# Lógica de programação

## 1. O que é?

Lógica de Programação é o modo como se escrevem programas de computador através de uma sequência de passos para executar uma ou várias funções, esta sequência é o algoritmo.

Algoritmo é basicamente uma “receita” para executar uma tarefa ou resolver um problema, ele tem começo meio e fim, assim como a receita de bolo, se você seguir os passos descritos, terá um resultado. Assim como na atividade de fazer um bolo, utilizando o algoritmo, você pode fazer atividades repetitivas ou ao mesmo tempo, pode delegar funções ou esperar um passo ser concluído para iniciar outro, desde que seja algo finito existem várias possibilidades.

Para se escrever um texto é necessário entender a linguagem. A língua que o computador entende são sequências lógicas de zeros e uns, ou seja, um sistema binário, apenas dois elementos. Para desenvolver a lógica de programação é necessário escrever utilizando uma linguagem de programação a qual será “traduzida”, nesse caso, será compilada e posteriormente transformada em código binário, para que a máquina entenda a sequência de comandos e execute uma tarefa.

A lógica de programação em geral baseia-se em uma lógica computacional compartilhada por humanos e máquinas, que é o que exploramos à medida que continuamos a interagir com as novas tecnologias. Com isso em mente, é possível desenvolver definições mais específicas de uma lógica de programação, à base de um trecho de código.

## 2. Linguagens de programações

Apesar de existir uma grande variedade de linguagens de programação, algumas podem ser mais adequadas para certos negócios ou empresas. É preciso cuidado, pois você pode se tornar um mestre em uma linguagem específica, mas, se ela não for muito requisitada no mercado de trabalho, isso significa que poderá ter dificuldades em arranjar um emprego na área.

Alguns Exemplos de Linguagens são:

- JavaScript
- Python
- Java
- C#
- PHP
- C++

E para esse conteúdo selecionamos a linguagem Python para trabalhar devido a sua facilidade de aprendizado e grande popularidade no mercado.

## 3. Introdução Python

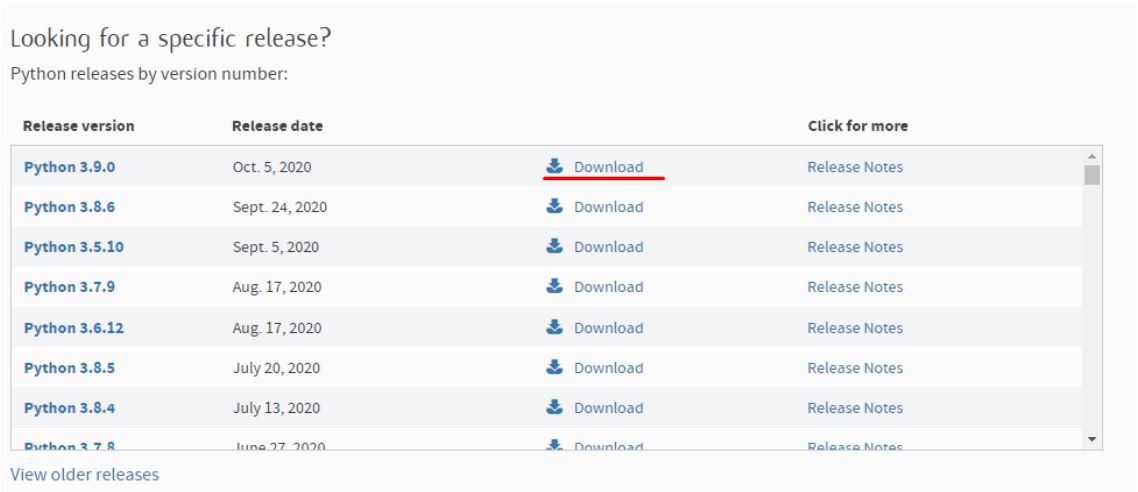
Python é uma linguagem interpretada e fracamente tipada. Além disso, é uma linguagem de propósito geral. Portanto, pode ser utilizada para solucionar qualquer tipo de problema, o qual pode ser atendido com um sistema desktop, para a web ou mobile.

## 4. Primeiros passos

- Download o Python

Para zer o download do arquivo de instalação que está no site principal ( https://www.python.org/ ).

Para baixar o arquivo, vá no item “Downloads” e clique no link que tem o nome da versão mais recente estável.




- Instalação do Python

Após o término do download, abra o executável para começar a instalação do Python. A tela inicial da instalação abrirá, como na Figura 3. Marque a opção “Add Python 3.* to Path”, para já deixar o Python configurado nas variáveis de ambiente do Windows. Clique em “Install Now” e aguarde até que a instalação seja concluída.

## 5. Começando com a linguagem

A sintaxe de uma linguagem são as regras que nos dizem como escrever um código para que ele seja compilado sem erros. A seguir, veremos algumas dentre as principais características da sintaxe do Python, com exemplos de código e explicações.

-Variável

Nas linguagens de programação, a variável é um local na memória que reservamos para armazenar dados. Para criar uma variável no Python basta informar nome e valor.

```
1 | total_alunos = 10
2 | print(total_alunos)
```

Neste código declaramos a variável total_alunos na Linha 1 e exibimos o seu valor na Linha 2 com o comando print.

O nome de uma variável deve começar com uma letra ou sublinhado e pode ser seguido por letras, números e o caractere de sublinhado _. No Código temos um exemplo de variável que recebeu um nome composto por duas palavras separadas por um sublinhado. Outras formas de se escrever variáveis são erradas ou incomuns.

```
1 | total_alunos = 10
2 | _total_alunos = 10
3 | 1total_alunos = 10
4 |
5 | print(total_alunos)
6 | print(_total_alunos)
7 | print(1total_alunos)
```

Na Linha 3, temos uma variável que se inicia com um número, o que gera um erro de sintaxe. A Linha 7 também contém um erro, pois está tentando exibir uma variável que tem um nome incorreto.

Python é uma linguagem case-sensitive, que faz diferenciação entre letras maiúsculas e minúsculas. Sendo assim, total_alunos e Total_alunos seriam nomes de variáveis diferentes.

```
1  total_aulas = 5
2  TOTAL_AULAS = 10
3
4  print(total_aulas)
5  print(TOTAL_AULAS)
```

Nesse exemplo, total_aulas é uma variável diferente de TOTAL_AULAS.

- Não é necessário usar ponto vírgula no fim de um comando

Em boa parte das linguagens de programação as linhas de código devem terminar com um ponto e vírgula (;). No Python isso é opcional. Um caso onde o ponto e vírgula seria necessário é quando temos mais de um comando em uma mesma linha

- Parênteses são opcionais

Vale ressaltar que no Python não há a necessidade de fazer o uso de parêntesis para as estruturas de controle de fluxo como if, for e while.

## 6. Tipos de Dados em Python

Uma característica que facilita muito a vida do desenvolvedor são as conversões de tipos em Python. Veja na Listagem 2 alguns exemplos.

Vamos entender melhor cada tipo:

- Inteiros

Para a declaração de números inteiros, necessitamos que estes estejam entre -2147483648 a 2147483647. Cada um ocupa quatro bytes na memória e pode armazenar tantos valores decimais (de base 10), quanto valores octais (de base 8) e hexadecimais (base 16). Para declarar um inteiro octal, o número 0 (zero) tem que ser prefixado ao número, como em 0123; e para definir um número hexadecimal, o prefixo 0x ou 0X deve ser utilizado, como 0xFFFFFF ou 0X006699.

- Long

Representa números inteiros longos e pode armazenar números tão grandes quanto a memória puder armazenar. Assim como o tipo int, o long também pode armazenar números tanto decimais, quanto octais e hexadecimais. Para declarar um valor long mesmo se o número a ser atribuído estiver na faixa de valores do tipo int é necessário sufixar a letra L (minúscula ou maiúscula) como em 524511574362l, 0xDDEFBDAEFBDAECBFBAEL e 0122L.

Exemplo: a = 0xDDEFBDAEFBDAECBFBAEL representa: hexadecimal Long

- Float

Representa números reais e que possuem sinal de expoente (e ou E). Esses números são comumente chamados de floating-point numbers ou números de ponto flutuante. Por exemplo: 0.0042, .005, 1.14159265 e 6.02e23 (o mesmo que 6.02 x $10^{23}$).

- Bool

O tipo bool foi adicionado na versão 2.2 do Python como uma especialização do tipo int. Os valores do tipo bool podem representar dois valores completamente distintos: True (igual ao int 1) e False (igual ao int 0) para, respectivamente, verdadeiro e falso. Exemplo: a = True e b = False.

- None Type

NoneType é o tipo de None, uma constante embutida do Python que, assim como True e False, e é frequentemente utilizada para representar a ausência de um valor, similar ao null na linguagem C e derivadas. Exemplo: a = None (o mesmo que null em Java).

- String

Para atribuirmos a uma variável uma referência do tipo string, basta que coloquemos entre aspas simples, duplas ou triplas

## 7. Entrada de Dados em Python

**No Python**, a leitura de dados do teclado é feita através das funções raw_input e input. Contudo, em versões superiores a 3.0, a função raw_input não funciona.

```
1 import sys
2
3 nome = input("Digite seu nome: ")
4 idade = input("Digite sua idade: ")
5 print("Digite seu sexo: ")
6 sexo = sys.stdin.readline()
7
8 print("Nome:"+nome + "\n" + "Sexo: %s Idade: %s" %(sexo,idade))
```

Na linha 8 são utilizadas duas maneiras para exibir as strings, e também são utilizadas duas maneiras de fazer a leitura do teclado: uma diretamente na variável com exibição de uma mensagem, e outra em que utilizamos um método readline(), que está contido na importação feita na linha 1.

## 8. Comandos de Condições

- IF, ELIF, ELSE

If, elif e else são declarações condicionais que fornecem a você a tomada de decisão necessária quando você deseja executar o código com base em uma condição particular.

A instrução if ... elif ... else usada em Python ajuda a automatizar o processo de tomada de decisão

```
1 if  CONDIÇÃO :
2   BLOCO DE CÓDIGO
3 elif  CONDIÇÃO :
4   BLOCO DE CÓDIGO
5 else :
6   BLOCO DE CÓDIGO
```

- IF

A condição if é considerada a mais simples das três e toma uma decisão com base no fato de a condição ser verdadeira ou não. Se a condição for verdadeira, ele imprime a expressão recuada. Se a condição for falsa, ele ignora a impressão da expressão recuada.

- ELSE

A condição if-else adiciona uma etapa adicional no processo de tomada de decisão em comparação com a instrução if simples. O início de uma instrução if-else opera de maneira semelhante a uma instrução if simples; entretanto, se a condição for falsa, em vez de não imprimir nada, a expressão indentada sob else será impressa.

- ELIF

A mais complexa dessas condições é a condição if-elif-else. Quando você se depara com uma situação em que tem várias condições, pode colocar quantas condições elif forem necessárias entre a condição if e a condição else.

```python
# coding:utf-8
dedos = int(input("Você tem quantos anos? "))

if dedos == 18:
  print("Você tem 18 anos")
elif dedos > 18:
  print("Você tem mais de 18 anos")
else:
  print("Você é menor de idade”)
```

- WHILE

O WHILE já se enquadra nas estruturas de repetição. Esse operador é utilizado para executar um bloco de código várias vezes, enquanto uma determinada condição for atendida. Traduzindo, esta estrutura funciona da seguinte forma: ENQUANTO (condição for atendida) FAÇA ALGO.

A sintaxe básica de uso dessa estrutura é apresentada abaixo.

```
while(condição)
{
  //comandos
}
```

Nesse caso, a condição avaliada também deve ser booleana, podendo assumir os valores true ou false.

Exemplo de While:

```
var $contador = 0;
while($contador < 10)
{
echo $contador;
$contador++;
}
```

Nesse exemplo, temos uma variável contador cujo valor é avaliado e, enquanto for menor que dez, um código é executado, imprimindo este valor e o incrementando em uma unidade.

Após a execução do bloco de comandos, o valor da expressão é novamente avaliado e, se for válido, os códigos são novamente executados

- FOR

A estrutura de repetição FOR é utilizada para se executar um conjunto de comandos por um número definido de vezes. Para esse operador, são passados uma situação inicial, uma condição e uma ação a ser executada a cada repetição.

Em geral informamos uma variável que serve como contador de repetições, com seu valor inicial, uma condição a ser atendida para que cada repetição seja executada e um incremento ao contador.

Observando a sintaxe a seguir fica mais fácil compreender o funcionamento.

```
for(valor inicial; condição; incremento)
{
  //comandos
}
```

Exemplo de uso do FOR:

```
for($contador = 0; $contador < 10; $contador++)
{
  //comandos
}
```

O código acima pode ser entendido como “com o contador partindo do zero e enquanto este for menor que 10, execute os comandos e incremente uma unidade em seu valor”.

## 9. LISTAS

Uma estrutura de dados que podem armazenar, inclusive simultaneamente, objetos de diferentes tipos, como strings, números e até mesmo outras listas.

- Criando uma lista

Em **Python**, uma **lista é representada como uma sequência de objetos** separados por vírgula e dentro de colchetes [], assim, uma lista vazia, por exemplo, pode ser representada por colchetes sem nenhum conteúdo.

```
1  01 >>> lista = []
2  02 >>> lista
3  03 []
4  04 >>> lista = ['O carro','peixe',123,111]
5  05 >>> lista
6  06 ['O carro', 'peixe', 123, 111]
7  07 >>> nova_lista = ['pedra',lista]
8  08 >>> nova_lista
9  09 ['pedra', ['O carro', 'peixe', 123, 111]]
```

Linhas 1 a 3: Declaração de uma lista sem nenhum elemento e sua impressão, que apresenta apenas os colchetes, indicando que a lista está vazia;

Linhas 4 a 6: Atribuição de duas strings e dois inteiros à variável lista e sua posterior impressão;

Linhas 7 a 9: Criação e impressão da variável nova_lista, que é inicializada com a string 'pedra', e uma outra lista.

As possibilidades de declaração e atribuição de valores a uma lista são várias, e a opção por uma ou outra dependerá do contexto e aplicação.

- Comprimento de uma lista

O comprimento de uma lista, ou o número de itens que a compõem, pode ser obtido a partir da função len(), como mostra o código abaixo, em que é impresso o valor 2, indicando que a variável nova_lista contém 2 elementos.

```
01 >>> len(nova_lista)
02 2
```

- Concatenação e multiplicação

Também é possível concatenar listas por meio do operador de adição + e multiplicá-las por um inteiro, o que gerará várias cópias dos seus itens. O código a seguir traz alguns exemplos dessas operações:

```
01 >>> lista+nova_lista
02 ['O carro azul', 'peixe', 123, 111, 'pedra', ['O carro azul', 'peixe', 123, 111]]
03 >>> lista*3
04 ['O carro azul', 'peixe', 123, 111, 'O carro azul',
'peixe', 123, 111, 'O carro azul', 'peixe', 123, 111]
```

- Verificando a existência de itens em uma lista

Em alguns casos é preciso verificar se um determinado valor está contido em uma lista. Para isso, em Python utilizamos o operador in, que indicará True se objeto pertencer ao conjunto, e False caso contrário. No código a seguir temos exemplos de uso desse operador:

```
01 >>> 'peixe' in lista
02 True
03 >>> 'gato' in lista
04 False
```

- Valores mínimos, máximos e soma

Python oferece ainda as funções min(), max() e sum(), através das quais é possível encontrar, respectivamente, o menor valor, o maior valor ou ainda realizar a soma de todos os elementos da lista. Na Listagem 3 podemos ver como utilizá-las.

```
01 >>> numeros = [14.55, 67, 89.88, 10, 21.5]
02 >>> min(numeros)
03 10
04 >>> max(numeros)
05 89.88
06 >>> sum(numeros)
07 202.93
```

## 9.1 Métodos das listas

Nas seções anteriores, vimos os operadores e funções que recebem uma lista como argumento, retornam um resultado, mas não efetuam alterações na sua estrutura. A partir de agora veremos métodos pertencentes ao tipo lista, e que nos permitem incluir ou remover elementos, bem como classificar as coleções.

O primeiro método a ser analisado é o append(), que tem por objetivo adicionar um novo elemento no final da lista, conforme mostra o código abaixo:

```
01 >>> livros = ['Java','SqlServer', 'Delphi','Python']
02 >>> livros.append('Android')
03 >>> livros
04 ['Java', 'SqlServer', 'Delphi', 'Python', 'Android']
```

Após criar, na primeira linha, uma lista com um conjunto de valores, adicionamos a ela um novo item: “Android”. Como podemos ver na linha 4, o elemento é adicionado após o último item. Caso desejemos fazer essa inserção em uma posição específica, podemos utilizar o método insert() que, além do elemento a ser inserido, recebe também o índice que ele deve assumir, como mostra o código abaixo:

```
01 >>> livros = ['Java', 'SqlServer', 'Delphi', 'Python', 'Android']
02 >>> livros.insert(0,'Oracle')
03 >>> livros
04 ['Oracle', 'Java', 'SqlServer', 'Delphi', 'Python', 'Android']
```

## 10.AGORA É COM VOCÊ

Programação é um universo amplo que está sempre em constante atualização. sendo assim os códigos, maneiras, macetes, estão em constante mudança, tornando-se assim umas das principais características dos programadores o estudo individual e busca por novas tecnologias.

Aproveite essa era de internet e busque novos cursos e conteúdo online.

Aproveite também nossos outros conteúdos disponibilizados especialmente para você.