

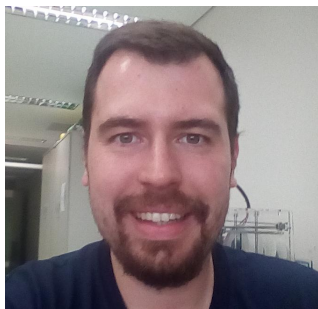
# CC1612

# Fundamentos de Algoritmos

---

Prof. Danilo H. Perico

# Teoria



**Prof. Danilo Hernani Perico**

dperico@fei.edu.br

## Formação:

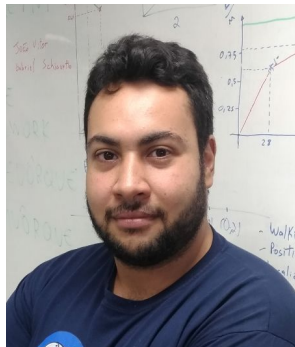
- Técnico em Informática (Etec lauro Gomes)
- Engenheiro Eletricista com Ênfase em Eletrônica (FEI)
- Mestre em Inteligência Artificial Aplicada à Automação (FEI)
- Doutor em Inteligência Artificial Aplicada à Automação (FEI)



**Currículo Lattes**

<http://lattes.cnpq.br/5676806579817092>

# Laboratório



**Prof. Isaac Jesus da Silva**

isaacjesus@fei.edu.br

## Formação:

- Técnico em Mecatrônica (SENAI Armando Arruda Pereira)
- Engenheiro Eletricista com Ênfase em Eletrônica (FEI)
- Mestre em Inteligência Artificial Aplicada à Automação (FEI)
- Doutor em Inteligência Artificial Aplicada à Automação (FEI)



**Currículo Lattes**

<http://lattes.cnpq.br/4632647003093335>

# Fundamentos de Algoritmos

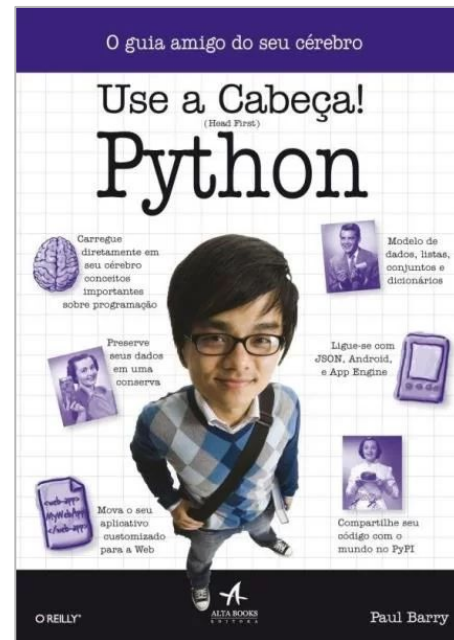
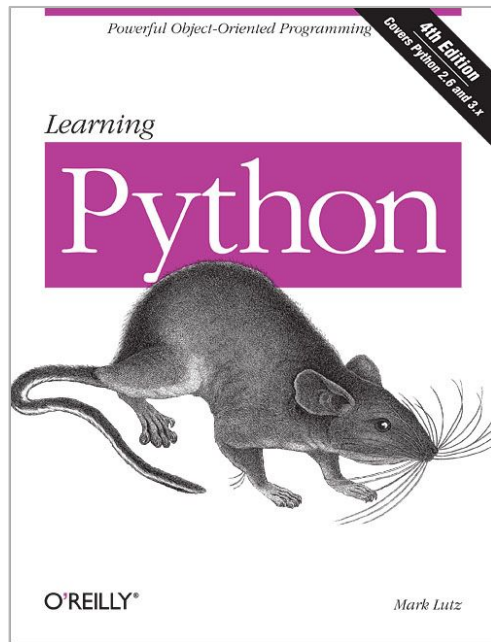
Objetivo da disciplina:

- Capacitar o aluno com a formação básica em programação de computadores, permitindo que possa converter e resolver problemas utilizando linguagem algorítmica, colaborando com o desenvolvimento do raciocínio lógico

# Conteúdo Programático

- Introdução à disciplina / Conceitos básicos: história da programação / computadores / Algoritmos / Python
- Variáveis / Entradas e saídas de dados
- Estruturas condicionais / operadores lógicos
- Estruturas de repetição
- Listas / Listas aninhadas (Matrizes)
- Funções / Arquivos
- PROJETO (PJ)
- Strings
- Dicionários
- Interface Gráfica
- PROVA (PV)
- SUB (Prova teórica)

# Bibliografia Básica



# Bibliografia Complementar



# Cr terios de Avalia  o

$$NF = (LAB + 2*PJ + 3*PV) / 6$$

- LAB = atividades realizadas no Laborat rio
- PJ = Projeto
- PV = Prova
  
- SUB = Substitutiva - caso a m dia n o seja alcan ada, atividade ou prova que substitui a pior nota entre PJ e PV



# Breve História da Programação

---

## 1801 – Tear de Jacquard (*Jacquard Loom*)



- Joseph Marie Jacquard inventa um tear mecânico automatizado;
- Utiliza cartões perfurados para o controle das operações



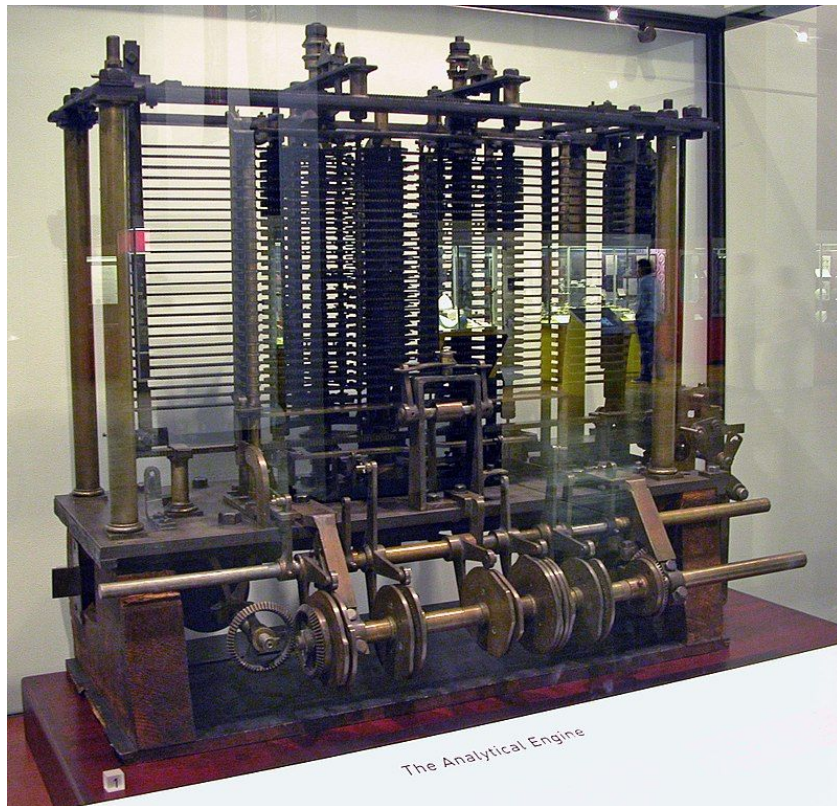
Holes in the cards control which threads are raised for weaving the pattern

## 1842/43 – Ada Lovelace



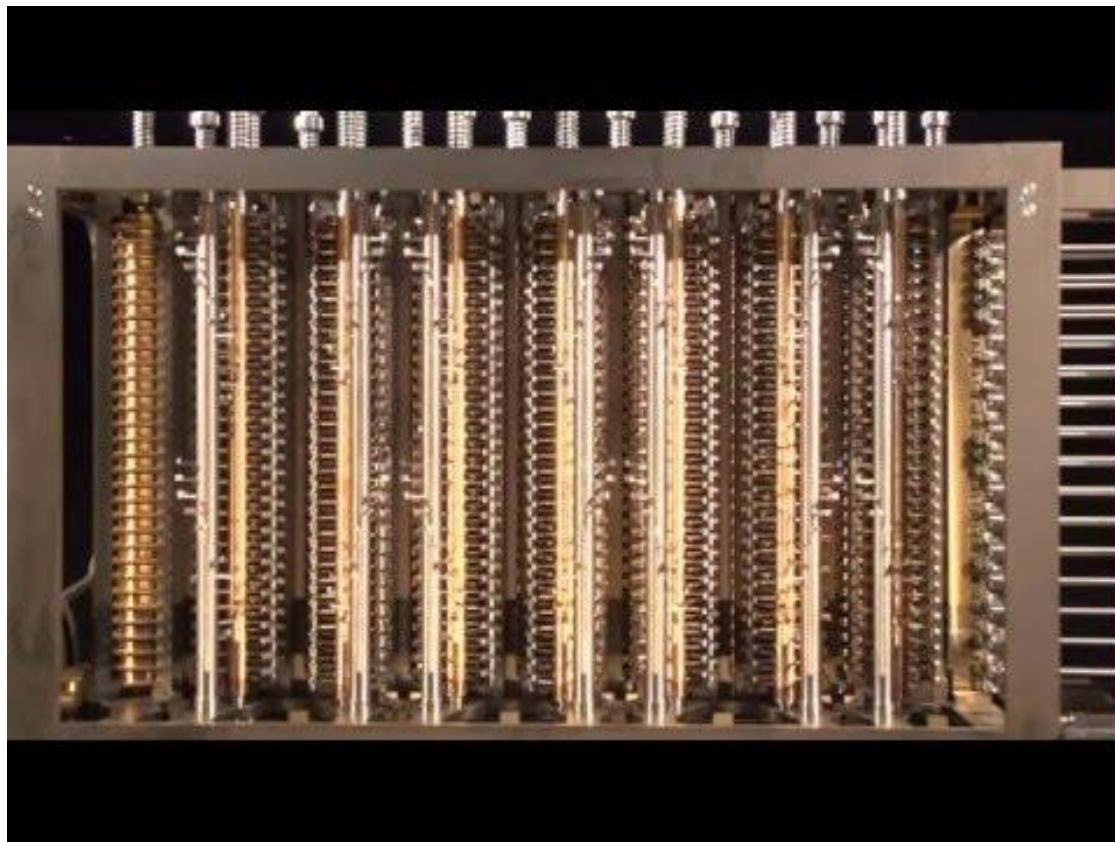
- Escreve o primeiro algoritmo para a máquina analítica de Charles Babbage.
- Calculava a sequência de Bernoulli

# Máquina Analítica de Babbage





# Máquina Diferencial de Babbage

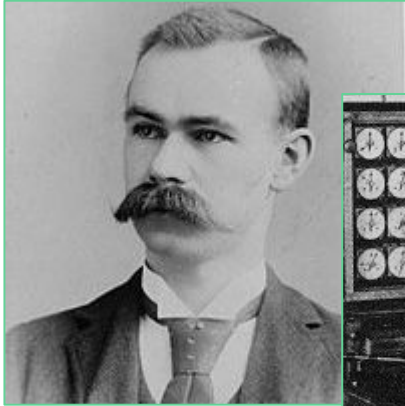


# Como fazer contas com engrenagens?



<https://youtu.be/v8BqNi9cj5M>

# 1889 – Herman Hollerith



- Cartões perfurados para codificar informações do censo de 1890 nos EUA.
- Máquinas elétricas que somavam a contagem dos dados.



# Hollerith

Fundou a empresa Tabulating Machine Company (TMC), que hoje é a IBM.



*IBM - International Business Machines*



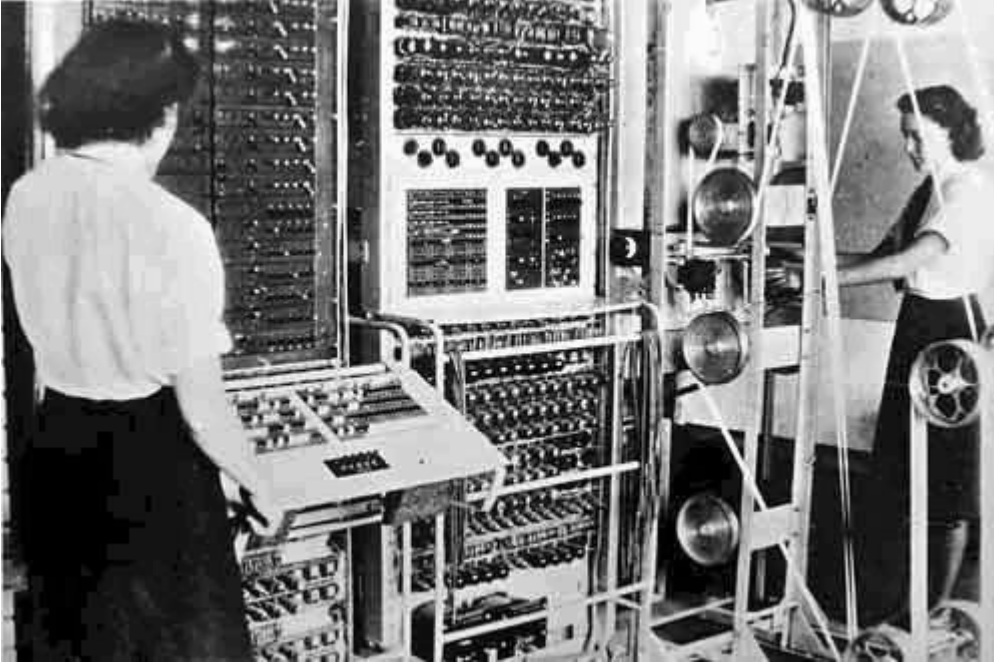
# 1937 – 1942 - Atanasoff-Berry Computer (ABC)

---



- Primeiro computador eletrônico digital

# 1943 - Colossus

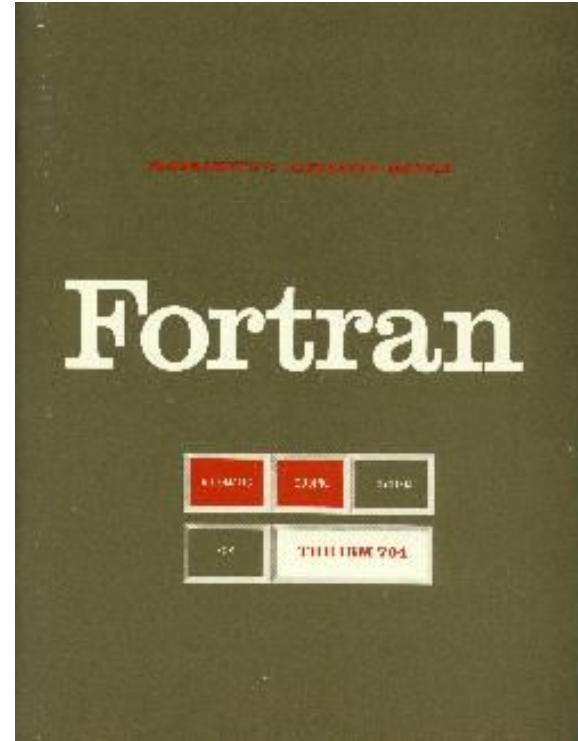


- Desenvolvido para decifrar e ler os códigos das mensagens alemãs durante a II Guerra Mundial.

# 1954 – Primeira linguagem de programação de alto-nível



Inventada por John Backus, da IBM



# 1961 – Primeiro jogo de computador: SPACEWAR!



<https://spacewar.oversigma.com/>

# 1983 – Primeiro vírus de computador

- Criado por Fred Cohen
- O programa podia se auto copiar e infectar outros computadores por meio dos disquetes
- O programa não prejudicava o computador... Cohen só queria provar que isso era possível



# Conceitos básicos: Computadores

---



# Computadores

- Para o computador funcionar, duas coisas muito importantes são necessárias:
  - **Peças (Hardware)**
  - **Programas (Softwares)**

# Hardware

- É a parte física da máquina (em inglês significa *equipamento pesado*).
- Como exemplo, podemos citar as memórias, placas de vídeo, placa mãe, processadores, entre outros.

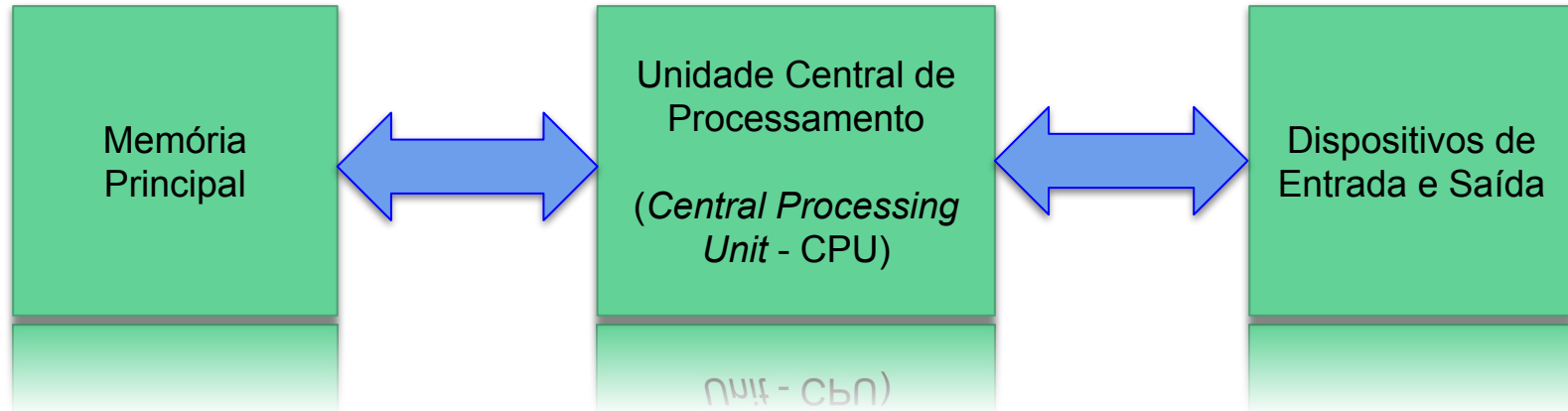


# Software

- São os programas que, utilizando o hardware do computador, executam as diferentes tarefas necessárias ao processamento de dados.



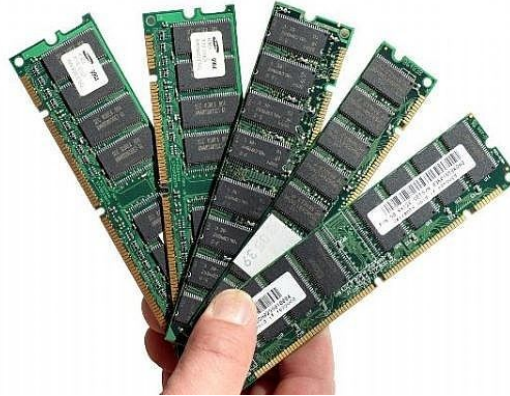
# Computador - Arquitetura Von Neumann



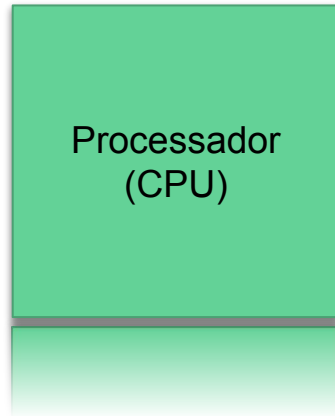
# Computador - Arquitetura Von Neumann

Memória  
Principal

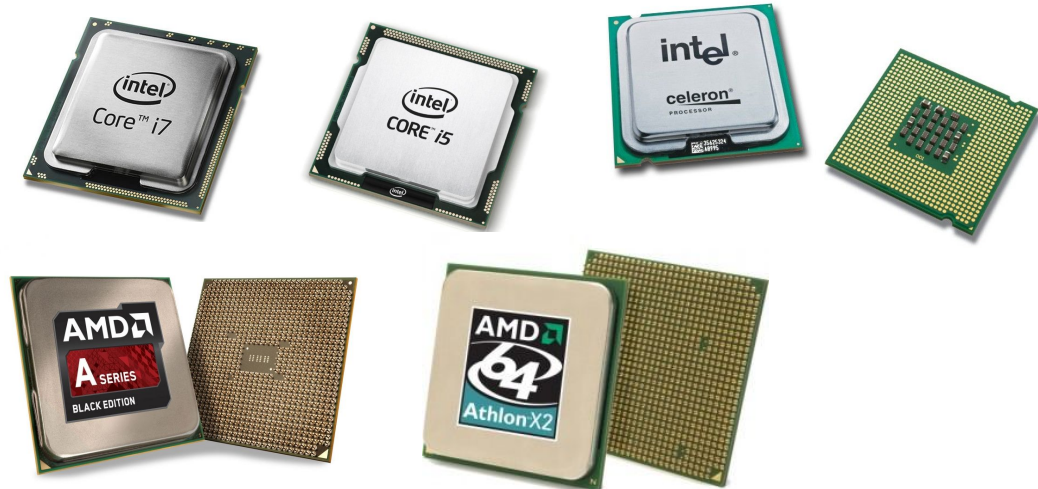
- Memória RAM (*Random Access Memory*)
  - Rápida e diretamente acessível pela CPU
  - Volátil (dados armazenados temporariamente)



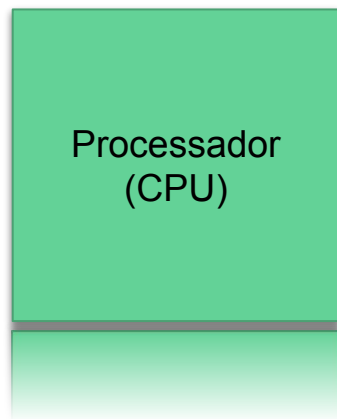
# Computador - Arquitetura Von Neumann



- Realiza operações que modificam o conteúdo armazenado na Memória Principal
- É o processador que executa as instruções de um programa



# Computador - Arquitetura Von Neumann



- Dividido em duas partes:
  - Unidade Lógica Artimética (ULA)
    - Realiza todas as operações lógicas e aritméticas (adição, subtração, divisão, multiplicação etc.)
  - Unidade de Controle (UC)
    - É responsável por controlar a comunicação das informações entre a Memória Principal e a ULA

# Computador - Arquitetura Von Neumann

Dispositivos de  
Entrada e Saída

- Entrada



- Saída





# Computador - Tipos de Armazenamento

Além da Memória Principal (RAM), o computador tem outros tipos de armazenamento:

- Memória Cache
- ROM (*Read-Only Memory*)
- Armazenamentos Secundários

# Computador - Tipos de Armazenamento

- Memória Cache:
  - Volátil
  - Trabalha em conjunto com o processador
  - Todos os processadores atuais trazem uma certa quantidade de memória cache embutida no encapsulamento.
  - Mais rápida do que a RAM
  - Ela foi criada para fornecer dados cruciais ao processador, pois a memória RAM é mais lenta do que o processador

# Computador - Tipos de Armazenamento

- ROM (*Read-Only Memory*)
  - Não volátil
  - A ROM é quase sempre utilizada para armazenar *firmwares*

*Firmwares*: pequenos softwares que funcionam apenas no hardware para o qual foram desenvolvidos e que controlam as funções mais básicas do dispositivo.

- Alguns tipos de memória ROM programáveis:
  - PROM, EPROM, EEPROM



# Computador - Tipos de Armazenamento

- Armazenamento Secundário:
  - Armazenamento não volátil (permanente)
  - Realizado em dispositivo externo:
    - *Hard-Disk (HD), CD, DVD, Solid-State Drive (SSD), Flash Drive*



# Armazenamento e Processamento de Dados

- Para armazenar e processar informações, o computador utiliza o conceito de ***bit***

# Bit

- Bit - simplificação de Dígito Binário (**Binary Digit**)
- **É a menor unidade de informação que pode ser armazenada ou transmitida**
- Um bit pode assumir somente 2 valores:
  - 0 - (representa *desligado*)
  - 1 - (representa *ligado*)
- Um conjunto de 8 bits é chamado de **byte**
- Cada byte tem 256 valores possíveis ( $2^8$ )

# Armazenamento e Processamento de Dados

- Bits são utilizados porque para o computador (formado por circuitos eletrônicos e transistores) tudo é interpretado como ***ligado (1)*** ou ***desligado (0)***, ***conduzindo corrente elétrica (1)*** ou ***não (0)***.

# Organização das Memórias

- As memórias são estruturadas em conjuntos ordenados de **bits**, denominadas células
- A maioria dos computadores utiliza **1 byte** como tamanho da célula
- Cada célula é identificada por um número, chamado endereço



# Organização das Memórias - Exemplo

- Memória com 8 células de 8 bits (1 byte), assim o sistema de endereçamento pode ser estruturado com valores binários de 3 bits:
  - endereço 0 - 000
  - endereço 1 - 001
  - endereço 2 - 010

endereço (3 bits)	célula (8 bits)
000	11001001
001	11011011
010	11101101
011	01011100
100	11100111
101	00100101
110	11111010
111	00001100

# Processador

- Como o processador (CPU) trabalha em conjunto com as memórias, ele também recebe, processa e retorna dados binários (0 ou 1).
- Os computadores pessoais atuais tem a capacidade de processar 32 ou 64 bits em um ciclo de processamento (sistemas operacionais de 32 ou 64 bits).

# Capacidade de Armazenamento

- O armazenamento das memórias não voláteis (permanente) é também medido em Bytes:
  - Kilo(byte): 1.000 ou 1.024 ( $2^{10}$ ) bytes
  - Mega(byte): 1.000.000 ou 1.048.576 ( $2^{20}$ ) bytes
  - Giga(byte): 1.000.000.000 ou 1.073.741.824 ( $2^{30}$ ) bytes
  - Tera(byte): 1.000.000.000.000 ou 1.099.511.627.776 ( $2^{40}$ ) bytes

# Bases de Numeração

- Vimos que o computador utiliza a base binária (**0 ou 1**) para processar e armazenar informações.
- Além da base binária, existem outras:
  - Decimal (**0, 1, 2, 3, 4, 5, 6, 7, 8 e 9**)
  - Octal (**0, 1, 2, 3, 4, 5, 6 e 7**)
  - Hexadecimal (**0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F**)

# Bases de Numeração

- Conversões numéricas entre estas bases são utilizadas em muitos casos na computação.
- Algumas razões:
  - Somos acostumados com a base numérica decimal
  - As bases octal e hexadecimal são muito utilizadas pela fácil representação (uma representação binária pode ter muitos dígitos).

# Bases de Numeração - Exemplo

- Decimal: 10759
- Binário:  $10101000000111_2$
- Octal:  $25007_8$
- Hexadecimal:  $2A07_{16}$
- O número subscrito representa a base em que o número está representado

# Conversões entre as Bases de Numeração

1ª Conversão: Decimal para Binário

Converter o número 34 para binário

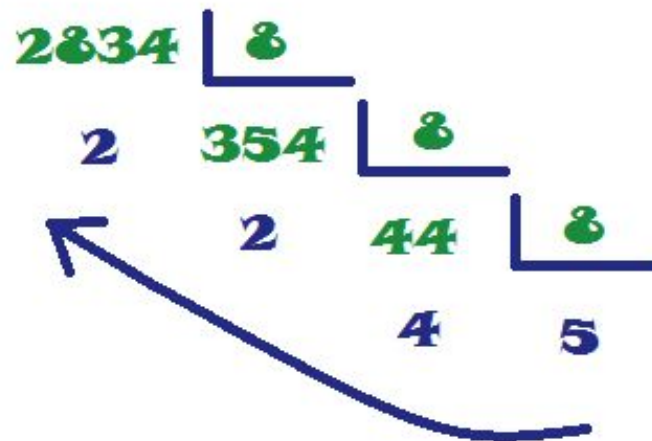


# Conversões entre as Bases de Numeração

2ª Conversão: Decimal para Octal

Converter o número 2834 para octal

**Conversão de Decimal para Octal**



**Resultado: 5422<sub>8</sub>**

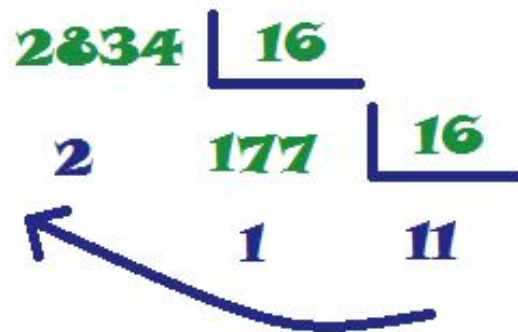


# Conversões entre as Bases de Numeração

3ª Conversão: Decimal para Hexadecimal

Converter o número 2834 para hexadecimal

**Conversão de Decimal para Hexadecimal**



**Resultado: B12<sub>16</sub>**

# Conversões entre as Bases de Numeração

## 4ª Conversão: Binário para Decimal

Converter o número  $100010_2$  para decimal

Primeiro: invertemos o número

$$100010 \Rightarrow 010001$$

Depois: somamos cada número multiplicado por 2 elevado a um número sequencial que começa de 0

$$0 * 2^0 + 1 * 2^1 + 0 * 2^2 + 0 * 2^3 + 0 * 2^4 + 1 * 2^5$$

Resultado da somatória:

**34**

# Conversões entre as Bases de Numeração

5ª Conversão: Octal para Decimal

Converter o número  $5422_8$  para decimal

Primeiro: invertemos o número

$$\mathbf{5422 \Rightarrow 2245}$$

Depois: somamos cada número multiplicado por 8 elevado a um número sequencial que começa de 0

$$\mathbf{2 * 8^0 + 2 * 8^1 + 4 * 8^2 + 5 * 8^3}$$

Resultado da somatória:

$$\mathbf{2834}$$

# Conversões entre as Bases de Numeração

6ª Conversão: Hexadecimal para Decimal

Converter o número  $B12_{16}$  para decimal

Primeiro: invertemos o número

$$\mathbf{B12 \Rightarrow 21B}$$

Depois: somamos cada número multiplicado por 16 elevado a um número sequencial que começa de 0

$$\mathbf{2 * 16^0 + 1 * 16^1 + 11 * 16^2}$$

Resultado da somatória:

$$\mathbf{2834}$$

# Conversões entre as Bases de Numeração

7ª Conversão: Binário para Hexadecimal

Converter o número  $10011011101_2$  para hexadecimal

Primeiro: separamos os dígitos em grupos de 4

**0100 1101 1101**

Depois: fazemos a conversão binário-decimal para cada grupo separadamente (4ª conversão)

**4 13 13**

Então: Trocamos os números maiores que 9 por letras

**4DD<sub>16</sub>**

# Conversões entre as Bases de Numeração

8ª Conversão: Hexadecimal para Binário

Converter o número  $4DD_{16}$  para binário

Primeiro: separamos os dígitos do número

**4 D D**

Depois: fazemos a conversão das letras para número

**4 13 13**

Então: fazemos a conversão de cada número separadamente para binário como se fosse número da base decimal (1ª conversão):

**$10011011101_2$**

# Exercícios

1. Converta  $160$  para binário e hexadecimal
2. Converta  $FA1_{16}$  para decimal e binário
3. Converta  $111010_2$  para decimal e hexadecimal
4. Converta  $100110101011110011011110_2$  para decimal e hexadecimal

# Algoritmos

---



# Algoritmos

- Algoritmo é uma sequência de passos que visa atingir um objetivo bem definido (FORBELLONE, 1999).
- Algoritmo é uma sequência finita de instruções bem definidas e não ambíguas
- O conceito de algoritmo existe há séculos e o uso do conceito pode ser atribuído aos matemáticos gregos: [Eratóstenes](#) e [Euclides](#), por exemplo.



Eratóstenes



Euclides

# Algoritmos

Executamos vários algoritmos no dia-a-dia, como por exemplo:

## **Somar dois números inteiros**

1. Precisamos receber os dois números
2. Então somamos o primeiro número com o segundo
3. Então, exibimos o resultado

# Algoritmos - Exercícios

1. Faça um algoritmo para trocar um pneu furado do carro.
2. Faça um algoritmo para trocar uma lâmpada.
3. Faça um algoritmo para sacar dinheiro no banco 24 horas.

# Algoritmos

- Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa.
- Sua implementação pode ser feita por um computador, por outro tipo de autômato ou mesmo por um ser humano.
- O conceito de algoritmo foi formalizado em 1936 pela Máquina de Turing de [Alan Turing](#) e pelo cálculo lambda de [Alonzo Church](#), que formaram as primeiras fundações da Ciência da Computação.

# Como representar um algoritmo?

- **Descrição Narrativa:** escrever em linguagem natural os passos que devem ser seguidos para solucionar um problema.
- **Fluxograma:** escrever os passos que devem ser seguidos utilizando símbolos gráficos predefinidos.
- **Pseudocódigo:** escrever por meio de regras predefinidas os passos necessários para se solucionar o problema.

# Python



# Python

- É uma Linguagem de Programação!
- Linguagens de programação são usadas como um meio de comunicação entre os computadores e os humanos
- Codificam os algoritmos para uma linguagem que o computador pode entender
- Língua de alto nível
- Interpretada

# Tipos de Linguagem de Programação

- Baixo nível:
  - mais próximas da maneira que o computador trabalha. São mais rápidas, porém mais difíceis de se trabalhar / programar.
- Alto nível:
  - são mais próximas da nossa linguagem; ações são representadas por palavras de ordem (faça, imprima etc). Mais fáceis de se trabalhar; mais lentas do que as de baixo nível.



# Linguagem Compilada vs. Interpretada

## Compilada:

Executam o código diretamente no Sistema Operacional ou Processador.

- São rápidas
- Ex.: C, C++, Fortran, Pascal

## Interpretada:

Necessitam de outro programa (interpretador) capaz de traduzir e executar o código fonte e executar os comandos no SO ou Processador.

- Mais lentas
- Ex.: JavaScript, Python, C#

# Python - História

- A Linguagem Python foi concebida no fim dos anos 80.
- A primeira ideia de implementar o Python surgiu mais especificamente em 1982 por [Guido Van Rossum](#)
- 1991: lançada a primeira versão do Python, então denominada de v0.9.0.



# Python - Origem do nome

- Guido sabia que não queria siglas;
- Ele queria que o nome da linguagem fosse marcante e forte
- Não fazia questão que o nome possuísse um significado profundo
- Foi então que Guido usou a primeira coisa que veio a sua cabeça: **Monty Python's**.



*Algumas vezes conhecido como os Pythons, foram um grupo de comédia britânico.*

# Python Software Foundation

- Em 2001 foi criada a Python Software Foundation (PSF):
  - uma organização sem fins lucrativos
  - tem como objetivo ser dona de qualquer propriedade intelectual relacionada ao Python
  - e como missão promover e proteger o avanço da linguagem Python, além suportar e auxiliar o crescimento de comunidades de programadores Python.

# Por que usar Python???

- Vantagens:
  - Fácil de programar e aprender a programar;
  - É portátil a quase todos os sistemas operacionais;
  - Rápida prototipagem;
  - Pode fazer integração com outras linguagens;
  - Produtividade.

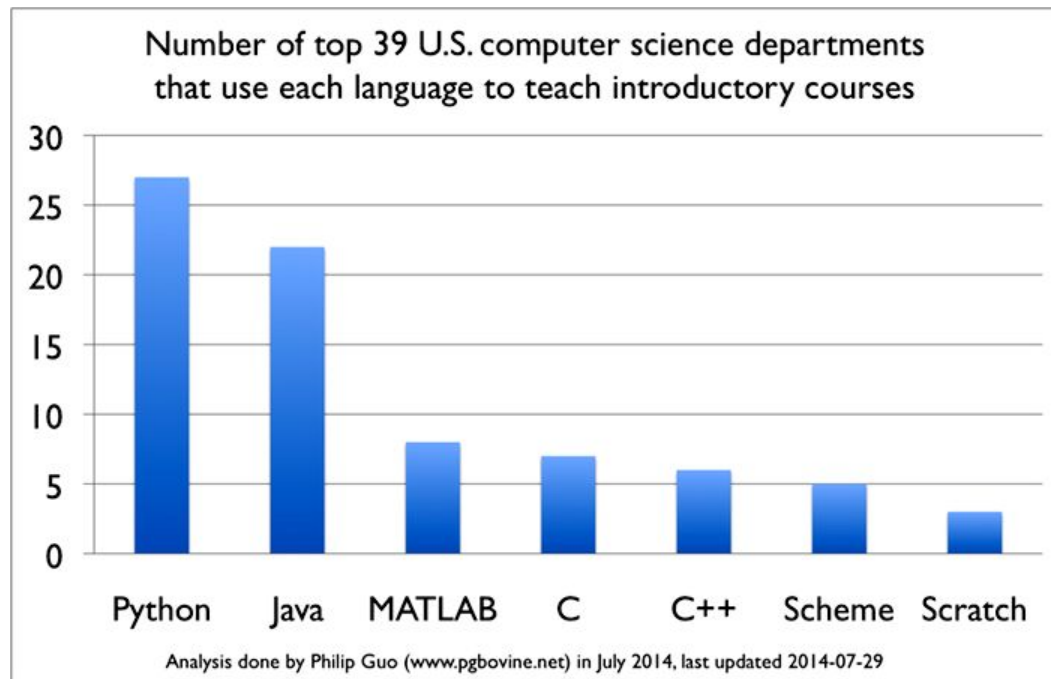
# Quem usa Python?



**NETFLIX**

***iRobot***<sup>®</sup>

# Quem usa Python?



Fonte :  
<http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext>

Comparado com as  
outras línguas, como  
está o Python???



# IEEE - 2020 Top Programming Languages

<https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020>

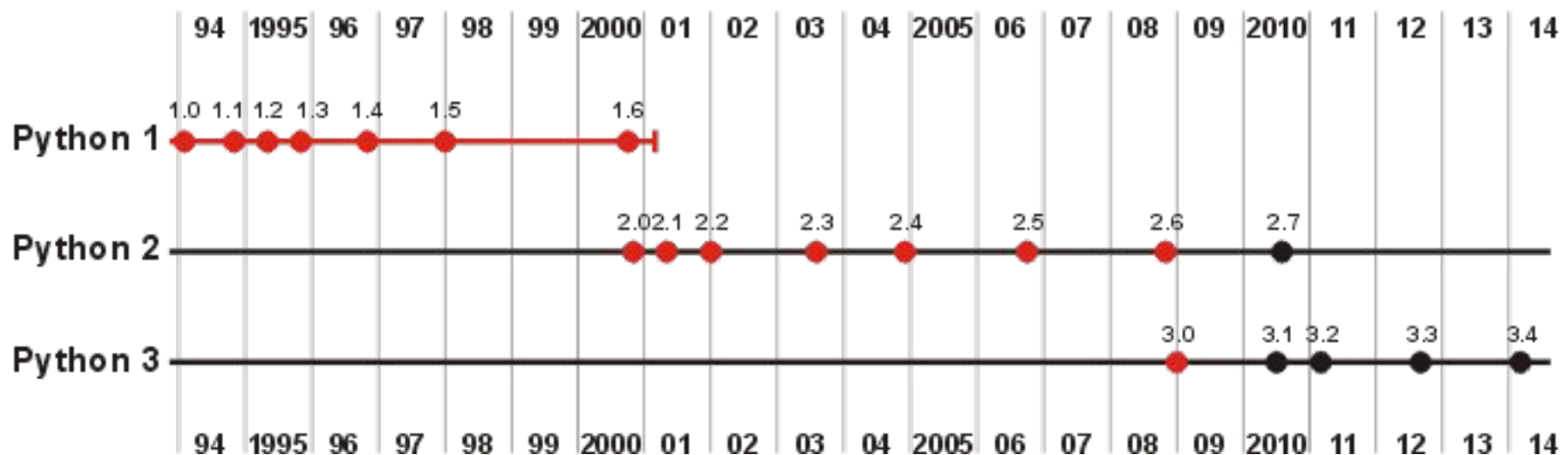
## Language Ranking: IEEE Spectrum

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

# Versões do Python

~~Python 2.x~~

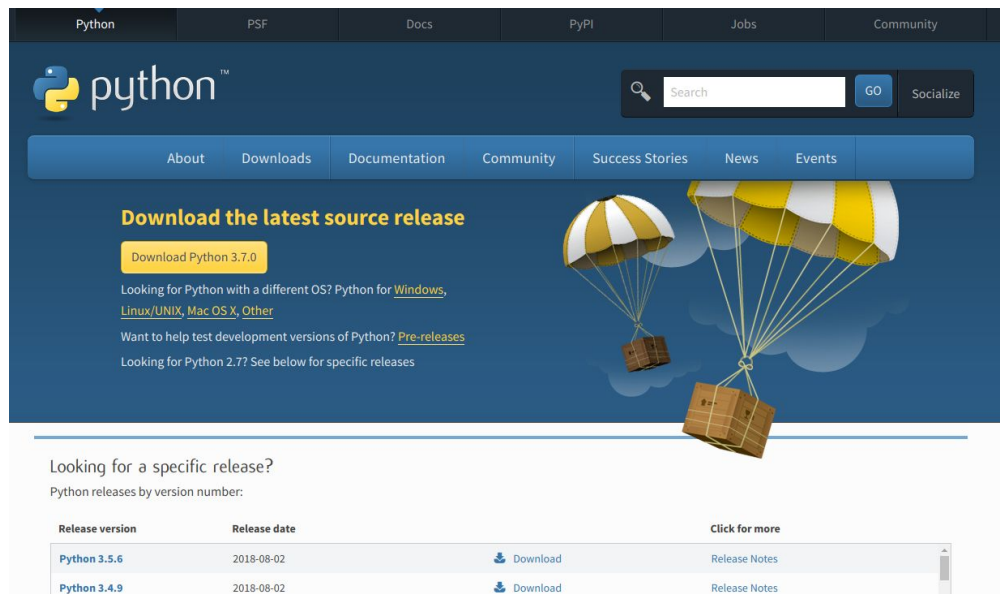
Python 3.x



# Como programar em Python?

- Você pode baixar o programa Python no site:

<https://www.python.org/downloads/>



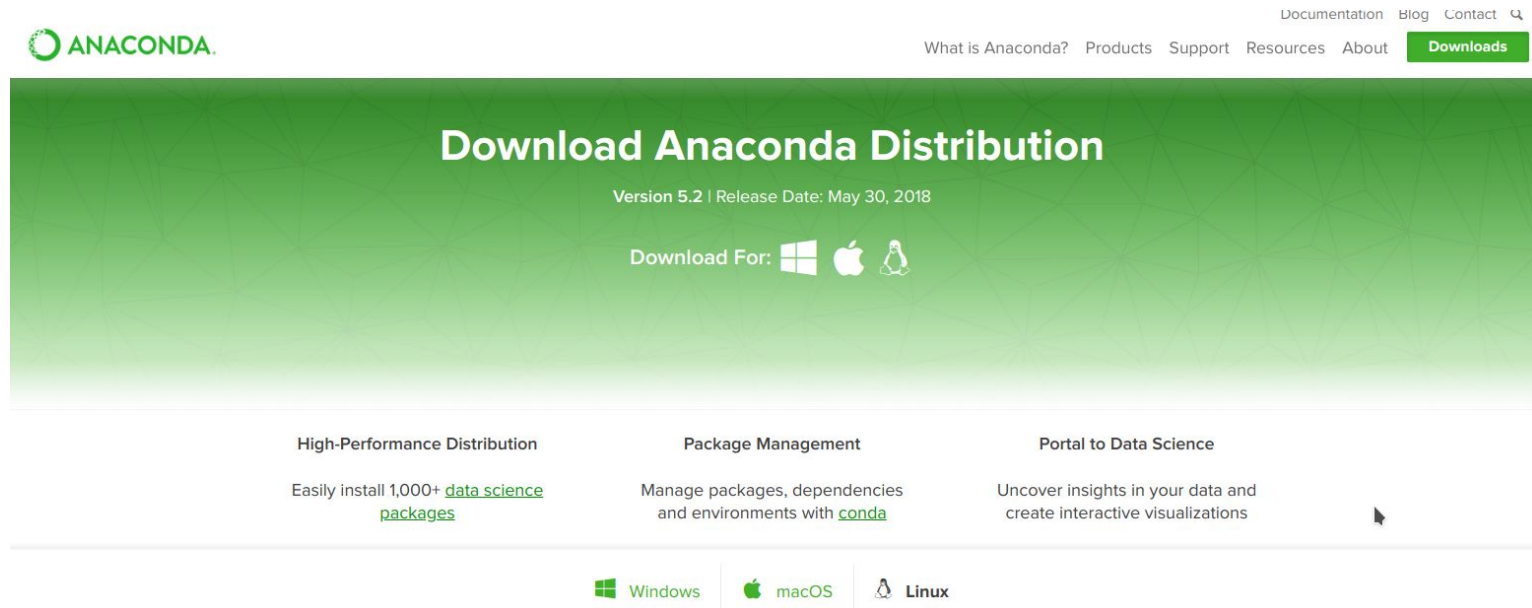
The screenshot shows the Python.org website. The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a search bar and a 'Socialize' button. The main content area features a 'Download the latest source release' section with a 'Download Python 3.7.0' button. It also includes links for other operating systems (Windows, Linux/UNIX, Mac OS X, Other) and a link for pre-releases. A large illustration of two parachutes carrying boxes is on the right. Below the main content, there is a section titled 'Looking for a specific release?' with a table of Python releases by version number.

Release version	Release date	Download	Click for more
Python 3.5.6	2018-08-02	<a href="#">Download</a>	<a href="#">Release Notes</a>
Python 3.4.9	2018-08-02	<a href="#">Download</a>	<a href="#">Release Notes</a>

# Como programar em Python?

- Ou baixar o Python já com vários pacotes no site:

<https://www.anaconda.com/download/>



# Python - Primeiros Programas - Saída de dados

Saída de dados:

```
print("Olá mundo!")
```

A função `print` informa que vamos exibir algo na tela.

# Python - Primeiros Programas - Entrada de Dados

Entrada de dados:

```
input("Entre com o dado:")
```

A função `input` informa que vamos realizar entrada de dados.

# Python - Primeiros Programas - Entrada de Dados

O problema do código de entrada exibido é que a entrada realizada pelo usuário não é salva em nenhum lugar!

Precisamos salvar a entrada na Memória Principal (RAM).

Como?

Com variáveis!

# Python - Primeiros Programas - Variáveis

Variáveis são usadas para criar uma região de memória em que vamos armazenar dados:

```
nome = input("Entre com o seu nome:")
```

`nome` é uma variável que vai armazenar a entrada que o usuário digitar



# Python - Primeiros Programas - Exemplo

Código:

```
1 nome = input("Entre com o seu nome: ")  
2 print(nome)
```

Saída:

```
Entre com o seu nome: Danilo  
Danilo
```