**Variáveis em Go**

Para declarar variáveis em go é usado a seguinte syntax:

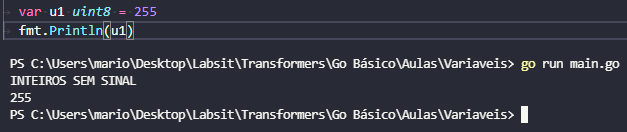


1. **Variáveis uint (unsigned integer)**

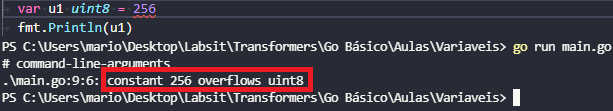
As variáveis do tipo uint (unsigned integers) são variáveis que encapsulam APENAS números inteiros positivos.

**Variáveis uint8**

Variáveis do tipo uint8 (8 bits) são variáveis q podem encapsular valores de 0 a 255.



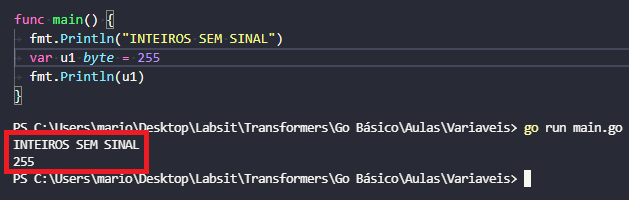
Caso o valor de uma variável do tipo uint8 não esteja entre 0 255 será retornado um erro de compilação.



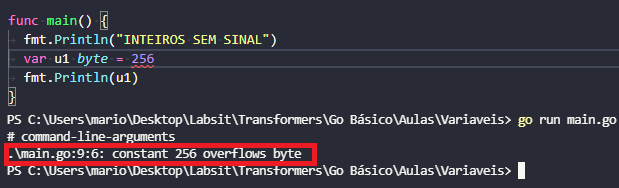
**Variáveis byte**

Variáveis do tipo byte (8 bits) têm o mesmo comportamento de variáveis do tipo uint8.

A variável do tipo byte É O *ALIAS do uint8, ou seja, é um “substituto” do uint8.*

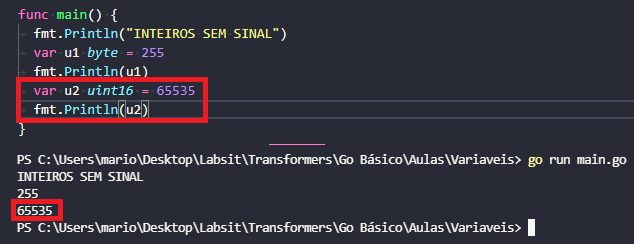


Caso o valor de uma variável do tipo uint8 não esteja entre 0 255 será retornado um erro de compilação.

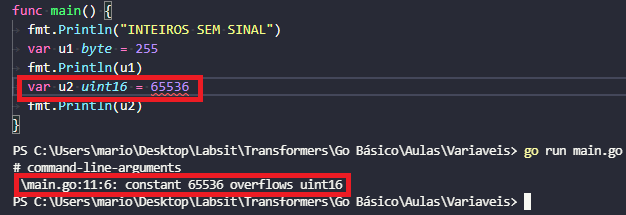
****

**Variáveis uint16**

Variáveis do tipo uint16 (16 bits) podem encapsular valores de 0 até 65535.

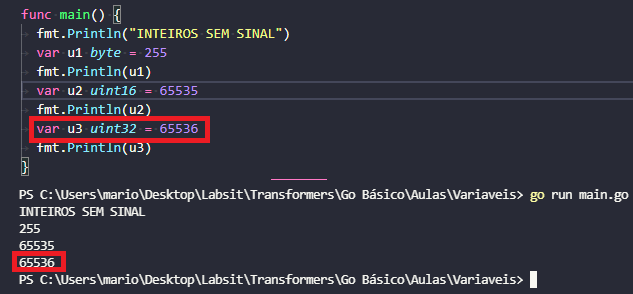
****

Caso o valor de uma variável tipo uint16 não esteja entre 0 e 65535 será retornado um erro de compilação.

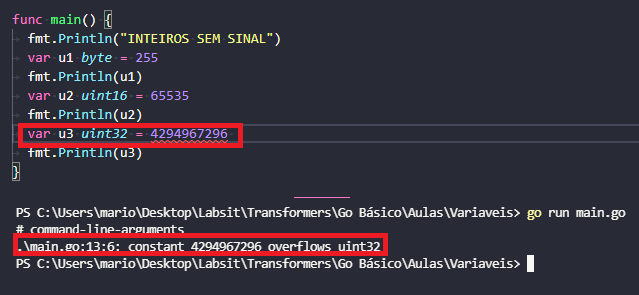
****

**Variáveis uint32**

Variáveis do tipo uint32 (32 bits) podem encapsular valores de 0 até 4.294.967.295.

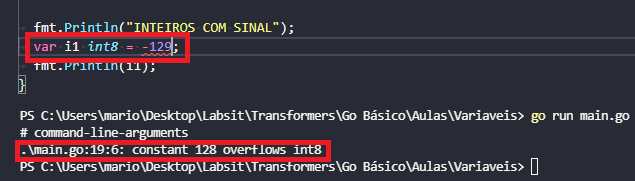
****

Caso o valor de uma variável tipo uint16 não esteja entre 0 e 4294967295 será retornado um erro de compilação.

****

**Variáveis uint64**

Variáveis do tipo uint64 (64 bits) têm o mesmo comportamento das demais variáveis já descritas anteriormente.

****

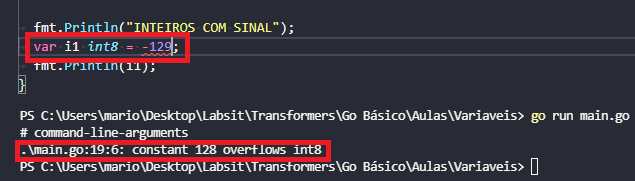
1. **Variáveis int (Integer)**

Variáveis do tipo int são variáveis que podem encapsular valores inteiros tanto positivos como negativos.

**Variáveis int8**

Variáveis do tipo int8 (8 bits) podem encapsular valores de -128 até 127.

Caso o valor encapsulado não esteja entre -128 e 127 será retornado um erro de compilação.



A mesma regra vale para variáveis int16, int32 e int64 respeitando suas respectivas capacidades.

**Variáveis rune**

Variáveis do tipo rune (32 bits) têm o mesmo comportamento de variáveis do tipo int32.

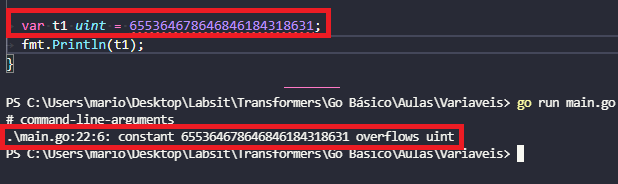
A variável do tipo rune é o *ALIAS* do int32, ou seja, é um “substituto” do int32.

**Variáveis uint**

Variáveis do tipo uint recebem a precisão de acordo com a arquitetura padrão do sistema operacional e do processador da máquina em que ele está rodando. Em caso de SOs de 32 bits ele terá precisão de 32 bits, em caso de SOs de 64 bits ele terá precisão de 64 bits.

Lembrando que as variáveis do tipo int encapsulam APENAS valores inteiros positivos, respeitando seus limites de precisão.

Caso o seu limite de precisão máximo seja execedido será retornado um erro de compilação como já vimos anteriormente.

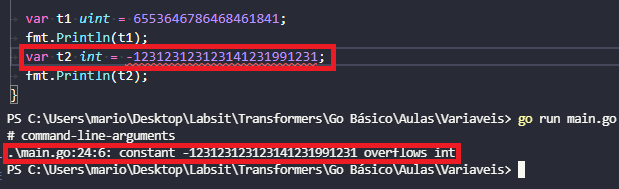


**Variáveis int**

Variáveis do tipo int também recebem a precisão de acordo com a arquitetura padrão do sistema operacional e do processador da máquina em que ele está rodando. Em caso de SOs de 32 bits ele terá precisão de 32 bits, em caso de SOs de 64 bits ele terá precisão de 64 bits.

Lembrando que as variáveis do tipo int podem encapsular valores inteiros positivos e negativos respeitando seus limites de precisão.

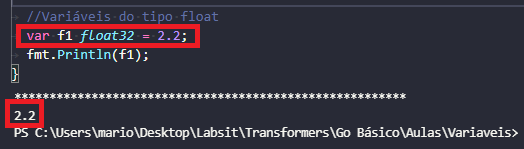
Caso o seu limite de precisão máximo seja execedido será retornado um erro de compilação como já vimos anteriormente.



1. **Variáveis float (Ponto Flutuante)**

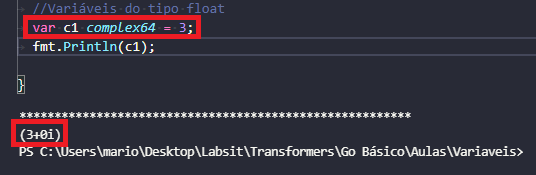
São variáveis que podem encapsular valores do tipo ponto flutuante.

Temos o float32 (32 bits) e o float64 (64 bits).

****

1. **Variáveis complex (Números Complexos)**

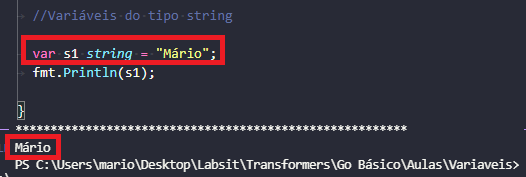
São variáveis que podem encapsular números complexos. Podem ser do tipo complex64 (64 bits e complex128 (128 bits).

****

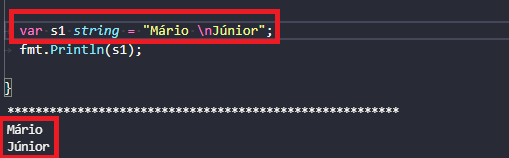
Ter variáveis do tipo complexo é muito interessante para fazer o uso em cálculos científicos, por exemplo.

**5. Variáveis String (Texto)**

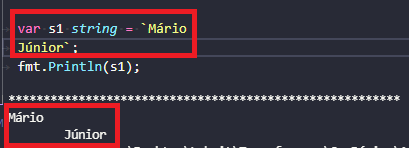
São variáveis que podem encapsular strings (texto).

****

Para fazer quebra de linha em strings podemos usar o \n assim como nas demais linguagens.



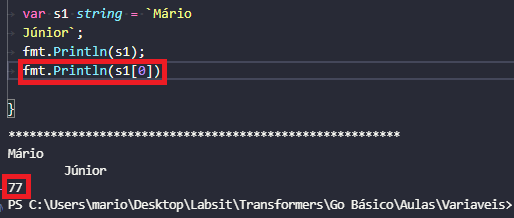
Ao encapsular uma string usando a crase (``) podemos fazer quebras de linha sem precisar do \n.

****

Perceba que até mesmo o tab foi executado dentro da string.

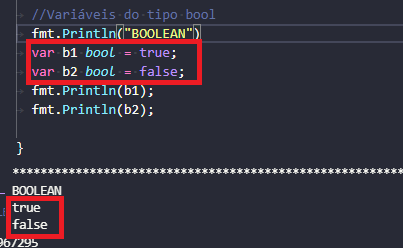
* 1. **. Acessando os caracteres de uma string através dos índices**

Ao acessarmos os caracteres de uma string através dos seus índices (ex.: s1[0]) será retornado o código UNICODE daquele caractere.



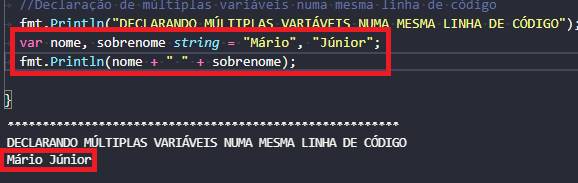
1. **Variáveis bool (Boolean)**

São variáveis que podem encapsular valores booleanos (true ou false).

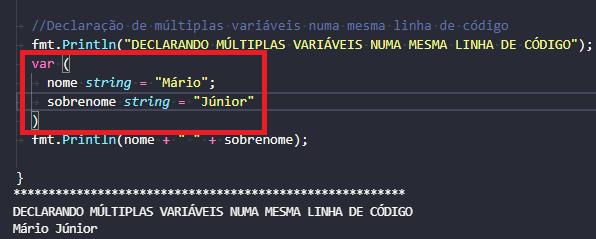


1. **Declaração de Múltiplas Variáveis numa mesma linha de código**

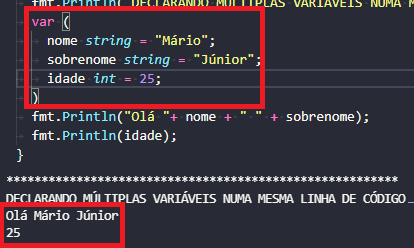
Com a linguagem Go é possível declarar múltiplas variáveis e atribuir valor a elas numa mesma linha de código usando a seguinte syntax:



Também é possível fazer declaração de múltiplas variáveis em “bloco” utilizando a seguinte syntax:

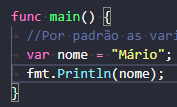


Também é possível declarar múltiplas variáveis de tipos diferentes em “bloco” utilizando a mesma syntax que vimos anteriormente:

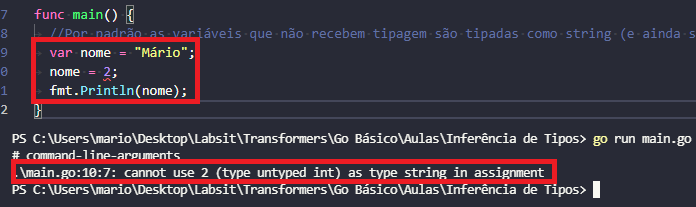


1. **Inferência de Tipos no Go**

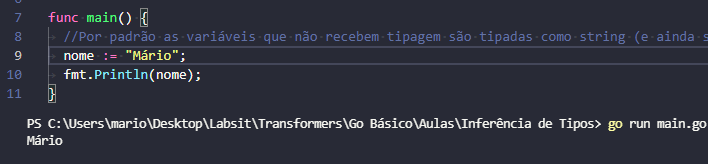
Em Go mesmo que você declare uma variável sem declarar o tipo dela, ela receberá, por padrão, o tipo de variável correspondente ao tipo do dado encapsulado. Isso não quer dizer, porém, que essa variável passa a ter tipagem fraca, quer dizer apenas que foi feita uma INFERÊNCIA de tipo, ou seja, o tipo foi declarado automaticamente.



Se tentarmos declarar um novo tipo para essa mesma variável de forma hard code teremos um erro de compilação.

****

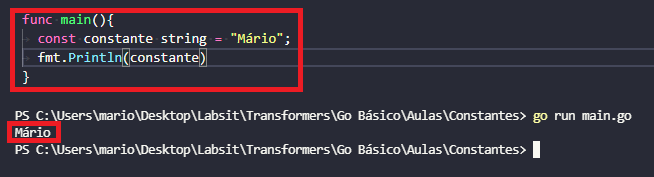
Também é possível inicializar uma variável sem o var e já inferir o tipo dela automaticamente usando a seguinte syntax:



1. **Constantes**

Constantes são “variáveis” cujos valores são fixos, inalteráveis, não é possível sobrescrever nem reatribuir valores para elas.

A syntax utilizada na declaração de constantes é muito similar à da declaração de variáveis:

****