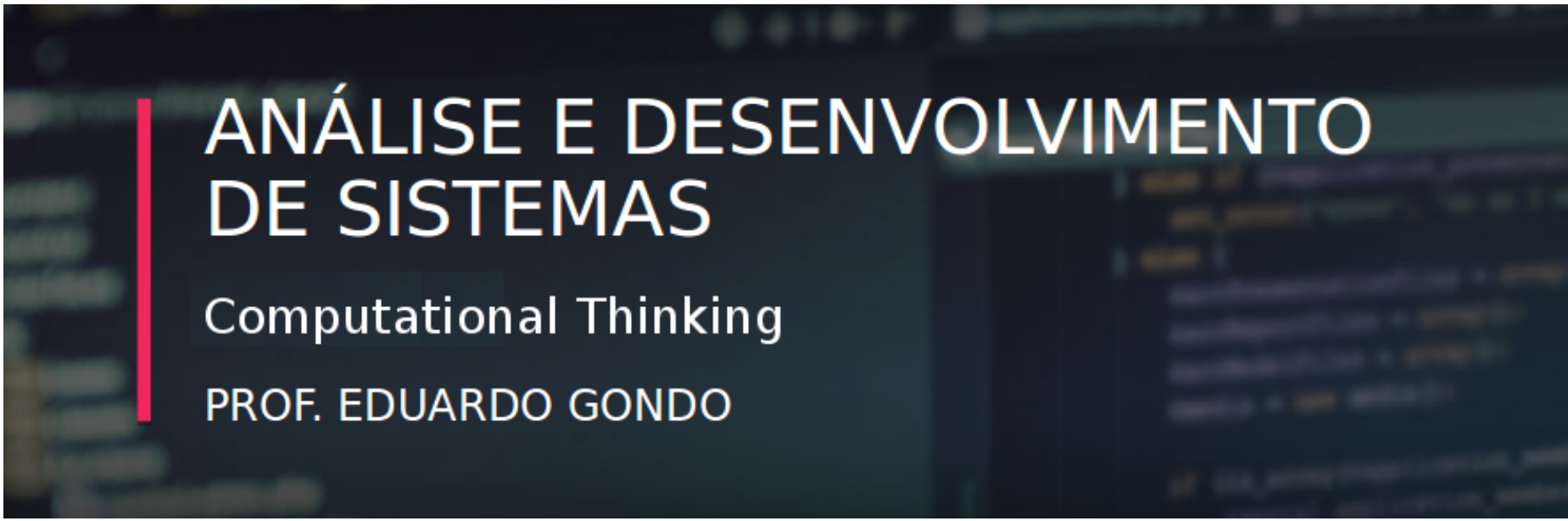
FIAP GRADUAÇÃO


ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
Computational Thinking
PROF. EDUARDO GONDO

Agenda

- introdução aos algoritmos
- lógica de programação
- conceito de variáveis
- entrada e saída
- operadores aritméticos
- tipos de dados

Introdução — Algoritmos

Algorimo

Como já dissemos, um algoritmo é uma sequência de ações finitas cuja execução resolve um problema ou realiza uma ação.

Diariamente, executamos diversas ações ou resolvemos vários problemas utilizando algoritmos. Por exemplo: escovar os dentes, dirigir um carro, tomar banho, ir para o trabalho, etc.

Desde o ensino fundamental também aprendemos diversos algoritmos em sala de aula: efetuar uma divisão, calcular o *mdc* entre 2 números inteiros, verificar se um número é primo entre outros.

Introdução — Algoritmos

Nesta disciplina, vamos construir soluções para problemas que podem ser executadas no computador. Essas soluções, ou melhor algoritmos, serão escritos usando o que chamamos de **linguagem de programação**.

Recapitulando

A linguagem de programação é uma linguagem escrita e formal que especifica um conjunto de instruções e regras usadas para gerar programas (software).

A linguagem de programação possui várias regras e construções semânticas que serão exemplificadas nesta disciplina.

Por quê usar uma Linguagem de programação

- para podermos usar o computador para executar nossos algoritmos
- executando no computador temos garantia que ele executará corretamente e rapidamente
- para isso, devemos usar algo que tanto nós como o computador entenda, ou seja a linguagem de programação
- damos o nome de **código-fonte** para o algoritmo escrito em uma linguagem de programação
- em algumas linguagens o código-fonte é interpretado diretamente pelo "computador"em outras o código-fonte é compilado para gerar um código binário que será executado dentro do computador

Conceitos e vocabulário

Vamos explorar alguns conceitos antes de prosseguirmos

- um **programa de computador** é um algoritmo escrito usando uma linguagem de programação já na forma de código de máquina
- o processo de transformação de um código-fonte em código de máquina é chamado de **compilação**
- o **teste de mesa** consiste em VOCÊ fazer o papel do computador e executar seu programa
- ele é muito importante para corrigirmos erros de lógica e certificarmos que o algoritmo está funcionando corretamente
- nessas primeiras aulas faremos uso de lápis e papel para resolver nossos problemas
- e depois transcreveremos o algoritmo para a linguagem Python

Pensando em soluções algorítmicas

Basicamente um programa de computador ou algoritmo pode ser representado pela figura abaixo:

Figura: Esquema de um programa

Para desenvolvermos um programa de computador é preciso determinar:

entrada de dados informações fornecidas para o programa pelo usuário

processamento as operações que são efetuadas sobre a entrada de dados para obter a resposta do problema

saída resposta fornecida para o usuário do programa

Passos para desenvolver um algoritmo

1) certifique-se que você entendeu bem o enunciado do problema
2) **resolva o problema usando papel e lápis** e use valores reais na sua solução
3) ainda no papel, tente **descrever** uma solução do problema:
4) teste sua descrição para outros valores reais e veja se a solução está correta
5) tente substituir os valores por incógnitas (como na matemática)
6) no computador, transcreva seu algoritmo em uma linguagem de programação
7) faça mais testes e, se necessário e possível, corrija os erros
8) caso você não consiga corrigir os erros, volte ao item 2

Atividades 1

Dada uma sequência de números inteiros onde o último termo dessa sequência é o número 0, calcule a soma de todos os números da sequência.

A ideia nesta atividade e nas próximas é tentar resolver o problema no papel. Será muito importante ser o mais detalhista possível, principalmente com relação às operações matemáticas que serão feitas.

Atividades 2

Dados um preço de um produto e o percentual de desconto, calcule o novo preço do produto.

Atividades 3

Dada uma sequência de 10 números inteiros, encontre aquele que é o maior número da sequência.

Atividades 4

Considere a equação de segundo grau $x^2 + 3x - 10 = 0$, encontre as raízes da equação

Entrada de Dados e variáveis

- já vimos no diagrama que a **entrada de dados** é a parte inicial do desenvolvimento dos nossos algoritmos
- somente com esses dados que conseguimos fazer as operações para resolver o problema
- para as atividades anteriores, note que, anotamos o(s) número(s) no papel para lembrarmos deles e isso não é diferente no computador
- a *memória RAM* é o dispositivo responsável para essa finalidade
- para ter acesso à memória RAM dentro do nosso código-fonte (ou programa) usamos as **variáveis**

Variáveis

- as **variáveis** são elementos responsáveis por armazenar informações dentro do programa
- elas podem ser entendidas como partes da memória RAM que são alocadas pelo seu programa
- elas podem armazenar vários **tipos de informação**: números inteiros, números reais, palavras (sequência de caracteres), valores booleanos (verdadeiro ou falso), etc
- normalmente as informações da entrada de dados dos problemas são armazenadas em variáveis
- em Python não necessitamos especificar o tipo para as variáveis, o tipo é inferido automaticamente

Variáveis (continuação)

Como criamos as variáveis dentro do nosso algoritmo em Python?

- as variáveis são declaradas dentro do código-fonte e elas são referenciadas através de um nome
- esse nome pode iniciar com uma letra qualquer ou o caracter underline
- o restante do nome pode ser composto por letras, números e, se desejar, mais _ (underline)
- use nomes significativos para nomear suas variáveis, por exemplo, suponha que você deseja armazenar um preço de um produto
- apesar de ser permitido você usar como nome da variavel o `p`, talvez você obtenha maior valor semântico usando `preco`

Exemplo de Problema

PROBLEMA 2.1: Dados uma sequência de 5 números inteiros. Calcule a soma de todos os números da sequência.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?

Exemplo de Problema

PROBLEMA 2.1: Dados uma sequência de 5 números inteiros. Calcule a soma de todos os números da sequência.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Vamos resolver usando papel e lápis, usando os seguintes valores: 3.14, 5, 10.9, 7 e 8

Exemplo de Problema

PROBLEMA 2.1: Dados uma sequência de 5 números inteiros. Calcule a soma de todos os números da sequência.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Vamos resolver usando papel e lápis, usando os seguintes valores: 3.14, 5, 10.9, 7 e 8
- Por que não podemos usar esses valores?

Exemplo de Problema

PROBLEMA 2.1: Dados uma sequência de 5 números inteiros. Calcule a soma de todos os números da sequência.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Vamos resolver usando papel e lápis, usando os seguintes valores: 3.14, 5, 10.9, 7 e 8
- Por que não podemos usar esses valores?
- E essa agora: 7, 16, 37, 28 e 19?

Exemplo de Problema

PROBLEMA 2.1: Dados uma sequência de 5 números inteiros. Calcule a soma de todos os números da sequência.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Vamos resolver usando papel e lápis, usando os seguintes valores: 3.14, 5, 10.9, 7 e 8
- Por que não podemos usar esses valores?
- E essa agora: 7, 16, 37, 28 e 19?
- Qual a resposta do problema?

Exemplo de Problema

PROBLEMA 2.1: Dados uma sequência de 5 números inteiros. Calcule a soma de todos os números da sequência.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Vamos resolver usando papel e lápis, usando os seguintes valores: 3.14, 5, 10.9, 7 e 8
- Por que não podemos usar esses valores?
- E essa agora: 7, 16, 37, 28 e 19?
- Qual a resposta do problema?
- **107**

Exemplo de Problema (cont)

- descrevendo o algoritmo no papel:

```
peça para o usuário informar cada um dos 5 números
vamos nomear cada um dos números informados como:
     numA, numB, numC, numD, numE
some os 5 números (numA + numB + numC + numD + numE)
              e guardar o resultado em soma
mostrar o valor da variável soma
```

Solução do Problema 2.1 - Python

```
 1  valor = input("Digite um numero: ")
 2  numA = int(valor)
 3  valor = input("Digite um numero: ")
 4  numB = int(valor)
 5  valor = input("Digite um numero: ")
 6  numC = int(valor)
 7  valor = input("Digite um numero: ")
 8  numD = int(valor)
 9  valor = input("Digite um numero: ")
10  numE = int(valor)
11
12  soma = numA + numB + numC + numD + numE
13
14  print(soma)
```

Analisando o Algoritmo

```
valor = input("Digite um numero: ")
numA = int(valor)
.
.
.
.
valor = input("Digite um numero: ")
numE = int(valor)

soma = numA + numB + numC + numD + numE

print(soma)
```

- Vamos analisar a primeira linha do programa
  - `input` coleta a informação através do teclado
  - essa informação é do tipo String
  - o operador = atribui a informação a variável `valor` do tipo String
- Na segunda linha, convertemos o conteúdo da variável `valor` para um número inteiro
- Na penúltima linha, somamos todos os valores inteiros e atribuímos na variável `soma`
- E na última linha, imprimimos o resultado na tela

Mais um pouco do Problema 2.1

- e se fossem mais números? teríamos que armazenar mais números e portanto criar mais variáveis

Mais um pouco do Problema 2.1

- e se fossem mais números? teríamos que armazenar mais números e portanto criar mais variáveis
- seria possível um algoritmo que usa menos variáveis não independente da quantidade de números?

Mais um pouco do Problema 2.1

- e se fossem mais números? teríamos que armazenar mais números e portanto criar mais variáveis
- seria possível um algoritmo que usa menos variáveis não independente da quantidade de números?
- usando papel e caneta, tente fazer a soma para a seguinte sequência de números: 37, -64, 75, -56 e -43

Mais um pouco do Problema 2.1

- e se fossem mais números? teríamos que armazenar mais números e portanto criar mais variáveis
- seria possível um algoritmo que usa menos variáveis não independente da quantidade de números?
- usando papel e caneta, tente fazer a soma para a seguinte sequência de números: 37, −64, 75, −56 e −43
- você conseguiu somar todos de uma vez?

Mais um pouco do Problema 2.1

- e se fossem mais números? teríamos que armazenar mais números e portanto criar mais variáveis
- seria possível um algoritmo que usa menos variáveis não independente da quantidade de números?
- usando papel e caneta, tente fazer a soma para a seguinte sequência de números: 37, -64, 75, -56 e -43
- você conseguiu somar todos de uma vez?
- pense em um caixa de supermercado, como é feita a soma total da compra?

Mais um pouco do Problema 2.1

- note que, se somarmos os números conforme eles vão sendo fornecidos, podemos reaproveitar a mesma variável várias vezes
- assim, nessa solução vamos precisar apenas de duas variáveis: **soma** e **num**

Solução do Problema 2.1 - Versão 2

```
1  soma = 0
2  valor = input("Digite um numero: ")
3  num = int(valor)
4  soma = soma + num
5
6  valor = input("Digite um numero: ")
7  num = int(valor)
8  soma = soma + num
9
10 ...
11
12 valor = input("Digite um numero: ")
13 num = int(valor)
14 soma = soma + num
15
16 valor = input("Digite um numero: ")
17 num = int(valor)
18 soma = soma + num
19 print(soma)
```

Figura: Segunda versão da solução em Python

Analisando o algoritmo em Python

- além da questão das variáveis o algoritmo fornece outros elementos que podemos analisar
- na linha 2, temos a instrução `input("Digite ...")` que lê uma informação através do teclado armazenando ela em `valor`
- essa instrução permite realizar a entrada do algoritmo/programa
- note que, o comando `input` devolve uma String (texto) digitado pelo usuário
- na linha 4, a instrução `soma = soma + num` é a responsável por acumular os valores
- essa instrução é muito utilizada nos algoritmos e deve ser interpretada da direita para a esquerda, ou seja,
- primeiro faço a adição do valor da variável `soma` com a variável `num` e depois **atualizo** o resultado na variável `soma`

Entrada e saída

- também conhecido como *input* e *output*
- parte do algoritmo/programa responsável pela comunicação com o usuário
- nos algoritmos que estudaremos usaremos como entrada o teclado e a saída o monitor

tipo	Python
output	`print(...)`
input	`variavel = input("texto")`

Tabela: Resumo dos comandos de entrada e saída

- a instrução `input` sempre retorna um texto (String), mesmo quando a pessoa digita um número.
- você precisa converter esse texto para número quando for necessário

Conversores

- usaremos funções (instruções) do Python para converter texto em números e números em texto (String)
- `int(<texto>)`: converte o `<texto>` em um número inteiro se possível
- `int(<num>)`: converte o `<num>` em um número inteiro
- `float(<texto>)`: converte o `<texto>` em um número real se possível
- `float(<num>)`: converte o `<num>` em um número real
- `str(<numero>)`: converte o `<numero>` em um texto (String)

Exemplo de Problema

PROBLEMA 2.2: Dados a base e a altura de um triângulo, calcule sua área.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?

Exemplo de Problema

PROBLEMA 2.2: Dados a base e a altura de um triângulo, calcule sua área.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Quais são os dados de entrada?

Exemplo de Problema

PROBLEMA 2.2: Dados a base e a altura de um triângulo, calcule sua área.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Quais são os dados de entrada?
- Por exemplo, a entrada poderia ser: 6.7 e 5.5?

Exemplo de Problema

PROBLEMA 2.2: Dados a base e a altura de um triângulo, calcule sua área.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Quais são os dados de entrada?
- Por exemplo, a entrada poderia ser: 6.7 e 5.5?
- Os números acima são números de que tipo?

Exemplo de Problema

PROBLEMA 2.2: Dados a base e a altura de um triângulo, calcule sua área.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Quais são os dados de entrada?
- Por exemplo, a entrada poderia ser: 6.7 e 5.5?
- Os números acima são números de que tipo?
- Como calculamos a área do triângulo?

Exemplo de Problema

PROBLEMA 2.2: Dados a base e a altura de um triângulo, calcule sua área.

Vamos tentar aplicar nosso roteiro para desenvolver um algoritmo para resolver o problema:

- Está claro o que o problema pede?
- Quais são os dados de entrada?
- Por exemplo, a entrada poderia ser: 6.7 e 5.5?
- Os números acima são números de que tipo?
- Como calculamos a área do triângulo?
- Qual a resposta do problema?

Exemplo de Problema (cont)

- descrevendo o algoritmo no papel:

```
ler 2 números reais: base e altura
obter área do triângulo: area = base * altura / 2
imprimir area na tela
```

Solução do Problema 2.2 - Python

```
1  auxiliar = input("Digite a base: ")
2  base = float(auxiliar)
3
4  auxiliar = input("Digite a altura:")
5  altura = float(auxiliar)
6
7  area = base * altura / 2
8
9  print("A area vale " + str(area))
```

Figura: Solução do problema 2 em Python

Operadores aritméticos

Segue a lista dos operadores aritméticos no Python:

operador	Python
soma	+
subtração	-
multiplicação	*
divisão real	/
potência	**
resto da divisão	%
divisão inteira	//

Tabela: operadores aritméticos

Operadores em String

No código anterior, usamos a seguinte instrução:

```
1  print("A area vale " + str(area))
```

- operador +, quando os argumentos são Strings, faz a concatenação delas
- também temos o operador *, cujos argumentos são uma String e um número natural *n*
- esse operador "n-plica"a String, execute o código abaixo:

```
1  nome = " FIAP "
2  print(nome * 4)
```

Mais um pouco sobre o print

- podemos passar mais de uma informação para o comando print
- basta separmos através de vírgulas as informações
- veja o código abaixo

```
1  idade = 34
2  nome = "Eduardo"
3  print("Meu nome e idade", nome, idade)
```

- teste o código no interpretador de comandos
- observe que o python já adiciona 1 espaço ao final de cada valor impresso

Declarando variáveis

Já mencionamos que as variáveis podem armazenar vários **tipos de dado**: números inteiros, números reais, palavras (sequência de caracteres), valores booleanos (True ou False)
Na linguagem Python, podemos declarar as variáveis da seguinte forma:

```
1   #número inteiro
2   num = 54
3
4   #número real
5   porcentagem = 0.25
6
7   #texto (String)
8   nome = "Computational Thinking"
9
10  #Bool
11  mudou = False
12
13  #atribuição múltipla
14  x = y = z = 1
```

Declarando variáveis

Sempre que você declarar uma variável em Python devemos inicializá-las, pois só assim, o interpretador conseguirá inferir o tipo da variável que você está criando.
Podemos criar variáveis através do valor retornado por algumas instruções:

```
1  endereco = input("Onde você mora?")
2
3  idade = int("18")
4
5  idade_texto = str(idade)
```

Variáveis - dicas

Variáveis

Use nomes relacionados ao problema que você está resolvendo. Não economize na quantidade de letras.

- a linguagem de programação é lida pelo computador e por seres humanos, escreva seu programa de maneira legível
- caso você precise saber qual o tipo da variável, use a instrução `type(<var>)`, onde `<var>` é uma variável do seu programa

Exercícios

Sua tarefa é fazer os exercícios propostos!

E lembre-se:

Importante!

Se eu não sei resolver um problema usando lápis e papel é muito improvável que eu consiga desenvolver um programa de computador para resolvê-lo

Referência Bibliográfica

- Puga e Rissetti - Lógica de Programação e Estrutura de Dados
- Ascêncio e Campos - Fundamentos da Programação de Computadores
- Forbelone e Eberspacher - Lógica de programação: a construção de algoritmos e estruturas de dados
- Documentação do Python - https://docs.python.org/3.8/
- Python Programming For Beginners: Learn The Basics Of Python Programming (Python Crash Course, Programming for Dummies) (English Edition). Kindle
- Python: 3 Manuscripts in 1 book: - Python Programming For Beginners - Python Programming For Intermediates - Python Programming for Advanced (English Edition). Kindle

Copyleft