Computação I - Python
Aula 1 - Teórica: Introdução

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ

Conhecendo a turma

Experiência com programação e uso do computador

- Quantos já programaram antes ?
- Quais linguagens ?
- Quantos tem computador em casa com acesso a Internet ?
- Qual Sistema Operacional ?
- Quantos são calouros ?
- Quem veio de outro curso ?
- Nível de inglês ?

Objetivo da Disciplina

- Desenvolver as competências necessárias para a construção de programas legíveis e modulares em Python.

A programação é uma atividade complexa, que envolve conhecimento, prática e proficiência.

Receber bons fundamentos no estágio inicial é essencial para a capacitação de um programador.

Objetivo da Disciplina

- Desenvolver as competências necessárias para a construção de programas legíveis e modulares em Python.

Exemplos de Programas ?

Objetivo da Disciplina

- Desenvolver as competências necessárias para a construção de programas legíveis e modulares em Python.

Exemplos de Programas ?

- Explorer, Firefox, Google Chrome
- Facebook
- Windows
- Word, Powerpoint
- Media Player, iTunes
- SIGA
- The Sims 4
- ...

Objetivo da Disciplina

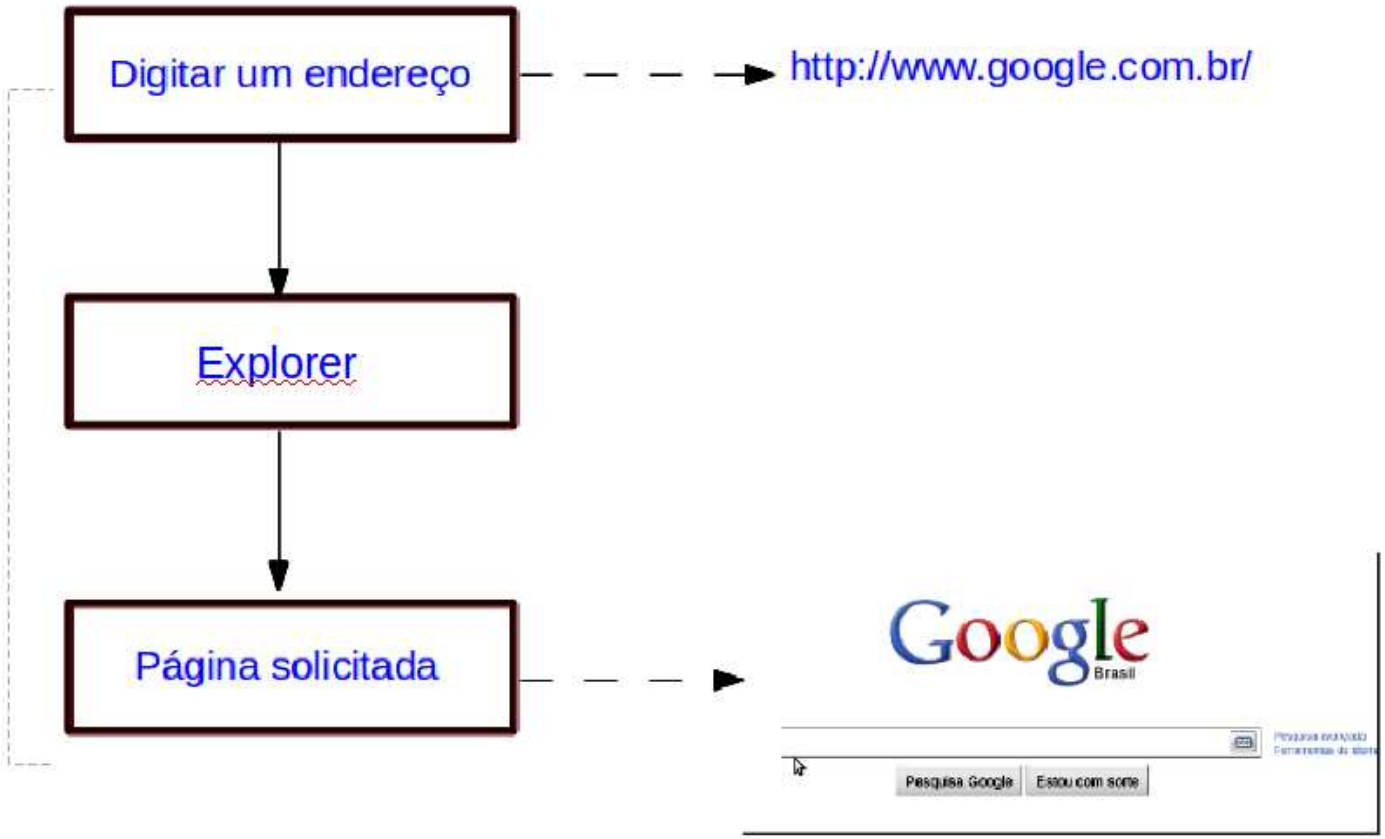
Exemplos de Programas

- Explorer, Firefox, Google Chrome

Qual a "tarefa" que o Explorer deve realizar?

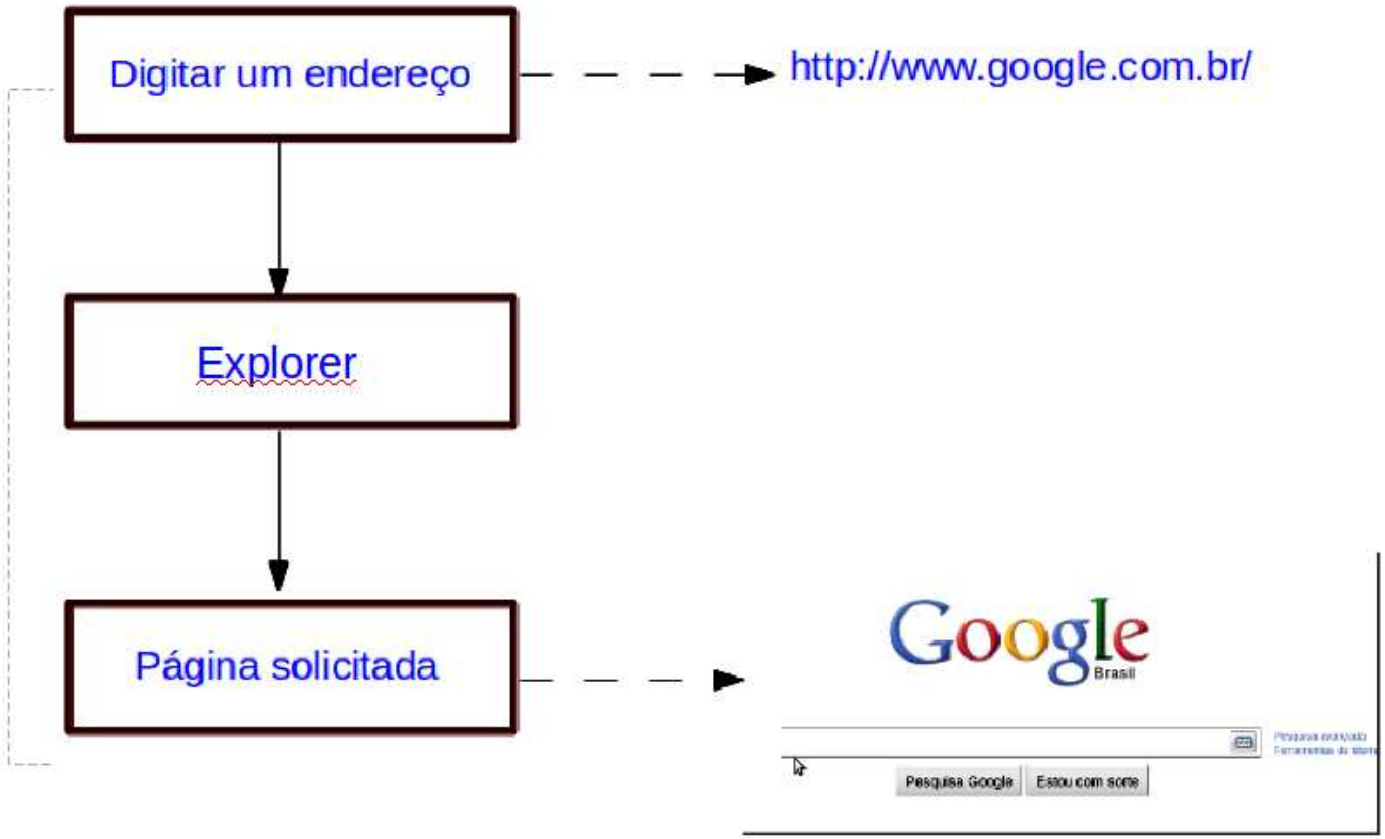
Objetivo da Disciplina

Qual a "tarefa" que o Explorer deve realizar?



Objetivo da Disciplina

Como o Explorer realiza esta "tarefa"?



Algoritmo

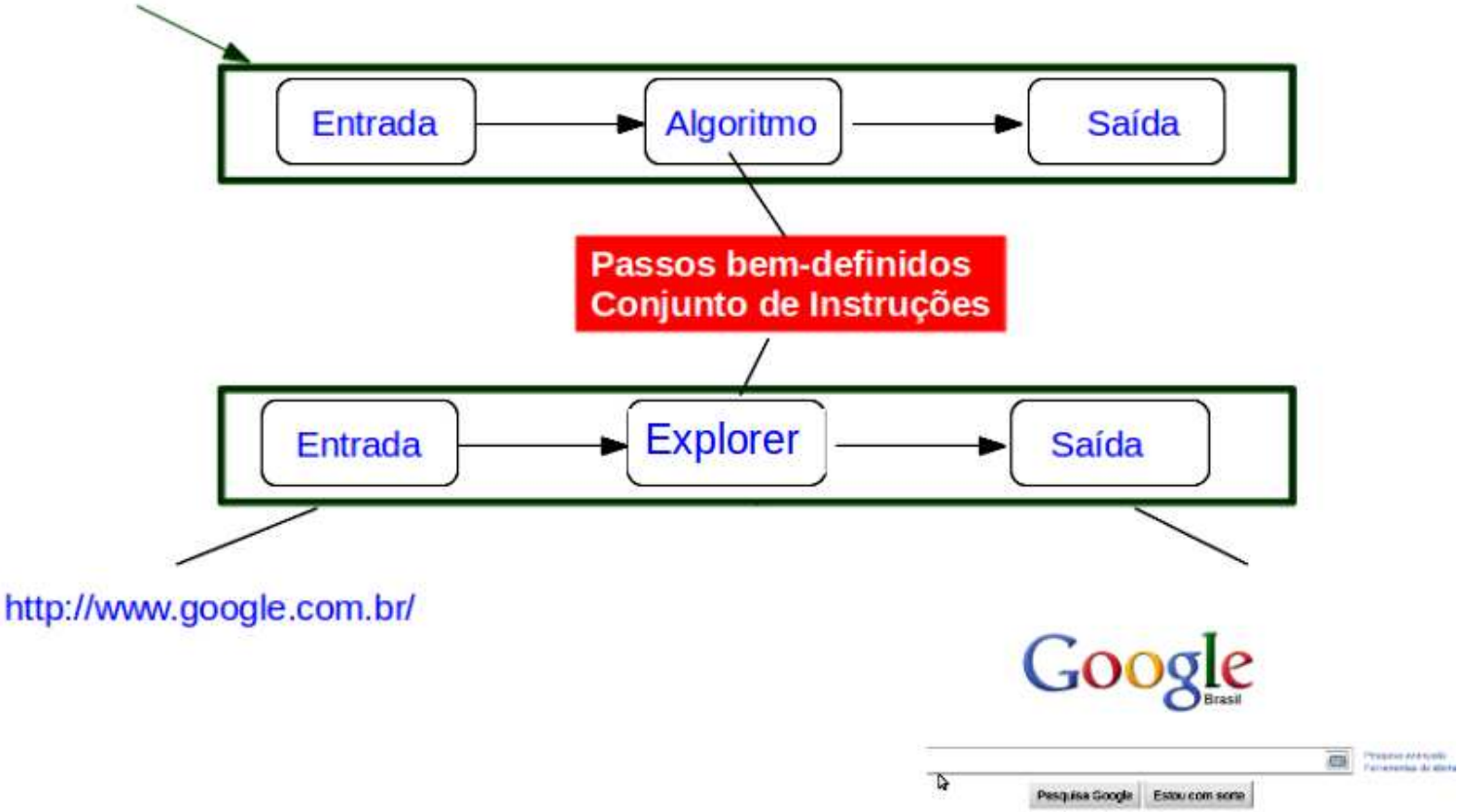
Método efetivo expresso como um conjunto de instruções que devem ser feitas para realizar uma tarefa.

Características

- **Finitude**: deve sempre terminar após um número finito de passos.
- **Bem-definido**: cada passo de um algoritmo deve ser precisamente definido (sem ambiguidades).
- **Entradas**: deve ter zero ou mais entradas (informações que são fornecidas antes do algoritmo iniciar).
- **Saídas**: deve ter uma ou mais saídas (resultado final do algoritmo).
- **Efetividade**: todas as operações devem ser suficientemente básicas de modo que possam ser executadas com precisão em um tempo finito por uma pessoa.

Algoritmo - Características

Finitude - (para um problema genérico)



Finitude - (para um problema específico): Dado um endereço na internet, exibir o conteúdo endereçado.

Exemplo - Jogo da Velha

Faça um algoritmo para jogar o jogo da velha.

Exemplo - Jogo da Velha

Faça um algoritmo para jogar o jogo da velha.

Representação

1	2	3
4	5	6
7	8	9

- **posição(n)**: Retorna o que tem na posição n.
- **jogue(n)** : Jogar na posição n.
- **faça2** : Retorna 5 se a posição 5 estiver vazia. Caso contrário, retorna qualquer uma das seguintes posições que esteja vazia: 2,4,6 ou 8.
- **ganha(p)** : Retorna (*verdade*, *posição*) se o jogador p puder vencer jogando em *posição*.

Exemplo - Jogo da Velha

- **Jogada = 1**: jogue(1)
- **Jogada = 2**: Se posição(5) = vazia
  então jogue(5)
  c.c. jogue(1)
- **Jogada = 3**: Se posição(9) = vazia
  então jogue(9)
  c.c. jogue(3)
- **Jogada = 4**: Se ganha(X)
  então jogue(ganha(X)) {bloqueia vitória adv}
  c.c. jogue(faça2)
- **Jogada = 5**: Se ganha(X)
  então jogue(ganha(X)) {vença}
  c.c. Se ganha(O)
  então jogue(ganha(O))
  c.c. Se posição(7) = vazia
  então jogue(7)
  c.c. jogue(3)

Exemplo - Jogo da Velha

- **Jogada = 6**: Se ganha(O)
  então jogue(ganha(O))
  c.c. Se ganha(X)
  então jogue(ganha(X))
  c.c. jogue(faça2).

- **Jogada = 7 ou 9**: Se ganha(X)
  então jogue(ganha(X))
  c.c. Se ganha(O)
  então jogue(ganha(O))
  c.c. jogue em qualquer posição vazia.

- **Jogada = 8**: Se ganha(O)
  então jogue(ganha(O))
  c.c. Se ganha(X)
  então jogue(ganha(X))
  c.c. jogue em qualquer posição vazia.

Exemplo - Jogo da Velha

- Podemos fazer um algoritmo semelhante ao anterior para jogar xadrez?
- Você consegue pensar em um outro algoritmo para jogar o jogo da velha?
- Este novo algoritmo poderia ser usado para jogar xadrez?

Exemplo - Jogo da Velha

Para fazer uma jogada:

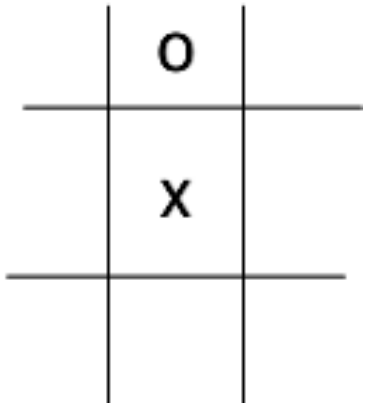
- Observe as configurações do tabuleiro resultantes de cada uma das possíveis jogadas que podem ser executadas;
- Decida pela melhor jogada. Para escolher qual a melhor jogada dentre um conjunto de configurações do tabuleiro, faça:
  - Verifique se é uma posição vencedora. Escolha esta.
  - Se não, considere todas as jogadas que o oponente pode fazer a seguir, atribuindo uma nota a cada um dos tabuleiros resultantes.
    Veja qual o pior para nós (menor nota).
    Suponha que o opositor escolherá tal jogada. Seja qual for a nota desta pior jogada, passe para cima como a nota da jogada que estamos considerando.
  - A melhor jogada é aquela com a nota mais alta.

Exemplo - Jogo da Velha

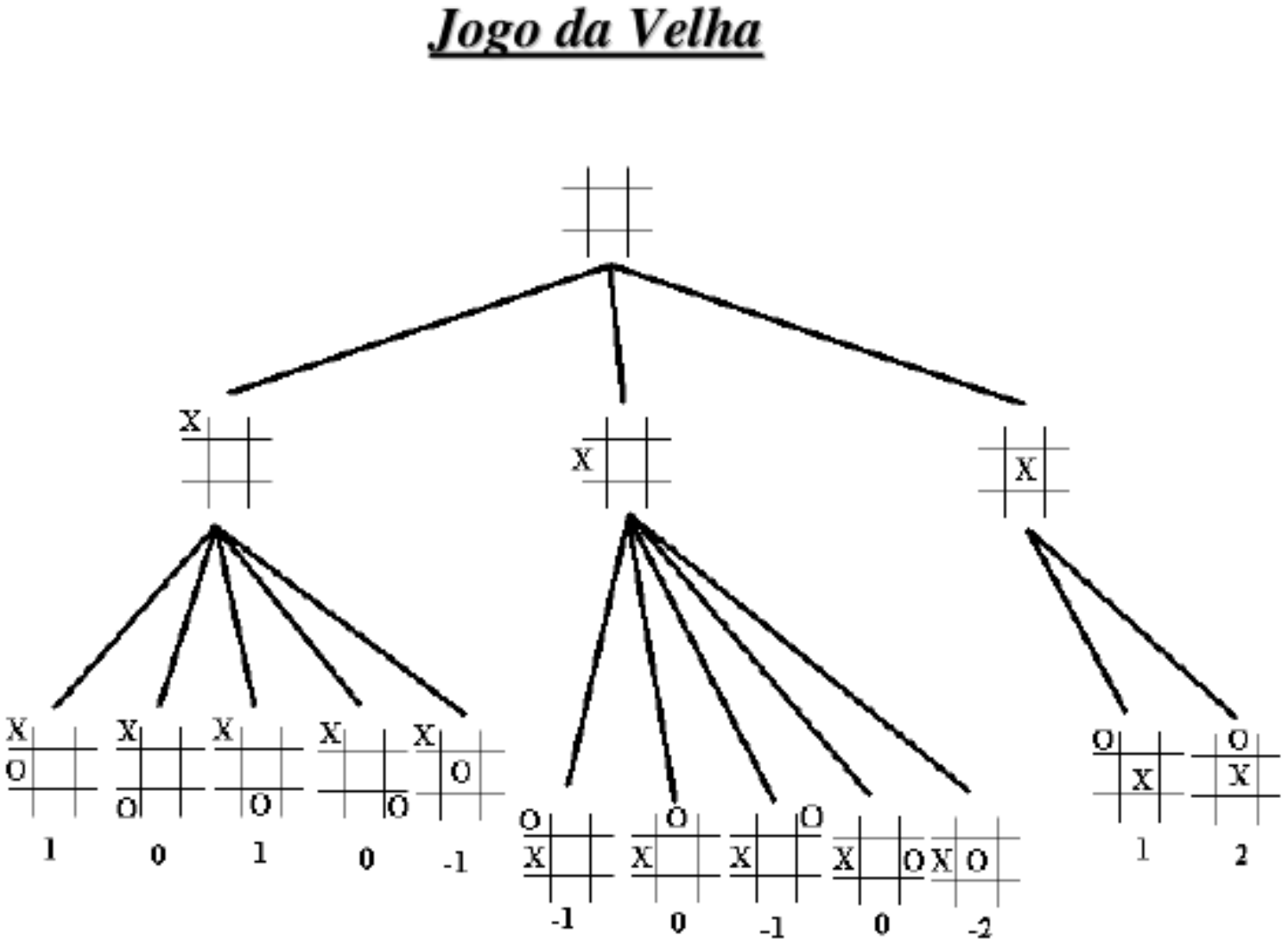
Avaliação: Como atribuir uma nota a uma jogada

$$f(n) = \begin{cases} \infty & \text{se } n \text{ é uma posição de vitória para MAX} \\ -\infty & \text{se } n \text{ é uma posição de vitória para MIN} \\ (\#\text{ de fileiras abertas para MAX} & \text{caso contrário} \\ \text{- }\#\text{ de fileiras abertas para MIN}) & \end{cases}$$

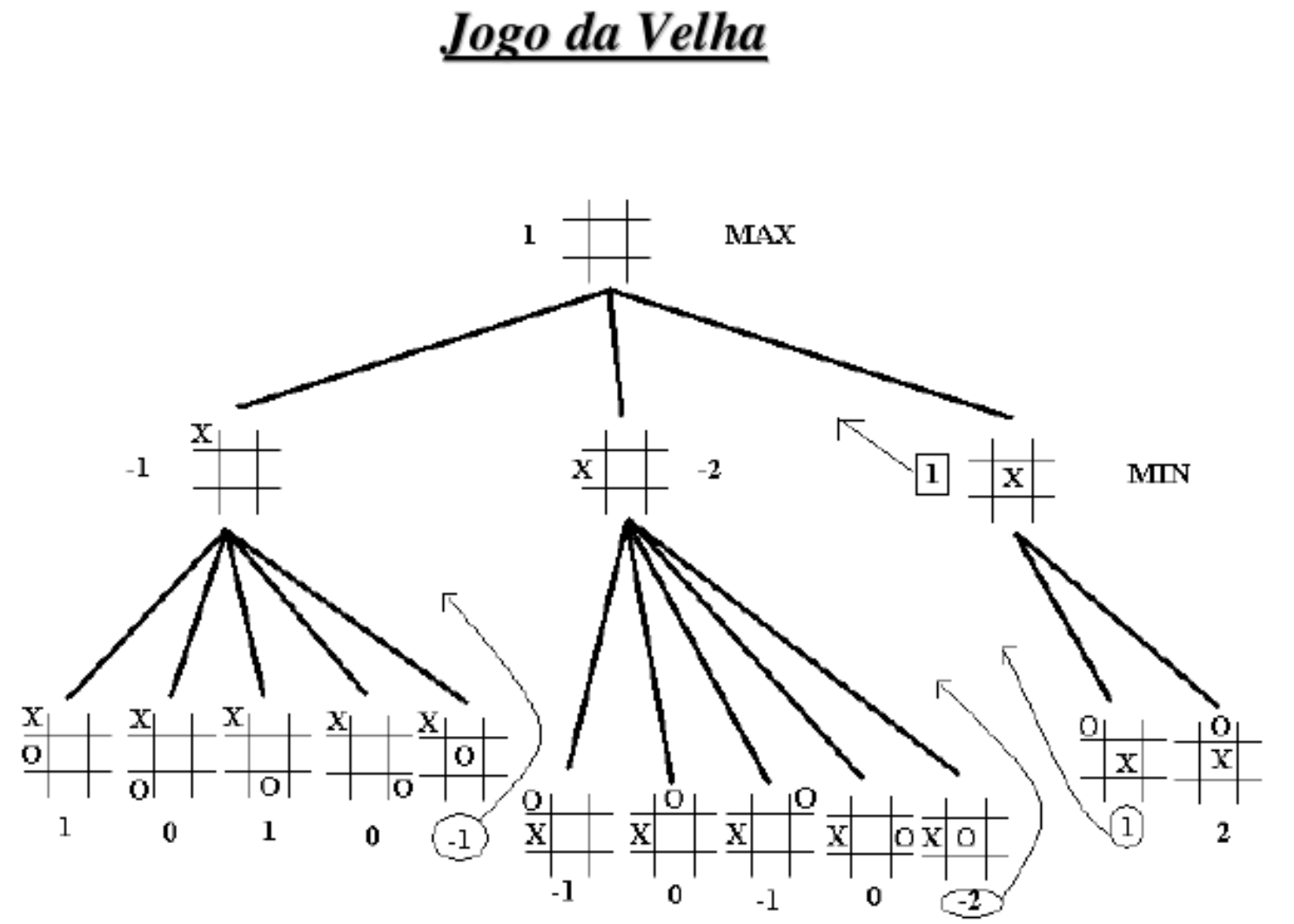
Considere : X = MAX e O = MIN $\Rightarrow f = 6 - 4 = 2$



Exemplo - Jogo da Velha



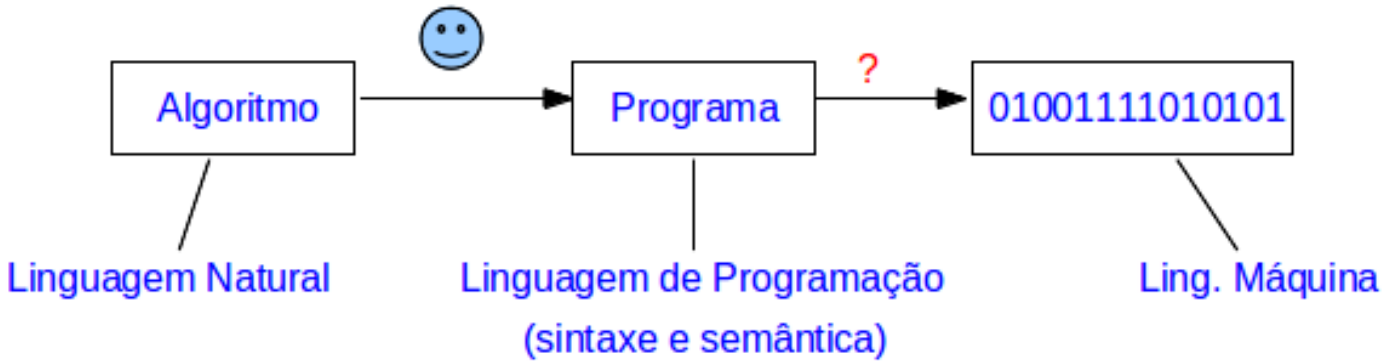
Exemplo - Jogo da Velha



Programa de Computador

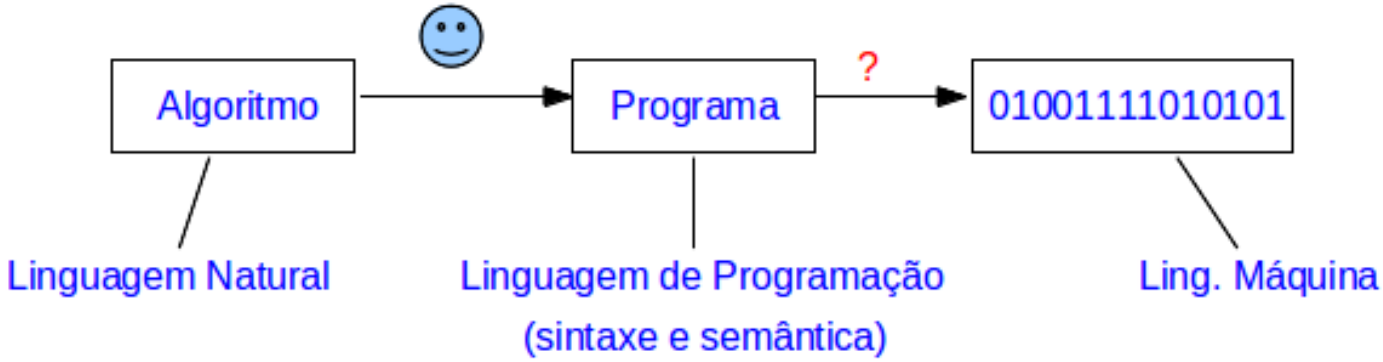
Conjunto de instruções que descrevem como uma tarefa deve ser realizada por um computador. Ou seja, o computador deve ser capaz de "entender" as instruções.

O computador "entende" linguagem de máquina: 01011100110.
Como traduzir um algoritmo para código de máquina?



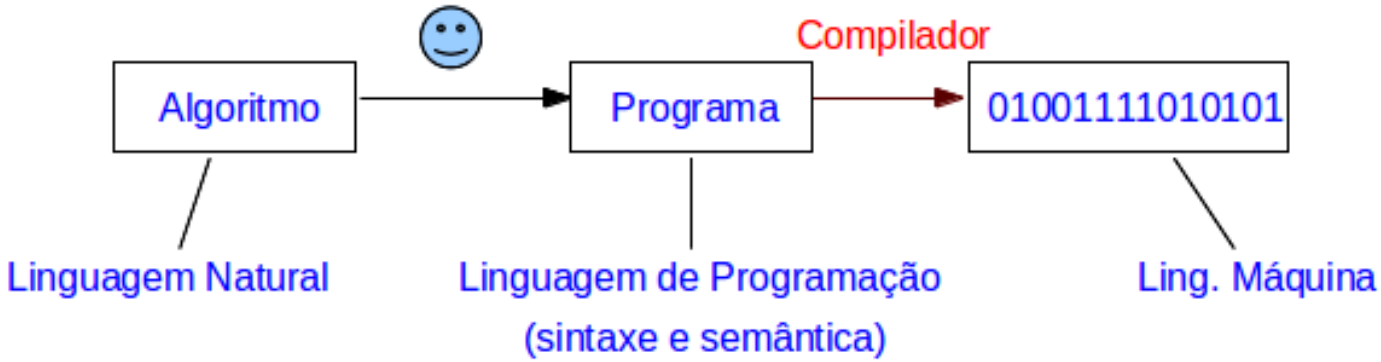
Linguagens e Paradigmas de Programação

1. **Programação Imperativa**: define sequências de comandos que um computador deve seguir para realizar uma tarefa.
2. **Programação Declarativa**: expressa o que deve ser realizado sem dizer como realizar.
3. **Programação Orientada a Objeto**
4. **Programação Funcional**

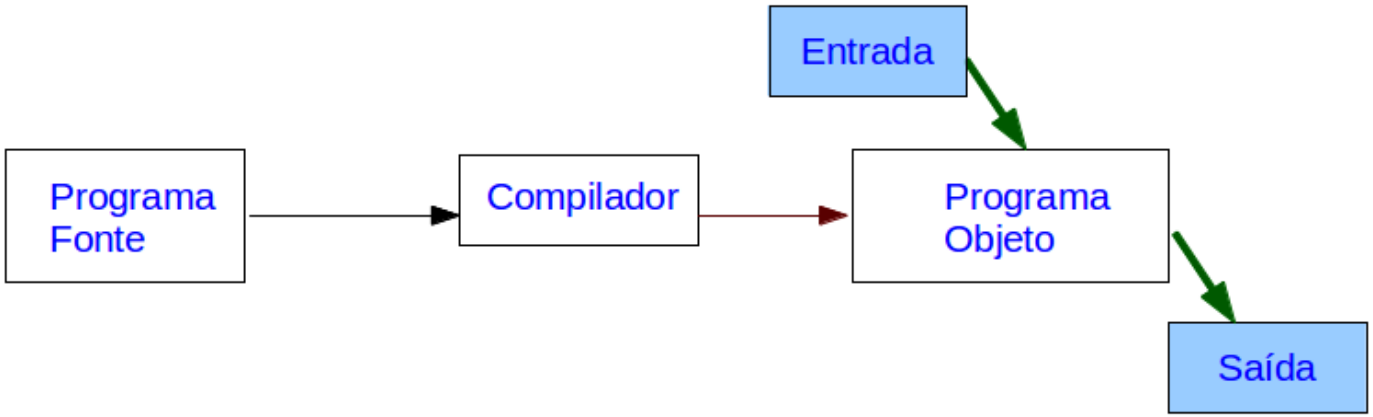


Linguagens e Paradigmas de Programação

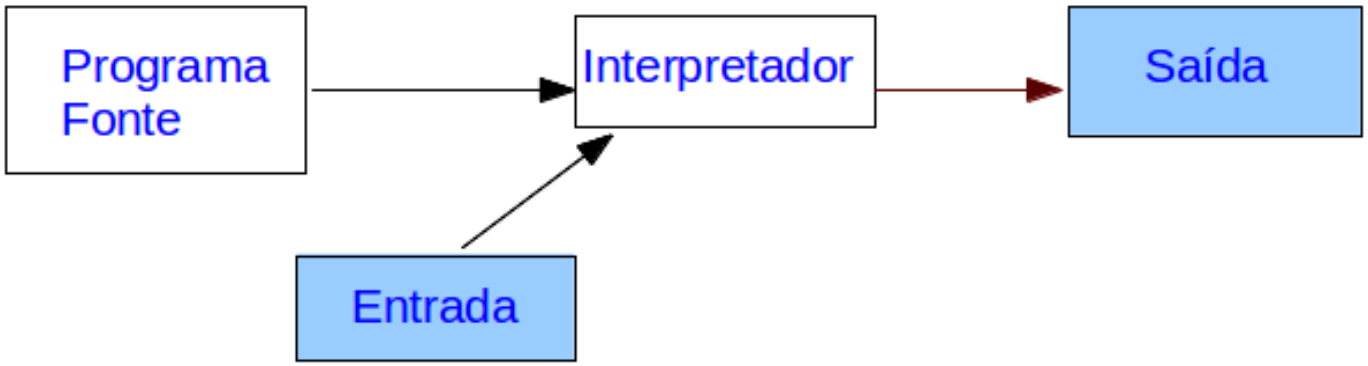
1. **Programação Imperativa**: define sequências de comandos que um computador deve seguir para realizar uma tarefa.
2. **Programação Declarativa**: expressa o que deve ser realizado sem dizer como realizar.
3. **Programação Orientada a Objeto**
4. **Programação Funcional**



Compilador



Interpretador

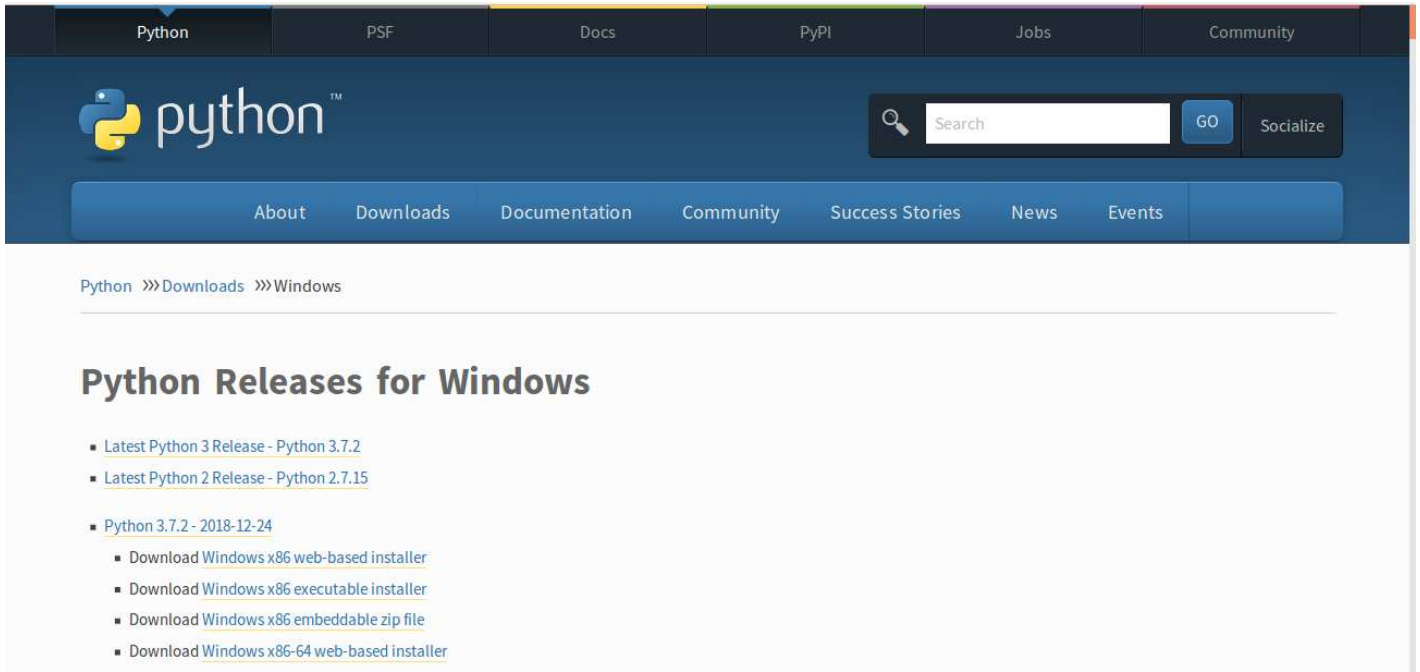


Por que Python ?

- Linguagem moderna, robusta, bem projetada
- Open source e gratuita
- Simples o suficiente para um curso introdutório
- Muitos recursos
  - Orientação a Objetos
  - Escalável (módulos, classes, controle de exceções)
  - Biblioteca embutida extensa e grande número de módulos fornecidos por terceiros
- Grande variedade de aplicações
- Linguagem interpretada (script)
- Multi-plataforma
- Muito usada no meio científico
- Comunidade bastante grande

Quem usa Python ?

Instalando o interpretador Python 3



https://www.python.org/downloads/windows/

Python na Nuvem

Ferramenta didática: **Pythontutor** - `http://pythontutor.com/`

Start using Online Python Tutor now

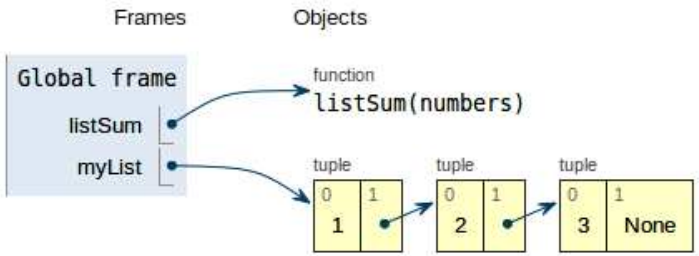
For instance, here is a visualization showing a program that recursively finds the sum of a (cons-style) linked list. Click the "Forward" button to see what happens as the computer executes each line of code.

```
1 def listSum(numbers):
2   if not numbers:
3     return 0
4   else:
5     (f, rest) = numbers
6     return f + listSum(rest)
7
8 myList = (1, (2, (3, None)))
9 total = listSum(myList)
```

Edit code

< Back Step 3 of 22 Forward >

line that has just executed
next line to execute



Material e Dicas...

http://www.dcc.ufrj.br/~pythonufrj

Computação I - Python
Aula 1 - Teórica: Introdução

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ