LEVERAGING PYTHON IN INFORMATION SYSTEMS COURSES

Ray Kresman
Department of Computer Science, Bowling Green State University
Bowling Green, OH 43403, USA

ABSTRACT

Spreadsheets are ubiquitous and tools like Excel and Minitab have been the workhorse of IS courses for decades. While these tools continue to excel at what they do, IS students need to be cognizant of the evolving skillsets needed in today's workforce, specifically the business skillsets needed to handle data that varies in size (big and small), format (potentially different from tabular structures such as spreadsheets), and medium (web, social media, etc.). We propose the use of certain educational pedagogies to help prepare IS students for this transition and argue the need for introducing elementary Python concepts to undergraduate students in IS. Students leverage what they already know, and by connecting to intuitive coding concepts in Python, can stay engaged, handle repeated tasks and simple workflow automation, and be prepared for the emerging needs of the information industry.

KEYWORDS

IS, Educational Pedagogy, Spreadsheets, Python

1. INTRODUCTION

Spreadsheets are ubiquitous, and Excel has been the workhorse in information systems courses for decades. Excel with VBA supports native integration of MS applications, for example: Excel with Power BI provides analytics and visualization capabilities; and one can spawn an application (such as email) in response to Excel cell values meeting a threshold (Maabo, 2020). Integration with external APIs, however, is difficult and VBA syntax is not all that intuitive. Google Sheets is another tool, but it needs an internet connection.

Many IS programs we reviewed include, among other items, business-facing and project management courses to design real-world systems (that use computing tools such as Excel, VBA, and PowerBI), and a general education computing course that is at best loosely coupled with the major. Recognizing that introductory Python is becoming a valuable skillset for students who graduate in IS/business (Khudov, 2020; Nye, 2022), we propose a constructivist approach using certain educational pedagogies to better integrate this computing course; students use their previous knowledge to build on their foundation, leverage what they already know, and by connecting to intuitive coding concepts in Python, stay engaged, handle repeated tasks and simple workflow automation, and be prepared for the emerging needs of the information industry.

1.1 Motivation and Business Case

Recent research (Coffman, et al, 2023) based on a randomized comparative study of students' perception of visual vs textual language (python) in an introductory general education computing course open to all majors shows that students overwhelmingly perceive Python to be more valuable. Whether it is data collection from multiple sources or a computation across time periods or market segments, business users are confronted with repeated tasks and Python provides mechanisms for automating such workflows with a robust, reusable pipeline. The web, social media and other digital data repositories house enormous amount of data that far exceeds the capability of traditional spreadsheet processing programs such as Excel. While serious analysis of such datasets is left to experts, entry level employees with a college degree in IS or business may still be expected to (or at least be cognizant and willing to learn to) do simple collection and processing of large datasets, and Python provides such capabilities. Python can help complement their Excel skillsets, for

example: a) use familiar GUI such as Excel as the front-end, and use back-end 3rd party Python tools to process the data; b) use openpyxl library to extract data from a database with Excel in the pipeline to do pre or post processing in Python; c) use native Pandas library to handle data (read, repair holes, cleanse duplicates, etc.) in a variety of formats including spreadsheets – Python can do this efficiently no matter what the data size is, a task that is particularly challenging for spreadsheet programs (Bosacki, n.d).

Students who graduate in IS with exposure to Python are in high demand and our proposal for better integration of the computing course into their major provides opportunities for reinforcement and helps them leverage what they already know. Even if these graduates don't write any code on the job, programming skillsets can help improve their participation in technical conversations (April, et. al, 2018) to gain credibility with their technical team members and to stay current with digital trends and technology developments. Additionally, the divide between IS, IT, and CS is narrowing, and college graduates are likely to encounter multidisciplinary teams as they enter the workforce; such teams help remove institutionalized silos and build team synergy, and the ability to work together with peers who have complementary skills and expertise is a critical asset for the future workforce.

Python is open source with a significantly large and continuously expanding ecosystem. Spreadsheet copy and paste operations can be tricky and formulas add an additional level of complexity, and intermediate results may be suppressed in such formulae though they can be more revealing to business analysts; consider using discount factor—information system students and even average consumers know inflation erodes value of money—to figure out whether a projected investment is profitable, i.e. net present value (NPV) is positive. As compared to XNPV of Excel, a benefit of coding (in Python, for example) is that can also see the present values of intermediate cashflows as opposed to only the final NPV (Moffatt, 2019; CFI Team, 2022).

2. EDUCATIONAL PEDAGOGY

2.1 Scaffolding

Our task as teachers is to help students build on their experiences and knowledge as they learn new skills; the help that we give can be relaxed as they progress in their learning. Scaffolding, a teaching pedagogy, creates the bridge between learning gaps and helps construct new knowledge (Harris, 2016). By breaking up a learning experience into manageable parts, we can give our students the assistance they need to learn each part. As instructors we must take it as a challenge to impress upon our IS undergraduates that coding is a creative endeavor that can produce a creatively generated work with relevance in the business world.

The power of jigsaw puzzles cannot be overstated. Students are already familiar with jigsaw puzzles and one approach to is to give them the code for a Python program but jumbled up, and their task is to put them together in the right sequence. Parsons problems (Borchert and Valia, 2022) are coding puzzles in which students drag-and drop code snippets to put them in the right order. The difficulty level can be varied; for example, one can do even simpler activities such as using a visual programming languages (Snap!, for example) where the blocks are visually rearranged to end up with the correct sequencing of the blocks. Conversely, one can increase the difficulty level by adding distractors. Research has shown that Parsons problems can help scaffold students' transition from learning in Snap! to Python (Shah, 2020).

2.2 Adaptive Programming

Sigmund Freud (Freud, 2012) laid the foundation for psychoanalysis and psycho-education when he noted that what was unconscious had to be made conscious to affect a change in symptomatic behavior. Adaptive programming attempts to let something be "known again," so it can be "known differently." This principle applies to any field including IS: reviewing past knowledge can lead to new, and beneficially updated understanding (Munshi, et al, 2022; Seltzer, 2019). This pedagogy can be applied to entry level classes for students in IS. Just as jigsaw puzzles help make the transition to Python, so can the spreadsheet itself. Students are exposed to spreadsheets as early as middle school so one approach is to start with doing the things in Excel and supplement it with a coding language such as Python (Shah, 2020); start with very simple examples (such an open an Excel file and print it) and progressively ramp up. Each such activity in Excel

provides a bridge to learning the comparable way of doing the tasks in Python. Ramped up excel activity may also be as simple as figuring out the standard deviation of student scores or other interesting statistic.

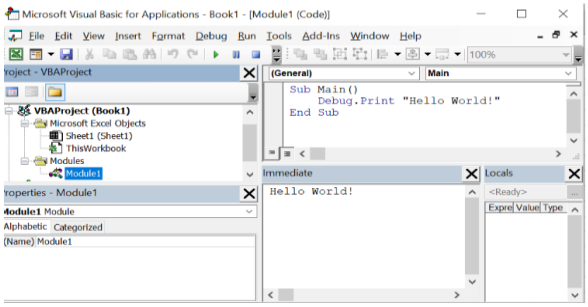
One student may not be at the same level as another, and the instructor may like to adapt to their varied abilities. This can help bring a spark back into the classroom and help students discover 'spreadsheets again,' so spreadsheets can be 'known differently.' As instructors we may be pleasantly surprised at how tech savvy today's information systems students are and that they may be eager and ready to engage in this adaptive programming pedagogy; together, we help them rediscover through Python what they already know.

Both pedagogies have the same end goal: by intentionally associating coding into the (spreadsheet) context they are already familiar with, we can ensure that the new content does not negatively affect their extraneous cognitive load. The illustrative examples in the next section attempt to employ these pedagogies.

3. ILLUSTRATIVE EXAMPLES

Jupyter Notebook, an interactive web tool available for all platforms including Windows, provides a convenient visual environment to explore Python. IS students already know how to write VBA code to output "Hello World" (Figure 1(a)). Corresponding one-line Python app in the Jupyter environment, shown in Figure 1 (b), is rather intuitive. Suppose the "Hello World" string is instead to be written to cell [1,2] of an Excel file, Test.xlsx, using VBA. As seen in the VBA (column 1) of Figure 2 (adapted from Tom (AnalystCave), 2022), one must open the spreadsheet and invoke a few functions to write the string to that cell. Corresponding Python code (column 2) is much simpler and it is instructive to compare the two.

(a) VBA



(b) Python

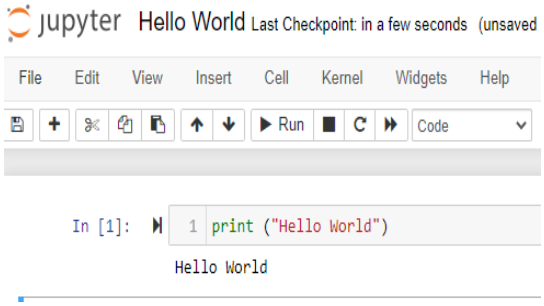


Figure 1. Output 'Hello World'

Sub Main()
  Dim app As Excel.Application
  Dim wb As Excel.Workbook
  Dim ws As Excel.Worksheet
  Dim fileName As String
  fileName = "C:\Test.xlsx"
  Set app = New Excel.Application
  Set wb = app.Workbooks.Add
  Set ws = wb.Sheets(1)
  ws.Cells (1, 2) = "Hello World"
  wb.SaveAs filename
End Sub
	import openpyxl as op
	wb = op.Workbook()
	ws = wb.active
	ws.cell (1,2).value = "Hello World"
	wb.save (r"C:\... \Test.xlsx") //replace cell [1,2] of Test

Figure 2. Write 'Hello World' to cell [1, 2] of an Excel file

As another example, let us recap some of the summary functions in spreadsheets that IS students experiment with like sum, mean, etc. One can compute these and more in Python; in fact, Python has a rich library, pandas, to manipulate tabular structure very efficiently and with minimal coding. An example of such

a computation is shown in Figure 3. Line 2 of Figure 3 Python code reads an excel file, 'sales data.xlsx,' while the next line prints it (see the output in column 2 with five rows with 'Order No,' 'Date,' Qty,' and 'Amount.'). Line 5 prints various stats (note the use of 'describe' function) on the 'Qty' and 'Amount' fields, such as count, mean, standard deviation. A spreadsheet program on the other hand would need many clicks through the formula icon to generate similar statistics. Another computation quite relevant to MIS/IS students is the SQL construct to do aggregates based on groups. Excel users may be familiar with its pivot operation. Line 6 of Figure 3 Python code does the (sum) aggregation of 'Amount' by date and the output is shown to the far right of Figure 3. Of course, aggregation can involve multiple categories and the like. Again, what appeals to students is the ability to do some useful computation in Python with minimal coding that is also intuitive and simple to understand.

Recall that information systems students use VBA to do SQL and one can do the same thing with the Pandas library. Figure 4 shows the Python code that runs a SQL against an MS Access database to retrieve all tuples in the table, 'student.' Line 4 connects to the database and the next line runs the query.

Python code

```
1 import pandas as pd
2 myData = pd.read_excel('sales data.xlsx')
3 print(myData)
4 # print stats on Qty and Amount
5 print (myData [["Qty", "Amount"]].describe())
6 pd.pivot_table(data=myData,
7     index='Date',values=['Amount'],aggfunc='sum')
```

Output

```
   Order No       Date  Qty  Amount
0       101 2022-12-01    2     120
1       103 2022-12-03    4     160
2       104 2022-12-01    7     200
3       105 2022-11-01    3      90
4       456 2022-12-03    4      40
              Qty      Amount
count    5.000000     5.000000
mean     4.000000   122.000000
std      1.870829    61.806149
min      2.000000    40.000000
25%      3.000000    90.000000
50%      4.000000   120.000000
75%      4.000000   160.000000
max      7.000000   200.000000
```

Date	Amount
2022-11-01	90
2022-12-01	320
2022-12-03	200

Figure 3. Process an Excel file and print some statistics

Code

```
1 import pyodbc
2 import pandas as pd
3
4 myConn = pyodbc.connect('DSN=studentData;')
5 myQuery=pd.read_sql_query
6   ('''select * from student''',myConn)
7 myResult = pd.dataFrame(myQuery)
8 print (myResult)
```

Output

```
   ID  SSN      Name
0  22    1   John S.
1  23    2  David T.
2  24    5  Cindy W.
3  25    4  Megan L.
```

Figure 4. Run a SQL query against an MS Access database

Now let us turn our attention to the metric noted in Section 1 - discount factor and the net present value (NPV). As the saying goes, money now is better than money in future. Business savvy students need to know if an investment is profitable (such as loaning money to someone or starting a small business). Net present value (NPV), the net after adding inflow and subtracting outflows, can help students pick an investment that is better among competing investments (CFI Team. 2022); for example, is investing $X now that returns $Y after n years better than, say putting the money in the bank at a rate of interest, r. Without belaboring the point, suffice to say that one can answer such cashflow question rather easily and with minimal coding using Python libraries. For brevity, the Python code is not shown here.

This paper proposed a constructivist approach in the design of a Python-based introductory computing course for IS majors. Couple of instructional pedagogies were introduced to help students make the transition from spreadsheets to doing similar computations in Python. Our illustrative examples use a constructivist approach to perk up students' interest in Python by connecting the new concepts to what they already know as spreadsheet users. We believe this approach should help IS students rediscover what they already know so it is 'known differently,' and prepare them to contribute to multidisciplinary teams of the future workforce by working together with peers who have complementary skills and expertise.

ACKNOWLEDGEMENT

We are grateful for the support through BGSU's Faculty Allies grant and for the constructive comments on an earlier draft from anonymous reviewers.

REFERENCES

April Y. et al, 2018. Mismatch of Expectations: How Modern Learning Resources Fail Conversational Programmers. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) (CHI '18). ACM, New York, NY, USA, pp. 1–13.

Borchert, O. and Valia, G. 2022. ScaffoldSQL: Using Parson's Problems to Support Database Pedagogy. *Proceedings of the 55th Hawaii International Conference on System Sciences*. Hawaii, USA. pp. 1032 - 1041.

Bosacki, A. How is Python Used in Finance? — Python Applications in Finance. https://www.softkraft.co/how-is-Python-used-in-finance/

CFI Team. (2022, November 29). What is a Discount Factor? corporatefinanceinstitute.com/resources/financial-modeling/discount-factor/

Coffman, J. et al, 2023. Visual vs. Textual Programming Languages in CS0.5: Comparing Student Learning with and Student Perception of RAPTOR and Python. *Proceedings of the SIGCSE 2023 Technical Symposium*. Toronto, Canada. *to appear*.

Freud, S. (2012). A general introduction to psychoanalysis. Wordsworth Editions

Harris, K. (2016, February 12). Creating a Bridge to Learning with Scaffolding. https://teaching.nmc.edu/creating-a-bridge-to-learning-with-scaffolding/

Khudov, N. (2020, November 24). Why Python is Essential for Business Analysts. https://towardsdatascience.com/why-Python-is-essential-for-business-analysts-ed3d5a2b194c

Maabo, J. (2020, September 12). VBA vs Python: Key Differences Plus a Dose of History. https://software-solutions-online.com/vba-vs-Python/

Moffatt, M. (2019, April 10). What Is a Discount Factor? https://www.thoughtco.com/discount-factor-definition-1146077

Munshi, A. et al, 2022. Analyzing Adaptive Scaffolds that Help Students Develop Self-Regulated Learning Behaviors. doi:10.48550/ARXIV.2202.09698. *under review*.

Nye, W. (2022, December 2). Transitioning From Excel To Python: Essential Functions For SEO Data Analysis" https://www.searchenginejournal.com/transitioning-from-excel-to-Python-essential-functions-for-seo-data-analysis/472147/

Seltzer, L. (2019, May 8). You Can Only Learn What You Already Know. Psychology Today. Online. https://www.psychologytoday.com/us/blog/evolution-the-self/201905/you-can-only-learn-what-you-already-know

Shah, M. (2020). Exploring the Use of Parsons Problems for Learning a New Programming Language. *Technical Report* No. UCB/EECS-2020-88. 2020. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-88.pdf

Tom (AnalystCave). (2022, June 3). VBA to Python – 10 Simple Python vs VBA Examples. https://analystcave.com/vba-to-Python-10-examples-Python-vs-vba/