

Introdução à Programação

CIÊNCIA DA
COMPUTAÇÃO

ATITUS
EDUCAÇÃO



ATITUS
EDUCAÇÃO



- Aula 01 -
Pensamento Computacional

Prof. Me. Lucas R. C. Pessutto

Na aula de hoje...

01

Apresentações

Conhecendo o professor
Apresentação da turma
A skill “Pensamento Computacional”

02

Ciência da Computação

O que é? O que faz?
Resolução de Problemas e Algoritmos
Limites da Computação

03

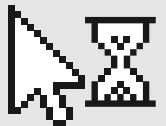
Programação

Processo de criar um Programa
Paradigmas de Programação
Linguagens de Programação

04

Python! 🐍

Características
Motivação: Quem usa Python?
Instalação
Primeiros Programas



Lucas R. C. Pessutto

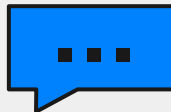
`lucas.pessutto@atitus.edu.br`

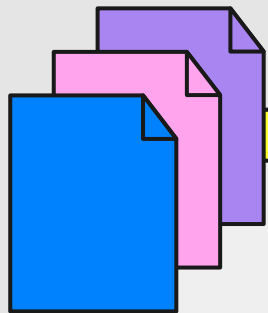
Formação:

- Bacharel em Ciência da Computação (UCS)
- Mestre em C. Computação (UFRGS)
- Aluno do Doutorado em Computação (UFRGS)

Trabalho com:

- Recuperação de Informações
- Processamento de Linguagem Natural
- Análise de Sentimentos

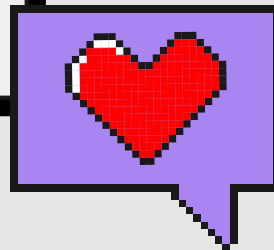
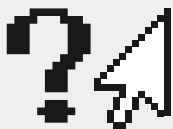




Apresente-se!

Quem é você?

Já teve alguma experiência com
programação? Qual linguagem você
utilizou?



Programação em Python

Objetivos Gerais:

- * Pensamento Computacional
(Computational Problem Solving)
- * Habilidades e Limites da Computação
- * Mapear problemas reais em “algo computacional”
- * Pensar como Cientistas da Computação!

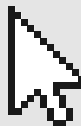


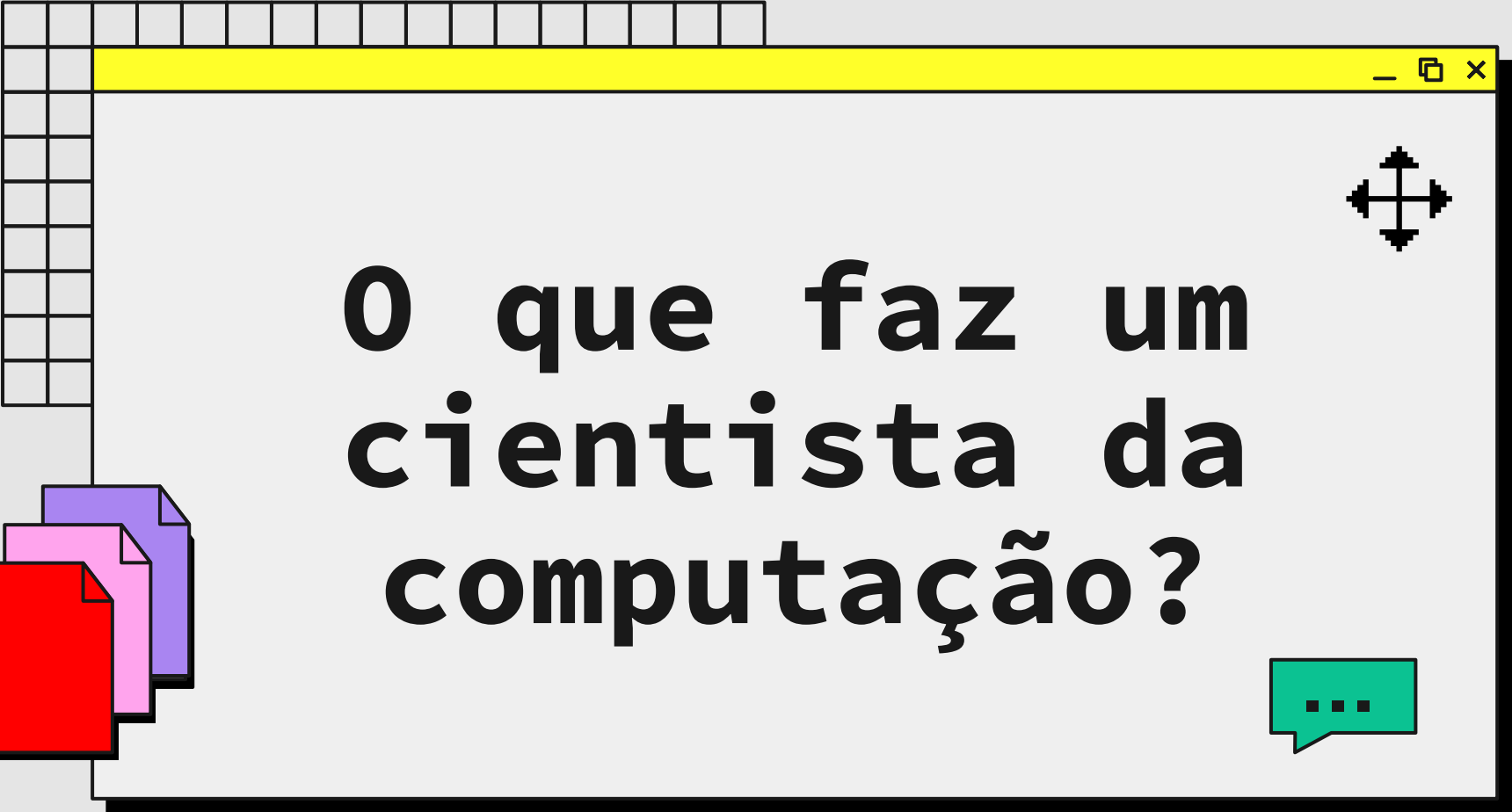
Programação em Python

Conteúdo Programático:



- * Introdução ao Python
 - Motivação / História
 - Instalação
 - Primeiros Programas
- * Programação Estruturada
 - Programas Sequenciais
 - Comandos Condicionais
 - Comandos Iterativos
- * Estruturas de Dados
 - Listas
 - Strings
 - Dicionários
- * Funções
- * Arquivos
- * Noções de Programação Orientada a Objetos
- * Noções de Banco de Dados
- * Biblioteca Pandas





O que faz um
cientista da
computação?



0 Computador

Escrever programas



Utilizar Aplicativos



Editor de Textos

Correio Eletrônico

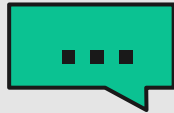
Jogos

Acesso à Internet

O que faz um computador?

- * Diferentes tarefas serão executadas por diferentes programas
- * Um programa é uma *receita*, que contém uma sequência correta de instruções, que são executadas pelo computador.
 - Quais instruções devem ser executadas pelo computador?
 - Em que ordem essas instruções devem ser executadas?
- * Precisamos uma linguagem para escrever as receitas.
 - Transformar uma ideia em uma sequência de instruções.



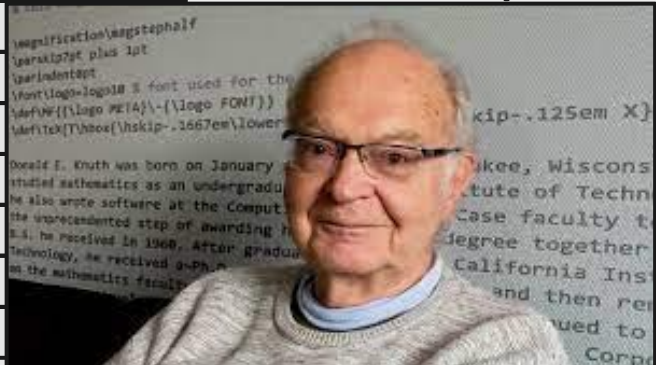


DEFINIÇÃO DE ALGORITMO

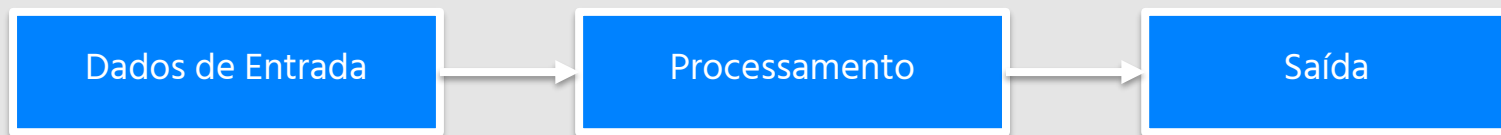
“Conjunto finito de regras que forma uma sequência de operações para resolver um tipo específico de problema.”

—Donald E. Knuth

“The art of Computing Programming” (1969)



Problemas e Algoritmos



- * **PROBLEMA** é uma relação entre entradas para saídas aceitáveis

Exemplo de Problema: Entrada: um conjunto de valores inteiros, todos diferentes entre si.

Saída : o menor de todos os valores de entrada

- * Um **ALGORITMO** resolve um problema, caso produza uma saída aceitável para QUALQUER entrada
- * Um algoritmo é **ÓTIMO** (no sentido de otimizado), quando a saída é a melhor solução para o problema

Problema da Travessia

- * Um homem vive no lado leste de um rio.
- * Ele precisa levar um repolho, uma cabra e um lobo para vender em uma vila que está no lado oeste do rio.
- * Entretanto, seu barco só tem capacidade de acomodar ele mesmo, e um único item, o repolho, cabra ou lobo.
- * Outro problema é que o homem não pode deixar a **cabra sozinha com o repolho**, porque ela irá comer o repolho, e ele também não pode deixar o **lobo sozinho com a cabra** pelo mesmo motivo.
- * Como este homem pode resolver este problema?



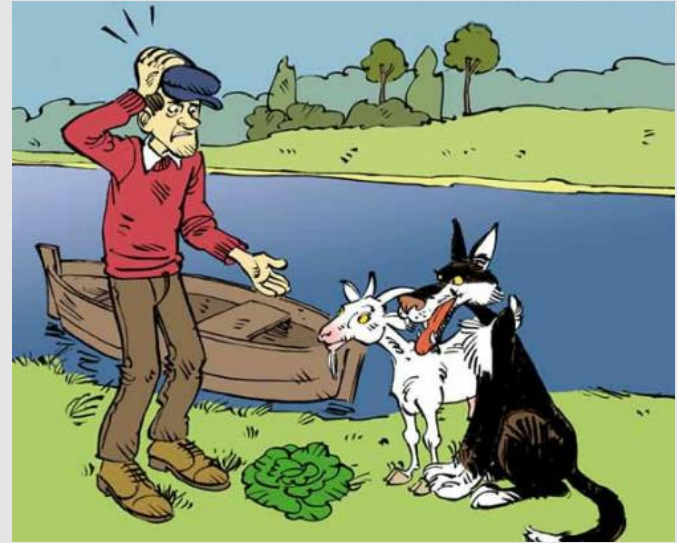
Problema da Travessia

Exercício:

Em grupos, elabore um algoritmo que resolva o problema da travessia.

Pense em:

- * Representação do problema (faça um algoritmo não textual)
- * Estado Inicial do problema (entrada)
- * Estado Final do problema (saída)
- * Passos para transformar a entrada em saída
 - Comandos necessários
 - Representação da solução



Problema da Travessia – Refletindo

- * Quais aspectos do problema precisaram ser representados?
- * Porque ninguém se preocupou em saber:
 - Nome do homem
 - Cor do barco
 - Profundidade do rio
 - etc.
- * **ABSTRAÇÃO**: a representação de um problema deve considerar somente seus aspectos relevantes, omitindo tudo o que for irrelevante àquela solução.



PROBLEMAS COMPUTACIONAIS = REPRESENTAÇÃO + ALGORITMO

- * **Representação**: deve capturar todos os aspectos relevantes do problema
- * **Algoritmo**: resolve o problema, de forma eficiente, utilizando a representação fornecida
- * Existem problemas que não podem ser resolvidos de maneira ótima e de forma eficiente!

Os limites da computação

PROBLEMAS COMPUTACIONAIS = REPRESENTAÇÃO + ALGORITMO

* Exemplo: **Caixeiro Viajante**

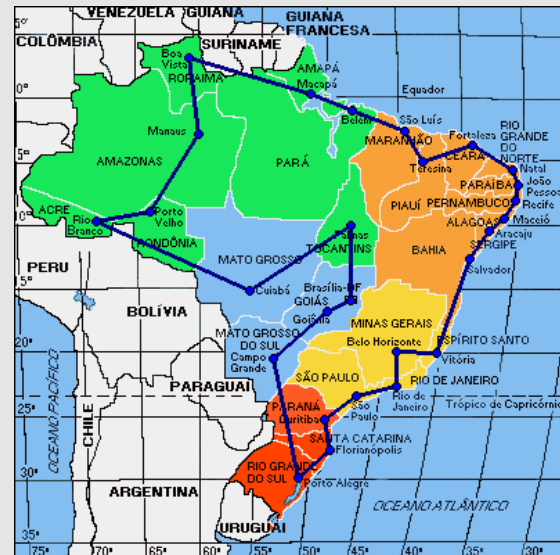
Encontrar a rota mais curta que permita que um vendedor realize um circuito em um conjunto de cidades

Representação:

* Tabela com distâncias entre as cidades

Algoritmo:

* Testar todas as combinações possíveis (força bruta)

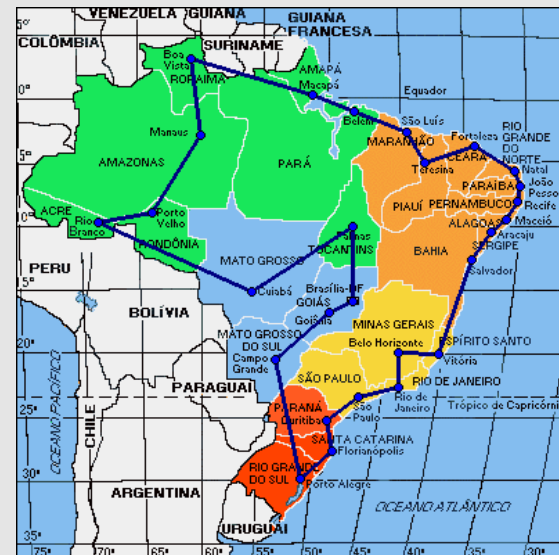


Os limites da computação

PROBLEMAS COMPUTACIONAIS = REPRESENTAÇÃO + ALGORITMO

Exemplo: Caixeiro Viajante

- * Para 10 cidades existem $10!$ possíveis rotas ~3.5 milhões de rotas
- * Se um computador calcula 1 milhão de rotas por segundo, ele levaria mais de 3.5 segundos para calcular todas essas rotas



Os limites da computação

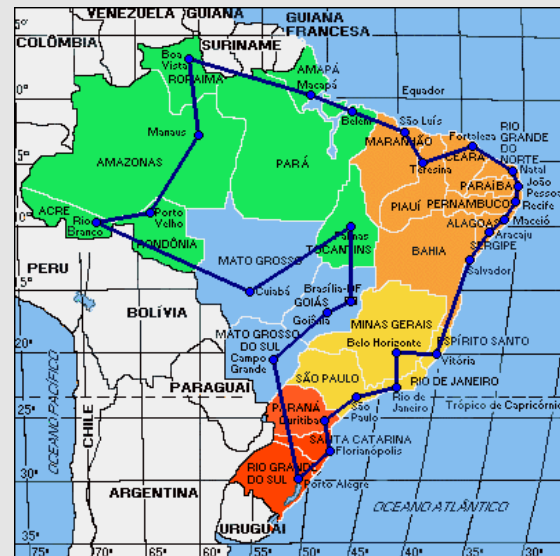
PROBLEMAS COMPUTACIONAIS = REPRESENTAÇÃO + ALGORITMO

Exemplo: Caixeiro Viajante

★ Para o mapa do Brasil:

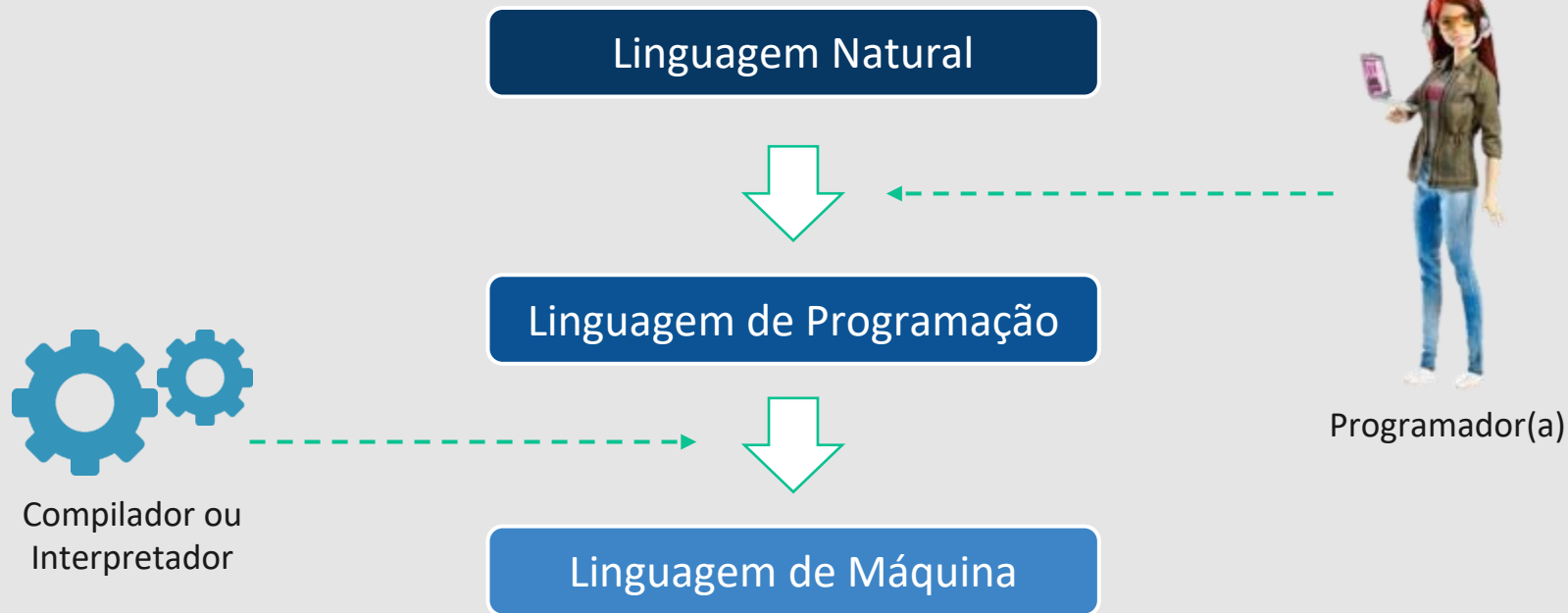
$27! = 10.888.869.450.418.352.160.768.000.000$ rotas
(>10 octilhões)

Se um computador calcula 1 milhão de rotas por segundo, ele levaria mais de 345 trilhões de anos para calcular todas essas rotas (25.000 vezes a idade do universo)



Processo de Desenvolvimento

Uma visão bem simplificada...



Modelos e Paradigmas de Linguagens

Objetivo: atender a diferentes tipos de problemas

Modelo Imperativo

Um programa é uma sequência de comandos que realizam transformações sobre os dados

Paradigmas: Procedural e Orientado a Objetos

Modelo Declarativo

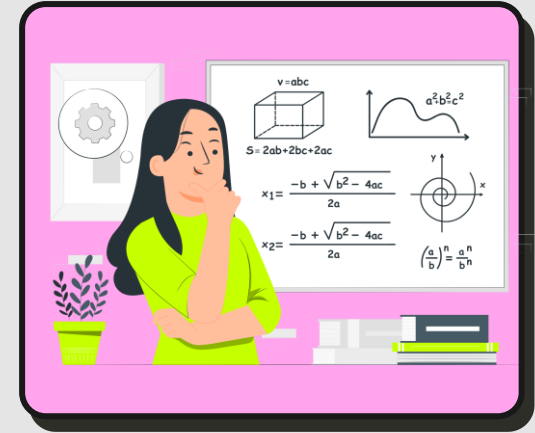
Linguagens que não possuem o conceito de sequência de comandos

Paradigmas: Funcional e Lógico



Modelo Imperativo

- * Uma solução é implementada através de uma série de comandos, que são executados **sequencialmente**.
- * Baseado nos princípios da **Programação Estruturada**.
- * Conceitos: variáveis, atribuição, sequenciação
- * Linguagens Procedurais: Python, C, Pascal, Java, Algol, Fortran, PL/I, Basic, Ada



Linguagem e paradigmas

Usaremos a Linguagem **Python**



Python é uma linguagem multiparadigma

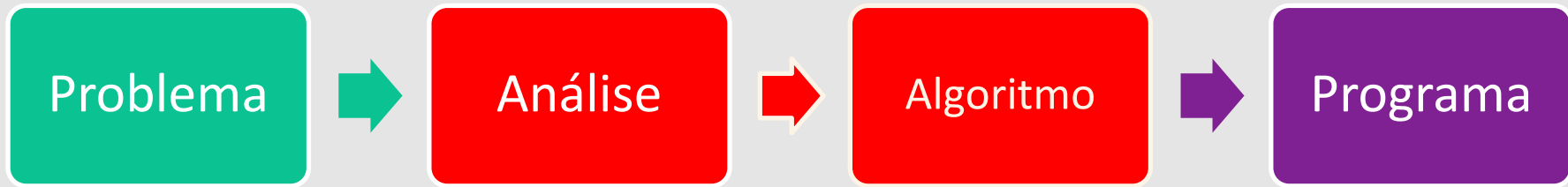
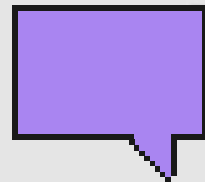
Nós usamos o **Paradigma Procedural** ou Imperativo

Solução implementada através de ações executadas **sequencialmente**

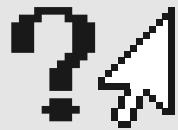
Princípios da **Programação Estruturada**

Estruturas de controle de fluxo como **sequenciação**, **seleção** e **repetição**

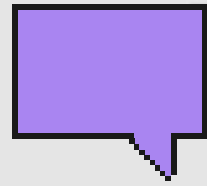
ELABORAÇÃO DE UM PROGRAMA



- Estruturação
- Decomposição



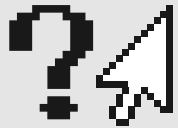
Como **NÃO** elaborar um programa



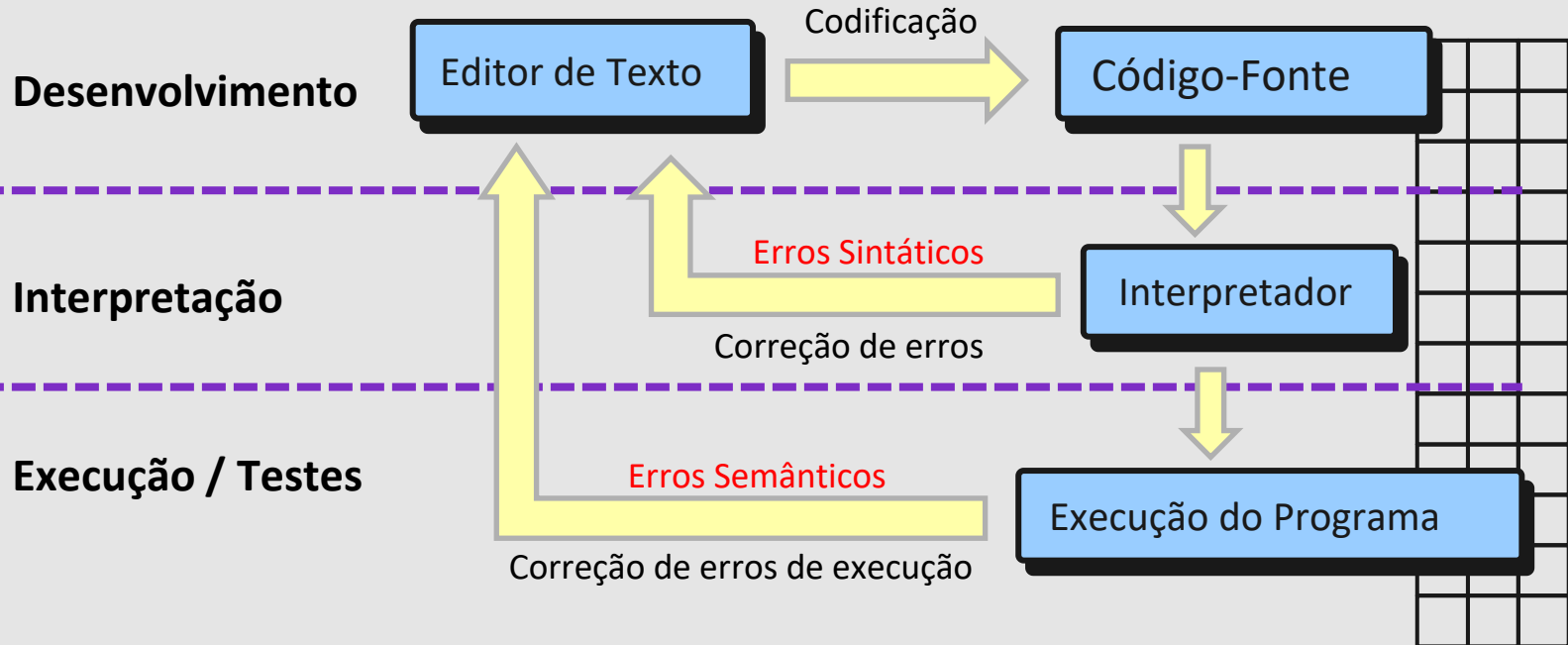
Problema



Programa



Ciclo de construção de um programa



Processo de desenvolvimento

- Análise e Definição do problema
- Projeto do Algoritmo
- Validação do Algoritmo (Teste de Mesa)

Sem computador

- Tradução do algoritmo para uma linguagem de programação (codificação)
- Compilação / Interpretação
- Teste e Depuração
- Execução

Com computador

Python

- * Linguagem de código aberto, lançada em 1991 por **Guido van Rossum**

- * **Simples:**

- Legibilidade
- Produtividade
- Indentação

- * **Flexível:**

- Tipos dinâmicos
- Script x Interativo
- Multiplataforma
- Multiparadigma

- * Página Oficial: <https://www.python.org/>

- * **Curiosidade:** Nome da linguagem inspirado em um grupo de comédia britânico: Monty Python



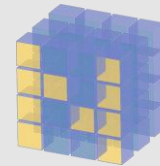
Python – Comunidade


pandas

 Keras

 TensorFlow



 NumPy

django

 Detectron2

pygame

matplotlib

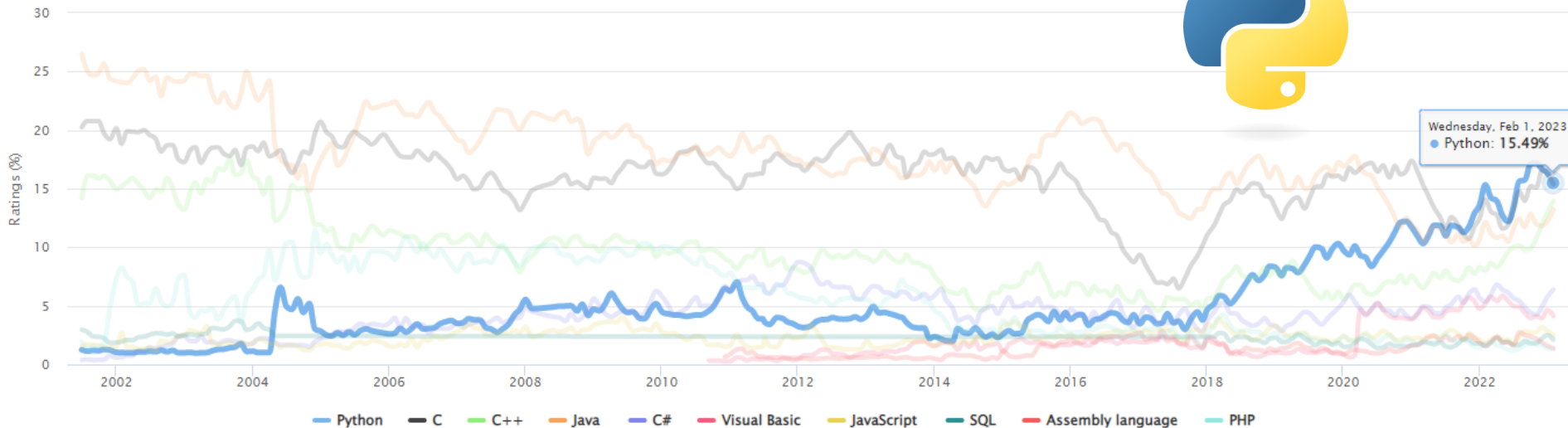
Python é Pop!

TIOBE Programming Community Index

Source: www.tiobe.com



Wednesday, Feb 1, 2023
● Python: 15.49%



Quem usa Python?



Instagram



Dropbox



Google



Pinterest



Spotify®

Uber

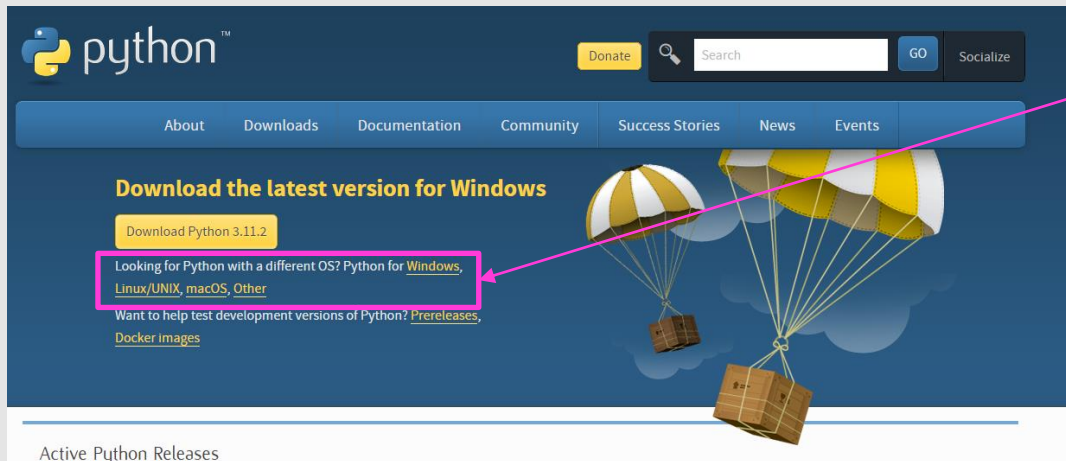
NETFLIX



reddit

Hands on: Instalando Python!

Acesse o site oficial do Python python.org/downloads e selecione a versão apropriada de acordo com o seu sistema operacional

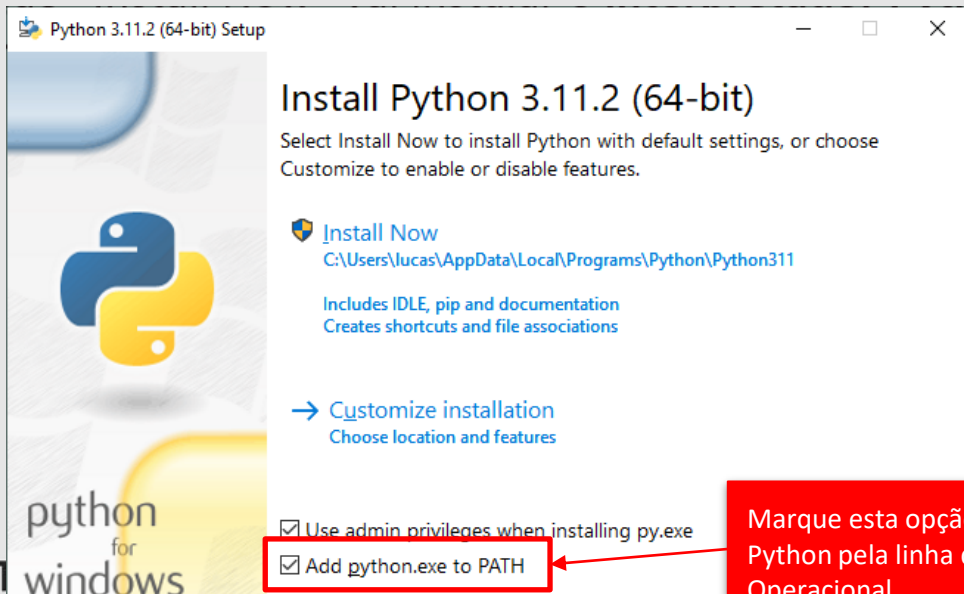


Se você não usa Windows, tem versões para **Mac OS X**, **Linux** e outras plataformas aqui!

Se você já possui o Python instalado em sua máquina, garanta que a versão instalada seja superior à versão 3.10

Hands on: Instalando Python!

A opção “Install Now” vai instalar o **interpretador Python** e o **ambiente de desenvolvimento IDLE** sem necessidade de muitas configurações manuais:

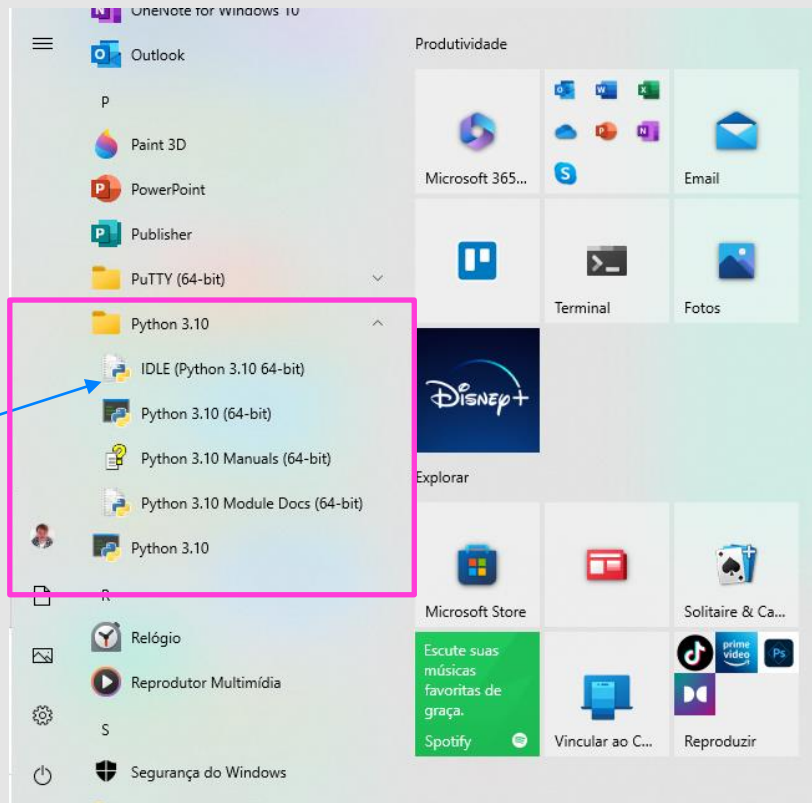


Marque esta opção! Ela vai permitir usar o Python pela linha de comando do Sistema Operacional

Hands on: Instalando Python!

Após instalados o interpretador Python e o IDLE podem ser acessados pelo menu do Windows

Normalmente vamos usar apenas o IDLE para editar e testar os nossos programas.



Hands on: Instalando Python!

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> print("hello world!")
hello world!
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline
    flush: whether to forcibly flush the stream.

>>> |
```

Usamos o editor para criar **scripts** (programas de várias linhas) e executá-los de uma só vez. Normalmente, os scripts são mais utilizados pois vamos resolver problemas que precisarão de várias linhas de código.

Na interface interativa (shell) é possível digitar os comandos um após o outro e ver o resultado imediatamente no console.

```
aula_pratica_01.py - C:\Users\JulianoWickboldt\Google Drive\Teaching\UFRGS\INF01041 - Introdução à Programação em Python\Semana 01 - Introdução...
File Edit Format Run Options Window Help
# Meu primeiro programa
print("Bem-vindos!")
print("Esse é um primeiro programa implementado como script Python.")
print("Tchau!")

Ln: 5 Col: 0
```

Hands on: Instalando Python!

- * Avalie as seguintes expressões usando o modo iterativo:

```
>>> 10 + 20
```

```
>>> 25.9 - 15.157
```

```
>>> 20 / 10
```

```
>>> 5 * 10
```

```
>>> 2 ** 4
```

```
>>> 5 * (3 + 4)
```

```
>>> 5 * 3 + 4
```

Documentação de Referência

A documentação completa de referência da linguagem está disponível em docs.python.org/3/:

Python » Brazilian Portuguese » 3.11.2 » 3.11.2 Documentation »

Download
Baixar esses documentos

Documentação por versão

- Python 3.12 (in development)
- Python 3.11 (stable)
- Python 3.10 (stable)
- Python 3.9 (security-fixes)
- Python 3.8 (security-fixes)
- Python 3.7 (security-fixes)
- Python 3.6 (EOL)
- Python 3.5 (EOL)
- Python 2.7 (EOL)
- Todas versões

Outros recursos

- Índice das PEPs
- Guia do Iniciante
- Lista de Livros
- Material Audiovisual
- Guia do Desenvolvedor do Python

documentação Python 3.11.2

Seja bem-vindol Esta é a documentação oficial do Python 3.11.2,

Partes da documentação:

- [O que há de novo no Python 3.11?](#)
ou todos os documentos "O que há de novo" desde a 2.0
- [Instalando módulos Python](#)
instalando a partir do Python Package Index e outras fontes
- [Distribuindo módulos Python](#)
publicando módulos para instalação por outros
- [Estendendo e Incorporando](#)
tutorial para programadores C/C++
- [API Python/C](#)
referência para programadores C/C++
- [FAQs](#)
perguntas frequentes (com respostas!)
- [Referência da Biblioteca](#)
mantenha isso debaixo do seu travesseiro
- [Referência da Linguagem](#)
descreve sintaxe e elementos da linguagem
- [Configurações e Uso do Python](#)
como usar o Python em diferentes plataformas
- [Python HOWTOs](#)
documentos aprofundados sobre tópicos específicos

Índices e tabelas:

Tem um tutorial bem completo para leitura que você pode usar para estudar.

Essa é a principal referência para as **funções** e **tipos** de dados nativos da linguagem. Como a própria documentação sugere: “*mantenha isso debaixo do seu travesseiro*”.

Alguns comandos para começar

```
print("um texto qualquer")
```

- * Imprime na tela uma mensagem.
- * Essa mensagem deve estar entre aspas duplas ou simples.
- * Cuidado pois imprimir na tela alguns caracteres especiais, como as próprias aspas, vai demandar o uso de caracteres de controle.
- * A documentação de referência da função print está em:
<https://docs.python.org/pt-br/3/library/functions.html#print>

Exemplo:

```
>>> print("Olá Mundo!")  
Olá Mundo!
```

Praticando prints

Utilizando o comando print, imprima as seguintes strings na tela:

`Olá Mundo`

`Olá "Mundo" (com as aspas)`

`'Olá' Mundo`

`'Olá' "Mundo"`

Caracteres Especiais:

`\n` – Quebra de Linha

`\t` – TAB

`\"` – Imprime as aspas

`\\` - Imprime a contra barra \

Praticando prints

Teste os seguintes comandos de print:

```
print("Olá\nMundo")
```

```
print("Olá\tMundo")
```

```
print("Olá\\Mundo")
```

Caracteres Especiais:

\n – Quebra de Linha

\t – TAB

\\" – Imprime as aspas

\\ - Imprime a contra barra \

Alguns comandos para começar

help(um_objeto_qualquer)

- * Imprime na tela uma mensagem de ajuda sobre um comando ou objeto qualquer da linguagem.
- * É uma função para ser usada sempre no modo interativo.
- * Mais detalhes sobre o uso da função em: <https://docs.python.org/pt-br/3/library/functions.html#help>

Exemplo:

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```


Alguns comandos para começar

`dir(um_objeto_qualquer)`

- * Apresenta uma **lista de atributos de um objeto** qualquer.
- * Útil para **explorar objetos** no terminal interativo no ao longo da execução e teste de programas.
- * Por exemplo, é possível descobrir que um objeto de texto ("hello") tem um método upper (além de vários outros) que converte o texto para maiúsculas.
- * Mais detalhes sobre o uso da função em: <https://docs.python.org/3/library/functions.html#dir>

Exemplo:

```
>>> dir("hello")
['_add_', '__class__', '__contains__', '__delattr__',
 '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
 '__getattr__', '__getitem__', '__getnewargs__',
 '__gt__', '__hash__', '__init__', '__init_subclass__',
 '__iter__', '__le__', '__len__', '__lt__', '__mod__',
 '__mul__', '__ne__', '__new__', '__reduce__',
 '__reduce_ex__', '__repr__', '__rmod__', '__rmul__',
 '__setattr__', '__sizeof__', '__str__', '__subclasshook__',
 'capitalize', 'casefold', 'center', 'count', 'encode',
 'endswith', 'expandtabs', 'find', 'format', 'format_map',
 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal',
 'isdigit', 'isidentifier', 'islower', 'isnumeric',
 'isprintable', 'isspace', 'istitle', 'isupper', 'join',
 'ljust', 'lower', 'lstrip', 'maketrans', 'partition',
 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex',
 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
 'translate', 'upper', 'zfill']

>>> "hello".upper()
```