Aprendendo e ensinando Python para "programadores alternativos"

Alexandre B A Villares

Introdução

1. De onde eu venho?
2. "*End user programmers*" e
   "Programadores(as) alternativos(as)"
3. Programação criativa e outros nomes estranhos
4. Assuntos para ensino introdutório de programação
5. Minhas ferramentas preferidas

Primeiro rápidos obrigados!

- Obrigado a organização da Python Brasil!
- Obrigado a vocês aqui!
- Obrigado Sesc-SP!

De onde eu venho?

- Comecei aprendendo BASIC e depois PASCAL, achei que nunca mais ia programar quando fui estudar arquitetura.
- Arquitetura e urbanismo na FAU-USP, 2000
- 15 anos em uma empresa de treinamento de programas de desenho para arquitetura e design (voltei a programação para fazer plugins...)
- 2016 Cursos e oficinas em vários lugares
  2018 Educador de tecnologias e artes no Sesc
  2019 Terminei um mestrado na FEC-Unicamp

"*End user programmers*" e "Programadores(as) alternativos(as)"

- Scaffidi, 2012

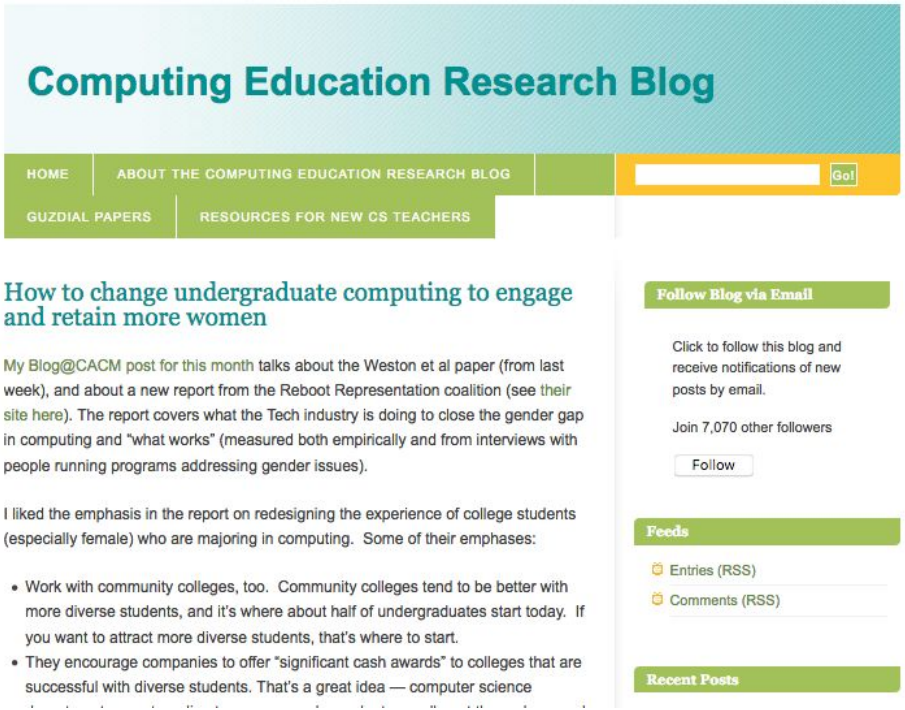
  (via Guzdial, acho)
- Schachman, 2012

  (via Patricia Oakim)

"*End user programmers*"

- Scaffidi, 2012

  (via **Guzdial**, acho)



https://computinged.wordpress.com/

"*End user programmers*"

- **Scaffidi, 2012**

  (via Guzdial, acho)

https://dl.acm.org/citation.cfm?id=2212421

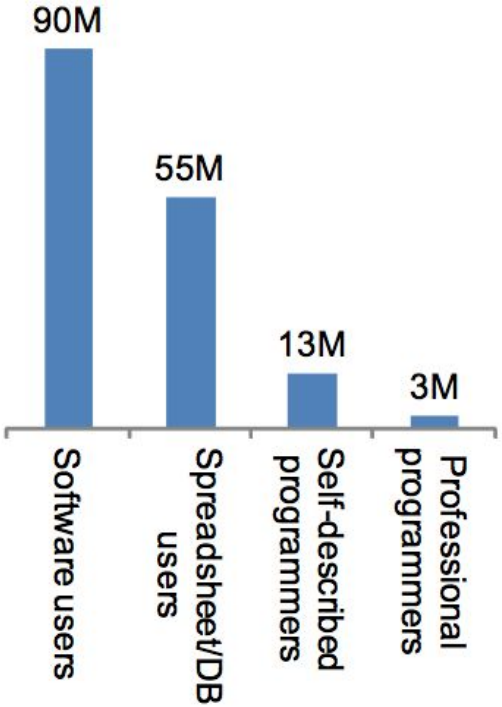


Figure 1: Estimates for the number of people in the US in 2012 who use computers at work, who use spreadsheets at work, who would describe themselves as programmers, and who would say they are professional programmers [25].

"Programadores(as) alternativos(as)"

- Schachman, 2012

  (via **Patricia Oakim**)

"Programadores(as) alternativos(as)"

- **Schachman, 2012**

  Patrícia Oakim traduz e ressalta trecho em que Schachman afirma estar surgindo uma nova geração de programadores alternativos com diferentes interesses:

- ***...esses programadores "alternativos" são pessoas que não se identificam como programadores, mas que regularmente programam computadores para alcançar seus objetivos. Programadores alternativos podem incluir músicos, performers, escritores, artistas visuais, designers, cientistas e ativistas.***

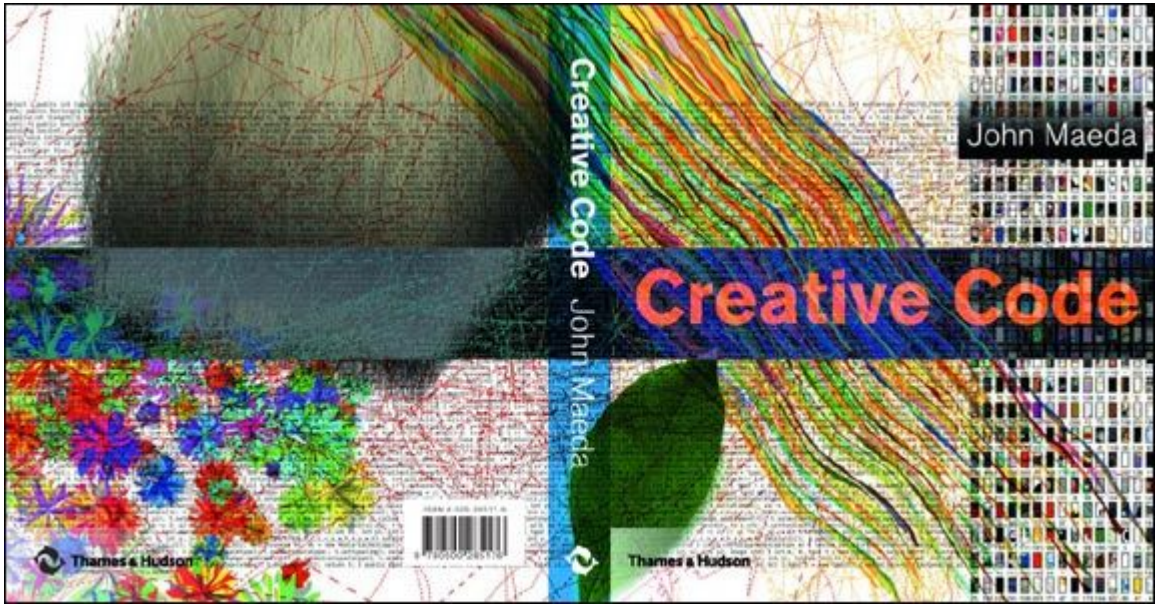
Programação criativa e outros nomes estranhos

Não sei se gosto muito desse nome (*creative coding*) afinal, quase toda programação não é criativa?

Mais nomes: *new media art*, arte computacional, arte algorítmica, arte gerativa...

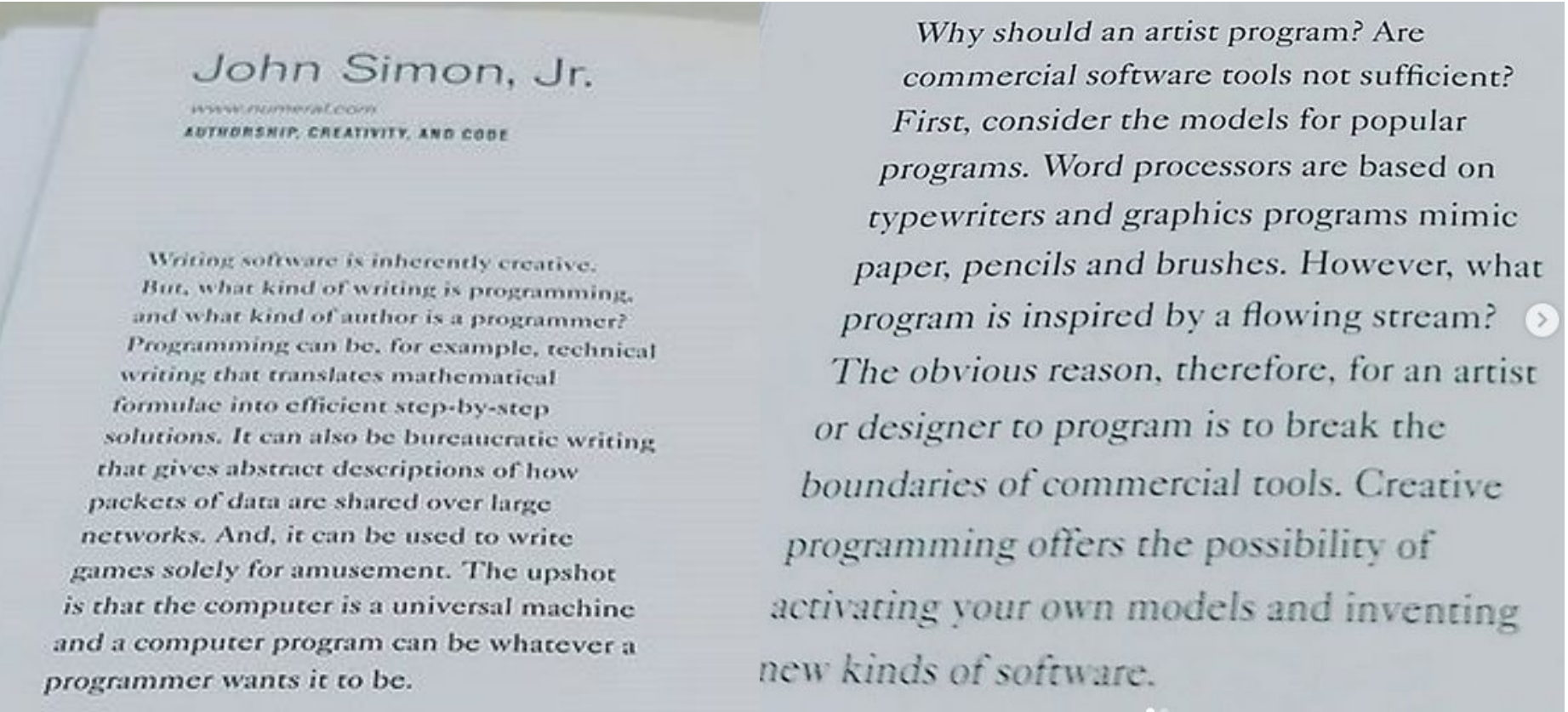
O pessoal do design costuma traduzir *generative* como generativo, eu prefiro gerativo. Tem design paramétrico (que também acho um nome complicado)

Programação criativa e arte computacional



Maeda, 2004 Creative Code: Aesthetics + Computation
http://maedastudio.com/2004/creativecode/

Programação criativa e arte computacional



John Simon, Jr.

www.numeral.com

AUTHORSHIP, CREATIVITY, AND CODE

Writing software is inherently creative. But, what kind of writing is programming, and what kind of author is a programmer? Programming can be, for example, technical writing that translates mathematical formulae into efficient step-by-step solutions. It can also be bureaucratic writing that gives abstract descriptions of how packets of data are shared over large networks. And, it can be used to write games solely for amusement. The upshot is that the computer is a universal machine and a computer program can be whatever a programmer wants it to be.

Why should an artist program? Are commercial software tools not sufficient? First, consider the models for popular programs. Word processors are based on typewriters and graphics programs mimic paper, pencils and brushes. However, what program is inspired by a flowing stream? The obvious reason, therefore, for an artist or designer to program is to break the boundaries of commercial tools. Creative programming offers the possibility of activating your own models and inventing new kinds of software.

Programação criativa e arte computacional

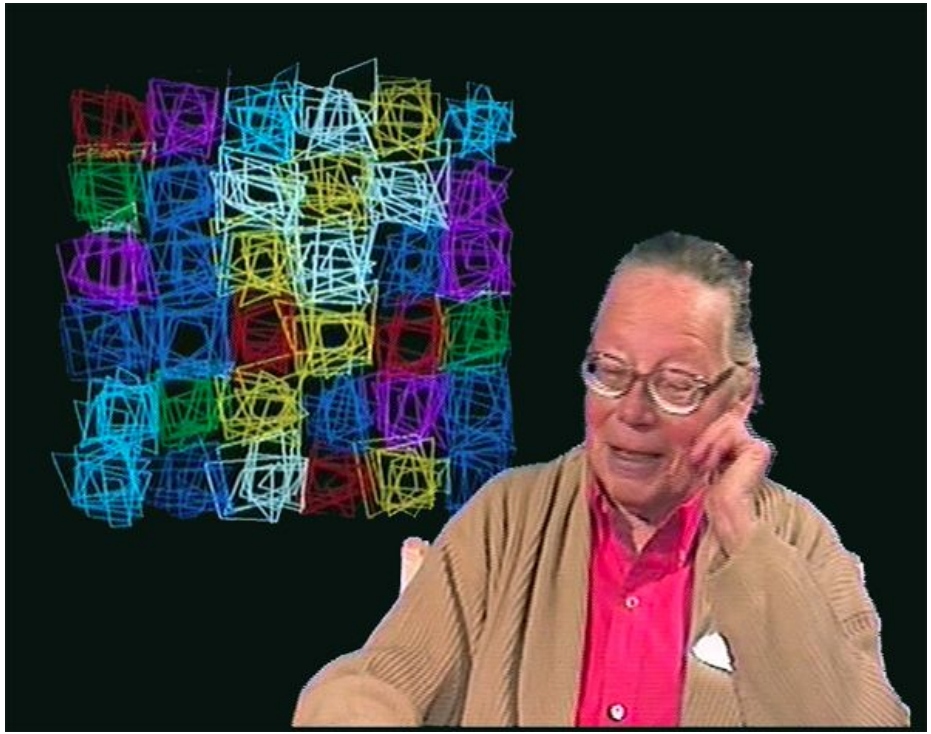
- Exemplos
  - Vera Molnár
  - Saskia Freeke
  - Alison Parish
  - Monica Rizzolli

Programação criativa e arte computacional

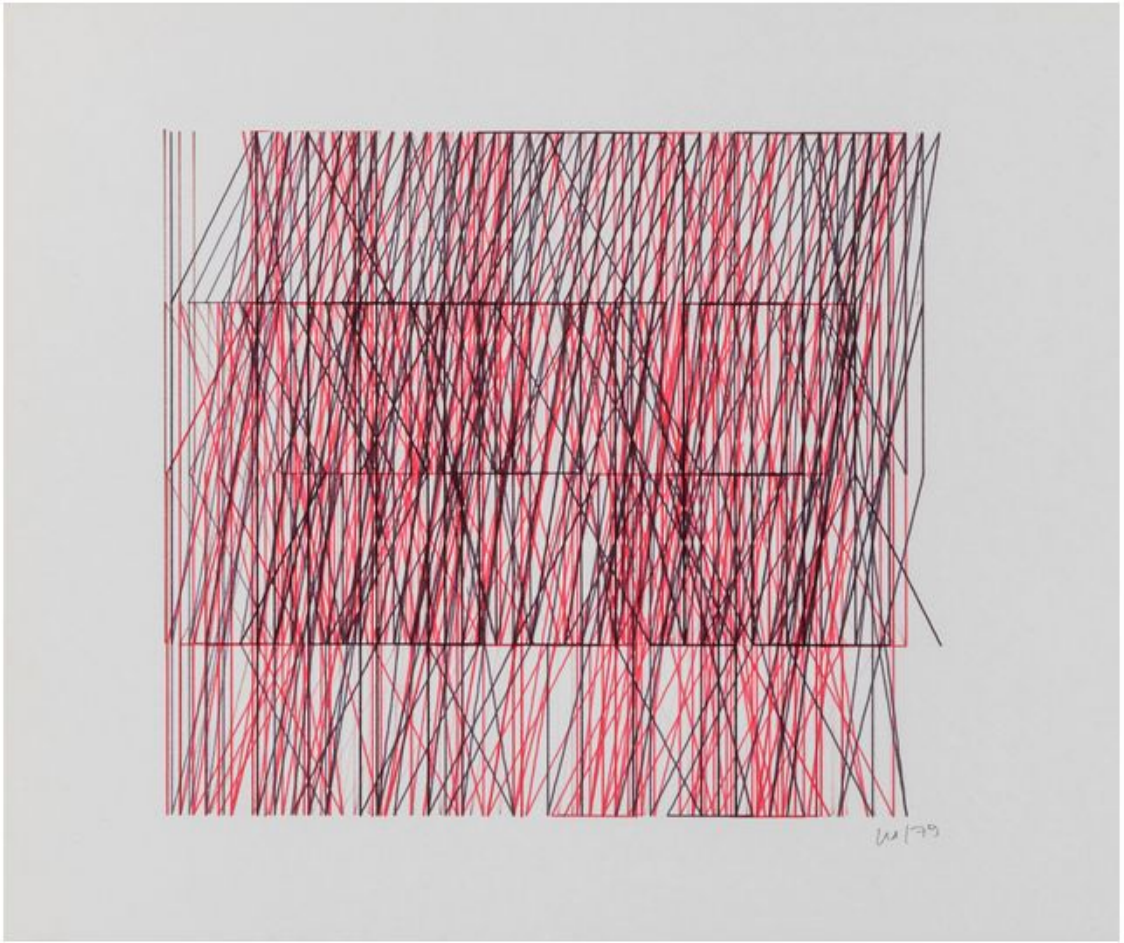
- Exemplos
  - **Vera Molnár** - em FORTRAN no final dos anos 50!
  - Saskia Freeke
  - Alison Parish
  - Monica Rizzolli

Programação criativa e arte computacional

- **Vera Molnár**



https://en.wikipedia.org/wiki/Vera_Molnár



Vera Molnar, *Tablotin 327 6*, 1979. Courtesy: Vintage Galéria, Budapest

Programação criativa e arte computacional

- Exemplos
  - Vera Molnár
  - **Saskia Freeke -** Processing / foi fellow da PF
  - Alison Parish
  - Monica Rizzolli

Programação criativa e arte computacional

- **Saskia Freeke**

https://twitter.com/sasj_nl/

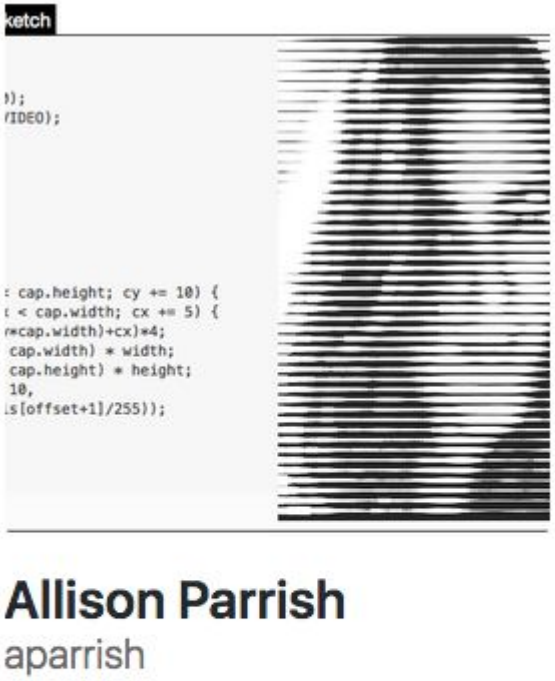
Saskia Freeke
sasj

Programação criativa e arte computacional

- Exemplos
  - Vera Molnár
  - Saskia Freeke
  - **Alison Parish** - Python, escrita algorítmica
  - Monica Rizzolli

Programação criativa e arte computacional

- **Alison Parish**



https://www.decontextualize.com/

Programação criativa e arte computacional

- Exemplos
  - Vera Molnár
  - Saskia Freeke
  - Alison Parish
  - **Monica Rizzolli** - Processing / P5js

Programação criativa e arte computacional

- **Monica Rizzolli**

Monica Rizzolli
MonicaRizzolli

www.instagram.com/monicarizzolli/

Assuntos para ensino introdutório de programação

- Luxton-Reilly et al. 2017
- Temas que eu levantei no meu mestrado

Assuntos para ensino introdutório de programação

- Luxton-Reilly et al. 2017

Tabela 2.1: Frequência de conceitos comuns em cursos introdutórios de ciência da computação (CS1)

Conceito	Freq.	Fonte
Variables & Assignment [Variáveis e atribuição]		
Variables	6	(9)(3)(5)(7)(2)(6)
Assignment	4	(9)(3)(4)(6)
Expressions	4	(9)(8)(5)(6)
Constants	3	(7)(2)(6)
Instance variables	2	(3)(2)
Scope of variables	6	(1)(3)(4)(7)(2)(6)
Lifetime	2	(7)(6)
Static & non-static variables	1	(2)
Data Types [Tipagem de dados]		
Data types	4	(8)(5)(7)(6)
Primitive data types	3	(9)(3)(4)
Integers	2	(3)(7)
Floating point	2	(3)(7)
Booleans	1	(3)
Strings	6	(8)(3)(5)(7)(2)(6)
Data Structures [Estruturas de dados]		
Arrays	6	(8)(3)(5)(4)(2)(6)
Matrices	1	(8)
Collections other than arrays	2	(7)(6)
Lists	1	(5)
Sets & relations	1	(8)
Structs & records	1	(7)
Linked list	3	(4)(7)(2)
Control Structures [Estruras de controle]		
Flow-control constructs	2	(8)(5)
Conditional control structures	5	(9)(3)(4)(2)(6)
Iterative control structures	4	(9)(7)(2)(6)
Loop (for)	1	(3)
Loop (while)	1	(3)
Loop (nested)	1	(3)
Events	2	(7)(6)
Exceptions	2	(8)(7)
Operations & Functions [Operações e funções]		
Arithmetic	2	(3)(7)
Relational operators	3	(3)(7)(6)
Logical operators	3	(3)(7)(6)
Boolean algebra	1	(8)
Functions, methods & procedures	6	(1)(9)(8)(5)(7)(6)
Function Definition	1	(3)
Parameters & param. passing	2	(3)(6)
Subroutines	2	(5)(6)
Accessor Methods	1	(3)
Mutator Methods	1	(3)
Return values	1	(3)
Static & non-static methods	1	(2)
Calling Functions		
Parameter passing	3	(3)(2)(6)
Dynamic binding	2	(1)(4)

Conceito	Freq.	Fonte
Recursion [Recursão]		
Recursion	7	(8)(3)(5)(4)(7)(2)(6)
Pointers & Memory Management [Ponteiros e gerenciam. memória]		
Pointers, references		
& memory management	2	(7)(2)
Input/Output [Entrada e saída]	2	(7)(6)
Simple input/output	5	(9)(3)(5)(7)(6)
File input/output	4	(9)(8)(7)(6)
Streams	2	(9)16
Graphical user interfaces	2	(7)(6)
Object-Oriented Concepts [Conceitos de orientação a objetos]		
Object-oriented concepts	2	(7)(6)
Objects & instances	4	(1)(3)(2)(6)
Classes	6	(1)(8)(3)(5)(2)(6)
Constructors	2	(3)(6)
Encapsulation & information hiding	4	(1)(3)(2)(6)
Message passing & object interaction	2	(1)(2)
Inheritance	5	(1)(8)(3)(5)(2)
Polymorphism	5	(1)(3)(4)(2)(6)
Interfaces (Java)	3	(8)(5)(7)
Abstract classes	1	(5)
Generics	2	(4)(2)
Instance variable types	1	(2)
Object identity	1	(6)
Libraries [Bibliotecas]		
Using language libraries	2	(7)(2)
Programming Process [Processo de Programação]		
Programming styles & standards[a]	4	(8)(5)(7)(2)
Reading code	4	(8)(5)(7)(6)
Debugging	2	(1)(2)
Design: algorithms	1	(2)
Design: classes[b]	2	(7)(2)
Design: methods[c]	2	(7)(6)
Design: programs	2	(1)(2)
Design: single class	1	(2)
Error handling	1	(2)
Testing	1	(7)
Abstract Programming Thinking		
Judgement[d]	1	(8)
Notional machine[e]	1	(8)
Problem solving strategies[f]	3	(5)(7)(6)

[a] Obedecendo a um conjunto de diretrizes ou padrões de estilo de programação.
[b] Identificando classes a partir de descrição de problemas. Schulte e Bennedsen (2) também incluem cartões CRC e "responsibility design" como categoria separada.
[c] Dadas as classes necessárias para resolver um problema, especificar os métodos requeridos.
[d] Escolher a estrutura de dados ou algoritmo adequado, e ser capaz de justificar a escolha.
[e] Modelo mental do computador.
[f] Incluído decomposição de um problema (dividir para conquistar) refinamento incremental outras estratégias de solução de problemas.

Fonte: Adaptado de Luxton-Reilly et al. (2017), p.53

Assuntos para ensino introdutório de programação

- Luxton-Reilly et al. 2017

Pode-se notar um certo grau de divergência entre as fontes, uma vez que dos 77 conceitos identificados por Luxton-Reilly et al. (2017), apresentados na Tabela 2.1, apenas 20 dos conceitos têm origem em 4 ou mais das 9 fontes. Todos os outros 57 conceitos foram relacionados em apenas uma, duas ou três fontes.

Tabela 2.1: Frequência de conceitos comuns em cursos introdutórios de ciência da computação (CS1)

Conceito	Freq.	Fonte
Variables & Assignment [Variáveis e atribuição]		
Variables	6	(9)(3)(5)(7)(2)(6)
Assignment	4	(9)(3)(4)(6)
Expressions	4	(9)(8)(5)(6)
Constants	3	(7)(2)(6)
Instance variables	2	(3)(2)
Scope of variables	6	(1)(3)(4)(7)(2)(6)
Lifetime	2	(7)(6)
Static & non-static variables	1	(2)
Data Types [Tipagem de dados]		
Data types	4	(8)(5)(7)(6)
Primitive data types	3	(9)(3)(4)
Integers	2	(3)(7)
Floating point	2	(3)(7)
Booleans	1	(3)
Strings	6	(8)(3)(5)(7)(2)(6)
Data Structures [Estruturas de dados]		
Arrays	6	(8)(3)(5)(4)(2)(6)
Matrices	1	(8)
Collections other than arrays	2	(7)(6)
Lists	1	(5)
Sets & relations	1	(8)
Structs & records	1	(7)
Linked list	3	(4)(7)(2)
Control Structures [Estruras de controle]		
Flow-control constructs	2	(8)(5)
Conditional control structures	5	(9)(3)(4)(2)(6)
Iterative control structures	4	(9)(7)(2)(6)
Loop (for)	1	(3)
Loop (while)	1	(3)
Loop (nested)	1	(3)
Events	2	(7)(6)
Exceptions	2	(8)(7)
Operations & Functions [Operações e funções]		
Arithmetic	2	(3)(7)
Relational operators	3	(3)(7)(6)
Logical operators	3	(3)(7)(6)
Boolean algebra	1	(8)
Functions, methods & procedures	6	(1)(9)(8)(5)(7)(6)
Function Definition	1	(3)
Parameters & param. passing	2	(3)(6)
Subroutines	2	(5)(6)
Accessor Methods	1	(3)
Mutator Methods	1	(3)
Return values	1	(3)
Static & non-static methods	1	(2)
Calling Functions		
Parameter passing	3	(3)(2)(6)
Dynamic binding	2	(1)(4)
Recursion [Recursão]		
Recursion	7	(8)(3)(5)(4)(7)(2)(6)
Pointers & Memory Management [Ponteiros e gerenciam. de memória]		
Pointers, references		
& memory management	2	(7)(2)
Input/Output [Entrada e saída]	2	(7)(6)
Simple input/output	5	(9)(3)(5)(7)(6)
File input/output	4	(9)(8)(7)(6)
Streams	2	(9)16
Graphical user interfaces	2	(7)(6)
Object-Oriented Concepts [Conceitos de orientação a objetos]		
Object-oriented concepts	2	(7)(6)
Objects & instances	4	(1)(3)(2)(6)
Classes	6	(1)(8)(3)(5)(2)(6)
Constructors	2	(3)(6)
Encapsulation & information hiding	4	(1)(3)(2)(6)
Message passing & object interaction	2	(1)(2)
Inheritance	5	(1)(8)(3)(5)(2)
Polymorphism	5	(1)(3)(4)(2)(6)
Interfaces (Java)	3	(8)(5)(7)
Abstract classes	1	(5)
Generics	2	(4)(2)
Instance variable types	1	(2)
Object identity	1	(6)
Libraries [Bibliotecas]		
Using language libraries	2	(7)(2)
Programming Process [Processo de Programação]		
Programming styles & standards **a**	4	(8)(5)(7)(2)
Reading code	4	(8)(5)(7)(6)
Debugging	2	(1)(2)
Design: algorithms	1	(2)
Design: classes **b**	2	(7)(2)
Design: methods **c**	2	(7)(6)
Design: programs	2	(1)(2)
Design: single class	1	(2)
Error handling	1	(2)
Testing	1	(7)
Abstract Programming Thinking		
Judgement **d**	1	(8)
Notional machine **e**	1	(8)
Problem solving strategies **f**	3	(5)(7)(6)

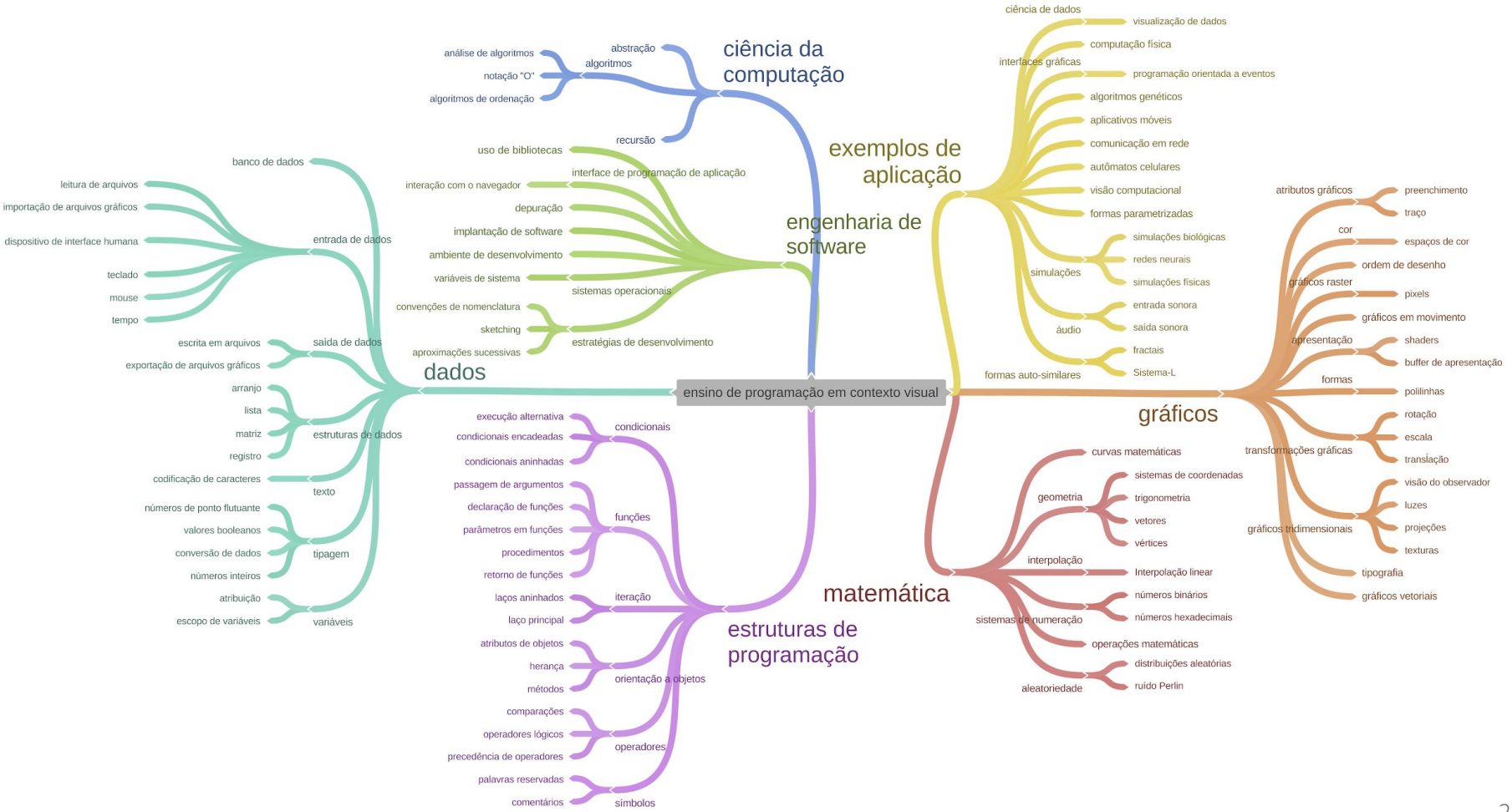
a Obedecendo a um conjunto de diretrizes ou padrões de estilo de programação.
b Identificando classes a partir de descrição de problemas. Schulte e Bennedsen (2) também incluem cartões CRC e "responsibility design" como categoria separada.
c Dadas as classes necessárias para resolver um problema, especificar os métodos requeridos.
d Escolher a estrutura de dados ou algoritmo adequado, e ser capaz de justificar a escolha.
e Modelo mental do computador.
f Incluído decomposição de um problema (dividir para conquistar) refinamento incremental outras estratégias de solução de problemas.

Fonte: Adaptado de Luxton-Reilly et al. (2017), p.53

Assuntos para ensino introdutório de programação

- Temas que eu levantei no meu mestrado

https://abav.lugaralgum.com/mestrado


ensino de programação em contexto visual
ciência da computação
algoritmos
abstração
análise de algoritmos
notação "O"
algoritmos de ordenação
recursão
engenharia de software
interface de programação de aplicação
uso de bibliotecas
interação com o navegador
depuração
implantação de software
ambiente de desenvolvimento
variáveis de sistema
sistemas operacionais
estratégias de desenvolvimento
convenções de nomenclatura
sketching
aproximações sucessivas
dados
entrada de dados
banco de dados
leitura de arquivos
importação de arquivos gráficos
dispositivo de interface humana
teclado
mouse
tempo
saída de dados
escrita em arquivos
exportação de arquivos gráficos
estruturas de dados
arranjo
lista
matriz
registro
texto
codificação de caracteres
tipagem
números de ponto flutuante
valores booleanos
conversão de dados
números inteiros
variáveis
atribuição
escopo de variáveis
estruturas de programação
condicionais
execução alternativa
condicionais encadeadas
condicionais aninhadas
funções
passagem de argumentos
declaração de funções
parâmetros em funções
procedimentos
retorno de funções
iteração
laços aninhados
laço principal
orientação a objetos
atributos de objetos
herança
métodos
operadores
comparações
operadores lógicos
precedência de operadores
símbolos
palavras reservadas
comentários
exemplos de aplicação
ciência de dados
visualização de dados
computação física
interfaces gráficas
programação orientada a eventos
algoritmos genéticos
aplicativos móveis
comunicação em rede
autômatos celulares
visão computacional
formas parametrizadas
simulações
simulações biológicas
redes neurais
simulações físicas
áudio
entrada sonora
saída sonora
formas auto-similares
fractais
Sistema-L
matemática
curvas matemáticas
geometria
sistemas de coordenadas
trigonometria
vetores
vértices
interpolação
Interpolação linear
sistemas de numeração
números binários
números hexadecimais
operações matemáticas
aleatoriedade
distribuições aleatórias
ruído Perlin
gráficos
atributos gráficos
preenchimento
traço
cor
espaços de cor
ordem de desenho
gráficos raster
pixels
gráficos em movimento
apresentação
shaders
buffer de apresentação
formas
polilinhas
transformações gráficas
rotação
escala
translação
gráficos tridimensionais
visão do observador
luzes
projeções
texturas
tipografia
gráficos vetoriais

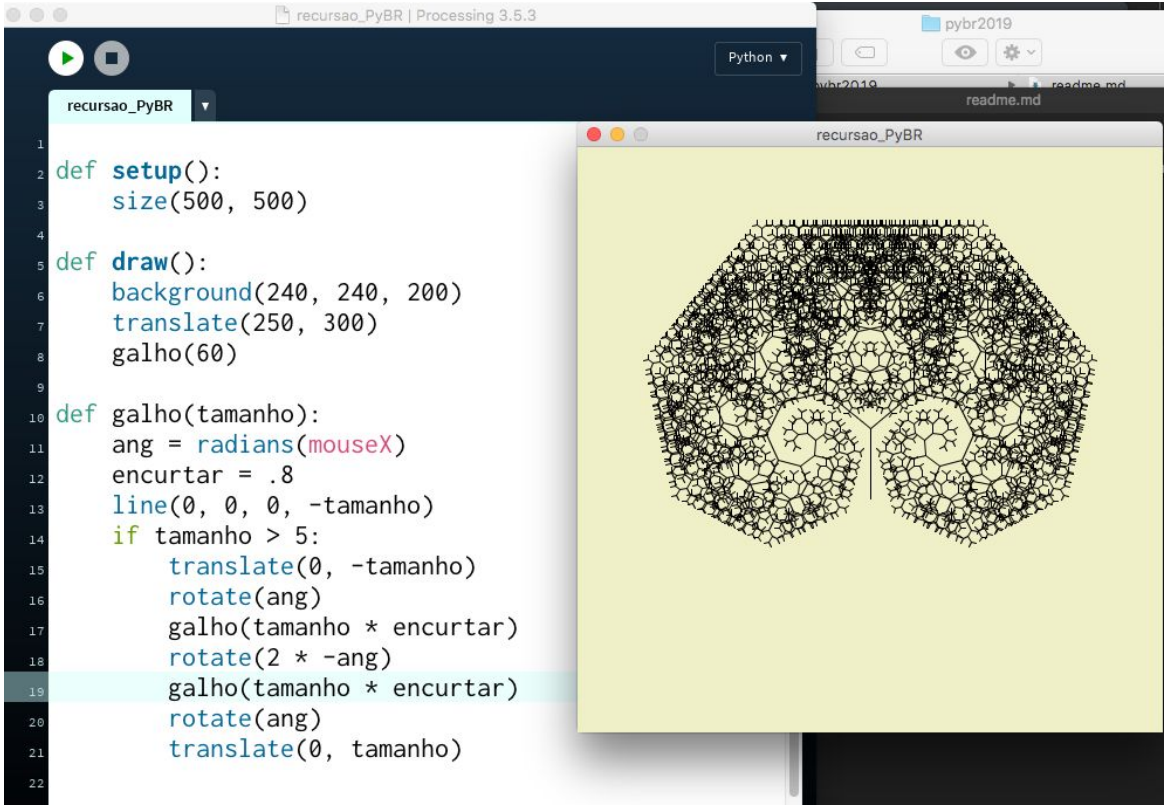
Minhas ferramentas preferidas

Processing + Python

https://github.com/villares/Resources-for-teaching-programming/

Nome	Recursos Processing	baseado em (& versão Python)	Python standard library	ecossistema de bibliotecas	features principais	limitações principais
Processing Python Mode	Processing Java	Jython (Python 2)	completa	Java & Processing Java	disponível dentro do Processing IDE, muito compatível com Processing	não tem como compartilhar na web, não tem bibliotecas de Python moderno.
SkulptIDE and trinket.io	ProcessingJS	Skulpt (Python 2)	parcial	não sei, possivelmente JavaScript	IDE web muito legal, permite sketches no browser	ProcessingJS está abandonado; não é extensível.
BrythonIDE	p5.js	Brython (Python 3)	bastante completa	JavaScript & p5.js	IDE web, permite sketches no browser, muito compatível com p5.js	download grande, pode ficar lento
p5py	nova inplementação (incompleta)	Python 3	completa	Python apenas	totalmente em Python	não tem com compartilhar na web, API bem diferente do Processing tradicional, ainda está bem incompleto
pyp5js	p5.js	Transcrypt (Python 3)	incompleta	JavaScript & p5.js	permite sketches no browser, muito compatível com p5.js	apenas com bibliotecas JS, command line interface
(futuro pyp5js)	p5.js	Pyodide (Python 3)	completa	Python, JavaScript & p5.js	permite sketches no browser, muito compatível com p5.js e com Python 3	(não está pronto)

Processing Modo Python



E agora?

- Google Groups Processing-Brasil
- Me procure estes dias para conversar!

Aprendendo e ensinando Python para "programadores alternativos"

Obrigado!

Alexandre B A Villares

https://abav.lugaralgum.com
https://twitter.com/villares