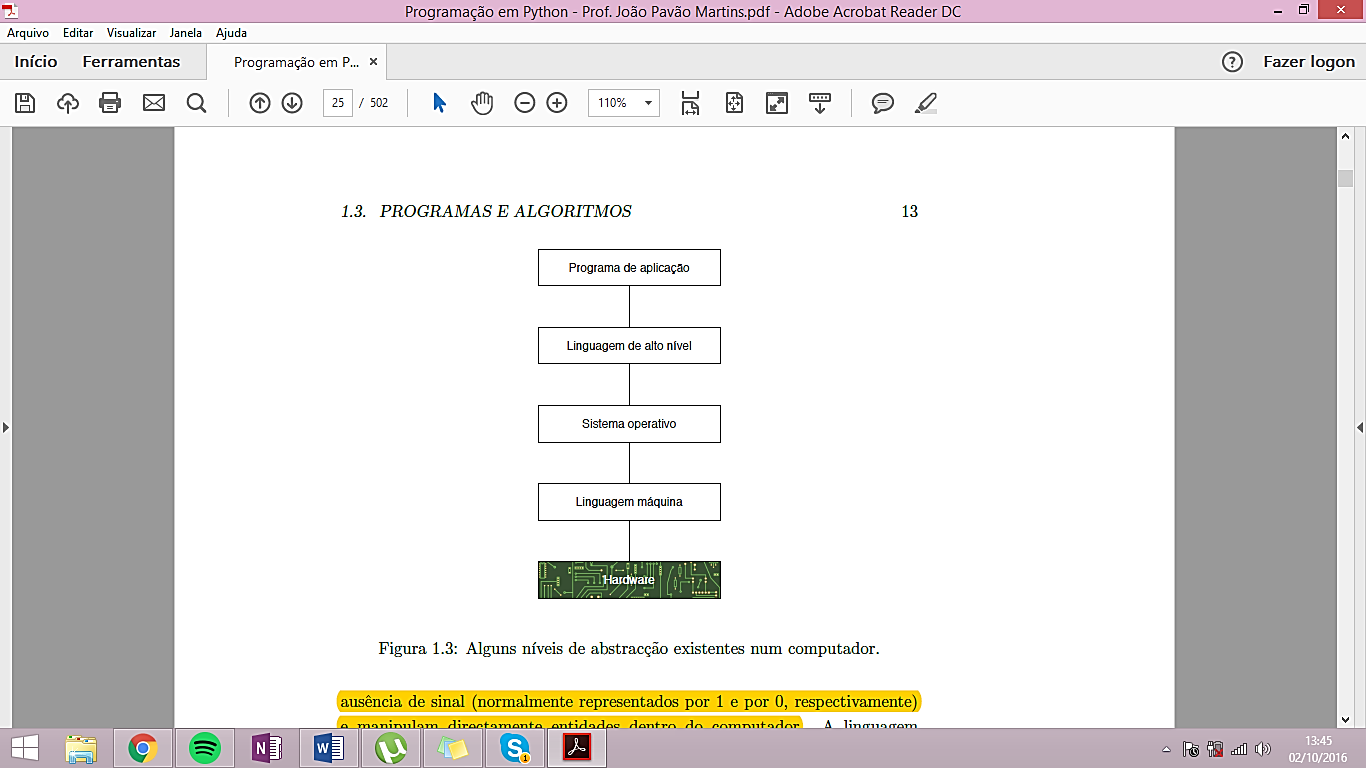
Programar consiste em dividir um problema complexo em vários mais simples que possam ser expressos em termos de operações elementares e fazer a depuração do programa (perceber quais os erros e corrigi-los).

A computação pode ser baseada nas operações elementares que correspondem:

1. Às operações de entrada de dados 🡪 obtêm valores do exterior do programa;
2. Às operações de saída de dados 🡪 mostram valores existentes no programa;
3. Às operações matemáticas 🡪 efetuam cálculos;
4. À execução condicional 🡪 teste de condições (True/False);
5. À repetição 🡪 execução repetitiva de instruções;

Um algoritmo é uma sequência executável num período de tempo e com uma quantidade de esforço finitos com instruções bem definidas e não ambígua. Um algoritmo tem de ser:

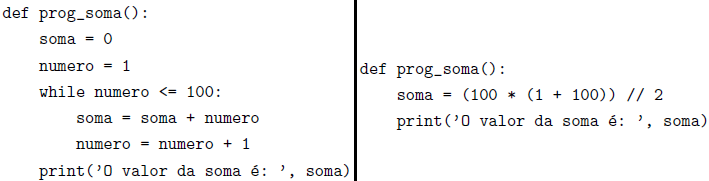
1. Rigoroso 🡪 não havendo lugar para ambiguidades;
2. Eficaz 🡪 cada instrução deve ser básica e bem compreendida, para ser executada num espaço de tempo finito;
3. Deve terminar 🡪 o objetivo deve ser atingido e não devem existir mais instruções a executar.

Uma linguagem de programação é utilizada para escrever programas de computador. Podem ser classificadas em:

1. Linguagens máquina 🡪 Comanda diretamente as ações do computador utilizando uma sequência de dois símbolos discretos (0 e 1) – Binários;
2. Linguagens *assembly* 🡪 Difere da linguagem máquina no sentido que uso nomes simbólicos com significado para os humanos;
3. Linguagens de alto nível 🡪 Aproximam-se da língua humana e, são, por isso, mais fáceis de utilizar. São necessários processadores de linguagem para a máquina entender as linguagens de alto nível, estes podem ser: compiladores ou interpretadores.

O “hardware pode ser diretamente comandada através da linguagem máquina ou *assembly*. Através do sistema operativo, pode-se usar linguagens de programação de alto nível que se utilizam para escrever programas que resolvam problemas específicos.

Exemplos de duas maneiras diferentes de fazer um programa em Python que some os 100 primeiros números inteiros positivos:

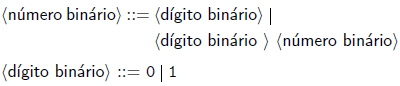


# Sintaxe e semântica

Sintaxe 🡪 determina qual a constituição das frases que podem ser fornecidas ao computador.

Regaras da Notação BNF:

1. Símbolos não terminais representam-se entre “‹›” e estão sempre associados a um conjunto de entidades da linguagem;
2. Os símbolos que aparecem nas frases da linguagem (exe: “+”) são chamados de símbolos terminais;
3. O símbolo “|” (ou) representa uma disjunção;
4. O símbolo “::=” (definido como) serve para definir componentes da linguagem;
5. A utilização de “+” após um símbolo não terminal significa que este pode ser repetido uma ou mais vezes (>0);
6. A utilização de “\*” após um símbolo não terminal significa que este pode ser repetido zero ou mais vezes (>=0);
7. A utilização de “{ }” englobando símbolos terminais ou não terminais, significa que estes são opcionais.
8. Exemplo:



Nota: A notação utilizada para definir formalmente uma linguagem, no caso da notação BNF, < >, |, ::=, { }, +, \*, os símbolos não terminais e os símbolos terminais, é denominada metalinguagem.

**Semântica** 🡪 determina o que o computador vai fazer ao seguir cada uma das frases.

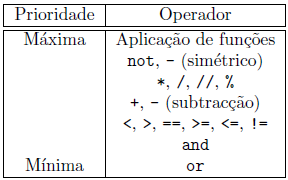
Podem ocorrer erros (*bugs*) de natureza:

* Sintática 🡪 o programador não escreveu as frases do programa de acordo com as regras da gramática da linguagem de programação utilizada;
* Semântica 🡪 são difíceis de detetar pois são de caráter lógico, um erro de raciocínio.

O processo de deteção e correção desses erros chama-se depuração (debugging).

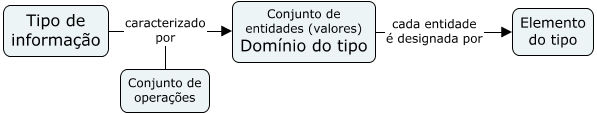
Expressões são entidades computacionais (algo que existe dentro de um programa) que têm um valor. Podem ser:

* Constantes 🡪 têm sempre o mesmo valor e têm uma representação externa (modo como a visualizamos) e interna (recorrendo a 0 e 1). Podem ser:
  1. Números inteiros (sem parte decimal);
  2. Números reias (com parte decimal);
  3. Valores lógicos (True / False);
  4. Cadeias de caracteres (ou strings são delimitadas por “plicas”;
* Expressões compostas 🡪 são funções que já estão pré-definidas na linguagem constituídas por um operador e por um certo números de operandos. Os operadores podem ser:
  + - Unários: se apenas têm um operando (exe: not);
    - Binários: se têm dois operandos (exe: +).
  + Lista de prioridades de operadores (quando têm a mesma prioridade a ordem é da esquerda para a direita ou, com o uso de parêntesis a ordem pode ser alterada):



* Um nome;
* Uma aplicação da função.

Tipos elementares de informação



Os tipos de informação variam de linguagem para linguagem e podem ser:

* Tipos elementares 🡪 as constantes são indecomponíveis (exe: true);
* Tipos estruturados 🡪 as constantes são constituídas por um agregado de valores.

Em Python, como os tipos elementares, existe também:

* O tipo inteiro 🡪 são designados por “int” e são números sem parte decimal que podem ser positivos, negativos ou zero.
* O tipo real 🡪 são designados por “float” e são números com parte decimal. Podem ser representados em notação decimal ou científica;
* O tipo lógico 🡪 são designados por “bool” e podem assumir apenas dois valores: True / False. Podem ser efetuadas operações sobre os valores lógicos de dois tipos:
  + Operações unárias: produzem um valor lógico a partir de outro (exe: not);
  + Operações binárias: aceitam dois argumentos do tipo lógico e produzem um valor do mesmo tipo (exe: and, or).

Nomes e atribuição

A instrução entre um nome e um valor é realizada através da instrução de atribuição que recorre à operação embutida = (operador de atribuição). Os nomes são utilizados para representar entidades usadas pelos programas.

A instrução de atribuição altera o valor de um nome (mantendo-o na memória que corresponde à associação de nomes e valores – memória ambiente – esta memória contém associações para todos os nomes que o Python conhece e, por isso, o mesmo nome não pode estar associado a dois valores diferentes). A instrução de atribuição apresenta duas formas distintas:

* atribuição simples 🡪 corresponde à primeira linha da expressão BNF que define <instrução de atribuição>;
* atribuição múltipla 🡪 corresponde à segunda linha da expressão BNF que define <instrução de atribuição>).

Predicados e condições

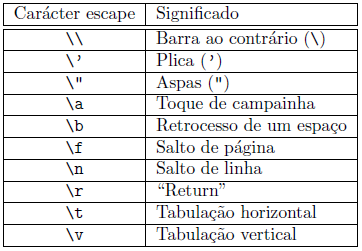
Predicado 🡪 é uma operação que produz resultados do tipo lógico

Condição 🡪 é uma expressão cujo valor é do tipo lógico .As condições podem ser combinadas através de operações lógicas.

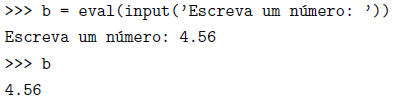
Leitura de dados

As operações de leitura de dados permitem transmitir informação do exterior para o programa. Para tal, em Python usamos a função “input”.

Os caracteres escape 🡪 são caracteres não gráficos com um significado especial para um meio de escrita. Exemplos:

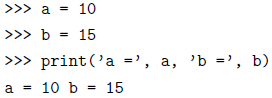


Como o valor da função “input” é uma cadeia de caracteres, temos que recorrer à função “eval” (função de avaliação) para avaliar a cadeia como sendo uma expressão. Exemplo:



Escrita de dados

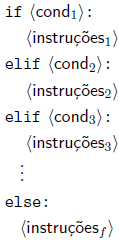
Permite que o computador comunique com o exterior sobre os resultados a que chegou, através das operações de escrita de dados. Para tal, em Python, usamos a função “print”. Exemplo:



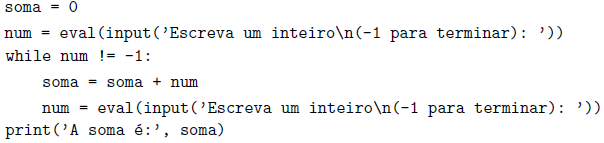
Programas, instruções e sequenciação

Ao nível da linguagem máquina, existem 2 tipos de estruturas de controlo: sequenciação e salto. Em Python (e nas linguagens de alto nível) para além destas duas estruturas têm também a seleção e a repetição. A utilização destas estruturas de controle facilita, consideravelmente, a leitura e manutenção dos programas.

* Sequenciação 🡪 é a estrutura mais simples e consiste na especificação de que as instruções de um programa são executadas sequencialmente (pela ordem que aparecem no programa);
* Seleção 🡪 consiste na especificação de uma execução condicional de instruções. Para construir programas mais complexos devemos ter a capacidade de decidir se uma instrução deve ser ou não executada, dependendo do valor de uma condição. Exemplo:



* Repetição 🡪 é necessária para a criação de ciclos (repetição de uma dada instrução). Em Python utiliza-se a instrução “while” e “for” para a especificação de ciclos.
  + A instrução “while” permite a execução repetitiva de um conjunto de instruções enquanto uma determinada expressão do tipo lógico tiver o valor verdadeiro (o número de vezes que o ciclo é repetido não pode ser calculado antecipadamente). Para evitar ciclos infinitos, utiliza-se a instrução “break” que faz com que o Python termine a execução do ciclo, independentemente do valor de outras condições. Exemplo:



* Instrução vazia: quando se inserem linhas em branco para aumentar a facilidade de leitura sem ter qualquer efeito sobre a execução do programa.
* Comentários: são frases em linguagem natural que aumentam a facilidade de leitura do programa. Para inserir um comentário utiliza-se o símbolo “#”, tal como as linhas m branco, os comentários são ignorados pelo computador.

Página: 85