

# **Web Python e Processing**

## **Uma experiência no ensino de programação**

Claudio Esperança – [esperanc@cos.ufrj.br](mailto:esperanc@cos.ufrj.br)

Programa de Engenharia de Sistemas e Computação / UFRJ

18 de Janeiro de 2019

# Histórico

- ▶ 2004 - Curso de Eng. Computação e Informação da UFRJ
- ▶ Disciplina de 1º período: *Algoritmos e Programação*
- ▶ 2005 → LISP
- ▶ 2006 → LOGO
- ▶ 2007-hoje → Python

# Características da disciplina

- ▶ Discentes = futuros profissionais de computação
- ▶ Ementa: de expressões a estruturas de dados e programação OO
- ▶ 2 aulas expositivas e 1 de laboratório por semana
- ▶ 90 horas

# Programação + gráficos

- ▶ Ideia introduzida com LOGO (1967)
- ▶ Scratch, Blockly, Alice ...
- ▶ Apelo visual alivia aridez dos problemas numéricos
- ▶ Conceitos de animação e interação

# Python + Gráficos

- ▶ turtle, graphics (bibliotecas Python)
  - ▶ Muito restritivos
  - ▶ Sem modelo de interação

# Python + Gráficos

- ▶ turtle, graphics (bibliotecas Python)
  - ▶ Muito restritivos
  - ▶ Sem modelo de interação
- ▶ Pygame, PyOpenGL, outros
  - ▶ Muito complicados

# Python + Gráficos

- ▶ turtle, graphics (bibliotecas Python)
  - ▶ Muito restritivos
  - ▶ Sem modelo de interação
- ▶ Pygame, PyOpenGL, outros
  - ▶ Muito complicados
- ▶ Processing
  - ▶ Abstração gráfica simples e poderosa
  - ▶ Python?

# Python + Processing

- ▶ `pyprocessing`
  - ▶ Biblioteca baseada em OpenGL
  - ▶ Instalação não trivial
  - ▶ Projeto defunto



# Python + Processing

- ▶ [pyprocessing](#)
  - ▶ Biblioteca baseada em OpenGL
  - ▶ Instalação não trivial
  - ▶ Projeto defunto
- ▶ [processing.py](#) (Processing modo Python)
  - ▶ Implementação de referência
  - ▶ Baseado em Jython (máquina virtual Java)
  - ▶ Relativamente simples de instalar
  - ▶ Infelizmente, Python 2.7

# Python + Processing

- ▶ [pyprocessing](#)
  - ▶ Biblioteca baseada em OpenGL
  - ▶ Instalação não trivial
  - ▶ Projeto defunto
- ▶ [processing.py](#) (Processing modo Python)
  - ▶ Implementação de referência
  - ▶ Baseado em Jython (máquina virtual Java)
  - ▶ Relativamente simples de instalar
  - ▶ Infelizmente, Python 2.7
- ▶ Recentemente (2017) [p5 for python](#)
  - ▶ Biblioteca para CPython
  - ▶ Ainda incipiente

# Python no navegador

- ▶ Plataforma “instantânea”
  - ▶ Nada a baixar, instalar ou atualizar
- ▶ Ambiente familiar para o iniciante
- ▶ Multiplataforma
- ▶ Facilmente integrável com material didático
- ▶ Aplicações portáteis

# IPython / Jupyter

- ▶ <https://ipython.org>
  - ▶ Python notebooks
  - ▶ Server side
  - ▶ Excelente interface para aprendizado
  - ▶ Requer servidor
  - ▶ Pouco adequado para programação de gráficos interativos

# Web Python

- ▶ Python no navegador  $\leftrightarrow$  “transpilado” para JS
  - ▶ <https://github.com/jashkenas/coffeescript/wiki/list-of-languages-that-compile-to-js>

# Web Python

- ▶ Python no navegador ↔ “transpilado” para JS
  - ▶ <https://github.com/jashkenas/coffeescript/wiki/list-of-languages-that-compile-to-js>
- ▶ Skulpt
  - ▶ Python 2.7 (mais ou menos)
  - ▶ Inclui uma biblioteca Processing incipiente baseada em [processing.js](#)
  - ▶ Parece abandonado...

# Web Python

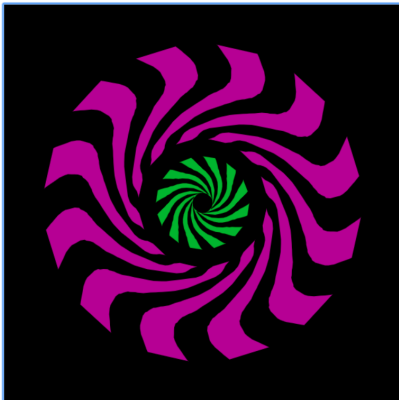
- ▶ Python no navegador ↔ “transpilado” para JS
  - ▶ <https://github.com/jashkenas/coffeescript/wiki/list-of-languages-that-compile-to-js>
- ▶ Skulpt
  - ▶ Python 2.7 (mais ou menos)
  - ▶ Inclui uma biblioteca Processing incipiente baseada em [processing.js](#)
  - ▶ Parece abandonado...
- ▶ Brython
  - ▶ Python 3.7
  - ▶ Implementação bastante madura
  - ▶ Fácil integração com bibliotecas JavaScript!

# SkulptIde

## Online Python development environment

See [Github site](#) for more info.

New Run Stop Clear Upload Download as mandala2.py



```
--
19 drawing = []
20
21 def setup():
22     size(600,600)
23     colorMode(HSB)
24     noStroke()
25
26 def mousePressed():
27     if mouse.button == RIGHT:
28         drawing[:] = []
29         colors[:] = []
30     elif mouse.button == CENTER:
31         if drawing != []:
32             drawing.pop()
33             colors.pop()
34     else:
35         colors.append( color(randint(0,255),255,255,alpha))
36         drawing.append([mouse.x,mouse.y])
37
38 def mouseDragged():
39     if mouse.button == LEFT:
40         ...
```

<https://esperanc.github.io/skulptIde/?program=demoSketches/mandala2.py>



# SkulptIde

- ▶ <https://esperanc.github.io/skulptIde/>
- ▶ Editor online + skulpt + processing.js
- ▶ Binding Processing-Python ligeiramente alterado
  - ▶ `mouse.x` ao invés de `mouseX`
- ▶ Suporte limitado para
  - ▶ Fontes
  - ▶ Imagens
  - ▶ 3D

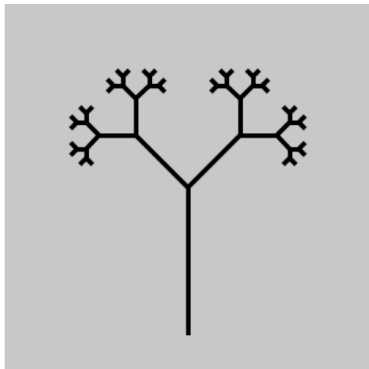
# SkulptIde - Exemplos de exercícios

## Árvore recursiva

### Online Python development environment

See [Github site](#) for more info.

New Run Stop Clear Upload Download as alignGrid.py



```
1 from processing import *
2 from math import *
3
4 w,h = 400,400
5
6 def arvore(x0,y0,tam,ang,n):
7     u"""Desenha uma arvore recursiva de nivel n com raiz
8     em (x0,y0), tamanho tam, crescendo com angulo ang"""
9     if n <= 1:
10         x1,y1 = x0+tam*cos(ang),y0+tam*sin(ang)
11         line (x0,y0,x1,y1)
12     else:
13         tam1 = tam*0.5
14         x1,y1 = x0+tam1*cos(ang),y0+tam1*sin(ang)
15         line (x0,y0,x1,y1)
16         arvore (x1,y1,tam1,ang-pi/4,n-1)
17         arvore (x1,y1,tam1,ang+pi/4,n-1)
18
19 def setup():
20     size(w,h)
21     strokeCap(SQUARE)
22     strokeWeight(5)
```

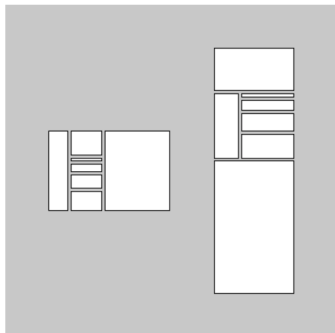
# SkulptIde - Exemplos de exercícios

## Treemap

### Online Python development environment

See [Github site](#) for more info.

New Run Stop Clear Upload Download as alignGrid.py



```
16
17
18 def retanguloDividido(x0,y0,dx,dy,l):
19     u"""Desenha um retangulo com canto superior esquerdo em x0,y0
20     largura dx e altura dy subdividido ao longo de sua maior dimensão
21     em trechos com comprimento proporcional aos valores em l"""
22     if type(l)!=list:
23         retangulo(x0+2,y0+2,dx-4,dy-4)
24         return
25     total = somaRecursiva(l)
26     if dx>dy:
27         for v in l:
28             dxv = 1.0*somaRecursiva(v)/total*dx
29             retanguloDividido(x0,y0,dxv,dy,v)
30             x0 += dxv
31     else:
32         for v in l:
33             dyv = 1.0*somaRecursiva(v)/total*dy
34             retanguloDividido(x0,y0,dx,dyv,v)
35             y0 += dyv
36
37 def setup():
38     size(400,400)
```

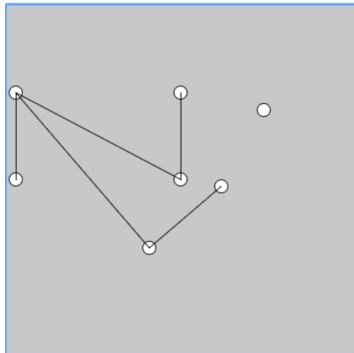
# SkulptIde - Exemplos de exercícios

## Editor de grafos

### Online Python development environment

See [Github site](#) for more info.

saveprogram.py



```
1 vertices = [0, 1, 2, 3]
2 posicao = {}
3 posicao [0] = [10,100]
4 posicao [1] = [10,200]
5 posicao [2] = [200,200]
6 posicao [3] = [200,100]
7 arestas = {}
8 arestas [0,1] = 1
9 arestas [0,2] = 1
10 sel = -1 # vertice selecionado
11
12 from processing import *
13
14 def setup():
15     size(400,400)
16
17 def achaVertice(x,y):
18     "retorna o vertice na posicao x,y. Se nao houver, retorna -1"
19     for p in posicao:
20         pos = posicao[p]
21         xp,yp = pos
22         d = ((x-xp)**2+(y-yp)**2)**0.5
```

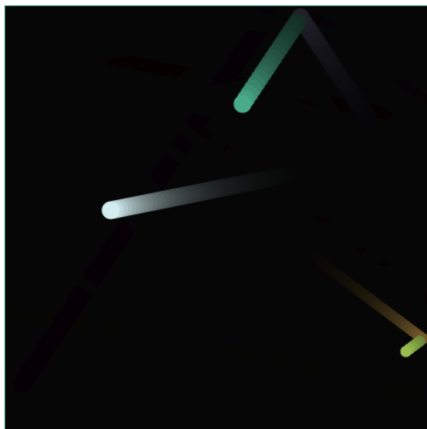
# SkulptIde - Exemplos de exercícios

## Partículas

Online Python development environment

See [Github site](#) for more info.

New Run Stop Clear Upload Download as saveprogram.py



```
1 from processing import *
2
3 width,height = 500,500
4
5 class Particula(object):
6     def __init__(self, x, y, r=10,vx=1,vy=1):
7         self.x = x
8         self.y = y
9         self.r = r
10        self.vx = vx
11        self.vy = vy
12        self.c = color(255,0,0)
13
14    def move(self):
15
16        if self.x > width-self.r or self.x < self.r:
17            self.vx = -self.vx
18            self.c = color(random(255),random(255),rar
19        self.x += self.vx
20
21        if self.y > height-self.r or self.y < self.r:
22            self.vy = -self.vy
```

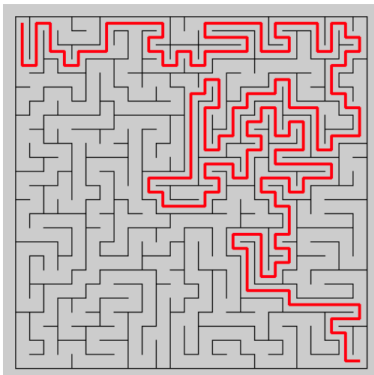
# SkulptIde - Exemplos de exercícios

## Labirinto

### Online Python development environment

See [Github site](#) for more info.

New Run Stop Clear Upload Download as alignGrid.py



```
1 from processing import *
2 from random import choice
3
4 class Labirinto:
5
6     def __init__(self,n,m):
7         "Labirinto com n linhas e m colunas"
8         self.sala = {}
9         self.parede = {}
10        for i in range(n):
11            for j in range(m):
12                self.sala[i,j]=False
13                for p in [(i+0.5,j),(i-0.5,j),(i,j-0.5)]:
14                    self.parede[p]=True
15
16        def vizinhosNaoMarcados(self,i,j):
17            u"Retorna lista com posições de salas ainda nã
18            return [p for p in [(i+1,j),(i-1,j),(i,j-1),(i,
19
20        def marca(self,i,j):
21            u"Marca sala i,j como visitada"
22            self.sala[i,j] = True
```

# SkulptIde - Experiência

- ▶ Módulo processing explicado em 1 aula expositiva
- ▶ Pequenos programas desenvolvidos nas aulas de laboratórios
- ▶ Excelente motivador para funções recursivas
- ▶ Projetos de jogos simples são populares
- ▶ Exercícios com problemas geométricos (leiaute)
- ▶ Alguns alunos não “curtem” computação gráfica

## Python 2.7 will retire in...

0	11	16	2	26	43
Years	Months	Days	Hours	Minutes	Seconds

[Create Quick Mock](#) [Join?](#)

### What's all this, then?

Python 2.7 [will not be maintained past 2020](#). Originally, there was no official date. Recently, that date has been updated to [January 1, 2020](#). This clock has been updated accordingly. My original idea was to throw a Python 2 Celebration of Life party at PyCon 2020, to celebrate everything Python 2 did for us. That idea still stands. (If this sounds interesting to you, email [pythonclockorg@gmail.com](mailto:pythonclockorg@gmail.com)).

Python 2, thank you for your years of faithful service.

Python 3, your time is now.

### How do I get started?

If the code you care about is still on Python 2, that's totally understandable. Most of PyPI's popular packages now [work on Python 2 and 3](#), and more are being added every day. Additionally, a number of critical Python projects have [pledged to stop supporting Python 2 soon](#). To ease the transition, the [official porting guide](#) has advice for running Python 2 code in Python 3.

<https://pythonclock.org/>



# Brython + p5

- ▶ Brython = Browser Python
- ▶ Fácil integração com bibliotecas JS
- ▶ Importar p5 é um caminho natural
- ▶ Prova de conceito Brython + p5 por Alexandre Villares
  - ▶ [Post no forum Processing](#)

# Brython + p5

```
<script src="//cdn.jsdelivr.net/npm/p5.js@0.5.0/p5.js"></script>
<script type="text/python">
# brython + p5js.org adapted from code from Kiko Correoso
# with correction from Pierre Quentel
from browser import document, window, alert

def sketch(p):

    def setup():
        p.createCanvas(700, 410)
        p.background(0)
        p.rectMode(p.CENTER)

    def draw():
        # p.background(0)
        p.fill(255,255,0,128)
        p.ellipse(p.mouseX,p.mouseY,50,50)

    def mousePressed():
        p.background(0)

    p.setup = setup
    p.draw = draw
    p.mousePressed = mousePressed

myp5 = window.p5.new(sketch)
</script>
```

jsfiddle

## p5 modo global

- ▶ Prova de conceito usa p5 em modo 'instância'
- ▶ Menos amigável que modo global
- ▶ p5 em modo global requer
  - ▶ Nomes definidos na instância levados para escopo global
  - ▶ Funções callback globais registrados na instância
  - ▶ Gerência de variáveis Python globais espelhando variáveis JS análogas (mouseX, mouseY, key, etc)
- ▶ Solução: módulo p5py

## Exemplo p5py

```
from p5py import run

def setup():
    createCanvas(700, 410)
    background(0)
    rectMode(CENTER)

def draw():
    fill(255,255,0,128)
    ellipse(mouseX,mouseY,50,50)

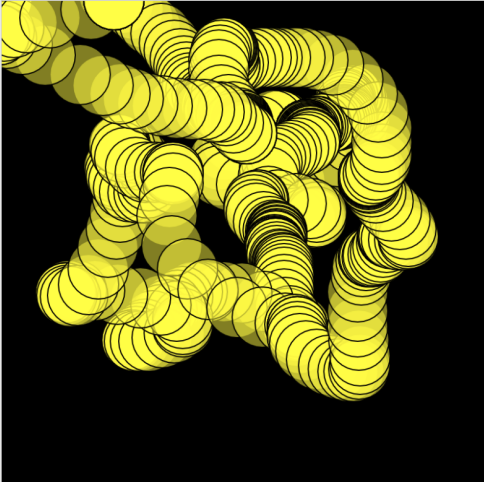
def mousePressed():
    background(0)

run()
```

edit with BrythonIDE

# Python/Processing Editor

New Run Clear Stop Console Upload Download as



```
1 from p5py import run
2
3 def setup():
4     createCanvas(400, 400)
5     background(0)
6     rectMode(CENTER)
7
8 def draw():
9     fill(255,255,0,128)
10    ellipse(mouseX,mouseY,50,50)
11
12 def mousePressed():
13     background(0)
14
15 run()
```

<https://esperanc.github.io/brythonide>

# BrythonIDE

- ▶ Ambiente integrado Brython + p5
- ▶ Facilita a edição online de sketches com Python
- ▶ Console Python incluído
- ▶ Pode-se também usar outros editores, como o [P5 Web Editor](#)
- ▶ Não tem suporte server side para armazenar sketches
  - ▶ Suporta carregar e salvar arquivos
  - ▶ Pequenos sketches para demo podem ser carregados via parâmetro URL, ex.: <https://esperanc.github.io/brythonide/?lsrc=GYJw9gtgBADgrDAnlAlhGYQBcoCoBQ+AJgKbBQDOJWArjABQCUAXPl01AMYgkCGWJAMK8AdgDdeFegDYALAAYANLIAC8x1lQAmop+VEFRJGEa+CA0IkxAA>
  - ▶ Use o parâmetro showurl para gerar link <https://esperanc.github.io/brythonide/?showurl>

# Brython + p5 - limitações

- ▶ Integração completamente funcional
- ▶ Problemas de performance
  - ▶ Muito mais lento que JS + p5
  - ▶ Mais lento que Skulpt + Processing.js
- ▶ Superposição de funções (*map*, *min*, *max*, etc)
- ▶ Material didático escasso
  - ▶ Exemplos processing.py ou p5 precisam ser adaptados
    - ▶ Alguns em <https://esperanc.github.io/brythonide/examples.html>
  - ▶ Tutorial sendo preparado para volta às aulas

# Reflexões finais

- ▶ Ensinar programação de computadores depende muito do público



## Reflexões finais

- ▶ Ensinar programação de computadores depende muito do público
- ▶ A linguagem usada deve ser escolhida com cuidado

## Reflexões finais

- ▶ Ensinar programação de computadores depende muito do público
- ▶ A linguagem usada deve ser escolhida com cuidado
- ▶ Uso de exemplos e problemas gráficos ajuda a motivar em geral, mas nem sempre

# Reflexões finais

- ▶ Ensinar programação de computadores depende muito do público
- ▶ A linguagem usada deve ser escolhida com cuidado
- ▶ Uso de exemplos e problemas gráficos ajuda a motivar em geral, mas nem sempre
- ▶ Python se torna crescentemente popular como 1a linguagem
  - ▶ Python 3 é sine qua non
  - ▶ Outras combinações Python+Processing precisam ser investigadas (pypy + p5?)

# Reflexões finais

- ▶ Ensinar programação de computadores depende muito do público
- ▶ A linguagem usada deve ser escolhida com cuidado
- ▶ Uso de exemplos e problemas gráficos ajuda a motivar em geral, mas nem sempre
- ▶ Python se torna crescentemente popular como 1a linguagem
  - ▶ Python 3 é sine qua non
  - ▶ Outras combinações Python+Processing precisam ser investigadas (pypy + p5?)
- ▶ Python no navegador ainda não maduro

# Reflexões finais

- ▶ Ensinar programação de computadores depende muito do público
- ▶ A linguagem usada deve ser escolhida com cuidado
- ▶ Uso de exemplos e problemas gráficos ajuda a motivar em geral, mas nem sempre
- ▶ Python se torna crescentemente popular como 1a linguagem
  - ▶ Python 3 é sine qua non
  - ▶ Outras combinações Python+Processing precisam ser investigadas (pypy + p5?)
- ▶ Python no navegador ainda não maduro
- ▶ JS pode tomar o lugar do Python como 1a linguagem?

**Obrigado!**