

Processamento

- Condições
- Lógica
- Desbastar
- Matemática
- Texto
- Listas
- Uso
- Variáveis
- Funções

Condições

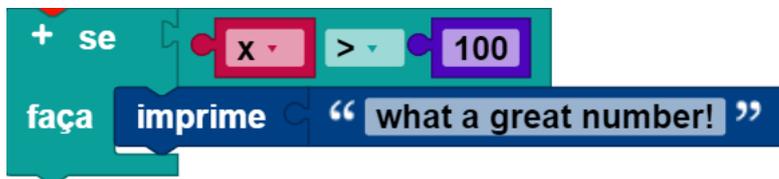
As instruções condicionais são fundamentais para a programação. Eles tornam possível formular distinções de caso, tais como:

- Se houver um caminho para a esquerda, vire à esquerda.
- Se pontuação = 100, imprima "Muito bem!".

Blocos

blocos **se**

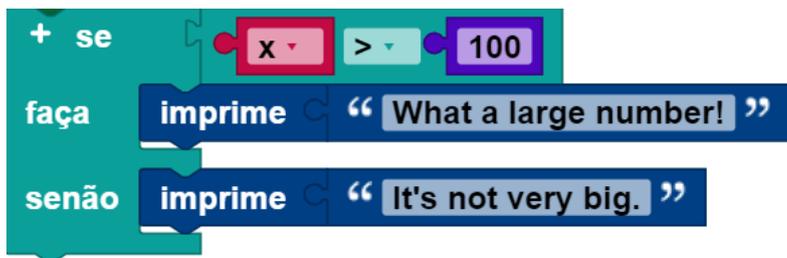
A condição mais simples é um bloco **se**:



Quando isso é executado, o valor da variável **x** é comparado a 100. Quando for maior, "Que grande número!" será emitido. Caso contrário, nada acontecerá.

blocos **caso contrário**

Também é possível indicar que algo deve acontecer se a condição não for verdadeira, como neste exemplo:

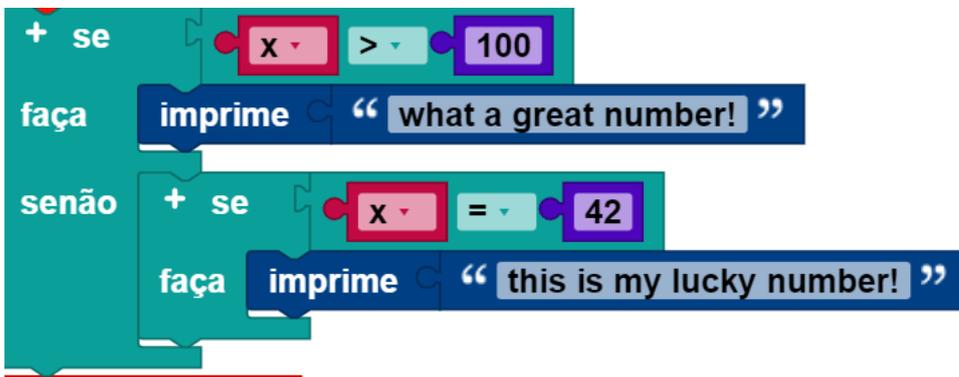


Como no bloco anterior, será emitido "Que grande número!" se $x > 100$. Caso contrário, "Isso não é muito grande." é especificado.

Um bloco **se** tem uma seção **caso contrário**, mas não mais de uma.

blocos **caso contrário, se**

Também é possível testar várias condições com um único bloco **se** adicionando cláusulas **caso contrário se**:

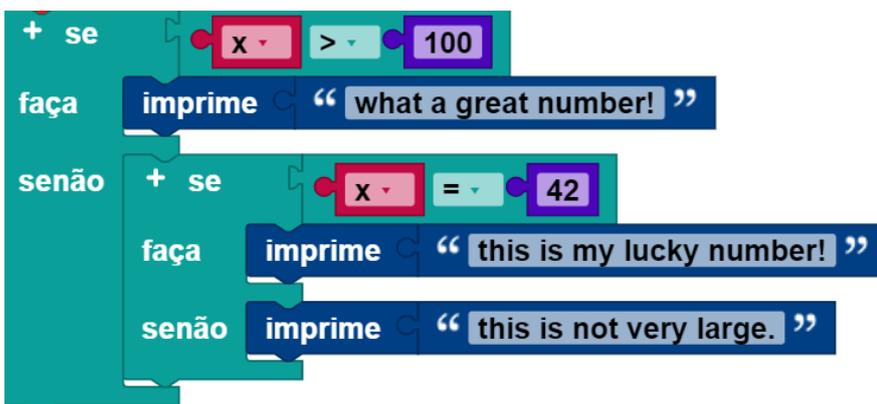


O bloco primeiro verifica se $x > 100$ e retorna "Que número grande!", se for esse o caso. Se este não for o caso, ele verifica se $x = 42$. Em caso afirmativo, ele diz "Este é meu número da sorte!". Caso contrário, nada acontecerá.

Um bloco **se** pode ter qualquer número de seções **caso contrário se**. As condições são avaliadas de cima para baixo até que uma seja satisfeita, ou até que não haja mais condições.

blocos **caso contrário**, **caso contrário**

Blocos **se** podem ter tanto seções **caso contrário se** como **caso contrário**:

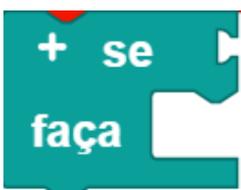


A seção **caso contrário** garante que uma ação será executada mesmo se nenhuma das condições anteriores for verdadeira.

Uma seção **caso contrário** pode aparecer após qualquer número de seções **caso contrário se**, incluindo zero, e então será obtido um bloco **se caso contrário** perfeitamente normal.

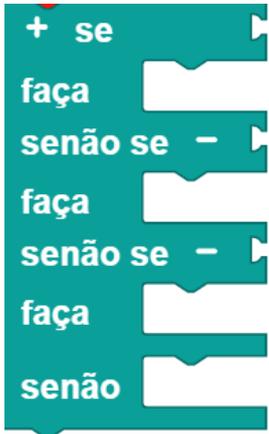
Modificação de bloco

Apenas o bloco **se** simples e o bloco **se caso contrário** aparecem na barra de ferramentas:





Para adicionar cláusulas **caso contrário se** e **caso contrário**, clique no símbolo (+). As cláusulas **caso contrário se** - podem ser removidas com o símbolo (-):



Observe que as formas dos blocos permitem qualquer número de sub-blocos **caso contrário se** a serem adicionados, mas apenas até um bloco **se**.

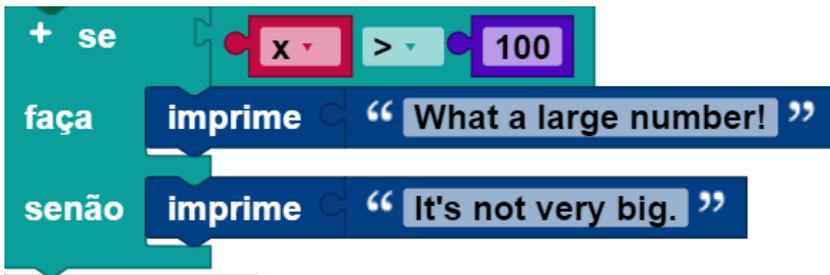
Lógica

A lógica booleana é um sistema matemático simples que possui dois valores:

- **verdadeiro**
- **falso**

Os blocos lógicos no ROBO Pro Coding geralmente existem para controlar condições e loops.

Aqui está um exemplo:



Se o valor da variável *x* for maior que 100, a condição é verdadeira e o texto "Que número grande!" é emitido. Se o valor de *x* não for maior que 100, a condição será falsa e "Isso não é muito grande." é emitido. Os valores booleanos também podem ser armazenados em variáveis e repassados para funções, assim como números, texto e valores de lista.

Blocos

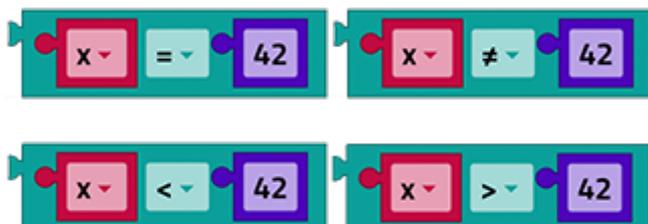
Se um bloco espera um valor booleano como entrada, uma entrada ausente é interpretada como **falsa**. Os valores não booleanos não podem ser inseridos diretamente onde os valores booleanos são esperados, embora seja possível (mas não aconselhável) armazenar um valor não booleano em uma variável e, em seguida, inseri-lo na entrada de condição. Este método não é recomendado e seu comportamento pode mudar em versões futuras do ROBO Pro Coding.

Valores

Um único bloco com uma lista suspensa indicando **verdadeiro** ou **falso** pode ser usado para obter um valor booleano:

Operadores de comparação

Existem seis operadores de comparação. Cada um recebe duas entradas (geralmente dois números) e o operador de comparação retorna **verdadeiro** ou **falso**, dependendo de como as entradas são comparadas.





Os seis operadores são: igual a, não igual a, menor que, maior que, menor ou igual a, maior ou igual a.

Operadores lógicos

O bloco **e** retorna **verdadeiro** se e somente se seus dois valores de entrada forem verdadeiros.



O bloco **ou** retorna **verdadeiro** se pelo menos um de seus dois valores de entrada for verdadeiro.



não

O bloco **não** converte uma entrada booleana em seu oposto. Por exemplo, o resultado é:



falso.

Se nenhuma entrada for feita, o valor é considerado **verdadeiro**, de modo que o bloco a seguir gera o valor **falso**:



No entanto, não é recomendável deixar uma entrada em branco.

três operadores

Os três operadores se comportam como um bloco **se-caso contrário** em miniatura. São necessários três valores de entrada: o primeiro valor de entrada é a condição booleana a ser testada, o segundo valor de entrada é o valor que deve ser retornado se o teste resultar em **verdadeiro**, o terceiro valor de entrada é o valor que deve ser retornado se o teste resultar em falso. No exemplo a seguir, a variável **cor** é definida como vermelho se a variável **x** for menor que 10, caso contrário, a variável **cor** é definida como verde.



Um bloco de três sempre pode ser substituído por um bloco **se-caso contrário**. Os dois exemplos a seguir são exatamente iguais.

```
imprime teste
  se verdadeiro "Game over!"
  se falso "It's going on!"
```

```
+ se x = 0
  faça imprime "Game over!"
  senão imprime "It's going on!"
```

Desbastar

A área "Controle" contém blocos que controlam se outros blocos colocados dentro deles são executados.

Existem dois tipos de blocos de controle: **blocos se caso contrário** (descritos em uma página separada) e blocos que controlam a frequência com que seus interiores são executados. Os últimos são chamados de loops porque seu interior, também conhecido como corpo ou corpo do loop, é (possivelmente) repetido várias vezes. Cada execução de um loop é chamada de iteração.

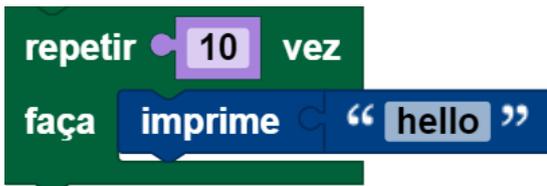
Blocos para criar loops

repetir permanentemente

O **bloco de repetição permanente** executa o código em seu corpo até que o programa termine.

repetição

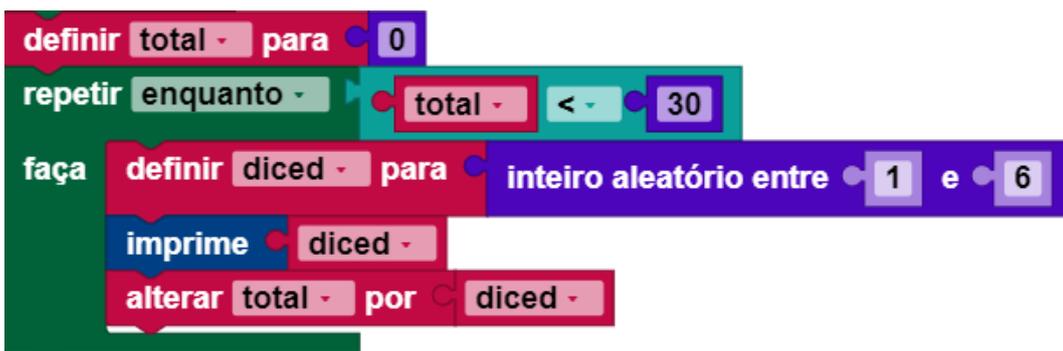
O bloco **repetição** executa o código em seu corpo quantas vezes forem especificadas. Por exemplo, o bloco a seguir diz "Olá!" Dez vezes:



repetir até

Imagine um jogo em que um jogador lança um dado e soma todos os valores lançados, desde que o total seja inferior a 30. Os blocos a seguir implementam este jogo:

1. Uma variável chamada **total** recebe um valor inicial de 0.
2. O loop começa verificando se o **total** é inferior a 30. Nesse caso, ele passará pelos bloqueios do corpo.
3. Um número aleatório no intervalo de 1 a 6 é gerado (para simular um lançamento de dados) e armazenado em uma variável chamada **lançamento de dados**.
4. O número lançado é dado.
5. A variável **total** é aumentada pelo número de **lançamento de dados**.
6. Quando o final do loop é alcançado, o controle volta para a etapa 2.



Após o término do loop, todos os blocos subsequentes (não representados) são executados. No exemplo, o ciclo do loop termina após um certo número de números aleatórios no intervalo de 1 a 6 terem sido emitidos, e o valor da variável **total** tem a soma desses números, que é pelo menos 30.

repetir até

Loops **repetir até** repetem seu corpo **até** que uma condição seja atendida. loops de **repetição** são semelhantes, exceto que eles repetem seu corpo **até** que uma determinada condição seja atendida. Os blocos a seguir são equivalentes ao exemplo anterior porque o loop continua até que o **total** seja maior ou igual a 30.

```
definir total para 0
repetir até total >= 30
  faça
    definir diced para inteiro aleatório entre 1 e 6
    imprime diced
    alterar total por diced
```

contagem de até

O loop **decontagem de até** incrementa o valor de uma variável, começando com um primeiro valor, terminando com um segundo valor e em etapas a partir de um terceiro valor, executando o corpo uma vez para cada valor da variável. Por exemplo, o programa a seguir imprime os números 1, 3 e 5.

```
contar com index de 1 até 5 por 2
faça
  imprime index
```

Como mostram os dois loops a seguir, que geram os números 5, 3 e 1, esse primeiro valor pode ser maior que o segundo. O comportamento é o mesmo, independentemente de o valor incremental (terceiro valor) ser positivo ou negativo.

```
contar com index de 5 até 1 por 2
faça
  imprime index
```

```
contar com index de 5 até 1 por -2
faça
  imprime index
```

para cada

O bloco **para cada** é semelhante ao loop de **contagem de até**, exceto que, em vez de usar a variável de loop em uma ordem numérica, ele usa os valores de uma lista em sequência. O programa a seguir gera cada elemento da lista "alfa", "beta" e "gama":



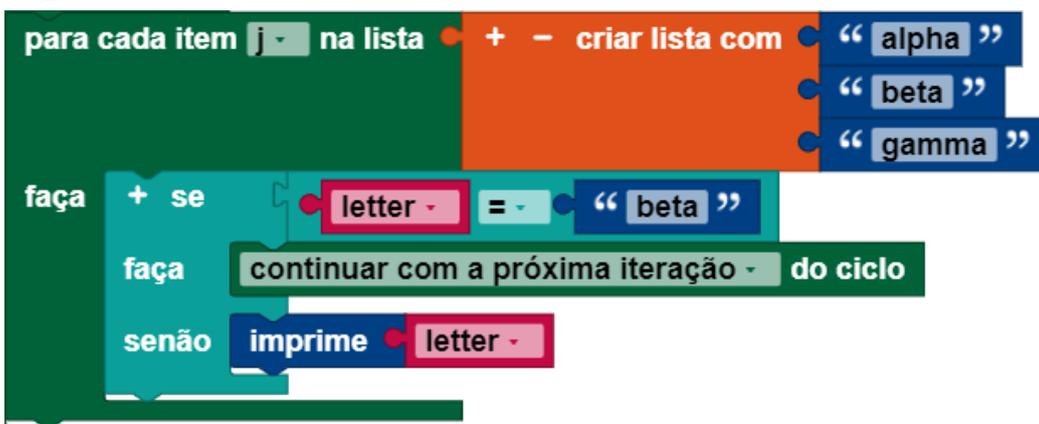
Blocos de cancelar loop

A maioria dos loops é executada até que a condição de término (para **blocos de repetição**) seja satisfeita ou até que todos os valores da variável do loop sejam aceitos (para loops **contar com** e **para cada**). Dois blocos raramente usados, mas ocasionalmente úteis, oferecem opções adicionais para controlar o comportamento do loop. Eles podem ser usados com qualquer tipo de loop, embora os exemplos a seguir mostrem seu uso com o loop **para cada**.

continuar com a próxima interação

continuar com a próxima interação faz com que os blocos restantes no corpo do loop sejam pulados e a próxima interação do loop comece.

O programa a seguir imprime "alfa" na primeira interação do loop. Na segunda interação, o bloco **continuar com a próxima interação** é executado, sendo a saída de "beta" ignorada. A última interação imprime "gama".



Cancelamento de loop

O bloco de **cancelamento de loop** permite uma saída antecipada de um loop. O programa a seguir gera "alfa" na primeira interação e interrompe o loop na segunda interação se a variável do loop for igual a "beta". O terceiro ponto da lista nunca é alcançado.

```
para cada item j na lista + - criar lista com "alpha"  
"beta"  
"gamma"  
faça + se letter = "beta"  
faça sair do ciclo  
senão imprime letter
```

Matemática

Os blocos da categoria Matemática são usados para fazer cálculos. Os resultados dos cálculos podem ser usados como valores para variáveis, por exemplo. A maioria dos blocos de matemática está relacionada a cálculos matemáticos gerais e deve ser autoexplicativa.

Blocos

Números

Use o bloco de números para adicionar qualquer número ao seu programa ou para atribuir esse número como um valor a uma variável. Este programa atribui o número 12 à variável **idade**:



Contas simples

Este bloco possui a estrutura valor-operador-valor. Os tipos de cálculo $+$, $-$, \div , \times e $^$ estão disponíveis como operadores. O operador pode ser selecionado no menu suspenso. Pode ser aplicado diretamente a números ou aos valores das variáveis. Exemplo:



Dieser Block gibt Ergebnis 144 (12^2) aus.

Este bloco retorna o resultado 144 (12^2).

Contas especiais

Este bloco aplica o tipo de cálculo selecionado por meio do menu suspenso ao número ou ao valor da variável colocada atrás dele. As operações aritméticas disponíveis são:

- Raiz quadrada,
- Montante,
- Logaritmo natural,
- Logaritmo decádico,
- Função exponencial com a base e (e^1 , e^2 ,...),
- Função exponencial com base 10 (10^1 , 10^2 ,...),
- Mudança de sinal (multiplicação por -1).

Aqui, e é o número de Euler. Este bloco obtém a raiz quadrada de 16 e define a variável **i** como o resultado.



Funções trigonométricas

Este bloco funciona de maneira semelhante ao bloco descrito acima, com a diferença de que as funções trigonométricas seno, cosseno, tangente e suas funções inversas são utilizadas como operações aritméticas. O número especificado ou o valor da variável especificada é usado na função selecionada no menu suspenso, e o resultado pode ser processado posteriormente no programa. Além disso, há o bloco **arctan2 de X: ... Y: ...**, que permite a saída de um valor de função do arctan2 na faixa de 360° com a ajuda de dois números reais (a serem inseridos como X e Y).

Constantes usadas com frequência

Este bloco funciona da mesma maneira que o bloco numérico, mas você não insere aqui o valor numérico. Em vez disso, constantes usadas com frequência (por exemplo, π) são pré-armazenadas. A constante pode ser selecionada no menu suspenso.

Restante de uma divisão

O bloco **restante do ...** é usado para produzir o restante de uma divisão. Este programa atribui o resto da divisão de 3:2, ou seja, 1, à variável **restante**:



Arredondar

Com o bloco **arredondar ...**, um número decimal especificado ou o valor de uma variável especificada pode ser arredondado para um inteiro. Existem três opções para escolher no menu suspenso:

- com "arredondar" comercialmente arredondado (por exemplo, 4,5 torna-se 5)
- com "arredondamento" é arredondado para cima (por exemplo, 5,1 torna-se 6)
- com "arredondar a partir de", é arredondado (por exemplo, 5,9 torna-se 5)

Avaliação de listas

Com o bloco **... da lista** você pode emitir

- com "Soma", a soma de todos os valores de uma lista,
- com "min", o menor valor de uma lista,
- com "max", o maior valor de uma lista,
- com "valor médio", a média de todos os valores em uma lista,
- com "mediana", a mediana de uma lista,
- com "valor modal", o valor de ocorrência mais frequente em uma lista,
- com "desvio padrão", o desvio do padrão de todos os valores de uma lista,
- com "valor aleatório" um valor aleatório de uma lista

. Todas essas opções podem ser selecionadas no menu suspenso do bloco:

ra **soma da** lista

- ✓ soma da
- menor de uma
- maior de uma
- média de uma
- mediana de uma
- moda de uma
- desvio padrão de uma
- item aleatório de uma

Restringir valores de entrada

O bloco de **restrição ... de ... a ...** permite que os valores de entrada sejam restritos a um determinado intervalo. Antes de um valor de entrada ser processado posteriormente, é testado se ele está dentro do intervalo especificado. Existem três opções de como proceder com um valor inserido:

- O valor está no intervalo, por isso é transmitido inalterado.
- O valor está abaixo do limite inferior do intervalo, portanto, esse limite inferior é transmitido.
- O valor está acima do limite superior do intervalo, portanto, esse limite superior é transmitido.

Neste exemplo, o bloco é usado para limitar o valor da **velocidade** variável às velocidades suportadas pelo motor:

restringe **speed** inferior **0** superior **512**

Gerar valores aleatórios

Ambos os blocos **número aleatório de ... a...** e **fração aleatória** geram um valor aleatório. Assim, o bloco **número aleatório de ... a...** gera um número do intervalo definido. O bloco de **fração aleatória**, por outro lado, produz um valor entre 0,0 (incluído) e 1,0 (excluído).

Texto

Exemplos de textos são:

"Coisa 1"

"12. Março de 2010 "

"" (texto vazio)

O texto pode conter letras (minúsculas ou maiúsculas), números, sinais de pontuação, outros símbolos e espaços.

Blocos

Criação de texto

O bloco a seguir cria o texto "Olá" e o salva na variável chamada **Saudação**:



O bloco de **criação de texto** combina o valor da variável **saudação** e o novo texto "mundo" com o texto "Olámundo". Observe que não há espaço entre os dois textos, uma vez que não havia nenhum em ambos os textos originais.



Para aumentar o número de entradas de texto, clique no símbolo (+). Para remover a última tarefa, clique no símbolo (-).

Alteração de texto

O bloco **em ... anexo** adiciona o texto dado à variável especificada. Neste exemplo, ele altera o valor da variável **saudação** de "Olá" para "Olá, você!":



Comprimento do texto

O **comprimento do** bloco conta o número de caracteres (letras, números etc.) contidos em um texto. A duração de "Nós somos#1!" é 12 e o comprimento do texto vazio é 0.



Verifique se há texto em branco

O módulo **está vazio** verifica se o texto especificado está vazio (tem comprimento 0). O resultado é **verdadeiro** no primeiro exemplo e **falso** no segundo exemplo.

“ ” está vazio

greeting está vazio

Busca de texto

Esses blocos podem ser usados para verificar se um texto aparece em outro texto e, em caso afirmativo, onde ele aparece. Por exemplo, a primeira ocorrência de "l" em "Olá" é perguntada aqui, o resultado é 2:

no texto “ hello ” primeira ocorrência do texto “ e ”

Este pede a última ocorrência de "l" em "Olá", que também resulta em 2:

no texto “ hello ” última ocorrência do texto “ e ”

Independentemente de a primeira ou última ocorrência ser selecionada, este bloco retorna o resultado 0, pois "Olá" não contém um "z".

no texto “ hello ” primeira ocorrência do texto “ z ”

Extração de texto

Extrair um único caractere

Isso resulta em "b", a segunda letra em "abcde":

no texto “ abcde ” obter letra n^o 2

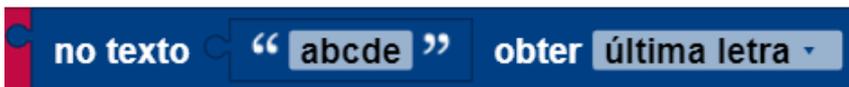
Isso dá "d", a penúltima letra em "abcde":

no texto “ abcde ” obter letra n^o a partir do final 2

Isso dá "a", a primeira letra em "abcde":

no texto “ abcde ” obter primeira letra

Isso dá "e", a última letra em "abcde":



Isso dá cada uma das 5 letras em "abcde" com igual probabilidade:



Nenhum deles altera o texto do qual é extraído.

Extrair um trecho de texto

Com o bloco **no texto ... corrente de caracteres**, um trecho de texto pode ser extraído, começando com:

- Letra #
- Letra # do final
- Primeira letra

começa e com:

- Letra #
- Letra # do final
- última letra

termina.

O exemplo a seguir extrai "abc":



Ajustar a caixa de texto

Este bloco cria uma versão do texto de entrada que pode ser salvo tanto em

- MAIÚSCULAS (todas as letras em maiúsculas) ou em
- minúsculas (todas as letras em minúsculas) ou em
- Substantivos próprios (primeiras letras maiúsculas, outras letras minúsculas).

O resultado do seguinte bloco é "OLÁ":



Os caracteres não alfabéticos não são afetados. Observe que este bloco não funciona no que se refere ao texto em idiomas que não diferenciam maiúsculas de minúsculas, como, por exemplo, o chinês.

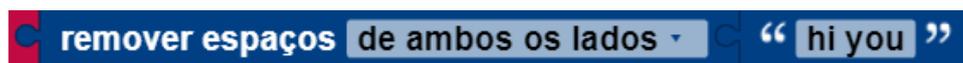
Cortar (remover) espaços

O bloco a seguir remove espaços, dependendo do que está definido no menu suspenso (triângulo pequeno):

- no início do texto

- no final do texto
- em ambos os lados do texto

O resultado do bloco a seguir é "Oi você".



Espaços no meio do texto não são afetados.

Emitir texto

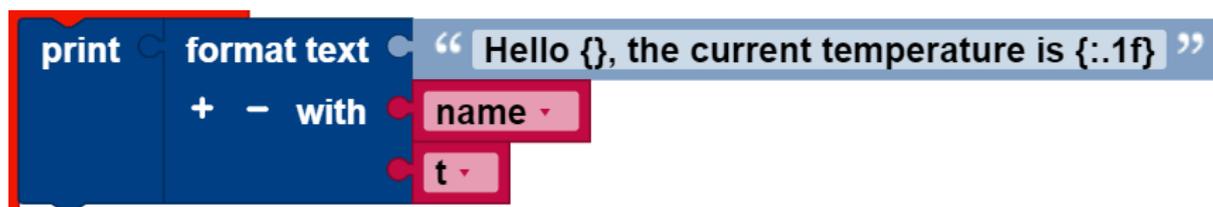
O bloco **emitir** faz com que o valor de entrada seja exibido na janela do console:



Em nenhum caso será enviado para a impressora, como o nome pode sugerir.

Emitir texto com formatação

Com o bloco **formatar texto**, os blocos podem emitir texto com conteúdo variável formatado. Todos os marcadores de posição `{}` no texto são substituídos pelo conteúdo das variáveis anexadas após o texto. A formatação pode ser especificada entre chaves. A formatação `{:.1f}`, por exemplo, exibe apenas a primeira casa decimal do ponto decimal na variável `t`.



Listas

Como na linguagem cotidiana, uma lista no ROBO Pro Coding é uma coleção ordenada de elementos, como, por exemplo, uma lista de "tarefas" ou uma lista de compras. Os itens em uma lista podem ser de qualquer tipo e o mesmo valor pode aparecer mais de uma vez em uma lista.

Criar uma lista

criar lista com

Com a o bloco **criar lista com**, você pode inserir os valores iniciais em uma nova lista. Este exemplo cria uma lista de palavras e as coloca em uma variável chamada **letras**:



Referimo-nos a esta lista como ["alfa", "beta", "gama"].

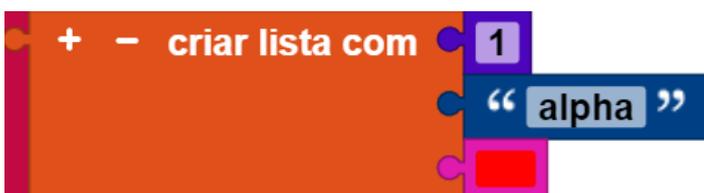
Isso mostra a criação de uma lista de **números** :



Para fazer uma lista de **cores**:



É menos comum, mas possível, fazer uma lista de valores de diferentes tipos:



Alterar o número de entradas

Para alterar o número de entradas, clique ou toque no símbolo da engrenagem. Isso abrirá uma nova janela. Você pode arrastar os sub-blocos do item do lado esquerdo da janela para o bloco da lista à direita para adicionar uma nova entrada:

Embora o novo item tenha sido adicionado abaixo no exemplo, ele pode ser adicionado em qualquer lugar. Da mesma forma, sub-blocos de itens indesejados podem ser arrastados para a esquerda do bloco de listas.

Criar lista com elemento

Com o bloco **criar lista com elemento**, você pode criar uma lista que contém o número especificado de cópias de um elemento. Por exemplo, os blocos a seguir colocam as variáveis **palavras** na lista ["muito", "muito", "muito"].



Verificar o comprimento de uma lista

está vazio

O valor de um bloco **está vazio** é **verdadeiro** se sua entrada for a lista vazia e **falso** se for qualquer outra coisa. Esta entrada é **verdadeira**? O valor do bloco a seguir seria **falso** porque as variáveis cores não estão vazias: Ele tem três elementos.



Observe a semelhança com o bloco **está vazio** para o texto.

Comprimento de

O valor do bloco **comprimento de** é o número de elementos que estão na lista usados como entrada. O valor do bloco seguinte seria, por exemplo, 3, uma vez que a **cor** tem três elementos:



Observe que o bloco **comprimento de** informa quantos elementos existem na lista, não quantos elementos diferentes existem nela. Por exemplo, o seguinte é 3, embora as **palavras** sejam compostas por três cópias do mesmo texto:



Observe a semelhança com o bloco **comprimento de** para o texto.

Buscar elementos em uma lista

Esses blocos encontram a posição de um elemento em uma lista. O exemplo a seguir tem o valor 1 porque a primeira ocorrência de "muito" está no início da lista de palavras (["muito", "muito", "muito"]).



O resultado do seguinte é 3 porque a última ocorrência de "muito" em **palavras** está na posição 3.



Se o elemento não aparecer em nenhum lugar da lista, o resultado será o valor 0, como neste exemplo:



Esses blocos se comportam da mesma maneira que os blocos para localizar letras no texto.

Obtenha elementos de uma lista

Recuperar um único item

Lembre-se da definição da lista de **cores**:



O bloco a seguir recebe a cor azul porque é o segundo elemento da lista (contando a partir da esquerda):



Ele fica verde porque é o segundo elemento (contando da extremidade direita):



Este recebe o primeiro elemento, vermelho:



Este resulta no último elemento, amarelo:



Este seleciona aleatoriamente um elemento da lista, com uma probabilidade igual de retornar um dos elementos vermelho, azul, verde ou amarelo.



Obter e remover um item

O menu suspenso é usado para alterar o bloco **recuperar da lista ...** para o bloco **recuperar da lista ... e remover**, que fornece o mesmo resultado, mas também altera a lista:



Este exemplo define a **primeira letra** da variável como "alfa" e deixa as letras restantes (["beta", "gama"]) na lista.



Remover uma entrada

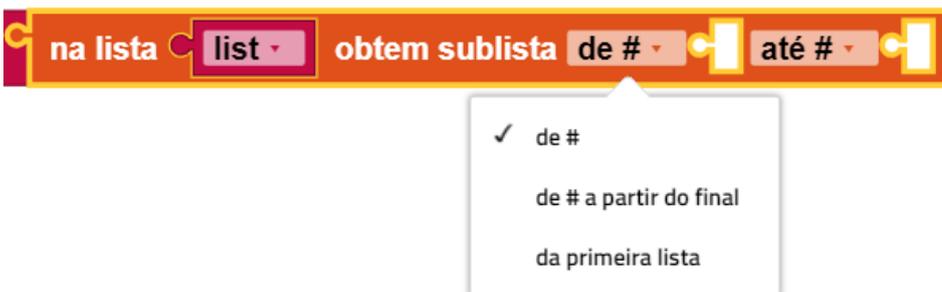
Ao selecionar **Remover** no menu suspenso, o item à esquerda do bloco desaparecerá:



Isso remove o primeiro elemento das **letras**.

Obter uma sublista

O bloco **da lista ... obter sublista** é semelhante ao bloco em **obter da lista ...**, exceto por extrair uma sublista em vez de um único elemento. Existem várias opções para especificar o início e o fim da sublista:



na lista ▾ list ▾ obtem sublista de # ▾ até # ▾

- ✓ até #
- até #, a partir do final
- para o último

Neste exemplo, uma nova lista de **primeiras letras** é criada. Esta nova lista tem dois elementos: ["alfa", "beta"].

definir letters ▾ para + - criar lista com " alpha " " beta " " gamma "

definir first letter ▾ para na lista letters ▾ obtem sublista da primeira lista ▾ até # ▾ 2

Observe que este bloco não altera a lista original.

Adicionar elementos a uma lista

Substituir elementos de uma lista

O bloco **na lista ... substituir** substitui o elemento em uma determinada posição em uma lista por outro elemento.

na lista ▾ words ▾ definir ▾ # ▾ como ▾

- ✓ #
- # a partir do final
- primeiro
- último
- aleatório

O significado das opções suspensas individuais pode ser encontrado na seção anterior.

O exemplo a seguir faz duas coisas:

1. A lista **palavras** é criada com 3 elementos: ["muito", "muito", "muito"].
2. O terceiro elemento da lista é substituído por "bom". O novo valor das **palavras** é ["muito", "muito", "bom"]

definir words ▾ para criar lista com o item " very " repetido 3 vezes

na lista ▾ words ▾ definir ▾ # ▾ 3 como " well "

Inserir elementos em um ponto específico de uma lista

O **bloco inserir na lista ... no bloco** é chamado por meio do menu suspenso do bloco **substituir na lista ...**:



Ele insere um novo elemento na posição especificada na lista, antes do elemento que estava anteriormente nesta posição. O exemplo a seguir (baseado em um exemplo anterior) faz três coisas:

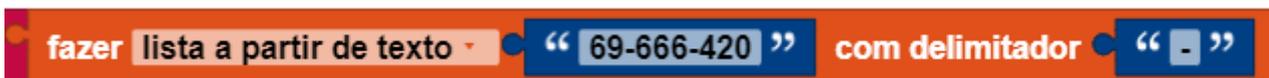
1. A lista **palavras** é criada com 3 elementos: ["muito", "muito", "muito"].
2. O terceiro elemento da lista é substituído por "bom". O novo valor das **palavras** é ["muito", "muito", "bom"].
3. A palavra "Seja" é inserida no início da lista. O novo valor das **palavras** é ["muito", "muito", "bom"].



Dividir correntes e unir listas

Criar uma lista a partir de um texto

O módulo **criar lista a partir do texto** divide o texto especificado em partes com a ajuda de um delimitador:



O exemplo acima retorna uma nova lista que contém três trechos de texto: "311", "555" e "2368".

Crie um texto a partir de uma lista

O módulo **criar texto a partir de uma lista** combina uma lista em um único texto com a ajuda de um separador:



No exemplo acima, um novo texto é retornado com o valor: "311-555-2368".

Blocos relacionados

Impressão de uma lista

O módulo **impressão** na categoria Texto pode gerar listas. O resultado do programa a seguir é a saída do console mostrada:



Console

O programa inicia ...

```
['alpha', 'beta', 'gamma']
```

Programa concluído.

Executar algo para cada elemento em uma lista

O bloco **para cada** na categoria Controle executa uma operação em cada item de uma lista. Por exemplo, este bloco imprime cada item da lista individualmente:



Isso não remove os itens da lista original.

Veja também os exemplos para os [blocos de cancelar loop](#) .

Uso

Com o ROBO Pro, a categoria de uso inclui blocos de codificação dos seguintes tipos:

- Seleção de cor
- Esperar
- Código Python
- Iniciar
- Execução de função

Seleção de cor

Este bloco é usado como um valor de entrada quando uma cor é solicitada (por exemplo, em equilíbrio da cor pela câmera). Ao clicar ou tocar na cor, uma das 70 cores pode ser selecionada de uma paleta de cores.

Esperar

Esperar até que o tempo acabe

O bloco de **espera [...]** evita que o programa continue pelo tempo de espera especificado. A unidade de tempo pode ser selecionada no menu suspenso (triângulo pequeno) e a duração desejada da pausa no campo de entrada atrás dela.

Esperar com condição

No bloco **esperar até**, a pausa não está ligada ao tempo, mas ao cumprimento de uma condição (por exemplo, pressionamento de um botão). A condição é anexada ao bloco **esperar até**.

Código Python

Caso deseje integrar o código Python existente no ROBO Pro Coding, você pode inseri-lo no bloco de **código Python**. O programa então executa tudo o que foi escrito em Python no bloco.

Iniciar

Além disso, o bloco **iniciar quando** é sujeito a uma condição. Somente quando esta condição é atendida o programa no corpo do bloco é iniciado.

Execução de função

Com a **função executar ... em uma thread**, a função selecionada pode ser executada em uma thread separada. Em alguns casos, essa medida pode permitir que um programa continue a responder às entradas e seja executado mais rapidamente.

Variáveis

Usamos o termo variável como é usado em matemática e em outras linguagens de programação: um valor nomeado que pode ser alterado (variado). As variáveis podem ser criadas de diversas maneiras.

- Alguns blocos, como **contar com** e **para cada** usam uma variável e definem seus valores. Um termo tradicional da ciência da computação para essas variáveis é o termo variáveis de loop.
- Funções definidas pelo usuário (também conhecidas como "procedimentos") podem definir entradas, criando variáveis que só podem ser usadas dentro daquela função. Tais variáveis são tradicionalmente chamadas de "parâmetros" ou "argumentos".
- Os usuários podem alterar as variáveis a qualquer momento usando o bloco **definir**. Essas são tradicionalmente chamadas de "variáveis globais". Elas podem ser usadas em qualquer lugar no código ROBO Pro Coding.

Menu suspenso

Ao clicar no símbolo suspenso (triângulo pequeno) de uma variável, o seguinte menu aparecerá:



O menu oferece as seguintes opções.

- a exibição dos nomes de todas as variáveis existentes definidas no programa.
- "Renomear variável ...", ou seja, alterar o nome desta variável onde quer que ela apareça no programa (selecionar esta opção abre uma consulta para o novo nome)
- "Excluir variável ...", ou seja, a exclusão de todos os blocos que se referem a esta variável, onde quer que ela ocorra no programa.

Blocos

Determinar

O bloco **definir** atribui um valor a uma variável e cria a variável caso ela ainda não exista. Por exemplo, o valor da variável **idade** é definido como 12:



Lembrar

O bloco **recuperação** fornece o valor armazenado em uma variável sem alterá-lo:



É possível, porém uma má ideia, escrever um programa em que um bloco **recuperação** sem um conjunto de blocos anterior correspondente.

Alterar

O bloco **alterar** adiciona um número a uma variável.



O bloco **alterar** é uma abreviatura para a seguinte construção:



Exemplo

Observe o seguinte código de exemplo:



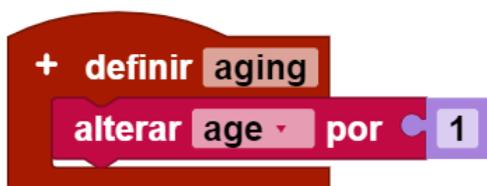
O primeiro conjunto de blocos cria uma variável chamada **idade** e **define** seu valor inicial para o número 12. A segunda linha de blocos **recupera** um valor de 12, adiciona 1 a isso e armazena a soma (13) na variável. A mensagem é emitida na última linha: "Parabéns! Agora você tem 13 anos".

Funções

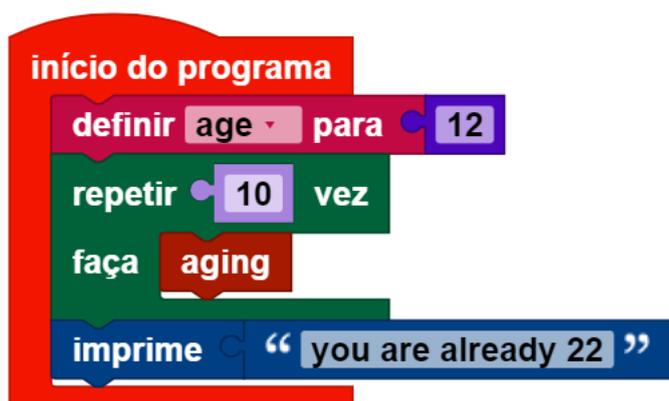
As funções são usadas para tornar partes do código reutilizáveis e, assim, estruturar o código como um todo. Caso você preencha um bloco de funções, um novo bloco aparecerá no menu de funções com o mesmo nome deste bloco. Agora é possível inserir apenas o bloco com o nome da função no programa principal. Quando o programa é executado, este bloco encaminha para o código na função de mesmo nome e o processa.

Função simples

O bloco de funções simples pode ser usado para criar uma função com o nome inserido no campo de texto. A função pode conter qualquer número de variáveis que podem ser adicionadas usando-se o símbolo de engrenagem. Esta função **envelhecimento** adiciona 1 à variável **idade**:



A função pode então ser usada no programa principal:



Função com valor de retorno

Este bloco permite que você crie uma função com um valor de retorno. Este valor de retorno pode então ser usado no programa principal. Aqui está um exemplo:

