Revisão de Python

Prof. Saulo Oliveira
Técnico em Informática para Internet
Instituto Federal do Ceará

Introdução

O algoritmo é uma sequência de passos lógicos e finitos que permite solucionar problemas;

- O objetivo de aprender a criar algoritmos é que este é a base de conhecimentos para as linguagens de programação;
- Em geral, existem muitas maneiras de resolver o mesmo problema. Ou seja, podem ser criados vários algoritmos diferentes para resolver o mesmo problema;
- Assim, ao criarmos um algoritmo, indicamos uma dentre várias possíveis sequências de passos para solucionar o problema.

Propriedades essenciais

Completo	Sem redundância	Determinístico	Finito
Todas as ações precisam ser descritas e devem ser únicas.	Um conjunto de instruções só pode ter uma única forma de ser interpretada.	Se as instruçõess forem executadas, o resultado esperado será sempre atingido.	As instruções precisam terminar após um número limitado de passos.

Instruções e Tipos de dados

As informações manipuladas pelo computador podem ser classificadas em:

Instruções	Dados
Coordenam o funcionamento do computador, determinando a maneira como os dados devem ser tratados.	São as informações a serem processadas pelo computador.

Em Python, os dados podem ser dos tipos: numéricos (`int` e `float`), lógicos (`bool`), literais (`string`), listas (`list`) e dicionários (`dict`). Há outros, mas deixemos para depois!

Exercício

Classifique os dados abaixo de acordo com seu tipo, assinalando com `I` os dados do tipo inteiro, com `R` os reais, com `L` os literais, com `B` os lógicos (booleanos), e com `N` aqueles para os quais não é possível definir a priori um tipo de dado.

	0.21		"josé"		"+3257"		True
	1		0,35		+3257.		False.
	V		TRUE		"-0.0"		"abc"
	"0"		+3257		"False"		+36
	1%		'a'		True		± 3

Declaração de variáveis

Em Python é necessário apenas o nome da variável, seguido do símbolo = e o valor que ela irá armazenar. O tipo da variável será o mesmo tipo de dado que ela armazena.

```
idade = 32
preco = 100.21
teste = True
nome = "SAULO"
```

Regras de nomenclatura (o nome das variáveis):

- Podem ter dígitos, letras maiúsculas e minúsculas, e underscore (_);
- Não pode ser iniciado por dígito e não são permitidos espaços em branco e nem podem conter caracteres especiais (@, $, +, -, %, !, /, ?, #);
- Nem palavras reservadas (`keywords`).

Saída de dados

> Usaremos a função `print`.

- A função `print` mostra em formato texto para o usuário o conteúdo de um variável;
- Também pode mostrar strings ou combinações de strings e variáveis, bastando separar por (,) vírgula.
- Podemos usar o print com valores dos tipos primitivos;
- Podemos executar expressões e só o valor do resultado vai ser impresso.

```
idade = 32
nome = 'Saulo'
print('Meu primeiro programa')
print(f'Meu nome é {nome} e minha idade é:', idade)
print('Faltam', 65 - idade, 'anos para eu me aposentar!')
```

Entrada de dados

> Usaremos a função `input`.

- A função `input` requer um texto que será mostrado para o usuário e retorna o que o usuário digitou (sempre do tipo literal `str`, precisamos converter depois, se necessário).
- Guardamos o valor retornado pela função `input` em uma variável.

```
nome = input('Digite seu nome: ')
print(f'Seja muito bem-vindo(a), {nome}!')

idade = int(input('Digite sua idade: '))
print(f'Legal, que você tem {idade} anos!')

altura = float(input('Digite sua altura: '))
print(f'E com altura de {altura} metros.')
```

Entrada de dados descasada

```
idade = input('Me diga novamente quantos anos tem:')
proxima_idade = idade + 1
print(f'Ano que vem, você terá {proxima_idade} anos')
```

QUAL O PROBLEMA COM O CÓDIGO ACIMA?

Conversão de tipos

A conversão de tipos é o ato de forçar uma expressão a utilizar e retornar um determinado tipo. Podemos ter dois tipos de conversões de tipos, pode ser implícita ou explicitas, que são conversões especificadas.

> Para saber o tipo de dado de uma expressão ou variável, basta usarmos a função `type`.

Exercício

Declaração	Tipo da variável	Conversão	Resultado	Tipo do resultado
`x = "42"`		`int(x)`		
`n = 123`		`str(n)`		
`pi = 3.14`		`int(pi)`		
`saldo = 567`		`float(saldo)`		
`tem_pix = True`		`str(tem_pix)`		
`poupanca = 0.0`		`bool(poupanca)`		
`nome = 'Saulo'`		`bool(nome)`		
`cpf = ''`		`bool(cpf)`		
`faltei = False`		`int(faltei)`		
`merenda = True`		`float(merenda)`		

Exercício prático

1. Faça um algoritmo para converter uma temperatura dada em Fahrenheit para Celsius.
2. Faça um algoritmo que receba duas notas e seus respectivos pesos, calcule e mostre a média ponderada.
3. Faça um algoritmo que receba um valor referente a uma compra em dólar no cartão de crédito, calcule e mostre o valor de conversão para real. Além disso, sabendo que em compras internacionais incide-se O IOF sobre o total, adicione o valor da taxa (valor de 6,38%). Ademais, adote o valor do dólar R$ 5,00.

Operadores e Expressões - I

Operadores aritméticos.

> São operadores matemáticos básicos que envolvem números e são utilizados para realizar cálculos e manipular quantidades numéricas.

Operador	Uso	Comentário
`+`	`x + y`	Soma o conteúdo de `x` e de `y`.
`-`	`x - y`	Subtrai o conteúdo de `y` do conteúdo de `x`.
`*`	`x * y`	Multiplica o conteúdo de `x` por pelo conteúdo de `y`.
`/`	`x / y`	Divide o conteúdo de `x` pelo conteúdo de `y`.
`%`	`x % y`	Obtém o resto da divisão de `x` por `y`.
`//`	`x // y`	Obtém o quociente inteiro da divisão de `x` por `y`.
`**`	`x ** y`	Eleva o conteúdo de `x` à potência do conteúdo de `y`.

Operadores e Expressões - II

Operadores relacionais.

> São operadores que comparam valores para determinar relações como igualdade, desigualdade, maior ou menor, retornando um valor lógico (`True` ou `False`).

Operador	Uso	Comentário
`==`	`x == y`	O conteúdo de `x` é igual ao conteúdo de `y`.
`!=`	`x != y`	O conteúdo de `x` é diferente do conteúdo de `y`.
`<=`	`x <= y`	O conteúdo de `x` é menor ou igual ao conteúdo de `y`.
`>=`	`x >= y`	O conteúdo de `x` é maior ou igual ao conteúdo de `y`.
`<`	`x < y`	O conteúdo de `x` é menor que o conteúdo de `y`.
`>`	`x > y`	O conteúdo de `x` é maior que o conteúdo de `y`.

Operadores e Expressões - III

Operadores Lógicos.

> Operadores lógicos permitem a realização de operações lógicas sobre valores booleanos. Esses operadores geralmente são utilizados para tomar decisões condicionais e controlar o fluxo de execução de um programa.

Operador	Uso	Comentário
`not`	`not x`	Equivale a modificar o conteúdo de `x` pela negação.
`and`	`x and y`	Retorna `True` se e somente se `x` e `y` forem `True`. Caso contrário, retorna `False`.
`or`	`x or y`	Retorna `False` se e somente se `x` e `y` forem `False`. Caso contrário, retorna `True`.

Precedência

A precedência determina a ordem em que os operadores são avaliados em uma expressão, quando ela envolve múltiplos operadores. É possível também alterar a ordem de avaliação usando parênteses, similar à Matemática.

Operadores	Descrição
**	Exponenciação.
+, -	Operadores unários (modificam o sinal).
*, /, %, //	Multiplicação, divisão, resto da divisão e divisão inteira.
+, -	Adição e subtração.
<=, <, >, >=	Operadores de comparação.
==, !=	Operadores de igualdade.
not, and, or	Operadores lógicos.

Exercícios

Dada a seguinte expressão:

```
2 - 2 ** 3 * 2
```

1. Adicione um conjunto de parênteses para que a expressão seja avaliada como -12.
2. Agora mova seus parênteses para que a expressão seja avaliada como -62.
3. Mova seus parênteses uma última vez para que a expressão seja avaliada como 0.

Exercícios

Dada a seguinte expressão:

```
2 / 3 * 4 ** 2
```

- Adicione dois conjuntos de parênteses que deixam o valor da expressão inalterado. Os parênteses não podem incluir a expressão inteira nem um único número.

QUANTO VALE A EXPRESSÃO ABAIXO?

```
-2 ** 2
```

Exercícios

Analise o programa abaixo e, para cada uma das saídas (comandos `print`), detalhe passo a passo como o Python (segundo suas prioridades) resolveria as equações e o resultado final obtido.

```
x = 2
y = 3
z = 0.5

print(x + x * x ** (y * x) / z)

print(not x + z < y or x + x * z >= y and True)

print(x + y == z)
```

Exercícios

Indique o resultado das expressões mostrando passo a passo a ordem de avaliação, sendo: `x = 6.0`, `y = 2`, `z = 4.0`, `a = 8`, `b = 7.5`, `c = 12`. Indique quando ela não puder ser realizada e informe por qual motivo.

a) `x - y * (a + 1) == z * -c`

b) `x - y * a > c % y`

c) `c % y <= y % c`

d) `(b * 4) >= (a + a * 2) and a >= c ** - 2`

e) `a + 3 > -b + -c`

f) `b + a > c + c and a != c < b != a`

g) `a // c < (b % 2) or (c ** b * 3) < a * 3`

Condicionais

- Condicionais em lógica de programação são estruturas que permitem a execução de diferentes blocos de código dependendo da avaliação de uma condição específica.
- Essas estruturas são fundamentais para controlar o fluxo de execução de um programa, permitindo que partes específicas do código sejam executadas ou ignoradas com base em condições lógicas;
- Eles avaliam uma expressão booleana e, com base no resultado (verdadeiro ou falso), direcionam o programa para diferentes caminhos de execução.

Condicional simples

> Permite a execução condicional de um bloco de código.

Avalia-se uma expressão booleana e, se a condição for verdadeira (`True`), o bloco de código indentado após o `if` é executado; caso contrário, o bloco é ignorado.

```
idade = 20

if idade > 18:
    print('Você é maior de idade')

saldo = 10

if saldo > 0:
    print('Você está liso!')
```

Referências

- Rafael Guimarães Sakurai. **Construtor**, 2020. Disponível em: http://www.universidadejava.com.br/java/java-construtor/. Acessado em 12 de set. de 2023.
- Tatiane Vieira. **O que são os padrões HEX, RGB e HSL de cores?** 2021, Disponível em: https://tecnoblog.net/responde/o-que-sao-os-padroes-hex-rgb-e-hsl-de-cores/. Acessado em 12 de set. de 2023.
- https://www.javaprogressivo.net/2012/09/O-que-sao-Construtores-em-Java-Como-Criar-e-Usar.html