MSEL
Modelling in Science Education and Learning

Modelling in Science Education and Learning
*Volume 11 (1), 2018* DOI: 10.4995/msel.2018.9152.
Instituto Universitario de Matemática Pura y Aplicada
Universitat Politècnica de València

# *Coding oriented learning in economics, business and finance*
# *Aprendizaje orientado a la programación en economía, negocios y finanzas*

**Francisco Salas-Molina, David Pla-Santamaria**
Universitat Politècnica de València
francisco.salas.molina@gmail.com, dplasan@upv.es.

**Abstract**

*As the relationship between both students (teachers) and information technology evolves, new tools are required to improve learning (teaching) in social sciences. Economics, business and finance are mainly based on data and dealing with data requires specific skills and techniques such as computer programming in order to get full potential of most quantitative models. In this paper, we propose a coding oriented learning method based on* `Python Notebooks` *which is specifically designed for students of degrees in economics, business and finance. We follow a learning-by-doing strategy that encourages students to implement economic models as a suitable way to improve the understanding of fundamental concepts. As an illustrative example, we also describe a case study in which Python Notebooks are the key tool to teach cash management in a Master in Business Administration program. Since students of today are the decision-makers of tomorrow, a further advantage of the use of a programming language as a teaching tool is the possibility to connect theory to practice by enabling students to implement their own decision support tools.*

*La evolución entre la relación entre los estudiantes (profesores) y la tecnología de la información, requiere nuevas herramientas para mejorar el aprendizaje (enseñanza) en las ciencias sociales. La economía, los negocios y las finanzas se basan principalmente en los datos y el tratamiento de los datos requiere habilidades y técnicas específicas, como la programación informática, para aprovechar al máximo el potencial de la mayoría de los modelos cuantitativos. En este documento, proponemos un método de aprendizaje orientado a la programación basado en* `Python Notebooks`*, que está diseñado específicamente para estudiantes de títulos en economía, negocios y finanzas. Nuestra estrategia de aprendizaje es eminentemente práctica motivando a los estudiantes a implementar modelos económicos como una forma adecuada de mejorar la comprensión de los conceptos fundamentales. Como ejemplo ilustrativo, también describimos un estudio de caso en el que* `Python Notebooks` *es la herramienta clave para enseñar gestión de efectivo en un programa de Máster en Administración de Empresas. Dado que los estudiantes de hoy son los que toman las decisiones del mañana, una ventaja adicional del uso de un lenguaje de programación como herramienta de enseñanza es la posibilidad de conectar la teoría con la práctica al permitir a los estudiantes implementar sus propias herramientas de apoyo a la decisión.*

Palabras clave: Python Notebooks, Economics, learning.
Keywords: Python Notebooks, economía, enseñanza.

## 1. Introduction

There is no doubt in the fact that students across all academic disciplines need to develop information technology skills to obtain their degrees. This fact is also true for teachers. However, in many classrooms, teaching means introducing and explaining concepts and learning means remembering these concepts the day of the examination. To improve the teaching-learning process, methodologies such as inquiry based learning [Eedelson1999addressing] have been developed to encourage students to play a more active role. Furthermore, technology is meant to be one of the key tools to succeed in any student-centered learning approach.

Technology supported teaching is not a new concept [Edelson et al. 1999]. Indeed, a large number of educational projects are currently exploiting the use of computers. A particularly interesting approach in Science and Engineering studies is the use of Python due to the growing number of educational projects using this tool [Barba 2013, Ketcheson 2014, Gutiérrez et al. 2016]. A particularly interesting online project in the field of economics is proposed by Sargent and Stachurski (2017), which presents a comprehensive series of lectures in quantitative economics using programming languages Python and Julia. Python is a free open-source general-purpose programming language with a remarkable increasing popularity [Sargent and Stachurski, 2017].

The development of technological skills for students in the field Social Sciences is usually focused on the use of Internet [Acikalin and Erdinç 2005] as a source for information and data. However, economics, business and finance (and possibly other Social Sciences) are mainly based on data and dealing with data to get full potential of quantitative models require specific skills that usually play a secondary role in economics, business and finance studies. Double degrees in Business Administration and Management and Computer Science aim to combine economics and technology training but there is still a technological gap to fill for the rest of students.

In this paper, we propose Coding Oriented Learning (COL) through the use of Python as a suitable way to enhance the understanding of economics, business and finance concepts, but also to provide students with data wrangling skills. Since a programming language is essentially a quantitative tool, we restrict ourselves to Social Sciences in which quantitative models predominate. To this end, we rely on Jupyter that directly derives from IPython Notebooks [Pérez and Granger 2007] as an interactive computational environment that includes, among others, support for data visualization, code execution, rich text, mathematics and media. As an interesting example of its versatility, VanderPlas (2016) claims that the entire manuscript for this book was composed as a set of IPython Notebooks. A further advantage of the use of Python is the wide range of available software libraries. This feature remarkably mitigates the necessity of using different software applications to implement all concepts included in a given course (e.g., see the case described in Mula and Poler (2011), for a quantitative methods for industrial organization program).

The main advantage for learning provided by COL is the opportunity to develop an improved understanding of economics, business and finance concepts. Implementing quantitative models derived from theoretical concepts contributes to knowledge acquisition process by encouraging students to:

1. Map theory to practice since they have to apply theoretical concepts to implement a model.
2. Verify their own understanding since it is difficult to implement a model that they do not understand.
3. Create their own solutions to a given problem since implementation requires creativity.

4. Experiment with quantitative models since they need to design numerical tests to validate implementations.

To sum up, we propose a coding oriented methodology based on Jupyter Notebooks which aims to enrich learning of students in economics, business and finance. We focus on the use of notebooks as a suitable tool to enhance the understanding of concepts and principles. The main contribution of notebooks as a teaching tool is a kind of double interactivity. First, because teachers are able to interact with notebooks to adapt examples and plots without difficulty on real time. Second, because students can interact with the same notebooks in their laptops at the time of teaching. A further advantage of our approach is the acquisition of other specific skills that are also useful for students such as data exploration, data cleaning and visualization.

In what follows, we first motivate in Section 2 the use of notebooks as a teaching tool. In Section 3, we propose a novel coding oriented learning method for students in economics, business and finance. In Section 4, we describe a real case study of the application of the method proposed in this paper. Finally, in Section 5, we provide some concluding remarks.

## 2. Teaching with Jupyter notebooks

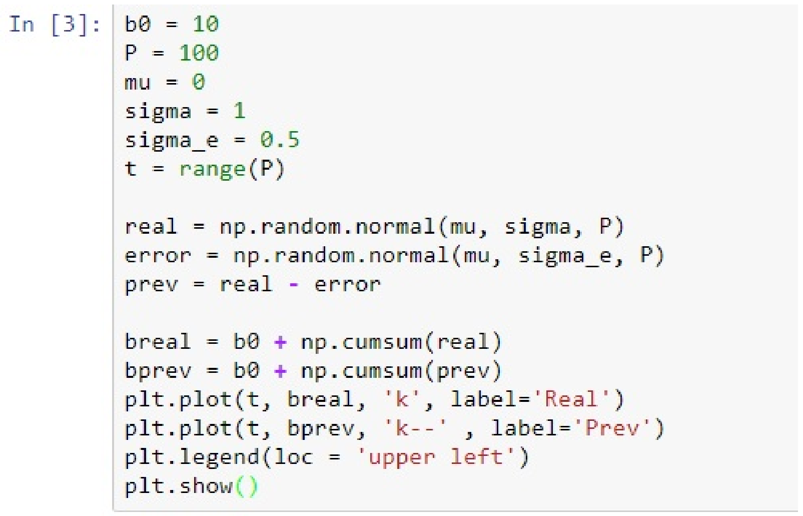
In this section, we provide useful background about Python Notebooks and its use as a teaching tool. Teachers across economics, business and finance academic disciplines often need to combine prose to introduce concepts, mathematics to express quantitative relationships between variables, numerical exercises to illustrate concepts, and figures to better visualize both data and results. The typical teaching approach relies on slides to introduce concepts, examples and plots. However, this method allows little or null interaction between teachers and materials at the time of teaching. Here, the notion of interaction means the possibility to adapt exercises, results and plots on real time and this feature is the main advantage of notebooks.

Jupyter extends IPython notebooks created by Pérez and Granger (2007) to several programming languages such as R, Julia and Octave within an open-source software framework. Python is a high-level general-purpose language conceived by Guido van Rossum in the 1980s. For a brief historical retrospective on Python as a scientific environment, we refer the interested reader to Rossant (2014). Python is built on a number of software libraries such as *numpy* for numerical computing, *pandas* for data analysis, *scipy* for scientific computing, or *matplotlib* for data visualization. The main advantage of notebooks is the combination of code, prose, mathematical expressions, plots, controls, results, images and video in a single stream or workflow that is interactive, meaning that can immediately respond to new instructions coded by users.

Notebooks are organized around independent cells of either code or text which can be individually modified and execute as many times as needed. The result (if any) of each code cell is shown just below it with the appropriate format. In the example shown in Figure 1, we first introduce some data about initial balance $b_0$, planning horizon $P$, and mean and standard deviation parameters to obtain two normally distributed variables, *real* and *error*. We later use these variables to derive the expected cash balance when affected by some error in predictions. Finally, we use the typical code instructions to visualize these time-series in a plot to compare the evolution of expected and real balances under some degree of uncertainty that can be controlled by the students to analyze its impact.

Initially devoted to scientific computing [Pérez and Granger 2007; Kluyver et al. 2016], we here argue that notebooks offer a suitable way to enrich both teaching and learning because:

1. teachers can interact with notebooks according to their preferences to teach;
2. students can follow the class (online) in their own laptops at the time of teaching and can also (offline) implement exercises and models to enhance the understanding of principles and concepts;
3. both teachers and students can experience teaching and learning processes within a rich environment combining many useful elements;
4. it is free, which is always a desirable feature for both teachers and students.



```
In [3]: b0 = 10
        P = 100
        mu = 0
        sigma = 1
        sigma_e = 0.5
        t = range(P)

        real = np.random.normal(mu, sigma, P)
        error = np.random.normal(mu, sigma_e, P)
        prev = real - error

        breal = b0 + np.cumsum(real)
        bprev = b0 + np.cumsum(prev)
        plt.plot(t, breal, 'k', label='Real')
        plt.plot(t, bprev, 'k--' , label='Prev')
        plt.legend(loc = 'upper left')
        plt.show()
```

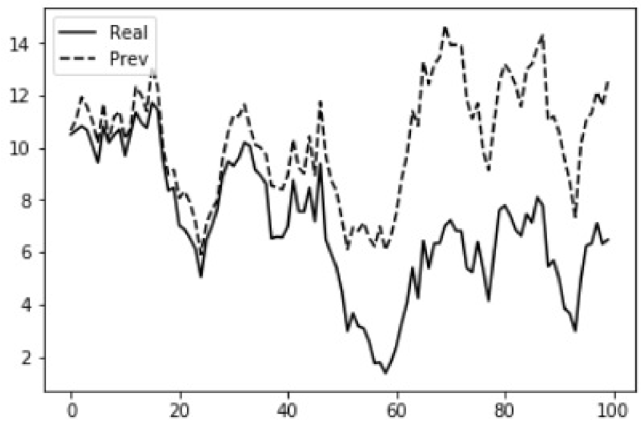


Figure 1: A cell in a Jupyter Notebook that outputs a graphic.

In what follows, we consider recent examples of the use of (IPython) notebooks for teaching purposes. The post in Barba (2013) describes a practical module of Professor Barba's Computational Fluid Dynamics class, as taught between 2010 and 2013 at Boston University. Ketcheson (2014) highlights the utility of combining mathematics, prose explanations, executable code and plots in an interactive environment as a learning tool for a numerical methods course. The author also recommends to make sure that the students type code from the start and to help them to discover concepts on their own. On the drawbacks side, the author comments on installation issues, scalability problems for large number of students, the difficulty to open notebooks and non-linear execution since code cells can be executed in any order resulting in different outputs. We will consider possible actions to mitigate these drawbacks in Section 4.

Sargent and Stachurski (2017) present a comprehensive series of lectures for upper level undergraduate students, graduate students and researchers in economics and finance using Python and Julia. The authors (one of them, Sargent, a Nobel laureate in economics) emphasize the utility of programming, mathematics and statistics as powerful tools to study economic theory. After providing background on the programming languages (Python or Julia) and additional tools and techniques, this project covers a wide range of single and multiple agent models implementations and economic dynamics.

Computer programming may be too challenging for students as pointed out by Guzdial (2010). However, we argue that combining code with text and plots in a familiar environment such as a browser facilitates the task of getting basic programming skills. Recall that the final goal is not transforming students in expert programmers but providing them with an additional useful tool to learn and experiment.

## 3. Coding oriented learning in economics, business and finance

In this section, we propose a method to enhance the understanding of economics, business and finance concepts and principles through the use of Jupyter notebooks. By doing so, we aim to foster computational literacy among students who are not supposed to develop this kind of skills in depth. Following the recommendations in Disessa (2001) and Wilensky et al. (2014), we advocate to introduce computational literacy through economics, business and finance classes rather than restrict these skills to science and engineering programs.

Both teaching and learning mainly deals with information and knowledge transmission. Providing students with a more prominent role in this process through increasing their participation in a learning-by-doing scheme is usually a desired feature for most teachers. In addition, firsthand experience happens to be very important in developing good intuitions about the underlying concepts and principles in many subjects. Thus, we here follow a coding approach to enrich learning in economics, business and finance fields in which implementing concepts plays a prominent role. In order to articulate this computational thinking in a suitable workflow for both teachers and students, we propose the following code-oriented method:

1. Select the concept to be taught.
2. Define the model that describes the relationship between the variables of interest within the selected concept.
3. Code or implement the model.
4. Experiment with the implementation of the model.

We classify this method as coding-oriented because the implementation step is the key aspect. This fact restricts the application of this method to teaching quantitative concepts that can be coded in one way or another. However, we encourage teachers to be creative in the search for alternative ways to implement concepts, hence enriching the learning process. As an illustrative example, consider a simple data exploratory exercise for a given time-series of daily observations. At first glance, there is no new concept to teach and there is nothing more to do rather than computing some statistics and visualizing its evolution. However, even though the concept of empirical mean is a known concept for most undergraduate students, one can enrich the learning process by introducing either the concept of moving average or the concept of filtered average considering only a subset of values from the original time-series such as the working days.

It is important to highlight that the final goal is to increase the understanding of principles and concepts in the field of economics, business and finance. To this end, we rely on the development of students' computational skills to become as usual as reading and writing. By both implementing and running economic models in a number of experiments, students can build up knowledge in an iterative way. Repeated cycles of implementation, execution and debugging provide students with the opportunity to achieve a better understanding of concepts both in the classroom (online) and out of the classroom (offline).

## 4. Case study: Python for cash management

In this section, we describe a real case study in which we apply the method introduced in Section 3 to teach cash management in the Master's degree in Business Administration at the Polytechnical University of Valencia (Campus of Alcoy) in 2017. Within the subject Advanced Techniques of Financial Management, we introduce several cash management concepts through the use of Python and Jupyter notebooks. In order to illustrate COL, we first select an important concept in cash management, we present a model to manage cash that we later code in Python to do some interesting experiments. Finally, we comment on the lessons learned in the classroom.

### 4.1. The concept

Cash managers deploy procedures to find the appropriate balance between what is held in cash for operational purposes and what is allocated in alternative investments in exchange for a profit. For a comprehensive review on cash management literature since the first research works, we refer the interested reader to Gregory (1976), Srinivasan and Kim (1986) and da Costa Moraes et al. (2015). The set of rules used by cash managers to control balances through a sequence of control actions is called a cash management model. One of the first models to control stochastic cash balances was proposed by Miller and Orr (1966).

### 4.2. The model

Miller and Orr (1966) proposed a simple policy based on three bounds to control cash balances. This policy implies that when some upper bound $U$ is reached a withdrawal transfer is made by selling available short-term investments to restore the balance to a target level $Z$. In the same way, when the cash balance reaches some lower bound $L$, a positive transfer is made by investing idle cash to restore the balance to $Z$. This model is mathematically expressed as:

$$x_t = \begin{cases} Z - b_{t-1} - f_t & \text{if} \quad b_{t-1} + f_t > U \\ 0 & \text{if} \quad L < b_{t-1} + f_t < U \\ Z - b_{t-1} - f_t & \text{if} \quad b_{t-1} + f_t < L. \end{cases} \tag{1}$$

As an example, let us consider as initial conditions a cash balance $b_0 = 8$ and an expected cash flow $f_1 = 1$. Assume also control bounds $U = 20$, $Z = 15$ and $L = 10$. Since $b_0 + f_1 = 8 + 1 = 9$ is below $L = 10$, the Miller and Orr model implies that $x_1 = Z - b_0 - f_1 = 15 - 8 - 1 = 6$, to restore the balance to the target balance $Z$ in view of the fact that $b_1 = b_0 + f_1 + x_1 = 8 + 1 + 6 = 15$. Here, control action $x_t$ is made at the end of time step $t$ when the current state is $b_t$ is updated with cash flow $f_t$ and the state transition law is given by:

$$b_t = b_{t-1} + x_t + f_t, \tag{2}$$

where $f_t$ is assumed to be a Gaussian, uncorrelated and stationary cash flow process. Miller and Orr set the lower limit $L$ to zero, and $Z$ and $U$ according to:

$$Z = L + \left( \frac{3 \cdot \gamma_0 \cdot \sigma^2}{4 \cdot h} \right)^{1/3} \tag{3}$$

and

$$U = 3 \cdot Z - 2 \cdot L, \tag{4}$$

where $\gamma_0$ is the fixed transaction cost, $\sigma$ is the standard deviation of net cash flows and $h$ is the holding cost per money unit.

## 4.3. The code

Starting at an initial balance, cash managers can control cash balances by deploying the [Miller and Orr 1966] model at each time step through the following piece of code:

```python
def transfer(f, b0, U, Z, L):
    """Obtains transfer x from cash flow, b0 and U, Z, L"""
    if b0 + f > U or b0 + f < L:
        x = Z - b0 - f
    else:
        x = 0
    return (x)
```

By implementing this cash management model, students need to understand how it works and what are its main characteristics. Even if students do not fully understand the model, repeated attempts of coding, execution and debugging will likely result in a progressive enhancement of the knowledge about the underlying concept.

## 4.4. The experiments

Once the model has been implemented, experimentation fulfills two fundamental goals: first, testing that the implementation is correct; and second, the improvement of the understanding by means of a what-if analysis strategy. On the one hand, every software implementation requires a verification that the code does what it is supposed to do, hence avoiding any unexpected behavior and detecting possible software errors (bugs). Details of about software testing [Myers et al. 2011] is beyond the scope of this paper, but we encourage teachers to design simple tests to verify that possibly different implementations lead to the expected result. For instance, the *transfer* function defined in Section 4.3 assumes that $L < Z < U$. This assumption implies that $x < 0$ when $b_0 + f > U$, and that $x > 0$ when $b_0 + f < 0$. However, if the user executes this function *transfer(f=0, b0 = 8, U=5, Z=10, L=3)* with $Z > U$, condition $b0 + f > U$ holds but the returned value 2 is unexpected since we expected a negative one. This simple test leads to consider improving the code by checking if $L < Z < U$ holds, warning the user otherwise.

In addition, experimentation allows what-if analysis, hence improving the understanding of the problem under consideration. As a simple example, consider the following question: what happens with control bounds $L$, $Z$, and $U$ if standard deviation of cash flows $\sigma$ doubles its value? Even though the analytical solution to this question is not difficult, a simple experiment giving alternative values to the variables that are necessary to compute bounds $L$, $Z$, and $U$ is likely to help students understand the impact of changes in the input parameters. Furthermore, this experiment can even help verify if the results obtained are coherent with the analytical solution.

### 4.5. Lessons learned in the classroom

In this section, we describe our teaching experiences when introducing cash management in a Master's degree in Business Administration. Since this was the first time that we used Jupyter Notebooks to teach, we paid special attention to the comments provided by students.

The first comments from students were related to installation problems. Even though students were previously provided with a short tutorial on installing the required software in their own laptops, some problems arose when they tried to get started with notebooks in the first class. Since installation problems are difficult to solve within time allocated for teaching, these issues need to be postponed to tutorials outside class hours. Fortunately, at the time of writing this paper, big technological companies such as Google and Microsoft provide web-based services including Jupyter Notebooks. Colaboratory [Google 2017] is a Google research project created to help disseminate machine learning education and research. Azure Notebooks [Microsoft 2107] is a similar project that provides online access to Jupyter Notebooks. Both services are free and represent a remarkable option to avoid installation problems since no setup is required. One can start using the service just after singing up in one of these platforms.

Introducing new concepts such as cash management models by means of a new tool such as Jupyter Notebooks is a double challenge for students. Then, a balance between background concepts about the teaching topic and Python essentials is a must. In this sense, one of the main advantages of Jupyter Notebooks is that you can combine text, equations, code and graph in the same interactive environment. In an attempt to follow a learn-by-doing strategy in the classroom, we decided to provide notebooks with the specific topic to teach after each class. Then, students were expected to write the same code that the teacher is writing and explaining. However, if a student fails to keep the same pace than the teacher, it is very difficult for her/him to adhere again to the learning process. We think that a suitable solution to this problem is providing the students with teaching materials in advance to be used in case they need.

Finally, we learned that allocating enough time to motivate both the topic and the tool is necessary. Within the bounds of the case study described here, answering the question why teaching cash management using Jupyter Notebooks is an important thing to do, it is a relevant message to send to the students. In this sense, we believe that the method described above is a suitable strategy: first, we introduce the concept to teach; second, we describe the model that illustrate this concept; third, we implement the model through Python code; and finally, we perform a number of experiments to test the code and also to study the model.

## 5. Concluding remarks

Information technology skills are important both for teachers and students. Indeed, economics, business and finance are full with concepts that can be expressed by means of quantitative models to allow a better exploration of the underlying problem. In this paper, we propose Coding Oriented Learning through the use of Jupyter Notebooks as a suitable tool to connect theory and practice within the teaching/learning process. We claim that interactivity and the possibility to combine code, text, mathematics and graphics are important advantages to enhance the understanding of economic concepts.

The method described here is mainly suitable for quantitative models that are likely to appear in the contents of many economics, business and finance degrees. This method is based on a key assumption, i.e., by implementing a model, students are somehow impelled to understand the underlying concept. To this end, we propose four main steps: (i) selecting the concept; (ii) defining the model; (iii) coding the model; and (iv) experimenting with the code.

As an illustrative example of the application of this method, we discuss relevant questions raised in the classroom by means of a real case study in which Python Notebooks are used to teach cash management. As a result, we provide teachers with a new method to improve teaching through the use of state-of-the-art technology. In addition, we provide students with an additional tool to learn both in the class and outside the class. Finally, it is important to highlight that the method proposed in this paper can also be used to implement their own decision support tools.

## Referencias

Acikalin, M. and Erdinc, D. (2005).
*The use of computer technologies in the social studies classroom.*
TOJET: The Turkish Online Journal of Educational 9 Technology, 4(2).

Barba, L. (2013).
*Cfd python: 12 steps to navier-stokes.*
http://lorenabarba.com/blog/cfd-python-12-steps-to-navier-stokes/

da Costa Moraes, M. B., Nagano, M. S., and Sobreiro, V. A. (2015).
*Stochastic cash flow management models: A literature review since the 1980s.*
In Decision Models in Engineering and Management, pages 11–28. Springer International Publishing.

DiSessa, A. A. (2001).
*Changing minds: Computers, learning, and literacy.*
Mit Press.

Edelson, D. C., Gordin, D. N., and Pea, R. D. (1999).
*Addressing the challenges of inquiry-based learning through technology and curriculum design.*
Journal of the learning sciences, 8(3-4), 391–450.

Google (2017). Colaboratory.

Gregory, G. (1976).
*Cash flow models: a review.*
Omega, 4(6), 643–656.

Gutiérrez, L. M. G., Pita, J. L. C., and Burgos, D. E. (2016).
*Implantación de metodologías de cálculo a través del lenguaje Python para asignaturas impartidas en las titulaciones de la ETSIN.*
Modelling in Science Education and Learning, 9(1), 151–160.

Guzdial, M. (2010).
*Why is it so hard to learn to program?*
In Making software: What really works, and why we believe it, pages 111–121. O'Reilly Media, Inc.

Ketcheson, D. I. (2014).
*Teaching numerical methods with iPython notebooks and inquiry-based learning.*
In Proceedings of the 13th Python in Science Conference. SciPy. org.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., et al. (2016).
Jupyter notebooks-a publishing format for reproducible computational workflows.
In ELPUB, pages 87–90.

Microsoft (2017).
*Azure notebooks.*

Miller, M. H. and Orr, D. (1966).
*A model of the demand for money by firms.*
The Quarterly journal of economics, 80(3),413–435.

Mula, J. and Poler, R. (2011).
*Applied mathematical modelling for industrial engineers.*
Modelling in Science Education and Learning 4, 307–318.

Myers, G. J., Sandler, C., and Badgett, T. (2011).
*The art of software testing.*
John Wiley & Sons.

Pérez, F. and Granger, B. E. (2007).
*Ipython: a system for interactive scientific computing.*
Computing in Science & Engineering, 9(3).

Rossant, C. (2014).
*IPython interactive computing and visualization cookbook.*
Packt Publishing Ltd.

Sargent, T. J. and Stachurski, J. (2017).
*Lectures in quantitative economics.*
https://lectures.quantecon.org/

Srinivasan, V. and Kim, Y. H. (1986).
*Deterministic cash flow management: state of the art and research directions.*
Omega, 14(2), 145–166.

VanderPlas, J. (2016).
*Python Data Science Handbook: Essential Tools for Working with Data.*
O'Reilly.

Wilensky, U., Brady, C. E., and Horn, M. S. (2014).
*Fostering computational literacy in science classrooms.*
Communications of the ACM, 57(8), 24–28.