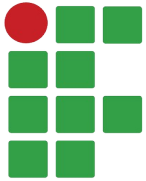
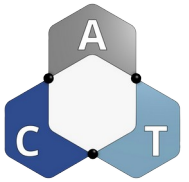


INSTITUTO FEDERAL
Rio Grande do Sul

Campus
Rio Grande



Python para Automação
Fundamentos

Prof. Carlos R Rocha
carlos.rocha@riogrande.ifrs.edu.br

ACT – Automação, Conectividade e Tecnologia

Roteiro

- ~~Apresentação do ecossistema Python~~
- ~~Primeiros passos: conhecendo elementos~~
- ~~Criação de scripts~~
- Modularização e componentização
- Estudo de caso
- E agora?

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Por quê modularizar?

- Ações podem se repetir para dados diferentes
- Códigos *monolíticos* são difíceis de depurar
- Reutilização uniforme de código que funciona
- Dividir para conquistar
- Passos de bebê: um problema simples por vez

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Por quê conhecer modularização?

- Para saber criar e utilizar bibliotecas
- Saber organizar o software
- Facilitar análises e testes de código
- Trabalho intelectual ao invés de braçal
- KISS

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Funções

- Blocos de código parametrizáveis
- Definição: cria o bloco de código
- Chamada: executa o bloco de código
- 1 definição → múltiplas chamadas
- Observar parâmetros e retornos

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Exemplo: Equação de 2º grau

```
def eq2Grau (a2, a1, a0, x):
    y = a2*x**2 + a1*x + a0
    return y
```

Parâmetros: a2, a1, a0, x

Corpo da função

Retorna o resultado

Define a função

```
[ ] import numpy as np
    import matplotlib.pyplot as plt

    x = np.linspace(-10,10, 101) # Cria um array de abcissas

    y1 = eq2Grau (1, -4.2, 18, x) # Obtém as abcissas da equação 1
    y2 = eq2Grau (-2, 18.4, -7, x) # Obtém as abcissas da equação 2
```

Chama a função para a2=1, a1=-4.2, a0=18, x definido

Chama a função para a2=-2, a1=28.4, a0=-7, x definido

INSTITUTO FEDERAL Rio Grande do Sul Campus Rio Grande

Modularização

Exemplo: Equação de 2º grau

```
[3]  def calcularRaizes(a2, a1, a0):
         if a2 == 0:
             print ("É uma equação de primeir
             return (-a0/a1, None)
         else:
           print ("É uma equação de 2o grau")
```

Tomada de decisão:
se expressão lógica for verdadeira … (if)
se não for verdadeira… (else)

INDENTAÇÃO É FUNDAMENTAL!!!

```
calcularRaizes(0, -5, 3)

É uma equação de primeiro grau
(0.6, None)
```

Resultado para teste verdadeiro:
Tupla com 2 valores (raiz e None)

```
[6]  calcularRaizes(1, -5, 3)

     É uma equação de 2o grau
```

Resultado para teste falso:
No próximo episódio

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Exemplo: Equação de 2º grau

```
[10] def calcularRaizes(a2, a1, a0):
         if a2 == 0:
             return (-a0/a1, None)
         else:
             delta = a1**2 - 4*a2*a0
             if delta == 0:
                 x = -a1/(2*a2)
                 return (x, x)
             else:
                 x1 = (-a1 + delta**0.5) / (2*a2)
                 x2 = (-a1 - delta**0.5) / (2*a2)
                 return (x1, x2)
```

Estrutura de controle aninhada:
Este if será executado como parte
Do else de nível superior a ele

Retorna tupla com 2 valores
Idênticos (idêntico ≠ igual)

Resultado tupla com
2 valores distintos

```
calcularRaizes(0, -5, 3)
```

```
calcularRaizes(1, -5, 3)
```

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Exemplo: Equação de 2º grau

- Problema com números reais: A == B
- Computadores tem problema com infinit...
- Dois cálculos distintos podem chegar a valores muito próximos, mas não iguais
- Comparações por tolerância

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

```python
def ehZero(x, tolerancia=1e-6):
    """ Recebe um valor e uma tolerância (por padrão, 1x10^-6)
        Retorna True se o absoluto de x for menor que essa tolerância
        O que significa que dá para considerá-lo igual a zero
    """
    return abs(x) < tolerancia
```

Parâmetro default: opcional

Docstring: documenta a função

Retorna o valor sem sinal do argumento

```python
def calcularRaizes(a2, a1, a0):
    if ehZero(a2):
        return (-a0/a1, None)
    else:
        delta = a1**2 - 4*a2*a0
        if ehZero(delta):
            x = -a1/(2*a2)
            return (x, x)
        else:
            x1 = (-a1 + delta**0.5) / (2*a2)
            x2 = (-a1 - delta**0.5) / (2*a2)
            return (x1, x2)
```

Chamando ehZero() em calcularRaizes()

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Modularizando a aplicação

```
a, b, c = lerCoeficientes()

x1, x2  = calcularRaizes(a, b, c)

apresentarResultados(x1, x2)
```

Obtém os coeficientes

Já é conhecida

Mostra resultados

Qual é a interface com o usuário? Preciso ter uma?

Modularização

Modularizando a aplicação

```python
def lerCoeficientes():
    print("Resolução de equações de 2o grau")
    print("---------------------------------\n")
    print("Para a equação ax² + bx + c = 0, forneça:")
    huguinho = float(input ("Coeficiente a: "))
    zezinho  = float(input ("Coeficiente b: "))
    luisinho = float(input ("Coeficiente c: "))
    return (huguinho, zezinho, luisinho)
```

Retorna uma tupla de coeficientes

Modularização

Modularizando a aplicação

```python
def apresentarResultados (x1, x2):
    if x2 is None: #Equacao de 1º grau
        print ("\n\nA equação é de 1º grau.")
        print (f"Sua única raiz é igual a {x1}.")
    else:
        print ("A equação é de 2º grau.")
        if x1 == x2:
            print (f"Ela tem 2 raizes reais iguais a {x1:2f}")
        elif isinstance(x1,float):
            print (f"Ela tem 2 raizes reais diferentes: {x1:.2f} e {x2:.2f}")
        else:
            print (f"Ela tem 2 raizes complexas: {x1:.2f} e {x2:.2f}")
```

Decisão múltipla

Modularização

Criando módulos reaproveitáveis

- Quando funções e definições são suficientemente parametrizadas e generalizadas, podem ser recicladas
- Copiar e colar é arriscado
- Criação de módulos importáveis é o mais confiável
- Módulo → script com definições apenas
- *There can be only one*... script principal (main)

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Hora de ir para o replit.com

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

```
37  if __name__ == "__main__":
38      a, b, c = lerCoeficientes()
39      x1, x2  = calcularRaizes(a, b, c)
40      apresentarResultados(x1, x2)
41
42      x = np.linspace(-10,10, 101) # Cria um array de
        abcissas
43      y = eq2Grau (a, b, c, x) # Obtém as ordenadas
44
45      fig = plt.figure()
46      plt.grid()                  # Cria grade
47      plt.axhline(0,color='darkgray') # Cria eixo
        horizontal
48      plt.axvline(0,color='darkgray') # Cria eixo vertical
49
50      plt.plot (x, y, 'b')    # Traça a equação
51      plt.show()
```

Verifica se o script é o *chosen one* (vulgo principal)

```
import numpy as np
import matplotlib.pyplot as plt

from equacoes import eq2Grau, calcularRaizes
```

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Orientação a objetos

- Modularização por funções cria blocos de ações
- A realidade é modelada por *dados* e ações
- Orientação a objetos: modelagem baseada na realidade
- Classe → define como um tipo de objeto É
- Objetos são instâncias (criados a partir) de classes
- Tudo em Python é objeto!

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

Refatorando para OO

- Classe Eq2Grau: Modela uma equação de 2º grau
- Atributos (dados): a2, a1, a0
- Métodos (ações): determinar raízes, calcular ordenadas
- Construtor: inicializa um objeto
- Objetos tem de ser consistentes

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Modularização

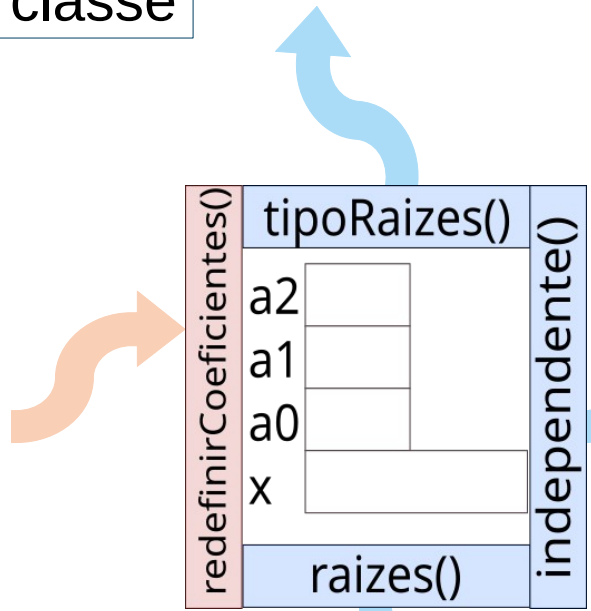
```
15  class Eq2Grau(object):
16      """ Representa uma equação de 2o grau
17          e suas funcionalidades
18      """
19
20      def __init__(self, a, b, c):
21          """ Inicializa a equação
22              Todo objeto deve ser consistente
23          """
24          self.a2 = a # coeficientes da equação
25          self.a1 = b
26          self.a0 = c
27          # raízes da equação. Se None, não foi calculada ainda
28          self.x = None
29
30
31      def tipoRaizes(self):
46
47      def raizes(self):
61
62      def redefinirCoeficientes(self, a, b, c):
67
68      def obterCoeficientes(self):
70
71      def independente(self, dependente):
73
```

Definição de classe

Construtor

Atributo

Método



self: referência ao objeto

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Estudo de caso

- Aquisição de dados de um sensor analógico
- Grandeza medida → sinal elétrico
- Conversão AD: sinal elétrico → inteiro
- Relação entre o valor discretizado e a grandeza não é exatamente linear
- Necessário identificar essa relação

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Estudo de caso

- Sensor: representa os sensores da aplicação
  - Mantém as configurações em arquivo
  - Faz a leitura diretamente do sensor
  - Converte valores discretos para valores das grandezas
  - Função de calibração em função de arquivo de leituras

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Estudo de caso

- VerificaSensor: classe da aplicação visual
  - Cria a tela do aplicativo em Kivy
  - Implementa ações para o apertar de botões
  - Interface com o sensor

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Estudo de caso

```
10  class Sensor(object):
11      """ Define o comportamento de um sensor para o app
12      """
13
14      def __init__(self, arqConfig=None):
27          |
28      def _salvarConfig(self, arqConfig):
33
34      def _linear1V(self, valor):
39
40      def _polinomial (self, valor):
45
46      def calibrar(self, arquivoCSV):
56
57      def ler(self):
```

Construtor

Funções de conversão

Calibração

Estudo de caso

```
 9  from sensor import Sensor
10
11  sensorNivel = Sensor()
12  print ('Antes da calibração')
13  print ('--------------------')
14  print (f'840 corresponde a {sensorNivel.converter(840)}')
15
16  sensorNivel.calibrar('sensorNivel.csv')
17
18  print ('\n\nDepois da calibração')
19  print ('--------------------')
20  print (f'840 corresponde a {sensorNivel.converter(840)}')
```

Cria um objeto sensor

Referência à função de conversão vigente

Esta função não é a mesma da linha 14

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Estudo de caso

Label
kivy.uix.label

GridLayout
kivy.uix.gridlayout

TextInput
kivy.uix.textinput

Button
Kivy.uix.button

BoxLayout
kivy.uix.boxlayout

Estudo de caso

```
 9  from kivy.uix.boxlayout import BoxLayout
10  from kivy.uix.gridlayout import GridLayout
11  from kivy.uix.label import Label
12  from kivy.uix.textinput import TextInput
13  from kivy.uix.button import Button
14  from kivy.uix.popup import Popup
15  from kivy.app import App
16
17  from sensor import Sensor
18
19  class VerificaSensor(App):
20      def build(self):
48
49      def atualizarNivel(self, *args):
53
54      def executarCalibracao(self, *args):
61
62
63  if __name__ == '__main__':
64      VerificaSensor().run()
```

Imports dos widgets

Import da classe Sensor

Especialização da classe da aplicação

Cria a tela (especialização)

Comportamento do botão (novo método)

Disparo da aplicação (loop de mensagens)

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

Retrospectiva do dia

- Modularização de código por funções
- Criação de módulos com funções reaproveitáveis
- Noções de orientação a objeto
- Leitura de dados de arquivos csv
- Polinômios em numpy
- Criação de uma interface gráfica

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

E agora?

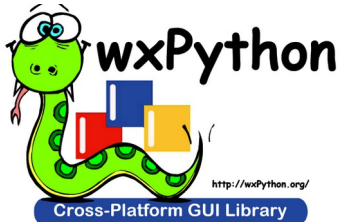
- Esta foi uma breve introdução ao Python
- Muitos *features* ficaram de fora
- Há uma infinidade de APIs e modelos
- Várias possibilidades de *backend* e *frontend*

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

E agora?

Desktop com interfaces gráficas

- PyQt e PySide2 → Qt
- Tkinter→ Tk
- WxPython → WxWindow
- Kivy



INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

E agora?

Ciência de dados e aprendizado de máquina

- Pandas
- Scikit-Learn
- Tensorflow
- PyTorch
- OpenCV

E agora?

Sistemas embarcados e IoT

- MicroPython, CircuitPython
- Flask
- Requests
- Paho MQTT

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

E agora?

Aplicativos para smartphone

- Kivy
- Python-for-Android
- Buildozer
- Pydroid3

E agora?

Desenvolvimento web

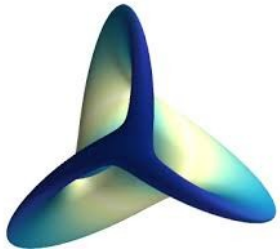
- Django
- Flask
- Requests

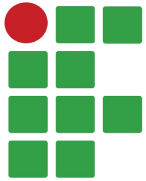
INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

E agora?

Aplicações gráficas

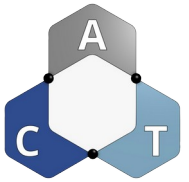
- Matplotlib/seaborn
- Mayavi2
- PyOpenGL
- Open Cascade
- Blender
- Freecad





INSTITUTO FEDERAL
Rio Grande do Sul

Campus
Rio Grande



Python para Automação
Fundamentos

Prof. Carlos R Rocha
carlos.rocha@riogrande.ifrs.edu.br

ACT – Automação, Conectividade e Tecnologia