TEACHING-LEARNING MULTI-TASKS PROGRAMMING USING PYTHON

Franco Ortega, Universidad Viña del Mar
Brandon General, Universidad Viña del Mar
Bairon Vega, Universidad Viña del Mar
Cristian Vidal, Universidad Católica del Norte
Claudia Jiménez, Universidad Viña del Mar
José Miguel Rubio, Universidad Bernardo O´Higgins

ABSTRACT

Algorithmic thinking is part of human nature. Nonetheless, writing algorithms in programming languages can be not trivial, mainly for syntax issues. Multi-tasks thinking seems natural for human teams. Nonetheless, implementing multi-tasks solutions can be a complex task for people and students who learn programming languages mainly for syntax issues. Python is a real solution to simplify their programming learning. This article presents and highlights the learning experience of Chilean students to implement multi-task solutions, that is, concurrent and parallel programs, in Python. Those benefits invite us to promote programming using Python in other courses, majors, and institutions. Programming for everybody is nearest now.

Keywords: Multi-task, Concurrent, Parallel, Programming, Python Solutions

INTRODUCTION

Sequential programming only allows the creation of solutions for sequential execution. The first computer systems were completely sequential; they would run only one program at a time until each of them finished to run another one (Oganjanyan, 2018). That happened even though each running task used only some system resources and others were free during execution. Concurrent computing was born to look for optimizing the use of computing resources. According to Schneider (Schneider, 2012), concurrent computing consists of a set of processes (running programs) and shared resources (CPU, shared memory, or network resources in use). Concurrent programming tries to improve responsiveness by reacting to the occurrence of possibly simultaneous events (network events, user interface, other computers or peripherals) (Sadowski, 2011). Parallel programming allows the parallel use of resources from a particular machine (such as multiple cores) or the set of machines in a cluster to try to improve the performance of computing tasks (Yazdeen, 2021). Thus, the concurrent execution of tasks can be seen as an apparent parallelism (Zhang, 2017).

Figure 1 (Terrell, 2016) illustrates concurrent task execution and parallel task execution. Algorithmic reasoning refers to the identification of tasks for their sequential performance. Thus, the main complexity of a programming solution is the translation of an algorithm into a language. Python is a programming language that reduces these difficulties (Srinath, 2017). Python is a high-level, interpreted programming language with a simple syntax to facilitate its learning and also emphasize the readability of its solutions to reduce its maintenance cost (Manolescu, 2021). For example, there are multiple positive experiences of using Python for teaching in various lines such as first programming language and web systems development (Pandey, 2020; Ellis, 2019). Likewise, there are experiences of teaching and using advanced computing topics in under- graduate courses such as big data (Edifor, 2021; Malik, 2021), machine learning (Lemenkova, 2019; Gupta, 2017) and deep learning (Lafuente, 2021; Raschka, 2017). This work aims to demonstrate the advantages and facilities of the practical teaching of

Citation Information: Ortega, F., Brandon, G., Vega, B., Vidal, C., Jiménez, C., & Rubio, J.M. (2020). Teaching-learning multi-tasks programming using python. *Journal of Management Information and Decision Sciences, 24*(S6), 1-7.

concurrent and parallel computing using the Python programming language. It is considered an experience in the Service Configuration (Conf-Ser) course of the current curriculum of Computer Engineering of the Viña del Mar University (CE-VMU).

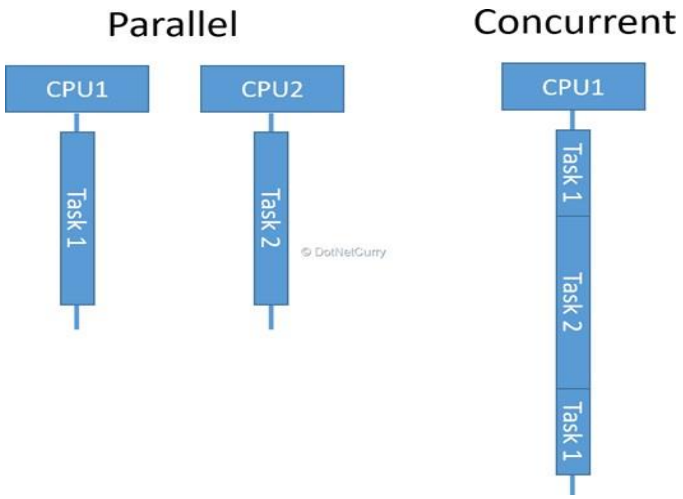

FIGURE 1
PARALLEL *VS* CONCURRENT COMPUTATION

HISTORY OF COMPUTING

Sequential Computing

Before the 1960s, hardware limited computing and programming performance. The optimization of efficiency in a specific machine when executing a specific algorithm (Sewak, 2018) characterized solutions at that time. Sequential programming emerged in the 1960s. Sequential or structured programming is a programming paradigm that allows the division of a program into blocks or programming subroutines for the sequential execution of tasks. Basic structures of sequential computing are the selection (if and switch) and iteration (for and while). The objective of structured programming is to make modular programs with more clarity, quality, easier to maintain, and with fewer errors (Botella López, 1979). In recent years, the speed of sequential computers has grown by orders of magnitude due to advances in their design and construction techniques (Moreno, 2014) and the development and standardization of software solutions (for example, the UNIX operating system (Aguilar, 2004). Although sequential computing has made extraordinary advances, it has not been able to demonstrate its successful application in complex problems that require machines that work autonomously, robustly, and efficiently in dynamic and threatening environments (Huerta, 2001). A clear example of this dynamism is working with large volumes of data (BigData) and time limits (Rojas Galeano, 1999).

Concurrent Computing

Concurrent computing refers to the fact that running pro- grams share the available computing resources for the same time, but their execution does not occur at the same time cites tanenbaum 2015 modern. Multiple processes can run on a single processor, giving each process a limited time to run. Thus, only one process runs at a given time. If a process P does not complete its work in a defined time, it stops, and another process starts or resumes its execution. Therefore, process P would run again until it finishes its work. Structuring software systems in a way in concurrent and communicated tasks is an excellent way to handle the complexity of an application cites rajsbaum 2019 mastering. According to Psycharis and Kallia cite psycharis

Citation Information: Ortega, F., Brandon, G., Vega, B., Vidal, C., Jiménez, C., & Rubio, J.M. (2020). Teaching-learning multi-tasks programming using python. *Journal of Management Information and Decision Sciences, 24*(S6), 1-7.

2017 effects, from the mid-1980s to 2004, increasing the number of instructions per second was the dominant reason for improvements in computer performance. The increasing complexity and time required of computing problems and the physical limitations of hardware demand new efficient forms of computing (Figure 2).

FIGURE 2
'HELLO WORLD' IN PYTHON

Parallel Computing

The physical limitation makes it impossible to obtain improvements in processing speed. One way of working to achieve potential improvements is with the use of multiple processors (Puyol Moreno, 2014). Parallel computing allows the simultaneous use of multiple processing elements for problem-solving. The processing elements include resources such as a multi-processor computer, a computer network, specialized hardware, or any combination of these (Naiouf, 2011). Thus, dividing a solution into sub-solutions that can be executed simultaneously for the parallel use of computing resources is required.

PROGRAMMING IN PYTHON

Python is a high-level, interpreted programming language with a simple syntax, properties that make it very suitable for the teaching-learning process (Koprawi, 2020; Vidal, 2021). Figure 2 shows the classic example of displaying a 'Hello World' message in Python where only one command or instruction is needed to write or display the message on the screen (print) and the message to display. The implementation of this example was in the online tool Google Colaboratory (Weiss, 2020). Python also supports modern computing features such as machine learning (Carneiro, 2018), and web application development (Chang, 2019). In addition, Python allows the development of multi-tasking solutions, concurrent and parallel computing solutions.

Advantages of the Multi-Tasking Approach

The multi-task approach is the ability of human beings to perform multiple tasks at the same time (Mele´, 2020). Performing multiple tasks at the same time is not a problem when those tasks are not in conflict. Working simultaneously on jobs can generate considerable advantages in increasing productivity and performance. With multi-task processing, it is possible to finish pending tasks in a single block of time, avoid wasting energy working separately on activities grouped into a single block of time, and save time and space in the schedule. Well-executed multi- tasking would allow large volumes of work thanks to increased performance without compromising quality. In addition to saving time, it translates into economic

Citation Information: Ortega, F., Brandon, G., Vega, B., Vidal, C., Jiménez, C., & Rubio, J.M. (2020). Teaching-learning multi-tasks programming using python. *Journal of Management Information and Decision Sciences, 24*(S6), 1-7.

performance. In computer science, as already described, we work with concurrent, parallel, and distributed computing as a form of multi-task solution. This work presents experiences of the first two.

Concurrent and Parallel Computing with Python

For process and thread creation in Python, we require the Process and Thread constructures of the multiprocessing and threading libraries, respectively. Figures 3 and 5 show example codes in C language for creating a process and a thread to send parent and child messages, respectively. Figures 4 and 6 present codes with the same purpose in Python. There are already differences in syntax for the syntactic rules of C and Python programming language in these examples. These rules in C usually disturb and distract students from understanding the purpose and usefulness of these functions. Furthermore, since the first programming courses develop sequential computing reasoning, thinking about parallel computing solutions is potentially complex, even though this way of reasoning is very typical for human beings for example, actions of daily life such as walking and talking are usually in parallel and teams of work usually distribute tasks in the team members.

```c
#include<sys/wait.h>
#include<stdio.h>

int main() {
    int pid;

    pid = fork();

   if (pid==0)
        printf("Soy el hijo: %d\n", getpid());
   else
        printf("Soy el padre: %d\n", getpid());
```

FIGURE 3
"HELLO WORLD" USING PROCESSES IN C

EXPERIMENTS AND RESULTS

In the Conf-Sev course of CE-VMU, the students were divided into four workgroups to develop solutions, first using concurrency and then parallel computing. As shown in table I and II, each group was assigned a classic interprocess communication problem (Sanchez, 2015). During the course, each group developed and presented their solutions effectively. At the end of the course, each group compared both solutions and presented their advantages and disadvantages.

In all groups, the concurrent solutions were more optimal in execution times, concluding that, in these problems, the multi- threaded solution is more optimal. The students also concluded that this result influenced the fact that these problems require access to shared resources, whose access is of a high cost for parallel solutions, which do not allow reaching the advantages of parallel computing to shine through (Table 1 & Figure 4, 5 and 6).

Citation Information: Ortega, F., Brandon, G., Vega, B., Vidal, C., Jiménez, C., & Rubio, J.M. (2020). Teaching-learning multi-tasks programming using python. *Journal of Management Information and Decision Sciences, 24*(S6), 1-7.

```c
#include<pthread.h>
#include<stdio.h>

void *Saludar(void *arg){
    printf("Soy la hebra hija\n");
}

int main()
{
    pthread_t hija;

    pthread_create(&hija, NULL, Saludar, NULL);
    pthread_join(hija, NULL);
    printf("Soy la hebra principal\n");
}
```

FIGURE 4
"HELLO WORLD" USING THREADS IN C

```python
from multiprocessing import Process
from os import getpid

def Saludar():
    print("Soy el hijo:", getpid())

if __name__ == "__main__":
    p1 = Process(target=saludar,)
    p1.start()

    print("Soy el padre:", getpid())
```

FIGURE 5
"HELLO WORLD" USING PROCESSES IN PYTHON

```python
from threading import Thread

def Saludar():
    print("Soy la hebra hija")

hebra = Thread(target=Sumar,)
hebra.start()

print("Soy la hebra principal")
```

FIGURE 6
"HELLO WORLD" USING THREADS IN PYTHON

Table 1
RESULTS OF CONCURRENT SOLUTIONS

Group	Running time
Group 1 (Readers and Writerrs)	18.0049s
Group 2 (Sleepomg Barber)	50s
Group 3 (Dining Philosophers)	12.0034s
Group 4 (Producer Consumer)	2.13s

CONCLUSIONS

This work demonstrates the utility of using Python to teach multi-task programming, concurrent and parallel computing, and allowing students to develop their solutions effectively.

Citation Information: Ortega, F., Brandon, G., Vega, B., Vidal, C., Jiménez, C., & Rubio, J.M. (2020). Teaching-learning multi-tasks programming using python. *Journal of Management Information and Decision Sciences, 24*(S6), 1-7.

Table 2 RESULTS OF PARALLEL SOLUTIONS	
Group	**Execution time**
Group 1 (Readers and Writers)	18.2471s
Group 2 (Sleeping Barber)	78s
Group 3 (Dinner of Philosophies)	12.2926s
Group 4 (Producer Consumer)	2.43s

This work is just an additional sample of the usefulness of Python for the development of algorithmic skills and, in this case, multi-tasking programming. According to the obtained results, each of the working groups effectively implemented and presented their solutions. As future work, we propose using Python from the first programming courses at the authors' universities and include examples of multi-task computing without shared resources.

REFERENCES

Aguilar, J., & Leiss, E. (2004). Introduction to parallel computing. Venezuela: Quintet Graphics.

Botella, P.L., & Rodriguez, H. (1979). Structured programming (an attempt at clarification). *Association of Computer Technicians, 28*, 5-11.

Carneiro, T., Da No´brega, R.V.M., Nepomuceno, T., Bian, G.B., & Reboucas Filho, P.P. (2018). Performance analysis of Google collaborator as a tool for accelerating deep learning applications. *IEEE Access, 6*, 677-685.

Chang, H.C., Wang, C.Y., & Hawamdeh, S. (2019). Emerging trends in data analytics and knowledge management job market: Extending KSA framework. *Journal of Knowledge Management, 23*(4), 664-686.

Edifor, E., Swenson, A., & Aiyenitaju, O. (2021). A virtual reality framework for upskilling in computer programming in the business context. Augmented Reality and Virtual Reality: New Trends in Immersive Technology.

Ellis, M.E., Hill, G., & Barber, C.J. (2019). Using python for introductory business programming classes.

Gupta, B., Negi, M., Vishwakarma, K., Rawat, G., & Badhani, P. (2017). Study of twitter sentiment analysis using machine learning algorithms on python. *International Journal of Computer Applications, 165*(9), 29-34.

Huerta, A.V. (2001). Introduction to the Unix operating system.

Koprawi, M. (2020). Parallel computation in uncompressed digital images using computer unified device architecture and open computing language. *PIXEL: Embedded Systems and Logic Computer Science Research, 8*(1), 31-38.

Lafuente, D., Cohen, B., Fiorini, G., Garc´ıa, A.A., Bringas, M., & Onna, D. (2021). A gentle introduction to machine learning for chemists: An undergraduate workshop using python notebooks for visualization, data processing, analysis, and modelling. *Journal of Chemical Education, 98*, 2892-2898.

Lemenkova, P. (2019). Processing oceanographic data by python libraries numpy, scipy and pandas. *Aquatic Research, 2*(2), 73-91.

Manolescu, D. (2021). The magic and mysteries of teaching ESL. *Middle Eastern Journal of Research in Education and Social Sciences, 2*(3), 122-128.

Malik, A., Scholar, U., Mahal, A., Kamboj, A., & Sharma, A. (2012). Big data & sentiment analysis using python. *Research Journal of Computer Science, 7*, 159-166.

Mele, A. (2020). Django 3 by example: Build powerful and reliable Python web applications from scratch. *Packt Publishing Ltd*.

Moreno, J. (2014). Structured Language Programming.

Naiouf, M. (2011). Parallel and distributed processing.

Oganjanyan, S.B., Shilov, V.V., & Silantiev, S.A. (2018). Armenian computers: First generations. In IFIP International Conference on the History of Computing: Springer.

Pandey, Y.N., Rastogi, A., Kainkaryam, S., Bhattacharya, S., & Saputelli, L. (2020). Python programming primer. In machine learning in the oil and gas industry: Springer.

Puyol Moreno, J. (2014). An approach to big data.

Raschka, S., & Mirjalili, V. (2017). Python machine learning: Machine learning and deep learning with python.

Rojas Galeano, S.A. (1999). Alternative computational paradigms: Parallelism and simplicity inspired by biology. *Engineering, 5*(1), 53-58.

Schneider, F. (2012). On concurrent programming. Texts in computer science: Springer, New York. Retrieved from https://books.google.cl/books?id=Mu7SBwAAQBAJ

Sadowski, C., Ball, T., Bishop, J., Burckhardt, S., Gopalakrishnan, G., Mayo, J., Musuvathi, M., & Toub, S. (2011). Practical parallel and concurrent programming. *In Proceedings of the 42nd ACM technical symposium on Computer science education*.

Citation Information: Ortega, F., Brandon, G., Vega, B., Vidal, C., Jiménez, C., & Rubio, J.M. (2020). Teaching-learning multi-tasks programming using python. *Journal of Management Information and Decision Sciences, 24*(S6), 1-7.

Sanchez, I. (2015). Multitasking or multitasking at work: More productive? Retrieved from https://www.rfi.fr/es/sociedad/20151203-el-multitasking-o-multitarea-en-el-trabajo-mas-productivo.

Sewak, M., Karim, M.R., & Pujari, P. (2018). Practical convolutional neural networks: Implement advanced deep learning models using Python. *Packt Publishing Ltd*.

Srinath, K. (2017). Python–the fastest growing programming language. *International Research Journal of Engineering and Technology (IRJET)*, *4*(12), 354-357.

Tanenbaum, A.S., & Bos, H. (2014). Modern operating systems. Boston, MA: Pearson.

Terrell, R. (2018). Concurrency in NET: Modern patterns of concurrent and parallel programming. USA: Manning Publications Co.

Vidal, C., Sanchez, A., Serrano, J., & Rubio, J. (2021). Academic experience in rapid development of web information systems with Python and Django. University education,14(6).

Weiss, C.J. (2020). A creative commons textbook for teaching scientific computing to chemistry students with python and Jupyter notebooks. *Journal of Chemical Education*, *98*(2), 489-494.

Yazdeen, A.A., Zeebaree, S.R., Sadeeq, M.M., Kak, S.F., Ahmed, O.M., & Zebari, R.R. (2021). Fpga implementations for data encryption and decryption *via* concurrent and parallel computation: A review. *Qubahan Academic Journal*, *1*(2), 8-16.

Zhang, Y., & Yang, Q. (2017). An overview of multi-task learning. *National Science Review*, *5*(1), 30-43.

Citation Information: Ortega, F., Brandon, G., Vega, B., Vidal, C., Jiménez, C., & Rubio, J.M. (2020). Teaching-learning multi-tasks programming using python. *Journal of Management Information and Decision Sciences, 24*(S6), 1-7.