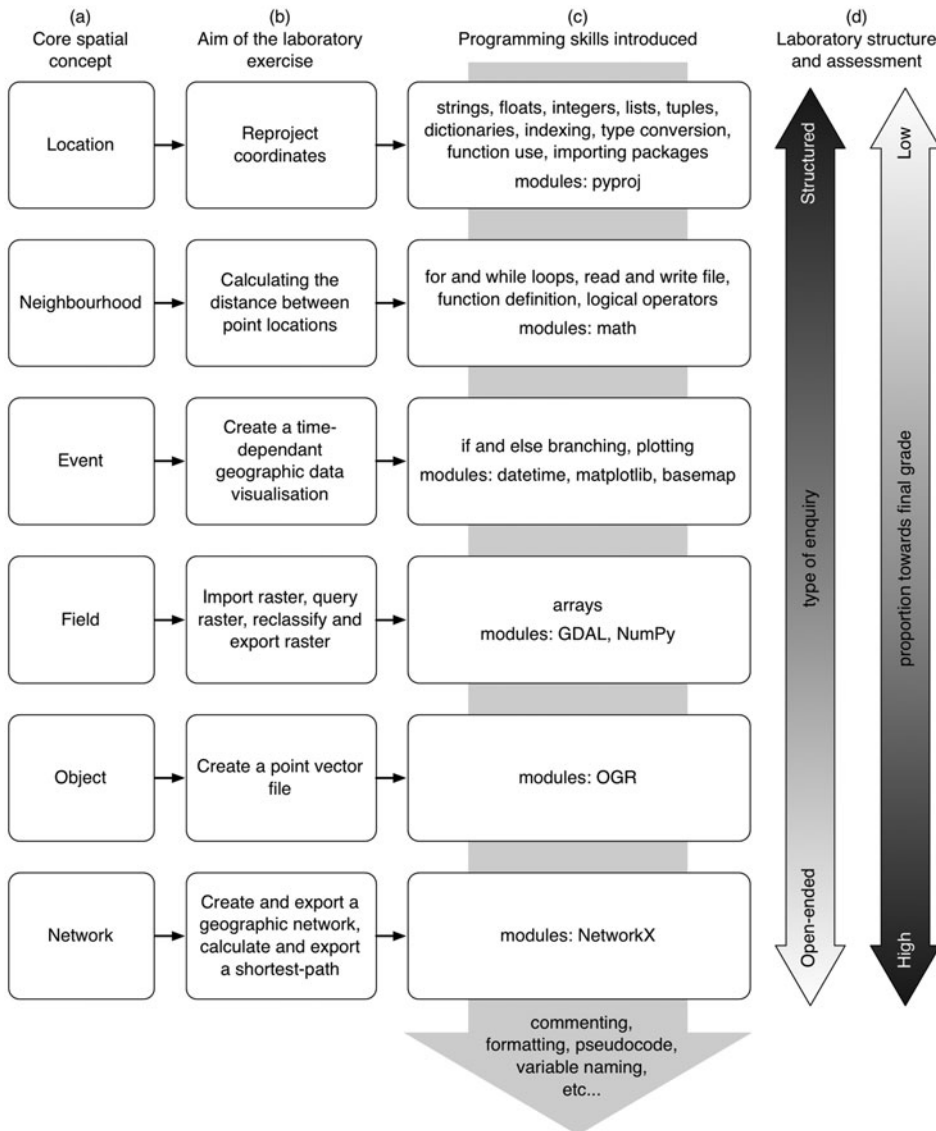


Figure 1. Six core spatial concepts that form the basis of laboratories to teach introductory GIS programming using an open source Python approach.

as an introductory programming course needs to focus on interacting with geographic data rather than seeking an understanding of geographic data.

### 2.1.1. Location

Location is fundamental to geography (Matthews & Herbert, 2008), so programmers have to work with coordinate systems that define locations. The pyproj Python package is an accessible and practical tool, as it provides a Python interface to the PROJ.4

library that is used by most GIS software for reprojecting locations (Lawhead, 2013). A first laboratory could involve a simple script that reprojects a location between coordinate systems. Whilst being a simple process, the laboratory would also introduce basic Python programming skills such as data types, containers, function use and package importing. Incorporating basic programming into the earlier exercises involving simpler spatial concepts will allow later labs to focus more time on learning to handle more complex spatial concepts (Figure 1c).

### *2.1.2. Neighbourhood*

Kuhn's (2012) neighbourhood spatial concept focuses on what is near, which emphasises the importance of calculating distances. As there is no Python function that will accept two projected coordinates as parameters and return a Euclidean distance, students can be challenged to write a Python function to do this. This exercise also demonstrates the importance of using functions for code that will be reused multiple times within a script, and allows for the introduction of recursion, logical operators and file input and output.

### *2.1.3. Event*

Spatial events are defined by changes over time (Kuhn, 2012), so GIS programmers need to be able to work with time data. The datetime Python module enables users to create time aware objects that can be compared in order to calculate differences in time (McKinney, 2013). On its own, a lab focussed on handling time may appear to lack relevance to geographers who are concerned primarily with space. Therefore, events are best introduced in conjunction with map making.

Graphicacy, or the ability to make maps to visualise geographic data, is a fundamental skill for geographers (Matthews & Herbert, 2008). Whilst the results of a program could be manually visualised in a GIS, computer programs can quickly produce more data than can be handled manually. Therefore, visualising programmatically becomes important for checking and correcting code. The matplotlib package and its associated basemap toolkit can visualise geographic data (Hunter, 2007). So the event laboratory requires students to plot a spatio-temporal set of locations based on some function of their time. This task also introduces basic Python programming such as conditional execution.

### *2.1.4. Field*

Fields that describe continuous phenomena are typically represented in a GIS using the raster data structure. The geospatial data abstraction library (GDAL) is a C++ library of functions that are accessed using the Python package (Warmerdam, 2008). GDAL is a fundamental geospatial library that provides read and write functionality for lots of raster data formats and is used by almost all geospatial software (Lawhead, 2013; Westra, 2013). GDAL also converts raster data into NumPy arrays. NumPy arrays form the basis of scientific computing in Python with the NumPy and SciPy packages (Oliphant, 2007), much of which is applicable to geographical analysis. Therefore, a laboratory based around the field spatial concept should cover reading and writing of raster data, and also the use of NumPy functions to extract summary information and modify the input data in some way.

### 2.1.5. *Object*

The spatial concept of objects relates to the vector data structure. As well as providing raster functionality, the GDAL package also contains another module called OGR that provides vector data functionality (Westra, 2013). Dealing with vector data programmatically can be quite challenging, so a laboratory introducing OGR will likely require a reasonably simple aim. Working through an input text file of point locations and associated attribute information in order to create a point vector file would demonstrate the principle of working with objects whilst reinforcing basic programming skills such as recursion that will have already been covered during earlier laboratories.

### 2.1.6. *Network*

Networks are a data structure that in a geographical context represents connectivity between locations (Kuhn, 2012). For teaching networks with Python the NetworkX package (Hagberg, Schult, & Swart, 2008) is well supported and documented, and has inbuilt functions to read and write shapefiles. As the final laboratory in the course, most of the basic Python requirements will have already been introduced. Therefore, with a view to demonstrating to students what the next step in programming would be, this is an opportunity to do something analytical. Such an analysis could include finding the shortest path between two locations in a network, and then exporting that path as a vector line file.

## 2.2. *General programming skills*

An introductory GIS programming course also needs to teach general programming skills. Simple practices such as appropriate commenting and variable naming within a script, or modularising code, are important practices to teach that will hopefully ensure that the students are capable of writing better and more robust code (Wilson et al., 2014). However, to try and teach such programming skills on their own may not appear too relevant to geography students. Therefore, the importance of these general aspects of programming are both introduced and reinforced throughout all of the exercises (Figure 1c), and included as part of the assessment criteria in order to provide feedback during the course.

## 3. *Teaching and assessment*

### 3.1. *Significant learning and constructive alignment*

Tertiary education is placing more emphasis on students developing graduate attributes that are more general skills and attitudes that students should develop along with discipline-specific knowledge and skills (Chalmers & Partridge, 2013). The taxonomy of significant learning (Figure 2a) developed by Fink (2003) provides a framework within which to consider graduate attributes. It recognises that the foundational knowledge that has traditionally formed the focus of university teaching is only one of six different types of learning. Therefore, this course has been designed to connect to each of Fink's (2003) six types of learning to achieve a more significant learning experience (Figure 2).

The course has also been designed using the principle of constructive alignment (Biggs, 2014), by breaking the course design into a series of discrete stages: (i) define

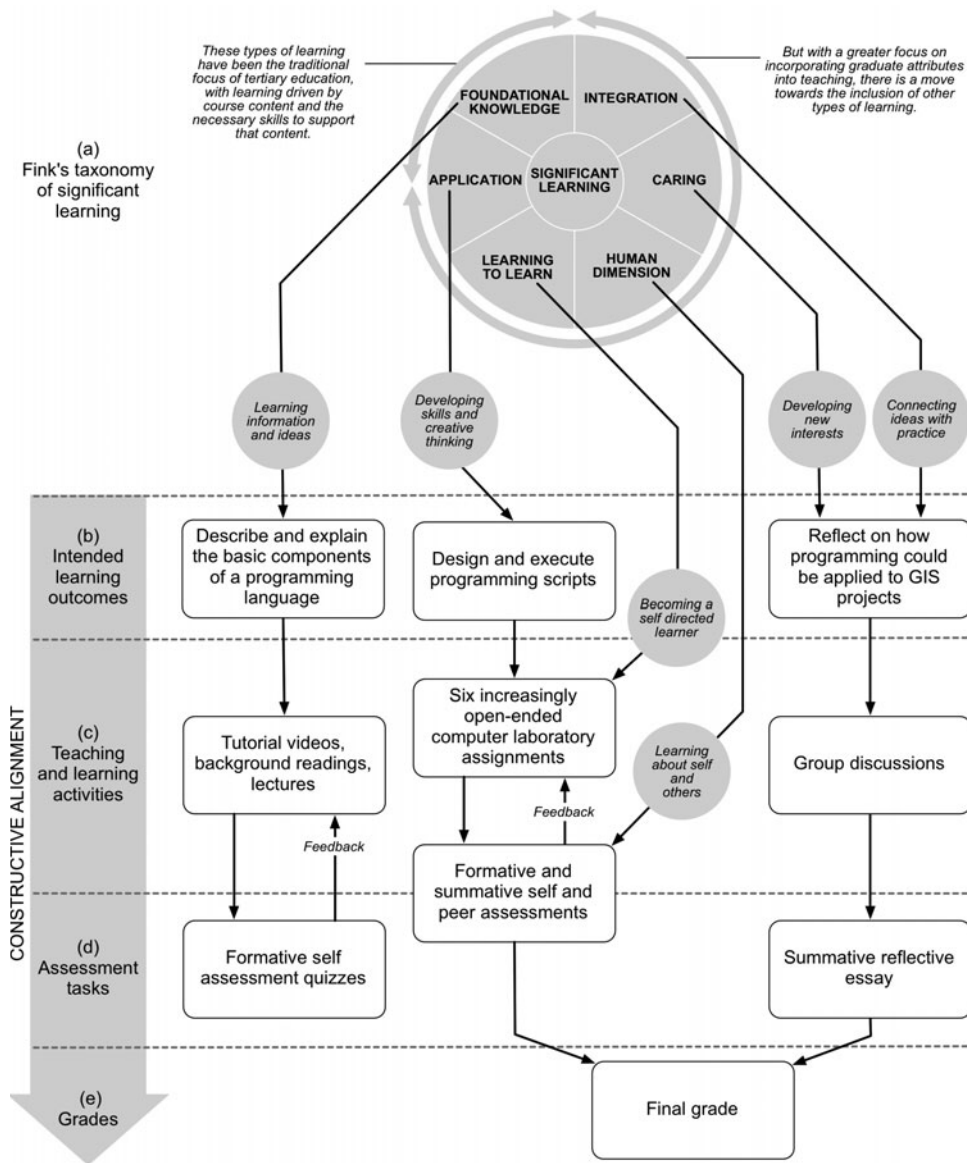


Figure 2. Using the pedagogical principles of significant learning and constructive alignment to design an introductory GIS programming course.

the intended learning outcomes, (ii) identify teaching and learning activities to support the intended learning outcomes, (iii) design assessment tasks that assess the teaching and learning activities and hence the intended learning outcomes and (iv) use the outcome of the assessment tasks to create final grades (Figure 2b–e). Constructive alignment is achieved by ensuring that the intended learning outcomes, teaching and learning activities, and the assessment tasks are aligned and that the learner is able to construct their own learning (Biggs, 2014). So having identified the intended learning outcomes



(Figure 2b) the learning and teaching activities need to be constructive, and the assessment tasks need to be aligned.

### **3.2. Teaching and learning activities – the construction**

#### *3.2.1. Computer laboratories*

The core teaching and learning activities (Figure 2c) are the six computer programming laboratories. As this is an introductory GIS programming course, the first laboratories will need to be highly structured in order to provide the necessary scaffolding for student learning. However, Unwin (1980) argued for open-ended laboratories, as highly structured laboratories do not teach research skills and result in students producing nearly identical work. To reconcile the need for structured laboratories to introduce programming and open-ended laboratories to better engage with students, laboratories for an introductory course could range from structured laboratories to more open-ended laboratories as the course develops (Figure 1d). This will enable students to build confidence by completing the simpler more structured laboratories, before then challenging them with more open-ended laboratories. The proportion of each laboratory that makes up a final grade would then also range so that the later more open-ended laboratories carry more weight (Figure 1d).

A simple way to make a laboratory open-ended would be to provide a simple example to create some basis of understanding, and then ask the students to apply this example to data of their own choice (Unwin, 1980). For example, a fully working Python script could be given to the students, and their task could simply be to modify the script in order to apply it to their own data. The main Python packages all come with examples, and there is a vibrant Python community that posts examples online. Therefore, this process of adapting an example to individual needs actually mimics the typical process that is followed when learning programming individually. Some other ways to make the laboratories more open-ended could include: providing code without comments and asking for comments explaining the following line of code to be inserted, providing code with poor variable names that need replacing, including deliberate mistakes that need to be corrected, putting lines of code in the wrong order and excluding lines of code but including a comment to describe what the missing line of code needs to do.

#### *3.2.2. Supporting activities*

Whilst the computer laboratories are the main teaching and learning activity, students will benefit from additional activities that provide scaffolding to help them construct their learning (Figure 2c). Therefore, before new programming concepts and language components are used in a laboratory, they should be introduced using tutorial videos, readings and lectures. Also, to reinforce the construction of learning, group discussions could be used to enable students to compare and contrast their experiences of applying programming methods to their own particular context.

### **3.3. Assessment tasks – the alignment**

#### *3.3.1. Self-assessment quizzes*

Computer programming does require a certain level of exactness, with specific tasks producing specific results. Therefore, being able to program efficiently does require the



memorisation of the basic components of a programming language. Multiple choice questions (Figure 2d) are an efficient and convenient method that students can use to formatively assess their level of comprehension of the basic components of computer programming (Roberts, 2006). If student involvement with multiple choice questions becomes an active process, with students creating as well as answering questions, a multiple choice approach can even aid deeper learning of computer programming (Denny, Hamer, Luxton-Reilly, & Purchase, 2008).

### *3.3.2. Self and peer laboratory assessments*

Computer programming is not just an exact science, it is also an art, as there is never one best programming solution to a problem (Knuth, 1974). This makes the assessment of computer programming difficult, especially for the open-ended enquiries advocated in this course, as there will be no fixed solution against which to assess. One approach to this challenge is to use it as an opportunity to adopt a sustainable assessment approach that views assessments as a learning tool (Boud, 2000).

Self and peer assessment lend themselves well to assessing computer programming laboratories (Figure 2d) as they maximise a student's exposure to different programming approaches, and help students develop critical evaluation skills. Research has shown that peer and tutor feedback on computer programming assignments does not differ much (Hamer, Purchase, Luxton-Reilly, & Denny, 2015; Reily, Finnerty, & Terveen, 2009), and that peer assessment actually improves student learning (Reily et al., 2009). With six laboratories to assess, the feedback from self and peer assessments can be applied to improve the quality of subsequent laboratories (Boud, 2000). Whilst peer assessment is a useful learning activity, successful implementation needs to focus on emphasising the relevance to learning, rather than on the production of grades (van Zundert, Sluijsmans, & van Merriënboer, 2010). Approaches such as ensuring anonymity in the assessment process, using a five-point Likert scale for rating assessment criteria, and providing tutor support during assessment process can help to create effective peer assessment of computer programming (Sitthiworachart & Joy, 2004). Using peer assessment will increase organisational workload, but there are online tools that can be used to help automate the process for large classes (Luxton-Reilly, 2009).

### *3.3.3. Reflective writing*

Whilst learning the technical skill of GIS programming is important, without the ability to identify how GIS programming can be integrated into their geographical work, students may miss opportunities to make use of their new skill. Even if students decide that programming is not for them, the ability to identify when they should be working with someone else who can program is a critical ability to develop. Reflective writing is one approach that can be used to increase the depth and relevance of geographical learning, by asking students to reflect on how their learning relates to their past experiences and future aspirations (Harrison, Short, & Roberts, 2003).

Although reflective writing has traditionally been used in the humanities, it has also been used to enhance learning of computer programming (George, 2002). So students could be encouraged to make notes on those occasions when something they see, hear, read or do connects a programming principle or practice to their past GIS experiences or future GIS aspirations. Students will then be able to draw on these connections when writing their assignment (Figure 2d). Reflective writing can be a challenge for those

who are unfamiliar with it, so this learning activity will require instructor support. A safe and supportive environment should be created, a clear and concise rubric should be provided, and students should be allowed the freedom to create their own context (Fernsten & Fernsten, 2005).

## 4. Implementation

### 4.1. Course delivery

The course delivery reported here aims to maximise time for hands-on programming help. Therefore, rather than spend timetabled class time lecturing about new ideas and concepts, introductory materials are provided ahead of time so that students can come to a weekly timetabled three-hour workshop having already been introduced to the new material. Whilst the focus of this paper is teaching Python GIS programming to postgraduate geographers, for whom this approach is effective, the course may require a different style of delivery if it was to be taught to undergraduates, who may lack the same level of commitment, engagement or ability, and who may benefit from more direct instruction.

Having six laboratories based around the core spatial concepts (Figure 1) results in plenty of time to support each laboratory within a single 12-week semester. The initial highly structured laboratories could be taught within a week, and the more open laboratories could run over two or more weeks. This would enable the first workshop for each laboratory to reinforce the introduction of new material with a short lecture, and then subsequent workshops could be used for group discussions, or to resolve issues that the students have discovered working on their assignment during the intervening week.

### 4.2. Feedback and reflections

This approach to teaching GIS to geographers has been developed from a discrete course at The University of Auckland, and a course that forms part of the Masters in Geographic Information Science program run between Auckland University of Technology, Victoria University of Wellington and University of Canterbury. Feedback from students at all four universities was acquired from standardised university course reviews that asked the students general questions about positive and negative aspects of course design, teaching activities and assessment tasks. The voluntary and anonymous nature of the feedback precludes any contextualisation of the comments, but the feedback was comprehensive with participation from 29 of 35 students. Students are quick to recognise the potential that Python GIS programming gives them, which I think helps to justify the argument that more geographers should learn to program:

- The power of Python is crazy and this paper has shown me that.
- So interested to learn how to harness the power and possibilities of its [programming] use within my area of interest.
- What I have learned makes me want to learn more programming!
- I now want to learn more programming!

Structuring the course around computer laboratories focused on the six core spatial concepts (Figure 1a) has been well received. This approach has relevance to geography

students, provides a supportive time frame for teaching within a single semester and allows programming skills to be reapplied in different contexts to strengthen student learning and confidence. Students have commented that:

- Lab sessions are the most constructive way to engage with the material due to the course being about programming.
- Favourite thing about the course was how well it was practically applied to GIS.
- The programming tasks necessitated use from other branches of the subject [core spatial concepts] and this helped me see how these could be expanded are used in different ways.
- Each task [laboratory] gave me a specific step to focus on and reused objectives from previous tasks [laboratories] so reinforcing their use.
- The consistent timing of fortnightly assignments [laboratories] helped me to be able to digest concepts.
- Giving two weeks for each lab was great, as it allowed me to explore the subject matter well.

A careful balance needs to be maintained between making the computer laboratories too structured or too open. When the laboratories are too open some students can be overwhelmed by the choice of possible applications. Therefore a few geographical applications are suggested to the class to try and give some direction for those students who need it. The more enthusiastic students may also need to be reminded about the specific requirements of the laboratory to prevent them from investing too much of their time pursuing to ambitious an application. When the laboratories are too structured some students, especially those who have some programming experience, do not find the tasks very engaging. But even students who have a lot of programming experience have generally found that the openness of the later laboratories allows them to pursue areas of GIS programming that are new to them in order that they can construct their own learning. So whilst it is a challenge to maintain a balance between structured and open laboratories, students have commented that they appreciate having open-ended laboratories so that they can contextualise the GIS programming skills being taught to a geographical application that is relevant to them:

- The lab assignments were good in that you could use your own data.
- I found the need to source data and how to use it within other areas of my course kept the subject alive and pointed out its immediate usefulness to my interest.
- The fact that to do the tasks [laboratories] I needed to explore other areas of my course, which I may not have looked at otherwise, and also therefore see the possibilities of its [programming] application.
- I particularly liked that he [the lecturer] did not promote “one way” ... but allowed for different approaches for different people and applications.

Regarding the assessment tasks, students consider peer assessment to be a valuable learning tool. Students have commented that the opportunity to see a peer's work allows them to appreciate different approaches to the same programming task and actually helps them learn new programming skills:

- ... peer assessment feedback was very effective. Viewing how others created their scripts was especially useful for wider learning.

- Great to use peer assessment as a tool to further demonstrate different methods and approaches to the tasks.
- As we were asked to review others work, I learned a lot and, he [the lecturer] also gave us good feedback.
- Peer assessments were a great learning tool however some direct feedback from the lecturer would provide more expert commentary and suggestions for improvement.
- Teaching was quite inspirational, with use of inclusive peer assessments to help introduce new ideas and possibilities to me ...

There proved to be a remarkable consistency between the self, peer and teacher assessments of students' programming, and self and peer assessments were used in generating final course grades. Students do still appreciate direct feedback from the instructor, so this should be provided when it is needed. The primary challenge with using peer assessment is from students who felt concerned that they or their peers might assess inaccurately. However, by providing a clear set of assessment criteria, and by maintaining a discourse with the students during the peer assessment process, misconceptions can be clarified and reassurance provided as needed. The reflective writing was mostly appreciated, but as might be expected given its rarity in the sciences, some students felt that this was perhaps not as useful as programming activities. This reinforces that reflective writing does require instructor support to ensure students appreciate the purpose and value of the task:

- Reflective writing is a good idea ...
- ... the self-reflection exercise as a means of tracking my learning whilst challenging has been interesting too.
- I'm not yet convinced by the reflective writing assessment, because practical programming skills are what I most wanted to get out of taking this course.

This student feedback highlights some key themes about their experience of the course. Framing the course around the six core spatial concepts did allow geography students to perceive the relevance of programming. Students also particularly enjoyed the open-ended laboratories that require individual creativity and the development of research skills (Unwin, 1980). These kind of open-ended exercises are a kind of problem-based learning, which has been used as a basis for teaching GIS at both a tertiary undergraduate (Drennon, 2005; Read, 2010) and even secondary level (Kinniburgh, 2010). Therefore, it should be quite feasible to adapt this approach to teaching GIS programming to geographers below the postgraduate level for which this course was developed – though doing so will probably require that the type of enquiry that is used for teaching becomes more structured than open-ended (Figure 1d). Some students were not convinced about the value of peer assessment and reflective writing as assessment tasks (Figure 2d). Whilst pedagogical benefits of peer assessment and reflective writing for learning computer programming have been clearly shown (George, 2002; Reily et al., 2009), some student's perceptions of these learning activities does highlight that a teacher needs to provide continual support to students who are not familiar with these processes.

#### **4.3. Python distributions and integrated development environments**

Perhaps the major challenge to adopting an open source approach to teaching Python will be getting a properly working installation of Python and associated geospatial packages

set-up. Python version 2.7 is recommended, as this is the version currently used by ArcGIS (Zandbergen, 2013) and should help ensure that the Python programming skills are as transferable as possible to ArcGIS. Setting up all the necessary geospatial Python packages can be a rather unintuitive task, especially if you are not overly familiar with installing open source software and require specific package versions. Using a scientific distribution of Python such as Anaconda (<https://store.continuum.io/cshop/anaconda/>) or Canopy (<https://www.enthought.com/products/canopy/>) is recommended, as they are both cross platform, free to download and come with all the Python packages most commonly required for scientific computing. They also both install a separate localised version of Python that eliminates problems associated with sharing Python between other software such as ArcGIS. The Anaconda distribution also comes with the Spyder integrated development environment (IDE). Spyder is an ideal IDE for writing and teaching Python, as it will identify and highlight simple errors and mistakes that would prevent code from working, and so actually helps to aid students with writing code by freeing them to worry more about the purpose and structure of the code they are writing.

## 5. Conclusion

Geographers have much to gain from learning GIS programming, and the student feedback presented here suggests greater emphasis on this aspect of a geographer's tertiary education is to be encouraged. To achieve this, an introductory GIS programming course needs to focus on emphasising the importance of programming and building confidence in the ability to program. This can be done using structured exercises of varying openness to equip students with a basic general programming vocabulary upon which they can build more advanced skills beyond the course. By designing the course around the core spatial concepts, this approach emphasises the geographical application to try to ensure relevance to all geography students. When programming there are usually multiple ways to go about achieving the same result, so there is little point in trying to identify "the" correct way of teaching GIS programming, as this will vary depending on an instructors personal experience. But in the hope of encouraging others, the laboratory teaching materials developed and reported here are available as supplementary material.

## Acknowledgements

Thanks to the students of: GEOG779 at The University of Auckland 2013; and GISC405 at Auckland University of Technology, Victoria University of Wellington and University of Canterbury 2015. Your successes, failures, reflections and feedback have been critical to the development of my approach to teaching GIS programming to geographers. I would also like to thank the reviewers and editor whose helpful suggestions significantly improved this paper.

## Disclosure statement

No potential conflict of interest was reported by the author.

## Supplemental data

Supplemental data for this article can be accessed at <http://dx.doi.org/10.1080/03098265.2015.1086981>.

## References

- Batty, M., Hudson-Smith, A., Milton, R., & Crooks, A. (2010). Map mashups, Web 2.0 and the GIS revolution. *Annals of GIS*, 16, 1–13.
- Biggs, J. (2014). Constructive alignment in university teaching. *HERDSA Review of Higher Education*, 1, 5–22.
- Bivand, R. S., Pebesma, E. J., & Gómez-Rubio, V. (2008). *Applied spatial data analysis with R*. New York, NY: Springer.
- Boud, D. (2000). Sustainable assessment: Rethinking assessment for the learning society. *Studies in Continuing Education*, 22, 151–167.
- Chalmers, D., & Partridge, L. K. (2013). Teaching graduate attributes and academic skills. In L. Hunt & D. Chalmers (Eds.), *University teaching in focus: A learning centered approach* (pp. 56–71). New York, NY: Routledge.
- Denny, P., Hamer, J., Luxton-Reilly, A., & Purchase, H. (2008). *PeerWise: Students sharing their multiple choice questions*. Proceedings of the Fourth international Workshop on Computing Education Research, Sydney, Australia.
- Drennon, C. (2005). Teaching geographic information systems in a problem-based learning environment. *Journal of Geography in Higher Education*, 29, 385–402.
- Fernsten, L., & Fernsten, J. (2005). Portfolio assessment and reflection: Enhancing learning through effective practice. *Reflective Practice*, 6, 303–309.
- Fink, L. D. (2003). *Creating significant learning experiences: An integrated approach to designing college courses*. San Francisco, CA: Jossey-Bass.
- George, S. E. (2002). Learning and the reflective journal in computer science. *Australian Computer Science Communications*, 24, 77–86.
- Golledge, R. G. (2002). The nature of geographic knowledge. *Annals of the Association of American Geographers*, 92, 1–14.
- Gorman, S. P. (2013). The danger of a big data episteme and the need to evolve geographic information systems. *Dialogues in Human Geography*, 3, 285–291.
- Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th Annual Python in Science Conference* (pp. 11–16). Pasadena, CA: SciPy.
- Hamer, J., Purchase, H., Luxton-Reilly, A., & Denny, P. (2015). A comparison of peer and tutor feedback. *Assessment & Evaluation in Higher Education*, 40, 151–164.
- Harrison, M., Short, C., & Roberts, C. (2003). Reflecting on reflective learning: The case of geography, earth and environmental sciences. *Journal of Geography in Higher Education*, 27, 133–152.
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9, 90–95.
- Janelle, D. G., & Goodchild, M. F. (2011). Concepts, principles, tools, and challenges in spatially integrated social science. In T. L. Nyerges, H. Couclelis, & R. B. McMaster (Eds.), *The SAGE handbook of GIS and society* (pp. 27–45). Thousand Oaks, CA: Sage.
- Kinniburgh, J. (2010). A constructivist approach to using GIS in the New Zealand classroom. *New Zealand Geographer*, 66, 74–84.
- Knuth, D. E. (1974). Computer programming as an art. *Communications of the ACM*, 17, 667–673.
- Kuhn, W. (2012). Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science*, 26, 2267–2276.
- Lawhead, J. (2013). *Learning geospatial analysis with Python*. Birmingham: Packt Publishing.
- Luxton-Reilly, A. (2009). A systematic review of tools that support peer assessment. *Computer Science Education*, 19, 209–232.
- Matthews, J. A., & Herbert, D. T. (2008). *Geography*. Oxford: Oxford University Press.
- McKinney, W. (2013). *Python for data analysis*. Sebastopol, CA: O'Reilly.
- Merali, Z. (2010). Error ... why scientific programming does not compute. *Nature*, 467, 775–777.
- Muller, C. L., & Kidd, C. (2014). Debugging geographers: Teaching programming to non-computer scientists. *Journal of Geography in Higher Education*, 38, 175–192.
- Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9, 10–20.

- Read, J. M. (2010). Teaching introductory geographic information systems through problem-based learning and public scholarship. *Journal of Geography in Higher Education*, 34, 379–399.
- Reily, K., Finnerty, P. L., & Terveen, T. (2009). Two peers are better than one: Aggregating peer reviews for computing assignments is surprisingly accurate. *Proceedings of the ACM 2009 International Conference on Supporting Group Work* (pp. 115–124). New York, NY: ACM.
- Roberts, T. S. (2006). *The use of multiple choice tests for formative and summative assessment*. Proceedings of the 8th Australasian Conference on Computing Education – Volume 52, Hobart, Australia.
- Sitthiworachart, J., & Joy, M. (2004). Effective peer assessment for learning computer programming. *ACM SIGCSE Bulletin*, 36, 122–126.
- Sui, D. (2014). Opportunities and impediments for open GIS. *Transactions in GIS*, 18, 1–24.
- Unwin, D. (1980). Make your practicals open ended. *Journal of Geography in Higher Education*, 4, 39–42.
- van Zundert, M., Sluijsmans, D., & van Merriënboer, J. (2010). Effective peer assessment processes: Research findings and future directions. *Learning and Instruction*, 20, 270–279.
- Warmerdam, F. (2008). The geospatial data abstraction library. In G. B. Hall & M. G. Leahy (Eds.), *Open source approaches in spatial data handling* (pp. 87–104). Berlin: Springer.
- Westra, E. (2013). *Python geospatial development*. Birmingham: Packt Publishing.
- Wikle, T. A., & Fagin, T. D. (2014). GIS course planning: A comparison of syllabi at US college and universities. *Transactions in GIS*, 18, 574–585.
- Wikle, T. A., & Fagin, T. D. (2015). Hard and soft skills in preparing GIS professionals: Comparing perceptions of employers and educators. *Transactions in GIS*. doi:[10.1111/tgis.12126](https://doi.org/10.1111/tgis.12126)
- Wilson, G., Aruliah, D. A., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., ... Plumbley, M. D. (2014). Best practices for scientific computing. *PLoS Biology*, 12, e1001745.
- Zandbergen, P. A. (2013). *Python scripting for ArcGIS*. Redlands, CA: Esri Press.