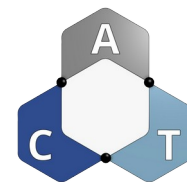


INSTITUTO FEDERAL
Rio Grande do Sul

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?

Introdução

O que é um programa?

- Sequência de ações a serem executadas por um sistema computacional
- Definidos por um conjunto de instruções escritas em uma linguagem de programação
- Podem ser de baixo ou alto nível

Introdução

Linguagens de programação de alto nível

- Baseadas nos idiomas humanos, mas com sintaxe e semântica claras e mais estritas
- Frases com significado claro e não ambíguo
- Centenas de linguagens e dialetos
- Não existe linguagem ideal, mas sim a mais adequada para um nicho de aplicações

Introdução

E as linguagens de baixo nível?

- No final, tudo se resume a circuitos eletrônicos
- Linguagem de máquina é uma relação de operações que os circuitos de um processador conseguem realizar
- Cada processador tem o seu conjunto de operações
- 1 instrução de alto nível → centenas de baixo nível

Introdução

O sistema operacional é nosso amigo

- Windows, Linux, Mac OS, Android, iOS...
- Escondem e padronizam o acesso ao hardware
- Multitarefa
- Controle de execução dos programas
- Interface gráfica

Introdução

Processos de tradução

- Na execução, o que escrevemos no programa deve ser legível pelo processador
- Diversidade de combinações de hardware e sistemas operacionais → variedade de problemas
- Processos
 - Compilação
 - Interpretação

Introdução

Compilação

- Tradução de uma só vez
- Gera uma versão executável
- Código fonte: versão original do programa
- Algumas linguagens: C/C++, Pascal, C#, Assembly, **Java, Kotlin**

Introdução

Interpretação

- Tradução simultânea
- O **interpretador** executa o fonte
- Script: outro nome dado ao fonte interpretado
- Algumas linguagens: Python, Javascript, PHP, BASIC

Introdução

Ranking 2020 das linguagens de programação IEEE Spectrum (CASS, 2020)

- Organizado de acordo com aplicação
 - Desktop
 - Web (*frontend e backend*)
 - Mobile
 - Embarcados
- É um entre vários outros publicados por diferentes entidades

Rank	Language	Type	Score
1	Python ▼	  	100.0
2	Java ▼	  	95.3
3	C ▼	  	94.6
4	C++ ▼	  	87.0
5	JavaScript ▼		79.5
6	R ▼		78.6
7	Arduino ▼		73.2
8	Go ▼	 	73.1
9	Swift ▼	 	70.5
10	Matlab ▼		68.4

Python

- Linguagem de programação de alto nível **interpretada**
- Criada por Guido van Rossum em 1990
- Características
 - Uso geral
 - Multiplataforma
 - Estruturada, modular e orientada a objetos
 - Expressão da lógica é mais importante que os aspectos computacionais
 - Grande base de recursos *built-in* e extensa oferta de bibliotecas
 - Comunidade de usuários e desenvolvedores imensa
 - *Flerte* com a comunidade *open source* e software livre



Python

Onde e como obter?

- Direto do site www.python.org (Python *vanilla*)
- Com uma *distribuição*:
 - Anaconda (www.anaconda.com/products/individual)
 - WinPython (winpython.github.io)
- Ou, se é um iluminado Linux, é só selecionar os pacotes
- Para Android: Pydroid3



Python

- APIs
 - Várias são parte da base do Python
 - APIs de terceiros disponíveis de acordo com a área
 - Distribuições vêm com várias e automatizam a busca
 - Instalações individuais com pip
- Ambientes de Desenvolvimento (IDE)
 - Além do editor (básico), ferramentas de apoio integradas a ele
 - Diversas IDEs independentes do Python
 - Distribuições costumam vir com IDEs



Python

- Spyder

Spyder (Python 3.6)

Arquivo Editar Pesquisar Código Executar Depurar Consoles Projetos Ferramentas Ver Ajuda

Editor - /home/carlos/Pesquisa/usv/nx01/testes/sockets/basics/servidor.py

testemapa.py cliente.py servidor.py

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Fri Jan 10 16:07:56 2020
5
6 @author: carlos
7 """
8
9 import socket
10
11 serv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12 serv.bind(('0.0.0.0', 8080))
13 serv.listen(5)
14
15 while True:
16     conn, addr = serv.accept()
17     from_client = ''
18
19     while True:
20         data = conn.recv(4096)
21         if not data:
22             break
23         from_client += str(data, encoding='utf-8')
24         print(from_client)
25         conn.send(b"I am SERVER\n")
26
27     conn.close()
28     print('client disconnected')
```

Unit testing

Run tests

Status	Name	Message	Time (ms)
--------	------	---------	-----------

Console IPython

Console 1/A

Python 3.6.9 (default, Nov 7 2019, 10:44:02)
Type "copyright", "credits" or "license" for more information.

IPython 5.5.0 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

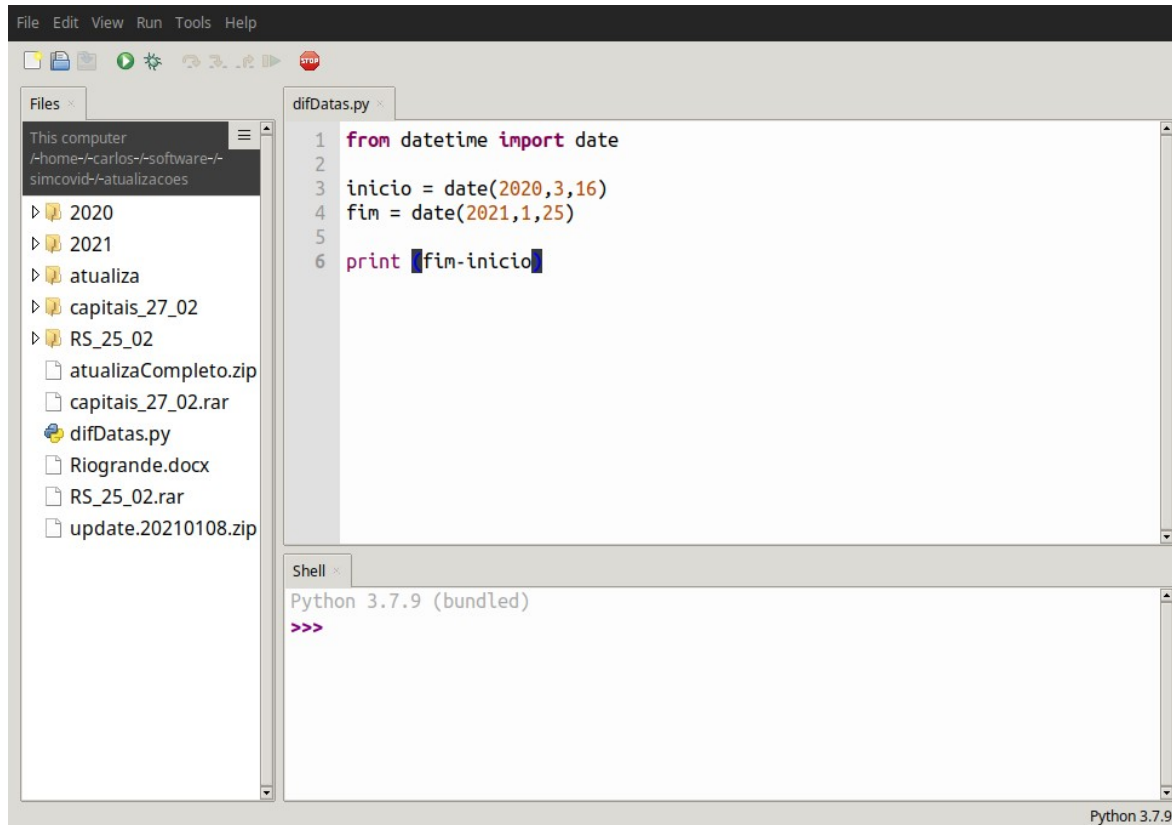
In [1]:

Console IPython Log do histórico

Permissões: RW Fim de linha: LF Codificação: UTF-8 Linha: 23 Coluna: 1 Memória: 25 %

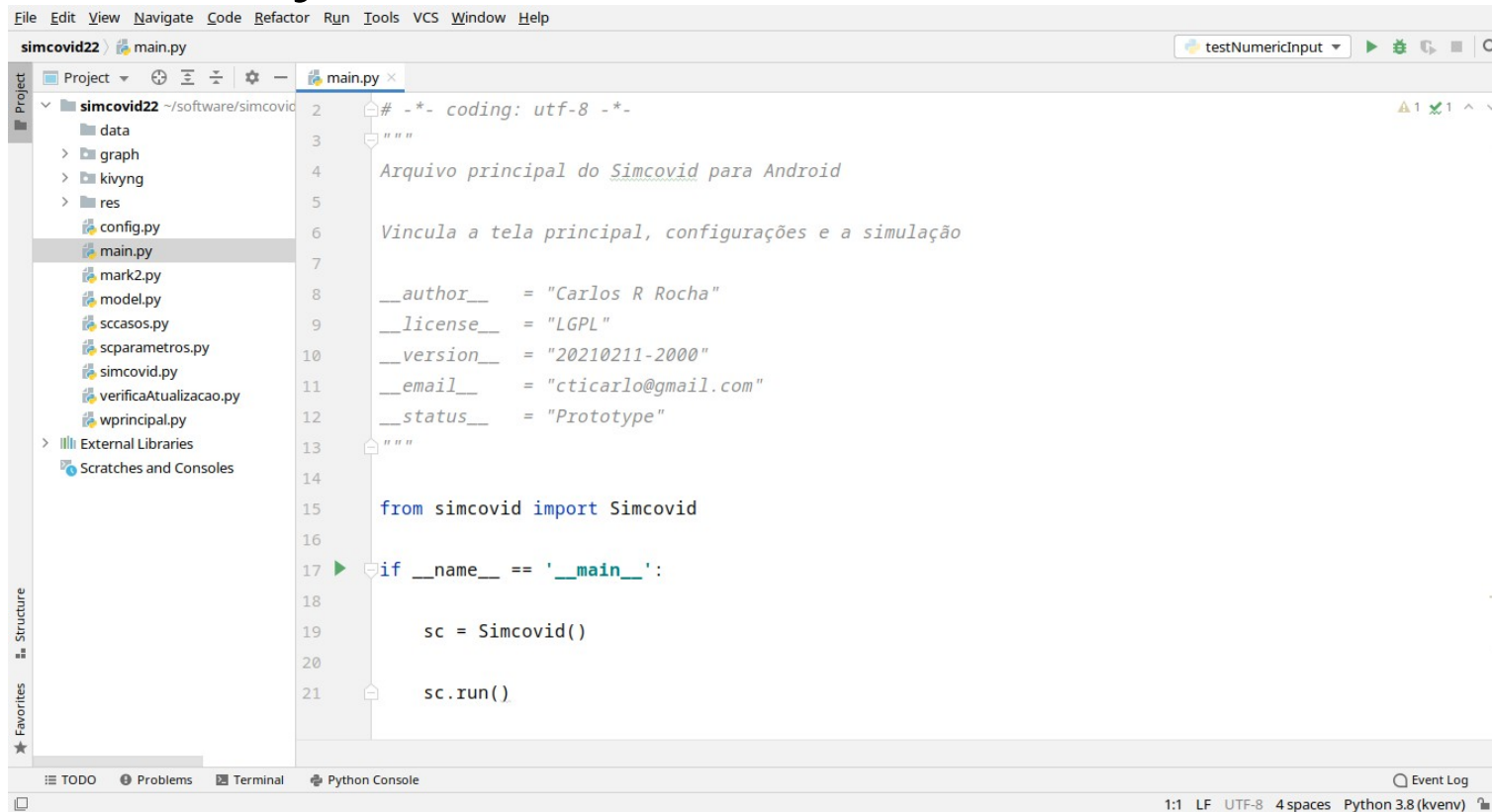
Python

- Thonny



Python

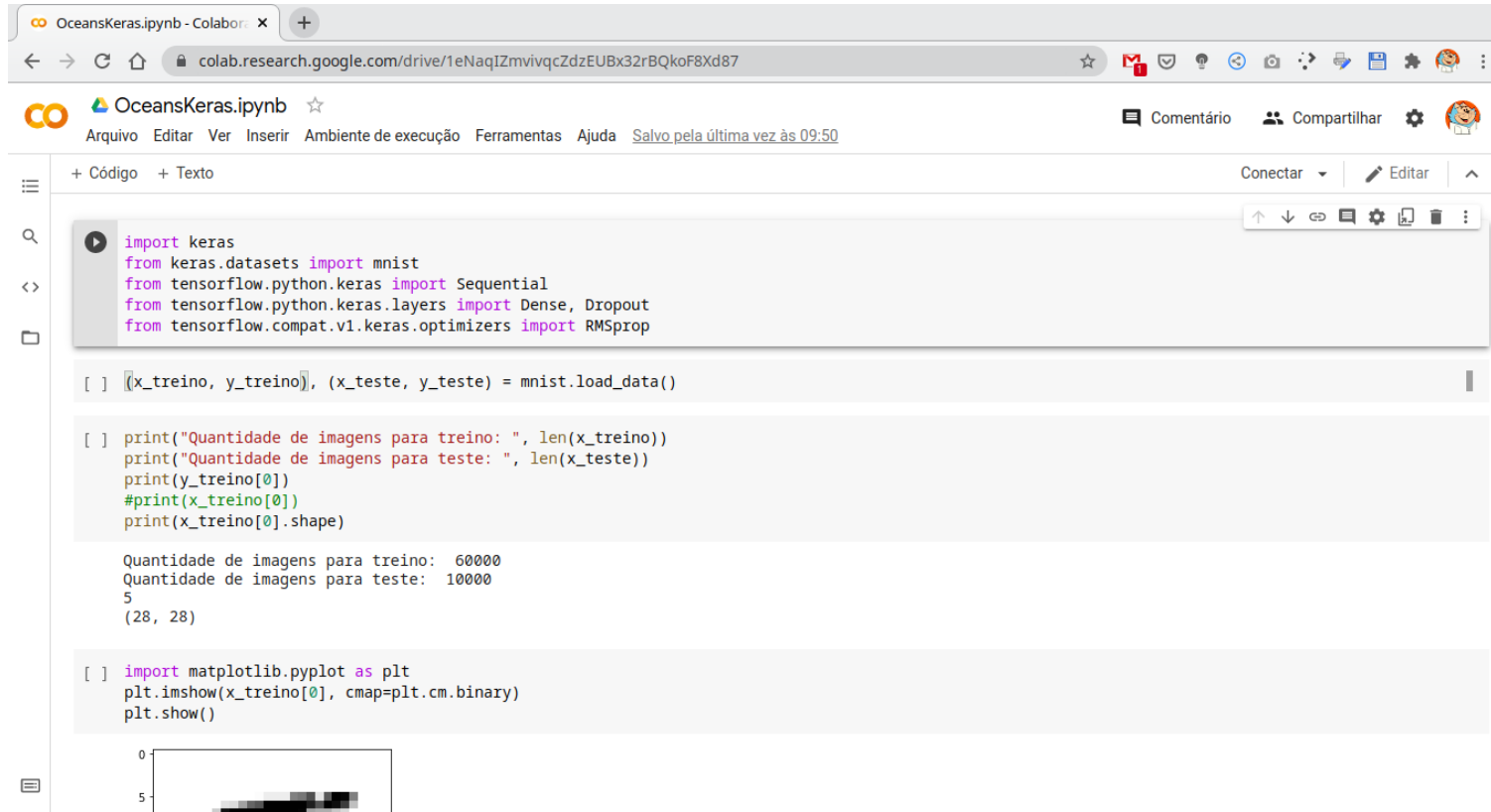
- IntelliJ PyCharm



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
simcovid22 main.py testNumericInput
Project
  simcovid22 ~/software/simcovid
    data
    graph
    kivyng
    res
    config.py
    main.py
    mark2.py
    model.py
    scasos.py
    scparametros.py
    simcovid.py
    verificaAtualizacao.py
    wprincipal.py
  External Libraries
  Scratches and Consoles
Structure
Favorites
2  # -*- coding: utf-8 -*-
3  """
4  Arquivo principal do Simcovid para Android
5
6  Vincula a tela principal, configurações e a simulação
7
8  __author__ = "Carlos R Rocha"
9  __license__ = "LGPL"
10 __version__ = "20210211-2000"
11 __email__ = "cticarlo@gmail.com"
12 __status__ = "Prototype"
13 """
14
15 from simcovid import Simcovid
16
17 if __name__ == '__main__':
18
19     sc = Simcovid()
20
21     sc.run()
```


Python

- Google Colab




```
import keras
from keras.datasets import mnist
from tensorflow.python.keras import Sequential
from tensorflow.python.keras.layers import Dense, Dropout
from tensorflow.compat.v1.keras.optimizers import RMSprop

(x_treino, y_treino), (x_teste, y_teste) = mnist.load_data()

print("Quantidade de imagens para treino: ", len(x_treino))
print("Quantidade de imagens para teste: ", len(x_teste))
print(y_treino[0])
#print(x_treino[0])
print(x_treino[0].shape)

Quantidade de imagens para treino: 60000
Quantidade de imagens para teste: 10000
5
(28, 28)

import matplotlib.pyplot as plt
plt.imshow(x_treino[0], cmap=plt.cm.binary)
plt.show()
```



Python

- REPLit

The screenshot displays the Repl.it web interface for a Python project named 'teste'. The browser address bar shows 'repl.it/@carlosRrocha/teste#main.py'. The interface includes a file explorer on the left, a code editor in the center, and a preview window on the right.

Files: The file explorer shows a 'main.py' file and a 'Package files' section with 'poe...' and 'pypr...'.

main.py: The code defines a Kivy application with an accordion widget.

```
1 from kivy.uix.accordion import
  Accordion, AccordionItem
2 from kivy.uix.label import Label
3 from kivy.app import App
4
5
6 class AccordionApp(App):
7     def build(self):
8         root = Accordion
9         (orientation='vertical')
10         for x in range(5):
11             item = AccordionItem
12             (title='Title %d' % x)
13             item.add_widget(Label
14                 (text='Very big content\n'
15                     * 10))
16             root.add_widget(item)
17         return root
18
19 if __name__ == '__main__':
20     AccordionApp().run()
```

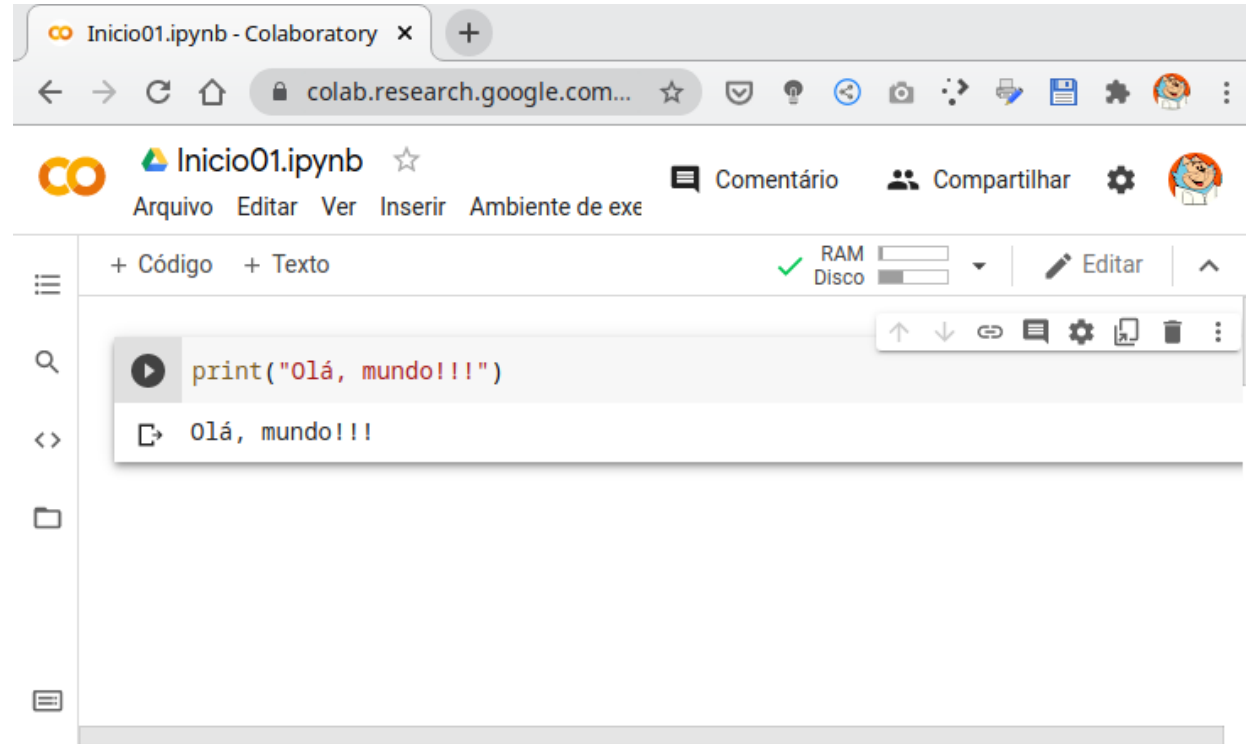
Accordion Preview: The preview window shows a vertical accordion with five titles: 'Title 0', 'Title 1', 'Title 2', 'Title 3', and 'Title 4'. 'Title 4' is selected and highlighted in blue. Below the titles, the text 'very big content' is repeated multiple times.

Console: The console shows the following output:

```
[INFO ] [GL      ] OpenGL version <b'3.1 Mesa 20.0.8'>
[INFO ] [GL      ] OpenGL vendor <b'VMware, Inc.'>
[INFO ] [GL      ] OpenGL renderer <b'llvmpipe (LLVM 10.0.0, 256 bits)'>
[INFO ] [GL      ] OpenGL parsed version: 3, 1
[INFO ] [GL      ] Shading version <b'1.40'>
[INFO ] [GL      ] Texture max size <8192>
[INFO ] [GL      ] Texture max units <32>
[INFO ] [Window  ] auto add sdl2 input provider
[INFO ] [Window  ] virtual keyboard not allowed, single mode, not docked
[INFO ] [Base    ] Start application main loop
[INFO ] [GL      ] NPOT texture support is available
```

Primeiros passos

- Usando Google Colab
- <https://colab.research.google.com/>
- Conceito de *notebooks*
- Células de código



Primeiros passos

```
[1] print("Olá, mundo!!!")
Olá, mundo!!!

[2] texto = 'Olá, mundo!!!'

[3] print(texto)
Olá, mundo!!!

[4] type(texto)
str

[5] texto = "Hi, mundo!"
print(texto)
Hi, mundo!
```

Função print()

String

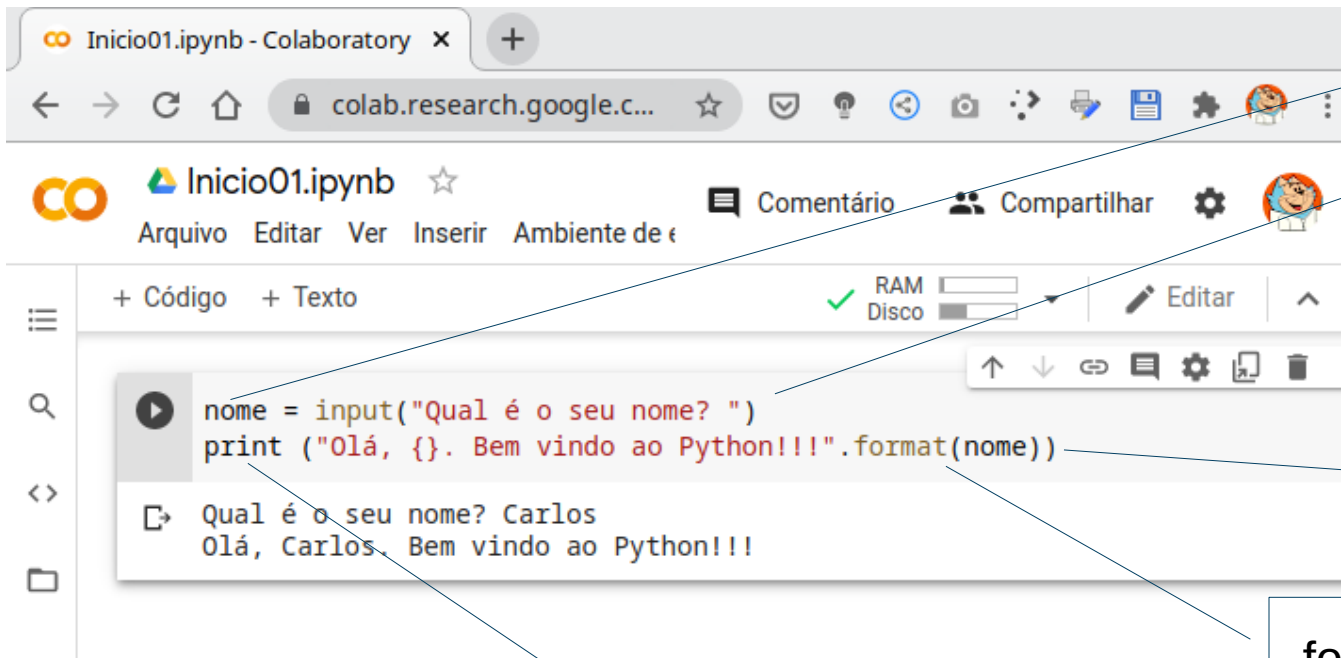
Variável texto
Atribuição (=)
Valor: "Olá, mundo"

Imprime o conteúdo
da variável texto

Mostra o tipo de dado
da variável texto

Modifica o conteúdo
da variável texto

Primeiros passos



The screenshot shows a Google Colaboratory notebook titled 'Inicio01.ipynb'. The code cell contains the following Python code:

```
nome = input("Qual é o seu nome? ")  
print ("Olá, {}. Bem vindo ao Python!!!".format(nome))
```

The output of the code cell is:

```
Qual é o seu nome? Carlos  
Olá, Carlos. Bem vindo ao Python!!!
```

Annotations with lines pointing to the code and output:

- Line from 'Variável nome' points to `nome` in the first line of code.
- Line from 'input(): entrada de dados interativa' points to the `input()` function in the first line of code.
- Line from 'String com parte Indefinida { }' points to the curly braces in the string of the `print` statement.
- Line from 'format(): método de String' points to the `.format()` method in the `print` statement.
- Line from 'Imprime a string após a execução de format()' points to the first line of the output.
- Line from 'Imprime o resultado de format()' points to the second line of the output.

Variável nome

input(): entrada de dados interativa

String com parte Indefinida { }

format(): método de String

Imprime a string **após** a execução de format()
Imprime o **resultado** de format()

Primeiros passos

- Dados numéricos

2 # inteiro de base 10

-20 # inteiro negativo de base 10

0x42 # inteiro de base 16 (hexadecimal): vale 66

0o42 # inteiro de base 8 (octal): vale 34

0b1101 # inteiro de base 2 (binário): vale 13

3.0 # real

-2.5 # real

23e-4 # real com expoente multiplicador: 23×10^{-4} ou
0.0023

Primeiros passos

float(): produz um número real a partir do argumento

input(): entrada de dados textual interativa

```
1_in = float(input("Entre com a medida em polegadas(in): "))  
  
l_cm = l_in * 2.54 # Relação entre centímetros e polegadas  
  
print ("{}in corresponde a {}cm.".format(l_in, l_cm))
```

Expressão aritmética: multiplicação

format(): completa os {} com elementos da lista

Primeiros passos

- Dados textuais

`'Exemplo número 1' # String`

`"Exemplo número 2" # String`

`"""`

`Exemplo com mais`

`De uma linha`

`""" # String (docstring)`

Primeiros passos

A variável s é criada e recebe o texto

Tamanho da string(quantos caracteres)

Menor caractere da string

Maior caractere da string

Posição do caractere 'C'

Versão com letras minúsculas

Versão com letras maiúsculas

Versão com primeiras letras maiúsculas

Cria uma lista com as palavras

```
s="Keep Calm and Python!"  
print(len(s))  
print(min(s))  
print(max(s))  
print(s.index("C")) # Maiúsculas e minúsculas são caracteres diferentes  
print(s.count('a'))  
print(s.lower())  
print(s.upper())  
print(s.title()) # Aqui só o a de and muda  
print(s.split(' ')) # Cria uma lista das palavras
```

Primeiros passos

- Dados textuais - *slicing*

```
▶ print(s[0:5]) # Um slice de 5 caracteres, do 1o(0) ao 5o(4)
print(s[7]) # O caractere da posição 7. Pode contar!
print(s[0]) # O primeiro caractere
print(s[-1]) # O último caractere (contado do final)
```

```
[ ] print(s[22]) # Deu erro! Não há tantos caracteres
```

Primeiros passos

- Dados lógicos

True # Valor lógico verdadeiro

False # Valor lógico falso

- Valores “nulos” são automaticamente assumidos como False: 0, 0.0, “”, None
- Valores “não nulos” são automaticamente assumidos como True

Primeiros passos

Outros tipos

- None
- Tuplas: (a,b,c)
- Listas: [a, b, c]
- Dicionários: {'nome': 'fulano', 'idade': 51, 10:False}
- E vários outros...

Primeiros passos


Expressões

- Aritméticas →
- Comparações
- Lógicas
- Textuais

Operador	Exemplo	Significado
+	$x+y$	Adição
-	$x-y$	Subtração
*	$x*y$	Multiplicação
/	x/y	Divisão
//	$x//y$	Divisão
%	$x\%y$	Resto da divisão
-	$-x$	Negação unária
+	$+x$	Positivo unário
**	$x**y$	Potenciação

Primeiros passos


Expressões

- Aritméticas
- **Comparações** 
- Lógicas
- Textuais

Operador	Exemplo	Significado
==	a == b	Igualdade
!=	a != b	Diferença
>	a > b	Maior que
>=	a >= b	Maior ou igual a
<	a < b	Menor que
<=	a <= b	Menor ou igual a

Primeiros passos

Expressões

- Aritméticas
- Comparações
- **Lógicas** 
- Textuais

Operador	Exemplo	Significado
not	not c	Negação
or	a or b	Ou lógico
and	a and b	E lógico
is	a is b	A é B
is not	a is not b	A não é B

Primeiros passos

Expressões

- Aritméticas
- Comparações
- Lógicas
- Textuais 

Operador	Exemplo	Significado
<code>[]</code>	<code>txt[i]</code> <code>txt[i:j]</code> <code>txt[:j]</code> <code>txt[i:]</code>	<i>Slicing</i>
<code>*</code>	<code>txt*n</code>	Retorna um texto com n vezes o conteúdo de txt
<code>+</code>	<code>txt+msg</code>	Concatena txt e msg
<code>in</code>	<code>s in txt</code>	s está contido em txt?
<code>not in</code>	<code>s not in txt</code>	txt não contém s?

Primeiros passos

Identificadores e palavras-chave

- Tudo tem que ter um nome (identificador)
- Eles devem ser únicos em um *contexto*
- Não se pode utilizar as palavras-chave da linguagem
- Deve-se evitar usar as palavras de elementos usuais, mas que não são palavras-chave (por exemplo, `print()`)
- Letras, algarismos e `_`

Primeiros passos

Identificadores e palavras-chave

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Scripts

- Arquivo com linhas de código
- Não há uma estrutura formal
- Código lido linha a linha, do topo até o fundo
- Declarações: para usar algo, ele já deve existir
- Imports: para usar algo, ele já deve existir
- **KISS: Keep It Simple, S...**

Scripts

- Arquivo com linhas de código
- Não há uma estrutura formal
- Código lido linha a linha, do topo até o fundo
- Declarações: para usar algo, ele já deve existir
- Imports: para usar algo, ele já deve existir
- **KISS: Keep It Simple, S...**

Scripts

Explicita a versão de Python

Define a codificação de caracteres

Docstring

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Criado em 14/3/2021 16:50
@author: Carlos R Rocha
"""
print ("Olá, mundo!")
```

Código do script

Scripts

Aproveite as APIs

Cria graus; adquire texto; converte para real; atribui a grau

```
▶ graus = float(input("Ângulo em graus: "))  
  
radianos = graus * 3.14159265 / 180.0 # Relação entre graus e radianos  
  
print (f"{graus} corresponde a {radianos}.")
```

PI !!! π

String formatadora: Tem campos para preencher

Scripts

Aproveite as APIs

Importa radians(), do módulo math



```
from math import radians
```

```
graus = float(input("Ângulo em graus: "))  
radianos = radians(graus)
```

Usando radians() para facilitar

```
msg = f"{graus} corresponde a {radianos}."  
print (msg)
```

Há vantagem no texto estar em variável?

Scripts

Explorando o potencial numérico e gráfico



```
tensao = float (input('Forneça o valor da tensão(V): '))  
freq = float (input('Forneça a frequência da rede(Hz): '))
```


Scripts

Explorando o potencial numérico e gráfico



```
import numpy as np
```

Numpy!!!



```
t = np.linspace(0, 1/freq, 361)
```

Array com 361 valores igualmente distribuídos entre 0 e $1/\text{freq}$

```
[ ] print (f"São {len(t)} valores em no array t.\n")  
    print (t)
```

Array com 361 valores igualmente distribuídos entre 0 e 2π

```
[ ] ang = np.linspace(0, 2*np.pi, 361)  
    v = tensao * np.sin(ang)
```

```
[ ] print(v)
```

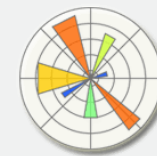
Scripts

Explorando o potencial numérico e gráfico



```
import matplotlib.pyplot as plt
```

Matplotlib!!!



Matplotlib

```
fig = plt.figure()
```

Define uma figura, onde gráficos poderão ser traçados

```
plt.plot (t, v)
```

Traça um gráfico de linha, com t como abcissas(x) e v como ordenadas(y)

```
plt.show()
```

Apresenta (renderiza) a figura corrente

Retrospectiva do dia

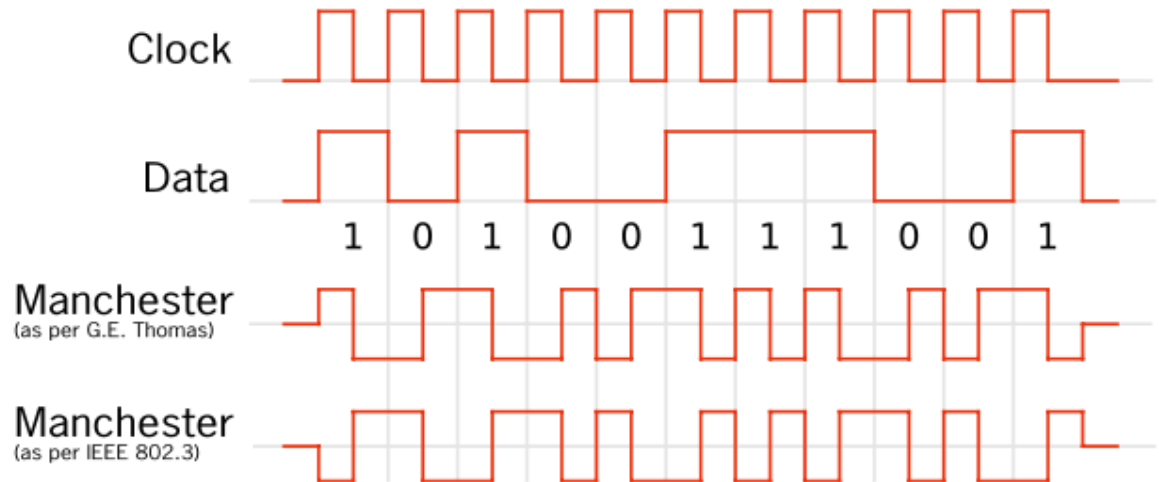
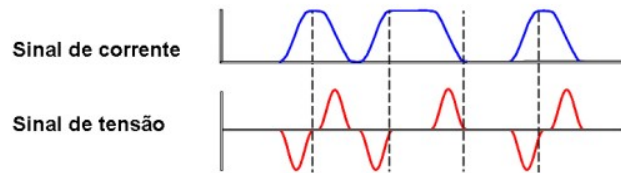
- Programação e suas linguagens
- Python e seu ecossistema
- Google Colab
- Dados, tipos, variáveis, expressões
- Entrada e saída
- Scripts
- Pacotes e APIs
- Numpy e Matplotlib

Roteiro

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

Modularização

- Modulação de corrente para bits 0 e 1
- Codificação Manchester (Detecção de borda) por período de tempo

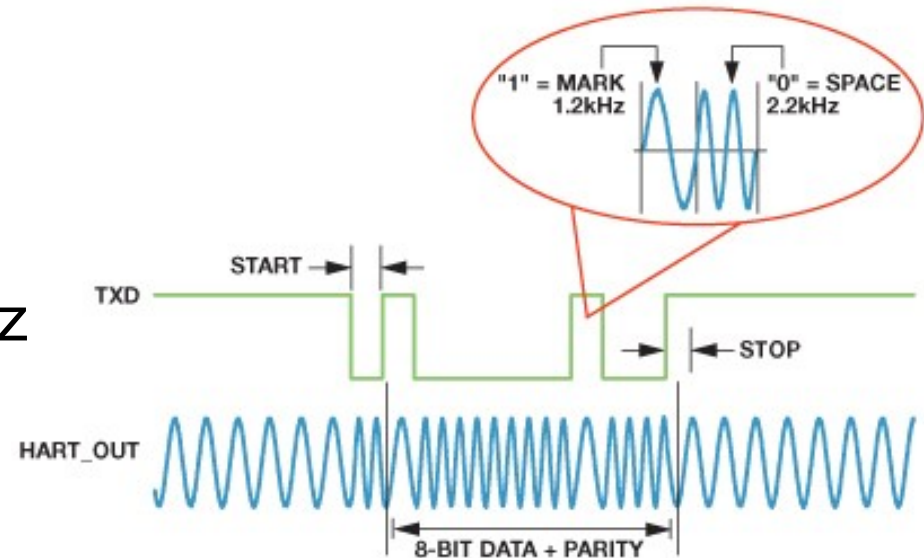


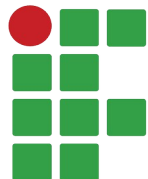
Estudo de caso

- *Highway Addressable Remote Transducer*
- Tecnologia que *recicla* malhas de instrumentação de 4-20mA (retrofit)
- Modelo cliente-servidor (mestre/escravo)
- É possível utilizar a malha para transmitir um sinal analógico, e vários digitais

E agora?

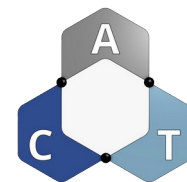
- Sinal digital é sobreposto ao analógico
- Velocidade de 1200bps
- Ciclo de atualização de 2Hz
- FSK (Frequency-Shift Keying)
 - 1mA pico a pico
 - Bit 0 → 2.2KHz ; Bit 1 → 1.2KHz





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

Referências

- ANACONDA. Anaconda – The World’s Most Popular Data Science Platform. 2021. Disponível em <<https://www.anaconda.com/>>. Acessado em fevereiro/2021.
- CASS, S. Top Programming Languages 2020. IEEE Spectrum, 2020. Disponível em <<https://spectrum.ieee.org/at-work/tech-careers/top-programming-language-2020>>. Acessado em julho/2020.
- PSF: Python Software Foundation. Welcome to Python.org. 2020. Disponível em <<https://www.python.org>>. Acessado em janeiro/2021.
- WINPYTHON Development Team. WinPython. 2021. Disponível em <<https://winpython.github.io/>>. Acessado em março/2021.