

BCM0505-22 – Processamento da Informação

Apresentação

Carla Negri Lintzmayer
carla.negri@ufabc.edu.br
<http://professor.ufabc.edu.br/~carla.negri/>

Outline

Sobre Mim

Sobre o Curso

Por que aprender a programar?

Como aprender a programar?

Linguagem do curso: Python

Como será esse curso?

Sobre Mim

Sobre Mim



Carla Negri Lintzmayer

- **E-mail:** carla.negri@ufabc.edu.br
- **Sala:** 508-2, Bloco A, Torre 2
- **Página pessoal:**
<http://professor.ufabc.edu.br/~carla.negri>

Sobre o Curso

Objetivos do Curso

Introdução a

- Algoritmos
- Programação de Computadores

Objetivos do Curso

Introdução a

- Algoritmos
- Programação de Computadores

Especificamente:

- Apresentar os fundamentos sobre manipulação e tratamento da informação
- Entender a lógica de programação de computadores
- Adquirir habilidade prática de desenvolver algoritmos básicos

Algoritmos

Um **algoritmo** é uma sequência finita de passos bem definidos que descrevem como executar uma determinada tarefa.

Algoritmos

Um **algoritmo** é uma sequência finita de passos bem definidos que descrevem como executar uma determinada tarefa.

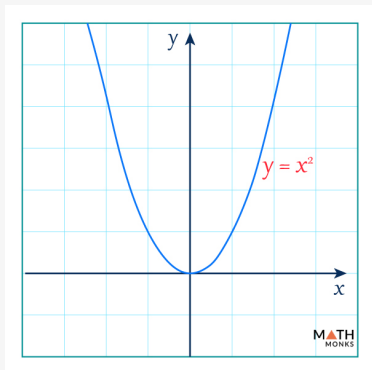
Bolo de Chocolate (Massa)



1. Em um liquidificador adicione 4 ovos, 4 colheres (sopa) de chocolate em pó, 2 colheres (sopa) de manteiga, 3 xícaras (chá) de farinha de trigo, 2 xícaras (chá) de açúcar e 1 xícara (chá) de leite, depois bata por 5 minutos.
2. Adicione 2 colheres (sopa) de fermento e misture com uma espátula delicadamente.
3. Em uma forma untada, despeje a massa e asse em forno médio (180 °C) preaquecido por cerca de 40 minutos.

Algoritmos

Um **algoritmo** é uma sequência finita de passos bem definidos que descrevem como executar uma determinada tarefa.



Resolver uma Equação Quadrática

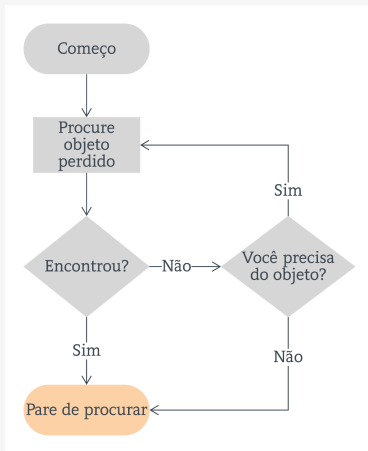
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Algoritmos

Um **algoritmo** é uma sequência finita de passos bem definidos que descrevem como executar uma determinada tarefa.



Buscar um objeto perdido



Algoritmos

Um **algoritmo** é uma sequência finita de passos bem definidos que descrevem como executar uma determinada tarefa.



Tocar uma música

EVIDÊNCIAS

www.superpartituras.com.br

CHITÃOZINHO & XORORÓ

SOM:
RITMO:
TEMPO:

Algoritmos

Um **algoritmo** é uma sequência finita de passos bem definidos que descrevem como executar uma determinada tarefa.



Tocar uma música

EVIDÊNCIAS

SOM:
RITMO:
TEMPO:

CHITÃOZINHO & XORORÓ

Algoritmo é a **ideia (método)** usado para resolver o problema que, em geral, abstrai detalhes computacionais

Algoritmos e Al-Khwarizmi

Al-Khwarizmi (c.780 - c.850)

- Matemático, astrônomo e geógrafo persa
- Conhecido como o pai da álgebra
- Publicou livros que descreviam vários métodos (algoritmos) para solucionar problemas algébricos
- Seu nome deu origem aos termos:
 - Algoritmo
 - Algarismo
 - Algol (linguagem de programação)



Programação de Computadores

- **Programar**: traduzir um algoritmo para um computador

Programação de Computadores

- **Programar**: traduzir um algoritmo para um computador
- Português, Inglês e outras linguagens naturais não são adequadas para programar computadores

Programação de Computadores

- **Programar**: traduzir um algoritmo para um computador
- Português, Inglês e outras linguagens naturais não são adequadas para programar computadores
 - Ambígua: “Vendo o carro”

Programação de Computadores

- **Programar**: traduzir um algoritmo para um computador
- Português, Inglês e outras linguagens naturais não são adequadas para programar computadores
 - Ambígua: “Vendo o carro”
 - Gramaticalmente complexa

Programação de Computadores

- **Programar:** traduzir um algoritmo para um computador
- Português, Inglês e outras linguagens naturais não são adequadas para programar computadores
 - Ambígua: “Vendo o carro”
 - Gramaticalmente complexa
 - Verbo (*transitivo direto, indireto, intransitivo*), Adjetivo, Pronome (*pessoal, possessivo, demonstrativos, interrogativos, relativos, indefinidos*), Substantivos (*abstratos, concretos*) e etc..

Programação de Computadores

- **Programar**: traduzir um algoritmo para um computador
- Português, Inglês e outras linguagens naturais não são adequadas para programar computadores
 - Ambígua: “Vendo o carro”
 - Gramaticalmente complexa
 - Verbo (*transitivo direto, indireto, intransitivo*), Adjetivo, Pronome (*pessoal, possessivo, demonstrativos, interrogativos, relativos, indefinidos*), Substantivos (*abstratos, concretos*) e etc..
 - Orações (*Subordinadas, Substantivas Subjetivas/Predicativas/Objetivas Diretas/Objetivas Indiretas/Completivas Nominais/Apositivas/Justapostas*)

Programação de Computadores

- **Programar**: traduzir um algoritmo para um computador
- Português, Inglês e outras linguagens naturais não são adequadas para programar computadores
 - Ambígua: “Vendo o carro”
 - Gramaticalmente complexa
 - Verbo (*transitivo direto, indireto, intransitivo*), Adjetivo, Pronome (*pessoal, possessivo, demonstrativos, interrogativos, relativos, indefinidos*), Substantivos (*abstratos, concretos*) e etc..
 - Orações (*Subordinadas, Substantivas Subjetivas/Predicativas/Objetivas Diretas/Objetivas Indiretas/Completivas Nominais/Apositivas/Justapostas*)
 - Muitas exceções

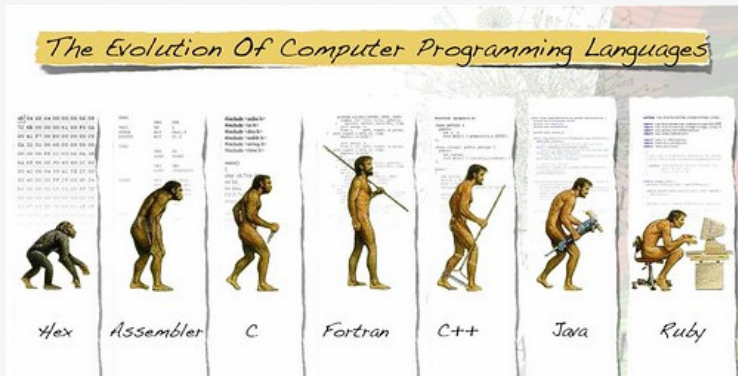
Programação de Computadores

- **Programar:** traduzir um algoritmo para um computador
- Português, Inglês e outras linguagens naturais não são adequadas para programar computadores
 - Ambígua: “Vendo o carro”
 - Gramaticalmente complexa
 - Verbo (*transitivo direto, indireto, intransitivo*), Adjetivo, Pronome (*pessoal, possessivo, demonstrativos, interrogativos, relativos, indefinidos*), Substantivos (*abstratos, concretos*) e etc..
 - Orações (*Subordinadas, Substantivas Subjetivas/Predicativas/Objetivas Diretas/Objetivas Indiretas/Completivas Nominais/Apositivas/Justapostas*)
 - Muitas exceções
- **Solução:** criar uma linguagem para este fim!

Linguagens de Programação



Evolução das Linguagens de Programação



Evolução das Linguagens de Programação

Mother Tonques

Tracing the roots of computer languages through the ages

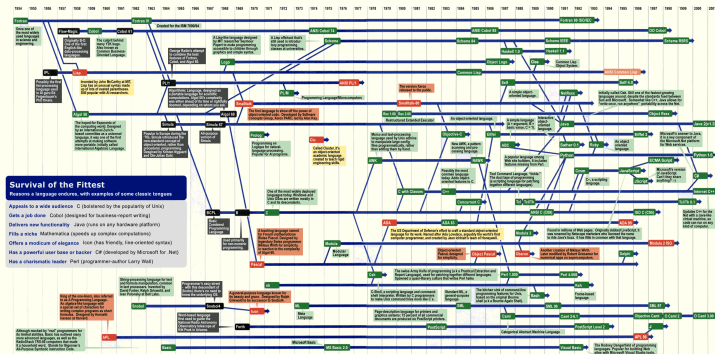
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic technologists, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua francas. Among the most endangered are Ada, APL, B (the predecessor of C), Lisp, Occam, RealTalk, and Simula.

Coder-archaeologist Grady Hall, National Software's chief scientist, is working with the Computer History Museum in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an after failure." Here's a peek at the strongest branches of the tree of code: <http://www.infocast.net/infocast/stories/story.do?storyId=1>. —Michael Mandel

Key

- Year introduced
- Active thousands of users
- Protected: taught at universities; compilers available
- Endangered: usage dropping off
- Extinct: no longer active users or up-to-date compilers
- Language extinction



Sources: Paul Boutin; Brent Hailpern, associate director of computer science at IBM Research; The Papyruscomputing Museum; Todd Probstman, senior researcher at Microsoft; Jim Wetherhold, computer scientist, Stanford University.

Problema: Resolver a Equação $ax^2 + bx + c = 0$

Algoritmo

- calcule o discriminante:
 $\Delta = b^2 - 4ac$
- determine a quantidade de raízes: se $\Delta = 0$, existe apenas uma; se $\Delta < 0$, existem duas raízes imaginárias; caso contrário, existem duas raízes reais
- calcule a(s) raiz(es): $\frac{-b \pm \sqrt{\Delta}}{2a}$

Problema: Resolver a Equação $ax^2 + bx + c = 0$

Algoritmo

- calcule o discriminante:
 $\Delta = b^2 - 4ac$
- determine a quantidade de raízes: se $\Delta = 0$, existe apenas uma; se $\Delta < 0$, existem duas raízes imaginárias; caso contrário, existem duas raízes reais
- calcule a(s) raiz(es): $\frac{-b \pm \sqrt{\Delta}}{2a}$

Algoritmo em Pseudocódigo

```
leia a
leia b
leia c
 $\Delta \leftarrow b^2 - 4ac$ 
Se  $\Delta > 0$  então
     $r_1 \leftarrow \frac{-b + \sqrt{\Delta}}{2a}$ 
     $r_2 \leftarrow \frac{-b - \sqrt{\Delta}}{2a}$ 
    escreva "Raízes:  $r_1, r_2$ "
Senão Se  $\Delta = 0$  então
     $r \leftarrow \frac{-b}{2a}$ 
    escreva "Raiz: r"
Senão
    escreva "Não há raízes reais."
```

Problema: Resolver a Equação $ax^2 + bx + c = 0$

Algoritmo em Pseudocódigo

```
leia a
leia b
leia c
 $\Delta \leftarrow b^2 - 4ac$ 
Se  $\Delta > 0$  então
     $r_1 \leftarrow \frac{-b + \sqrt{\Delta}}{2a}$ 
     $r_2 \leftarrow \frac{-b - \sqrt{\Delta}}{2a}$ 
    escreva "Raízes:  $r_1, r_2$ "
Senão Se  $\Delta = 0$  então
     $r \leftarrow \frac{-b}{2a}$ 
    escreva "Raiz: r"
Senão
    escreva "Não há raízes reais."
```

Python

```
from math import sqrt

a = float(input())
b = float(input())
c = float(input())
D = b**2 - 4*a*c

if D > 0:
    r1 = (((-b) + sqrt(D))/(2*a))
    r2 = (((-b) - sqrt(D))/(2*a))
    print("Raízes: %f, %f" % (r1, r2))
elif D == 0:
    r = (-b) / 2*a
    print("Raiz: ", x)
else:
    print("Não há raízes reais.")
```

Problema: Resolver a Equação $ax^2 + bx + c = 0$

Algoritmo em Pseudocódigo

```
leia a
leia b
leia c
 $\Delta \leftarrow b^2 - 4ac$ 
Se  $\Delta > 0$  então
     $r_1 \leftarrow \frac{-b + \sqrt{\Delta}}{2a}$ 
     $r_2 \leftarrow \frac{-b - \sqrt{\Delta}}{2a}$ 
    escreva "Raízes:  $r_1, r_2$ "
Senão Se  $\Delta = 0$  então
     $r \leftarrow \frac{-b}{2a}$ 
    escreva "Raiz: r"
Senão
    escreva "Não há raízes reais."
```

Java

```
import java.util.Scanner;
public class EquacaoSegundoGrau {

    public static void main(String[] Strings) {
        Scanner input = new Scanner(System.in);
        double a = input.nextDouble();
        double b = input.nextDouble();
        double c = input.nextDouble();
        double D = b * b - 4 * a * c;
        double r1, r2;

        if (D > 0.0) {
            r1 = (-b + Math.pow(D, 0.5)) / (2.0 * a);
            r2 = (-b - Math.pow(D, 0.5)) / (2.0 * a);
            System.out.println("Raízes: " + r1 + ", " + r2);
        } else if (D == 0.0) {
            r1 = -b / (2.0 * a);
            System.out.println("Raiz: " + r1);
        } else {
            System.out.println("Sem raízes reais.");
        }
    }
}
```

Problema: Resolver a Equação $ax^2 + bx + c = 0$

Algoritmo em Pseudocódigo

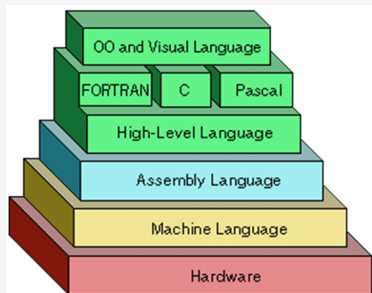
```
leia a
leia b
leia c
 $\Delta \leftarrow b^2 - 4ac$ 
Se  $\Delta > 0$  então
     $r_1 \leftarrow \frac{-b + \sqrt{\Delta}}{2a}$ 
     $r_2 \leftarrow \frac{-b - \sqrt{\Delta}}{2a}$ 
    escreva "Raízes:  $r_1, r_2$ "
Senão Se  $\Delta = 0$  então
     $r \leftarrow \frac{-b}{2a}$ 
    escreva "Raiz: r"
Senão
    escreva "Não há raízes reais."
```

C

```
#include <stdio.h>
#include <math.h>

int main() {
    double a, b, c, r1, r2;
    scanf("%lf %lf %lf", &a, &b, &c);
    double D = b * b - 4 * a * c;
    if (D > 0.0) {
        r1 = (-b + sqrt(D)) / (2.0 * a);
        r2 = (-b - sqrt(D)) / (2.0 * a);
        printf("Raízes: %lf, %lf\n", r1,
            ↵ r2);
    } else if (D == 0.0) {
        r1 = -b / (2.0 * a);
        printf("Raiz: %lf\n", r1);
    } else {
        printf("Não há raízes reais.\n");
    }
}
```

Categorização das Linguagem de Programação: Nível de Abstração



- **Linguagem de Máquina:**
Linguagem compreendida pelo computador
 - Depende da arquitetura (processador) utilizada
- **Linguagem de Baixo Nível:**
Usamos mnemonicos para representar as instruções de máquina
 - Exemplo: Assembler
- **Linguagem de Alto Nível:**
Utiliza instruções próximas da linguagem humana
 - Exemplo: C, C++, Java, Python, Lua, Javascript

Exemplos de Nível de Abstração

Linguagem de Máquina

```
00010000 01010010 01011110 00011010
00111011 00011010 00111110 00101010
01010101 00111110 01010010 01101010
01001001 11001010 01110111 11101010
00000000 01010010 01010110 01001010
```

Baixo Nível

```
.data
prompt: .asciiz "Resultado: "
.text
.globl main
main:
    li $t0, 5
    li $t1, 7
    add $t2, $t0, $t1
    li $v0, 4
    la $a0, prompt
    syscall
    li $v0, 1
    move $a0, $t2
    syscall
    li $v0, 10
    syscall
```

Alto Nível

```
#include <stdio.h>

int main() {
    int num1 = 5, num2 = 7;
    int soma = num1 + num2;
    printf("Resultado: %d\n", soma);
    return 0;
}
```

Categorização das Linguagem de Programação: Paradigma

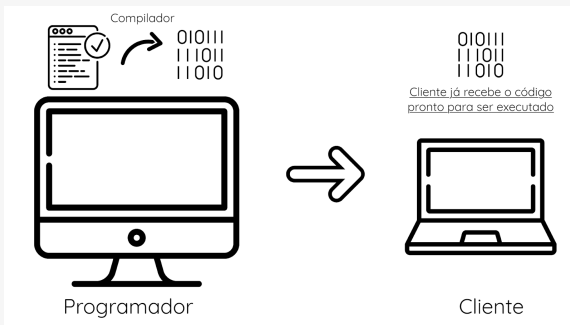
- **Imperativo/Estruturado:** Instruções sequências que modificam o estado do programa
 - Exemplos: C, Python, Java, Assembly
- **Funcional:** Baseado em funções matemáticas puras e imutabilidade
 - Exemplos: Haskell, Lisp, Scala
- **Lógico:** Baseado em regras e inferências lógicas
 - Exemplos: Prolog

Categorização das Linguagem de Programação: Modo Execução

- **Compilada:** análise e transformação do código ocorrem antes da sua execução
- **Interpretada:** a execução do código se dá junto à sua análise
- **Híbrida** (meio termo entre as outras duas)

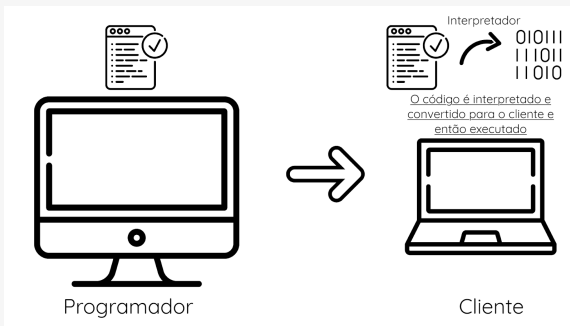
Linguagens Compiladas

- O código-fonte, em linguagem de alto nível, é convertido por um programa **compilador** em linguagem de baixo nível (linguagem de máquina) gerando um **programa**
- O programa é executado (lido) diretamente pelo processador
- **Exemplos:** C, C++, Rust, Go
- **Vantagens:** execução mais rápida; economia de recursos; proteção do código; minimização de erros
- **Desvantagens:** dependem de arquitetura



Linguagens Interpretadas

- O código-fonte é lido e executado *linha por linha* por um programa **interpretador**
- **Exemplos:** Python, JavaScript, Lua, Perl, R
- **Vantagens:** não depende da arquitetura; flexibilidade
- **Desvantagens:** costumam ser mais lentas; consomem mais recursos; o cliente precisa de um interpretador



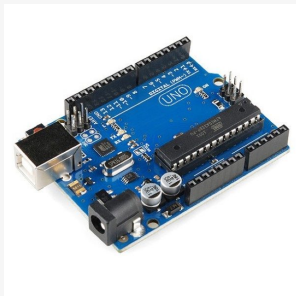
Por que aprender a programar?

Programar?

- Necessário para quem vai fazer Ciência da Computação e/ou vai ser programador
- Importante para automatizar algum processo
- Importante para fazer simulações e testes antes de criar ferramentas e protótipos
- Importante para usar bem as funcionalidades de planilhas (ou mesmo para substituí-las)
- Importante para tratar dados de forma automatizada
- Importante para fazer um website
- Importante para extrair informações de websites
- Importante para detectar situações onde uma solução computacional pode trazer benefícios

Automatizar tarefas no mundo físico

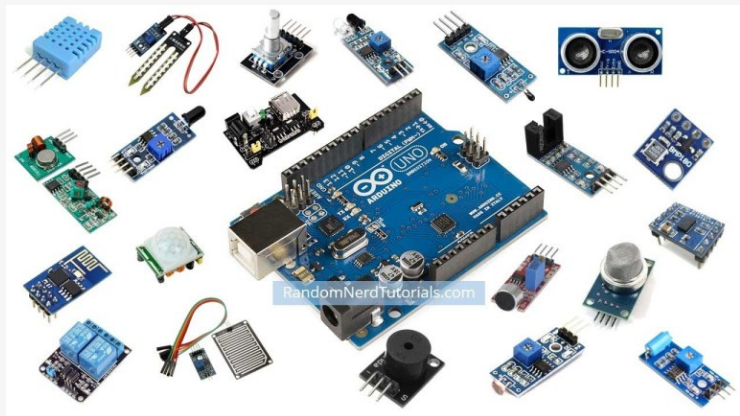
Arduino



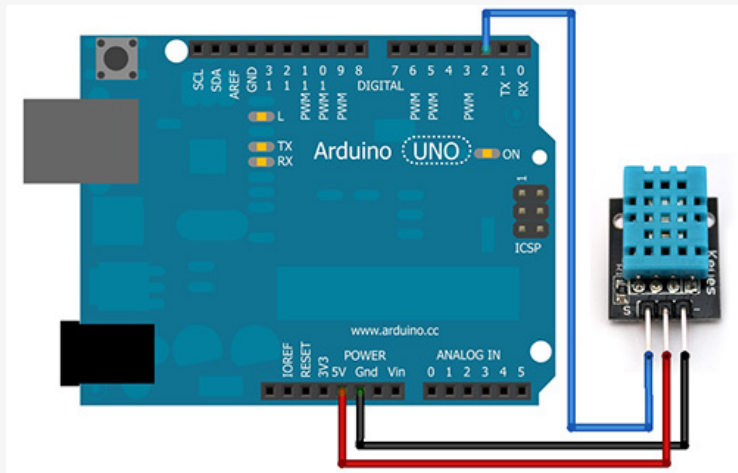
R\$ 26,58

```
1  #include <LiquidCrystal.h>
2  #include "DHT.h"
3  #define DHTTYPE DHT11
4  #define DHTPIN 8
5
6  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
7  DHT dht(DHTPIN, DHTTYPE);
8
9  void loop() {
10     delay(500);
11     lcd.setCursor(0, 1);
12     float h = dht.readHumidity();
13     float f = dht.readTemperature(true);
14     if (isnan(h) || isnan(f)) {
15         lcd.print("ERROR");
16         return;
17     }
18     lcd.print(f);
19     lcd.setCursor(7,1);
20     lcd.print(h);
21 }
```

Arduino: Sensores e Módulos



Arduino: Sensores e Módulos



Como aprender a programar?

Paciência

- Programar exige muita paciência e atenção a detalhes
 - A maioria dos problemas não são resolvidos na primeira tentativa
 - Uma vírgula ou parênteses errados podem estragar tudo

Paciência

- Programar exige muita paciência e atenção a detalhes
 - A maioria dos problemas não são resolvidos na primeira tentativa
 - Uma vírgula ou parênteses errados podem estragar tudo
- Programar exige começar com “baby steps”
 - Primeiros programas muito simples
 - Resolver vários problemas simples

Paciência

- Programar exige muita paciência e atenção a detalhes
 - A maioria dos problemas não são resolvidos na primeira tentativa
 - Uma vírgula ou parênteses errados podem estragar tudo
- Programar exige começar com “baby steps”
 - Primeiros programas muito simples
 - Resolver vários problemas simples
- Aprender a programar **não é** o mesmo que decorar todos os comandos e nomes estranhos
 - Ao trocar para outra linguagem de programação, se você realmente aprendeu algoritmos, terá pouca dificuldade para aprender a nova linguagem

Perseverança

- Ninguém consegue correr uma maratona sem treinar



Perseverança

- Ninguém consegue correr uma maratona sem treinar



- Não recorra a outros meios que não o seu próprio treinamento
 - Mais claramente: não copie códigos dos amigos, não utilize ferramentas de IA

Linguagem do curso: Python

Sobre Python



Python é uma linguagem de programação de alto nível e multipropósito (web, GUI, CLI, mobile, etc..)

Sobre Python

- Código aberto (open source)
- Inventada nos Países Baixos no começo dos anos 90 por Guido van Rossum
- *Benevolent Dictator for Life* (BDFL)
- Nomeada em homenagem ao programa de TV **Monty Python's Flying Circus**
- É interpretada e multi-paradigma



Sobre Python

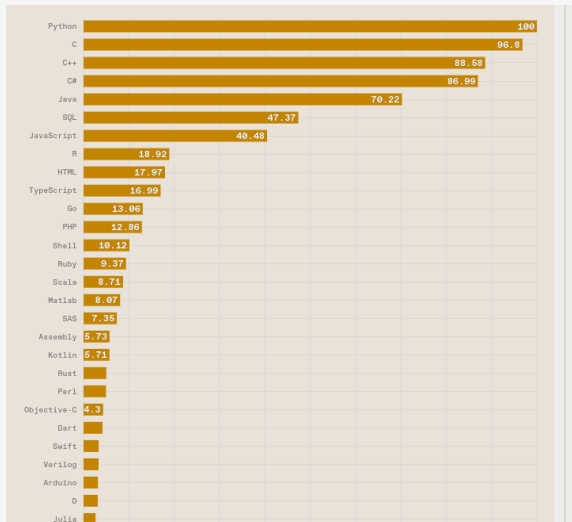
- Código aberto (open source)
- Inventada nos Países Baixos no começo dos anos 90 por Guido van Rossum
- *Benevolent Dictator for Life* (BDFL)
- Nomeada em homenagem ao programa de TV **Monty Python's Flying Circus**
- É interpretada e multi-paradigma



Por que Python?

- Fácil de aprender
- Fácil de usar
- Versátil (vem com baterias!)

Por que Python?



Fonte: <https://spectrum.ieee.org/top-programming-languages-2022>

Quem usa Python?

- Wikipedia
- Google
- Yahoo!
- Microsoft
- Meta (Facebook/Instagram/Whatsapp)
- Amazon
- CERN
- NASA
- Spotify

Como será esse curso?

Todas as informações

[http://professor.ufabc.edu.br/~carla.negri/cursos/2025Q1-PI/
index.html](http://professor.ufabc.edu.br/~carla.negri/cursos/2025Q1-PI/index.html)

Aulas

- Durante a primeira hora, aproximadamente:
 - Aula expositiva
 - Monitores (ou mesmo computadores) desligados
- Restante da aula:
 - Exercícios (juntos ou não)

Faltas

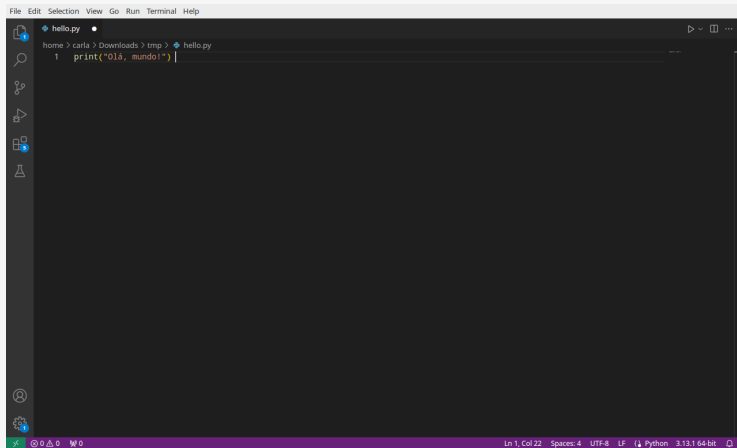
- Serão feitas duas chamadas durante a aula, uma para cada hora
 - Realizadas por meio de leitura de um QR-code através do seu celular, via Moodle
 - Ou, com apresentação da carteirinha para mim, caso dê algum problema
 - Quando eu avisar que haverá chamada, você deve se preparar, logando no Moodle pelo seu celular
- Essa disciplina tem carga didática de 48 horas
 - Você pode faltar em 12 dessa horas, sem necessidade de justificativa
- Faltas **não** são abonadas, exceto em dias de prova
 - Não há necessidade de me enviar justificativa se você faltou menos de 12 horas
 - Se você faltou 13 ou 14 horas e não desistiu do curso, converse comigo
 - E garanta que você tem vale moral...

Usaremos Python

- Nós não temos como objetivo aprender tudo sobre Python
 - Nem teríamos tempo para isso...
- Nós vamos usar Python como ferramenta para implementar os algoritmos que vamos criar
 - Criar algoritmos usando lógica de programação é nosso objetivo!
- Usaremos, sim, várias facilidades oferecidas pelo Python
 - Porém evite utilizar comandos diferentes daqueles vistos em sala de aula

Como usaremos Python?

- Visual Studio Code (VSCode)

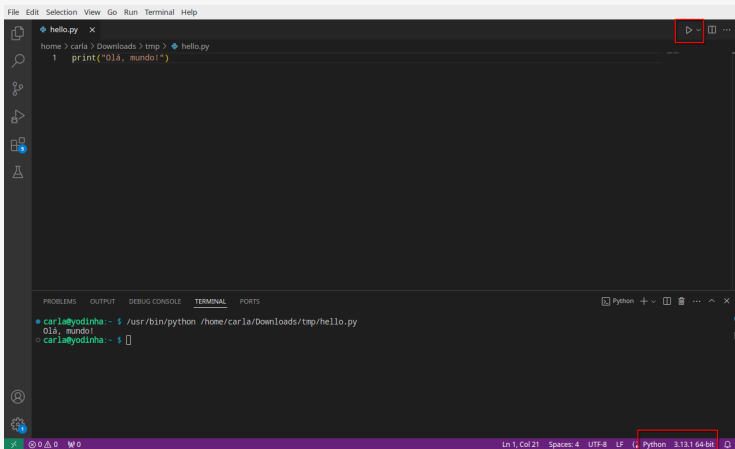


Meu Primeiro Programa

```
1  print("Olá, mundo!")
```

Meu Primeiro Programa

```
1 print("Olá, mundo!")
```



The screenshot shows a code editor window with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The editor has a tab for 'hello.py'. The code in the editor is:

```
home > carla > Downloads > tmp > hello.py
1 print("Olá, mundo!")
```

Below the editor is a terminal window. The terminal shows the command to run the script and its output:

```
carla@yodinha:~$ /usr/bin/python /home/carla/Downloads/tmp/hello.py
Olá, mundo!
carla@yodinha:~$
```

At the bottom of the editor, the status bar shows 'Ln 1, Col 21', 'Spaces: 4', 'UTF-8', 'LF', and 'Python 3.13.1 64-bit'. There are also icons for running, debugging, and other actions.

Meu Segundo Programa

```
1 nome = input("Qual o seu nome? ")
2 print("Olá, ", nome)
```

Por hoje é isso

