

Introductory Computer Science Course by Adopting Many Programming Languages

Francisco de Assis Zampiroli*, Fernando Teubl, Guiou Kobayashi,
Rogério Neves, Luiz Rozante, Valério Ramos Batista

Universidade Federal do ABC, Brasil

Simpósio Brasileiro de Informática na Educação, 2021

*Grant #2018/23561-1, São Paulo Research Foundation (FAPESP)

Contents

- Motivation
- Methods
 - Didactic Material in Colab
 - Conversions
 - Example
 - Free Access
 - Automatic Correction of Codes
- Experiments and Conclusions
- Future Work

Motivation

- Literate Programming

- "The main idea is to treat a program as a piece of literature, addressed to human beings rather than to a computer" [Knuth 1984]

- Using Google Colab or Jupyter

- In Didactic Material with Colab in six Language Programming
- Automatically convert notebooks into IPYNB, HTML, LATEX and PDF

- Automatic Correction of Codes

- MCTest+Moodle+VPL [SBIE-2020]

Method: Didactic Material in Colab

The screenshot displays a file browser interface with two columns. The left column shows a directory structure with folders and files, and the right column shows a list of individual files. Each item is accompanied by a green checkmark, indicating that all files are present or verified.

Item	Status
all (folder)	✓
figs (folder)	✓
filterNotebook.py (file)	✓
gen (folder)	✓
latex_begin.txt (file)	✓
README.md (file)	✓
cap1.part1.ipynb (file)	✓
cap1.part2.lab.ipynb (file)	✓
cap2.part1.ipynb (file)	✓
cap2.part2.lab.ipynb (file)	✓
cap3.part1.ipynb (file)	✓
cap3.part2.lab.ipynb (file)	✓
cap4.part1.ipynb (file)	✓
cap4.part2.lab.ipynb (file)	✓
cap5.part1.ipynb (file)	✓
cap5.part2.lab.ipynb (file)	✓
cap5.part3.lab.ipynb (file)	✓
cap6.part1.ipynb (file)	✓
cap6.part2.lab.ipynb (file)	✓
cap6.part3.lab.ipynb (file)	✓

Method: Didactic Material in Colab

Table 2. Code cells before applying the filter. After executing `filterNotebook.py` the first line containing `#[typei]#` will be removed. Moreover, cells whose first line indicate another PL will not appear in the notebooks created automatically.

code	compile/run
<pre>#[py]# %%writefile cap01exem01.py print("Hello, World!")</pre>	<pre>#[py]# !python cap01exem01.py</pre>
<pre>#[r]# %%writefile cap01exem01.r cat("Hello, World!")</pre>	<pre>#[r]# !Rscript cap01exem01.r</pre>
<pre>#[js]# %%writefile cap01exem01.js console.log("Hello, World!")</pre>	<pre>#[js]# !node cap01exem01.js</pre>
<pre>#[java]# %%writefile cap01exem01.java class cap01exem01 { public static void main (String[] args) { System.out.println("Hello, World!"); } }</pre>	<pre>#[java]# !javac cap01exem01.java !java cap01exem01</pre>
<pre>#[c]# %%writefile cap01exem01.c #include <stdio.h> int main(void) { printf("Hello, World!"); return 0; }</pre>	<pre>#[c]# !gcc cap01exem01.c -o output !./output</pre>
<pre>#[cpp]# %%writefile cap01exem01.cpp #include <iostream> using namespace std; int main(void) { cout << "Hello, World!" << endl; }</pre>	<pre>#[cpp]# !g++ cap01exem01.cpp -o output !./output</pre>

Method: Conversions

```
python filterNotebook.py file type format
```

file: can be a file as `all/cap1.part1.ipynb` or even the whole folder `all`;

type: can be one of the extensions `py`, `js`, `java`, `c`, `cpp`, `r` or even `all` (the six extensions). The user can also choose `type1+...+typen` as explained above, e.g. `py+js`, but they will come in alphabetical order `js+py`;

format: `html`, `slides`, `latex`, `all` or `none` (if omitted).

- For example
 - `python filterNotebook.py all all all`

Method: Example - `python filterNotebook.py all all all`

The image shows a file explorer interface with three panes. The left pane shows a directory structure with folders 'all', 'figs', and 'gen' (selected), and files 'filterNotebook.py', 'latex_begin.txt', and 'README.md'. The middle pane shows a list of files and folders, with '1.py' highlighted. The right pane shows a list of files, with 'ALL.pdf' highlighted by a red box and a yellow arrow pointing to it.

File Name	Icon
all	Folder
figs	Folder
filterNotebook.py	Python File
gen	Folder
latex_begin.txt	Text File
README.md	Markdown File
1.c	Folder
1.cpp	Folder
1.java	Folder
1.js	Folder
1.py	Folder
1.r	Folder
2.c+cpp	Folder
2.c+java	Folder
2.c+js	Folder
2.c+py	Folder
2.c+r	Folder
2.cpp+java	Folder
2.cpp+js	Folder
2.cpp+py	Folder
ALL.html	HTML File
ALL.ipynb	IPYNB File
ALL.pdf	PDF File
ALL.tex	TeX File
cap1.part1.py.html	HTML File
cap1.part1.py.ipynb	IPYNB File
cap1.part1.py.pdf	PDF File
cap1.part1.py.slides.html	HTML File
cap1.part1.py.tex	TeX File
cap1.part2.lab.py.html	HTML File
cap1.part2.lab.py.ipynb	IPYNB File
cap1.part2.lab.py.pdf	PDF File
cap1.part2.lab.py.slides.html	HTML File
cap1.part2.lab.py.tex	TeX File

Processando a Informação: um livro prático de programação independente de linguagem

Rogério Perino de Oliveira Neves

Francisco de Assis Zampirolli

EDUFABC
editora.ufabc.edu.br

Notas de Aulas inspiradas no livro

Utilizando a(s) Linguagem(ns) de Programação:

PY

Exemplos adaptados para Correção Automática no Moodle+VPL

Francisco de Assis Zampirolli

5 de setembro de 2021

Sumário

1	Processando a Informação: Cap. 1: Fundamentos	3
1.1	Instruções	3
1.2	Sumário	4
1.3	Introdução à Arquitetura de Computadores	4
1.4	Algoritmos, Fluxogramas e Lógica de Programação	5
1.4.1	Pseudocódigo	5
1.4.2	Fluxogramas	6
1.5	Conceitos de Linguagens de Programação	8
1.5.1	Níveis de Linguagens	8
1.5.2	Linguagem Compilada vs Interpretada	8
1.5.3	Estruturas de Código	9
1.5.4	Depuração de Código (DEBUG)	9
1.5.5	Ambientes de Desenvolvimento Integrados	9
1.6	Variáveis	10
1.6.1	Uso de Variáveis	10
1.7	Operadores e precedência	11
1.7.1	Operadores aritméticos	11
1.7.2	Operadores relacionais	13
1.7.3	Operadores lógicos	13
1.7.4	Operadores de atribuição	14
1.7.5	Precedência de Operadores	14
1.8	Teste de mesa	16
1.9	Aprendendo a programar	17
1.9.1	Programação sequencial	19
1.9.2	Entrada de dados	19
1.9.3	Divisão de um código em três partes	20
1.10	Exercícios	21
1.11	Revisão deste capítulo de Fundamentos	21
1.12	Processando a Informação: Cap. 1: Fundamentos - Prática 1	21
1.12.1	Instruções	22
1.12.2	Sugestões para envio dos exercícios com avaliação automática no Moodle+VPL	22
1.12.3	Exercícios	22
1.13	Processando a Informação: Cap. 1: Fundamentos - Prática 2	23
1.13.1	Exercícios	24
2	Processando a Informação: Cap. 2: Organização de código	27
2.1	Sumário	27
2.2	Revisão do capítulo anterior (Fundamentos)	27
2.3	Programas sequenciais	28
2.4	Comentários	28
2.5	Desvio de Fluxo	29
2.6	Programas e Subprogramas	29
2.7	Bibliotecas	29
2.8	Funções ou Métodos de Usuário	31
2.8.1	Tabulação	34

Method: Free Access

- The reader may use this `filterNotebook.py` and other files to simulate examples like the one just presented here.
- Such files are available at:
 - <https://github.com/fzampirolli/filterNotebook> and
 - <https://editora.ufabc.edu.br/matematica-e-ciencias-da-computacao/58-processando-a-informacao>
- Install: [python3](#), [latex](#), [nbconvert](#), [git](#)
- Get filterNotebook:
 - `git clone` <https://github.com/fzampirolli/filterNotebook> or
 - <https://github.com/fzampirolli/filterNotebook/archive/refs/heads/main.zip>

Contents

- Motivation
- Methods
 - Didactic Material in Colab
 - Conversions
 - Example
 - Free Access
 - Automatic Correction of Codes
- Experiments and Conclusions
- Future Work

Method: Automatic Correction of Codes

- Weekly Lists with MCTest+Moodle+VPL
- Individualized through parametric questions
- Paper improvement [SBIE-2020](#)
- Lists are generated in PDF and emailed to each student automatically using MCTest (vision.ufabc.edu.br)
- Evaluation material was supplied in six PLs:
 - Python, Java, JavaScript, C, C++ and R, at the student's choice.

Contents

- Motivation
- Methods
 - Didactic Material in Colab
 - Conversions
 - Example
 - Free Access
 - Automatic Correction of Codes
- Experiments and Conclusions
- Future Work

Experiments and Conclusions

- Applied in an Introductory Computer Science Course
- 223 matriculated students in five classes
- Weekly lectures (5h) synchronous or asynchronous
- Student's Feedback
 - 53 students responded to this questionnaire
- Average pass rate was 90%

Future Work

- Produce and publish more explanatory videos of the whole content of the course in many PLs
- We need to produce more didactic material compatible with the level of difficulty faced by the students during the evaluations

Using our method

- Get filterNotebook:
 - git clone <https://github.com/fzampirolli/filterNotebook> or
 - <https://github.com/fzampirolli/filterNotebook/archive/refs/heads/main.zip>

Thanks!

Questions?

{fzampirolli, fernando.teubl, guiou.kobayashi,
valerio.batista, rogerio.neves, luiz.rozante}
@ufabc.edu.br