



INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 01 – LÓGICA DA PROGRAMAÇÃO

O QUE IREMOS APRENDER

- 01** IMPORTÂNCIA DA LÓGICA
- 02** ALGORITMO
- 03** VARIÁVEIS
- 04** TIPOS DE DADOS
- 05** FUNÇÕES INTERNAS MAIS USADAS
- 06** FORMATAÇÃO COM F STRING
- 07** OPERADORES ARITMÉTICOS
- 08** INTRODUÇÃO AULA 02

Introdução

Um programador é conhecido, também, como desenvolvedor. Tem a função de escrever códigos que posteriormente se tornam comandos/instruções para os computadores. Os comandos são transformados para a linguagem digital, e criam um contexto de funcionamento, conforme o que foi projetado.

Esse agrupamento de instruções se transformará em aplicação tecnológica. A preocupação dos desenvolvedores é com a codificação, a experiência do usuário e a segurança de dados.

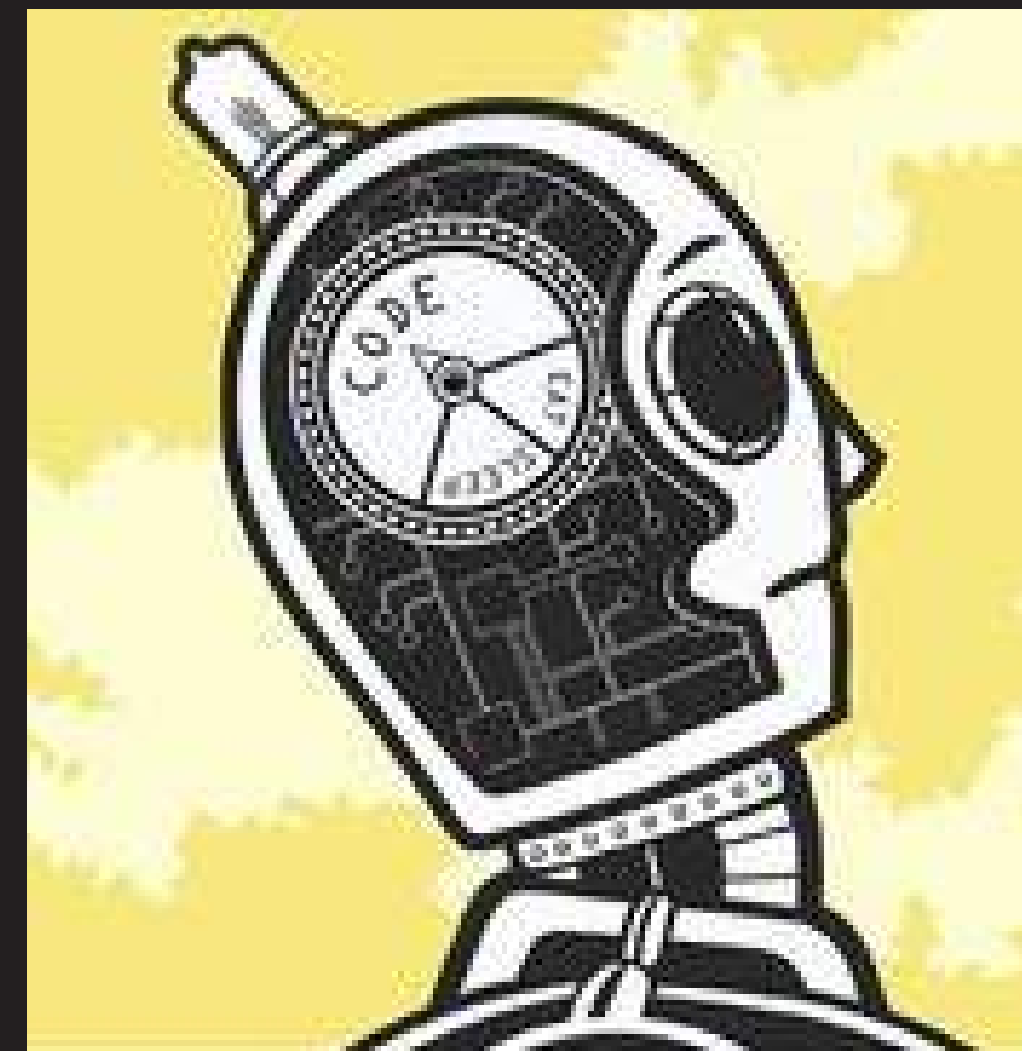
E tudo isso sempre começa com algoritmos.



Importância da lógica.

A lógica de programação, conhecida também como lógica computacional, é o elemento essencial para o processo de desenvolvimento de software. É ela quem organiza de maneira coerente a sequência de ações que um algoritmo precisa executar.

Lógica de programação é a técnica utilizada para desenvolver instruções em uma sequência para atingir determinado objetivo. É a organização e planejamento de instruções, em um algoritmo, com o objetivo de tornar viável a implementação de um programa ou software.



Algoritmo

Um algoritmo é uma sequência de instruções bem definidas, normalmente usadas para resolver problemas de matemática específicos, executar tarefas, ou para realizar cálculos e equações.

Quando se fala em algoritmo, muitas pessoas pensam rapidamente em computadores, tecnologia e até mesmo códigos difíceis de serem compreendidos. No entanto, o conceito e a aplicação são bem mais simples do que parecem.

Os algoritmos datam de tempos babilônicos, mas tornaram-se mais conhecidos na modernidade, principalmente, quando associados aos computadores e às estratégias de otimização para buscadores.



Lógica X Algoritmo

Os algoritmos são conjuntos de instruções que orientam a execução de tarefas específicas, enquanto a lógica de programação é a maneira como esses algoritmos são organizados para solucionar problemas e atingir um resultado desejado.



Você é programador todos os dias

A ação de sair de casa torna-se comum quando precisamos estudar ou trabalhar, com os horários e locais (dados) estabelecidos criamos uma rotina em cima dessas tarefas. Desta forma, inconscientemente, nós criamos algoritmos:

1. Acordar na hora estabelecida
2. escolher a roupa e se arrumar pra sair.
3. Verificar se pegou todos os itens que irá usar durante o dia.
4. Definir qual o meio de transporte.
5. Sair de casa
6. Trancar a porta
7. Ir até ponto de ônibus.
8. Esperar o ônibus chegar
9. Entrar no ônibus
10. Esperar chegar no local definido
11. Sair do ônibus
12. Se dirigir para o local da tarefa
13. Realizar as tarefas
14. Voltar para a casa



ATIVIDADE PRÁTICA

Crie um algoritmo para criação de um programa que instrua o usuário a fazer café.

Variáveis

Uma variável é um espaço na memória do computador (caixinha) destinado a um dado que é alterado durante a execução do algoritmo.

As variáveis são elementos básicos na programação, pois são aquelas que armazenam os dados que precisamos no nosso algoritmo. Cada dado pode ser de um tipo diferente e, por isso precisamos mostrar ao nosso algoritmo logo no início o que esperar armazenar.

Imagine, por exemplo, que precisamos separar roupas do vestuário de acordo com o seu tipo dentro de um armário. Esse é quem arruma a bagunça, por isso o consideramos nosso algoritmo, como vemos na ao lado.



Variáveis



Uma variável é um nome que definimos para armazenar dados de forma simples. O valor de uma variável pode ser alterado no andamento do algoritmo, por isso o nome de variável.

Construindo variáveis em Python

Assim como em outras linguagens, o Python pode manipular variáveis básicas como strings (palavras ou cadeias de caracteres), inteiros e reais (float). Para criá-las, basta utilizar um comando de atribuição (=), que define seu tipo e seu valor, conforme vemos no código abaixo:



```
1 msg = 'Exemplo de mensagem!' # String
2 p = 25 # Int
3 numero = 3.141592653589931 # Float
```

Nesse trecho foram feitas três atribuições. Linha 1 foi atribuída uma string para uma nova variável chamada msg. Linha 2 foi atribuído o valor inteiro 25 para p. Linha 3 foi atribuído um valor decimal para a variável numero.

Construindo variáveis em Python

msg = 'Exemplo de mensagem!'

↑ variável ↑ atribuição ↑ dado armazenado

numero = 3.141592653589931

↑ variável ↑ atribuição ↑ dado armazenado

Construindo variáveis em Python

Ao nomear variáveis em Python, é importante seguir algumas regras para garantir a legibilidade do código e evitar possíveis erros.

Siga essas dicas:

1. Utilize nomes descritivos: escolha nomes que sejam significativos e descrevam o propósito da variável. Evite utilizar nomes genéricos ou abreviações que possam dificultar a compreensão do código.
2. Utilize letras minúsculas: é recomendado utilizar apenas letras minúsculas ao nomear variáveis em Python. O uso de letras maiúsculas pode gerar confusões, pois Python é sensível a maiúsculas e minúsculas.
3. Utilize underscores para separar palavras: para melhorar a legibilidade do código, utilize underscores (_) para separar palavras em nomes de variáveis compostos. Por exemplo, "nome_completo" em vez de "nomecompleto".
4. Evite palavras reservadas: evite utilizar palavras reservadas da linguagem Python, como "if", "for", "while", entre outras, como nomes de variáveis.
5. Seja consistente: mantenha uma convenção de nomenclatura consistente ao longo do código. Escolha um padrão e siga-o em todas as suas variáveis.



Tipos de variáveis

Python é uma linguagem dinamicamente tipada, o que significa que não é necessário declarar o tipo de variável ou fazer casting (mudar o tipo de variável), pois o Interpretador se encarrega disso para nós! Isso significa também que o tipo da variável poder variar durante a execução do programa. Os tipos de dados padrão do Python são:



```
1  idade = 3 # Inteiro (int)
2  altura = 1.80 # Ponto Flutuante ou Decimal (float)
3  nome = "Pedro" # String (str)
4  solteiro = True # Boolean (bool)
```

Tipos de variáveis (int)

Tipo Inteiro (int)

O tipo inteiro é um tipo composto por caracteres numéricos (algarismos) inteiros.

É um tipo usado para um número que pode ser escrito sem um componente decimal, podendo ter ou não sinal, isto é: ser positivo ou negativo.

Por exemplo:



```
1  n = 21
2  n1 = 4
3  n2 = 0
4  n3 = -2048
```

Tipos de variáveis (float)

Tipo ponto flutuante (float)

É um tipo composto por caracteres numéricos (algarismo) decimais.

O famoso ponto flutuante é um tipo usado para números racionais (números que podem ser representados por uma fração) informalmente conhecido como “número quebrado”.



```
1  n = 3.15
2  n1 = 4.0
3  n2 = -3000.1
4  n3 = 1001.2
```

Tipos de variáveis (str)

Tipo string(str)

É um conjunto de caracteres dispostos numa determinada ordem, geralmente utilizada para representar palavras, frases ou textos.



```
1  n = "olá mundo"  
2  n1 = "Infinity School"  
3  n2 = "123456"  
4  n3 = "3.85 yeah"
```

Obs: Tudo que estiver dentro das aspas(string) é identificado como string.

Para o computador, isso não é número, é apenas um terminal que você apontou(apertou).

Tipos de variáveis (bool)

Tipo boolean(bool)

Tipo de dado lógico que pode assumir apenas dois valores:

falso ou verdadeiro (False ou True em Python).

Na lógica computacional, podem ser considerados como 0 ou 1.



```
1  casado = True
2  tem_filho = False
```


Primeiras funções internas (print)

A função `print()` é uma das funções mais usadas no Python, sendo utilizada para imprimir valores na tela.

A sintaxe da função é muito simples, basta escrever `"print()"` seguido dos valores que você deseja imprimir. Por exemplo:

```
valor = 123
```

```
print("Hello World")  
print("sapato")  
print("Céu Azul")  
print(valor)
```



Primeiras funções internas (print)

Execute a função no vs.Code e a mensagem será exibida no terminal.



```
1 valor = 123
2
3 print("Hello World")
4 print("sapato")
5 print("Céu Azul")
6 print(valor)
```

Terminal.




```
Hello World
sapato
Céu Azul
123
```


Primeiras funções internas (type)

Type() método retorna o tipo de classe do argumento (objeto) passado como parâmetro.
A função type() é usada principalmente para fins de depuração.

Neste exemplo, pedimos apenas para mostrar o tipo de dados que está armazenado em cada variável.



```
<class 'int'>  
<class 'str'>  
<class 'float'>  
<class 'bool'>
```



```
1  n1 = 15  
2  print(type(n1))  
3  
4  n2 = "Olá mundo"  
5  print(type(n2))  
6  
7  n3 = 3.93  
8  print(type(n3))  
9  
10 n4 = True  
11 print(type(n4))
```

Primeiras funções internas (input)

A função input permite que apresentemos um texto (ou prompt).

Quando a função é executada, o prompt é exibido. O usuário do programa pode digitar seu nome e pressionar return . Quando isso acontece, o texto digitado é retornado pela função input , e nesse caso, é associado à variável n .

```
n = input("Digite o seu nome: ")  
print(n)
```

```
n1 = int(input("Digite sua idade: "))  
print(n1)
```

```
n2 = float(input("Digite sua altura: "))  
print(n2)
```



Primeiras funções internas (input)

Execute no VsCode.



```
1 n = input("Digite o seu nome:")
2 print(n)
3
4 n1 = int(input("Digite sua idade: "))
5 print(n1)
6
7 n2 = float(input("Digite sua altura: "))
8 print(n2)
```

```
Digite o seu nome:Infinity School
Infinity School
Digite sua idade: 6
6
Digite sua altura: 1.70
1.7
```


F string

É basicamente uma ferramenta que temos para formatação de textos em Python.

As f-strings vão servir para que você consiga colocar uma variável dentro de um texto, e isso é feito utilizando a letra "f" antes do texto e colocando a sua variável dentro de {} chaves.

Isso é muito útil para que você não tenha que ficar concatenando o seu texto com as variáveis e tenha que fatiar seu texto várias vezes por conta disso.



```
1 nome = input("Informe seu nome")  
2 print(f"Olá {nome}, é um prazer te ver por aqui")
```

```
Informe seu nomeInfinity School
```

```
Olá Infinity School, é um prazer te ver por aqui
```

ATIVIDADE PRÁTICA

1. Faça um Programa que mostre a mensagem "olá mundo" na tela.
2. Faça um Programa que peça um número e então mostre a mensagem O número informado foi [número].
3. Faça um programa que peça a idade, a altura, o cpf e o nome do usuário e imprima na tela a mensagem:
Bem-vindo (usuário), seus dados foram cadastrados com sucesso. Idade = "", altura = "", cpf = "".

Operadores aritméticos

Os operadores aritméticos são usados para realizar operações matemáticas em Python. Existem seis operadores aritméticos em Python:

- + (adição): soma dois valores.
- - (subtração): subtrai o segundo valor do primeiro valor.
- * (multiplicação): multiplica dois valores.
- / (divisão): divide o primeiro valor pelo segundo valor.
- % (módulo): retorna o resto da divisão do primeiro valor pelo segundo valor.
- ** (exponenciação): eleva o primeiro valor à potência do segundo valor.
- // (divisão inteira): divide o primeiro valor pelo segundo valor ignorando a parte decimal.

Operadores aritméticos

Os operadores aritméticos têm uma ordem de precedência específica em Python, o que significa que eles são executados em uma ordem específica. A ordem de precedência dos operadores aritméticos é a seguinte:

****** (exponenciação)

***, /, %** (multiplicação, divisão, módulo)

+, - (adição, subtração)

Você pode usar parênteses para alterar a ordem de precedência e forçar que uma operação seja executada antes de outra. Por exemplo, a expressão $(2 + 3) * 4$ é avaliada primeiro a adição dentro dos parênteses e, em seguida, a multiplicação fora dos parênteses.

+	soma	$3 + 2 = 5$
-	subtração	$3 - 2 = 1$
*	multiplicação	$3 * 2 = 6$
/	divisão	$3 / 2 = 1.5$
%	módulo	$3 \% 2 = 1$
**	exponenciação	$3 ** 2 = 9$
//	floor division	$3 // 2 = 1$

ATIVIDADE PRÁTICA

1. Faça um Programa que peça dois números e imprima a soma.
2. Faça um Programa que peça as 4 notas bimestrais e mostre a média.
3. Faça um Programa que converta metros para centímetros.
4. Faça um Programa que calcule a área de um quadrado, em seguida mostre o dobro desta área para o usuário.
5. Faça um Programa que pergunte quanto você ganha por hora e o número de horas trabalhadas no mês. Calcule e mostre o total do seu salário no referido mês.

SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 02 DE LÓGICA DA PROGRAMAÇÃO.
ESTRUTURA CONDICIONAL (IF, ELSE E ELIF)

IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

Estrutura condicional

A estrutura condicional é uma seção que ajuda a definir condições para a execução de determinados blocos de comandos. Em vez de executar tudo de vez, sem nenhuma interrupção, o programa deve parar para executar um teste e decidir qual caminho seguir.

É como uma bifurcação, um momento importante para tomada de decisão. Nesse caso, a decisão é o resultado de uma análise, de uma comparação. Funciona assim: Se essa condição X é satisfeita, então execute esse bloco de comandos; senão, execute esse outro bloco de comandos.

Parte-se do pressuposto que o “senão” traz um bloco diferente, em contraponto com o bloco do “se”.

Assim, você pode direcionar o fluxo de execução para pular blocos e executar exatamente o que se precisa.

Estrutura condicional

```
1  x = 2
2  y = 5
3  z = 0
4  resultado = 0
5  valor = int(input('Digite 1, 2 ou 3: '))
6  if (valor == 1):
7      resultado = x * valor
8      valor = 2
9
10 if (valor == 2):
11     resultado += y
12     valor = 3
13
14 if (valor == 3):
15     resultado += z
16
17 print(resultado)
```



INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 01 – LÓGICA DE PROGRAMAÇÃO