


# Pensamento Computacional



2022/23

Aula 2

---

# Objetivos

- Conceitos Básicos de Algoritmos:
    - Estruturas Condicionais e de Repetição;
    - Testes de algoritmos;
  - Linguagens de Programação:
    - Introdução ao Python
-

# Introdução

- Considere o problema de cálculo da média aritmética das notas de dois testes de um estudante.

**Algoritmo** Média

**Var**

num1, num2, média: real

**Início**

Leia(num1, num2);

média  $\leftarrow$  (num1+num2) / 2;

Escreva(média);

**Fim**

# Introdução

- Como alteramos este algoritmo para calcular a média de 2 alunos?
  - E de um número indeterminado de alunos?
-

# Estruturas de Repetição

- Permitem a repetição de um bloco de instruções um número pré-determinado de vezes, ou até à satisfação de uma dada condição.
  - Também chamados de **ciclos** ou *loops*.
  - Tipos de estruturas:
    - Repetição contada (estrutura para)
    - Repetição condicional
      - Com teste no início (estrutura enquanto)
      - Com teste no final (estrutura repita)
-

# Estruturas de Repetição

- **Ciclos sem fim** ou **deadlock** – quando a estrutura de repetição executa interminavelmente!
  - **As estruturas de repetição devem ter sempre um fim**, ou seja, devem ter um limite pré-definido de repetições ou alguma condição que determine o fim da repetição.
- É umas das causas comuns de bloqueio ou impasse no software (“*Este programa não está a responder!*”)



# Estrutura de Repetição CONTADA

- Utilizada quando se conhece previamente a quantidade de vezes que queremos executar o bloco de comandos.
    - *A quantidade de vezes pode ser um valor constante ou a definir pelo utilizador.*
  - É representada pelo comando **Para**
-

# Estrutura de Repetição CONTADA

## Sintaxe

```
Para <variável> de <valor inicial> até <valor final>  
    [passo <incremento>] faça
```

```
    <bloco de comandos>
```

```
FimPara
```

Opcional.

- Valor incrementado à variável em cada repetição.
- Se não for informado, é assumido o incremento 1.



# Estrutura de Repetição CONTADA

*Exercício: Listar os primeiros dez números inteiros positivos*

```
Algoritmo DezPrimeiros  
Variáveis  
    i: inteiro  
Início  
    Para i de 1 até 10 faça  
        Escreva (i)  
    FimPara  
Fim
```

# Estrutura de Repetição CONTADA

*Exercício: Algoritmo que escreva os números inteiros pares menores ou iguais do que um valor determinado pelo utilizador.*

**Algoritmo** NúmerosPares

**Variáveis**

i, limite: inteiro

**Início**

Escreva("Listar números pares até:")

Leia(limite)

**Para** i **de** 2 **até** limite **passo** 2 **faça**

    Escreva(i)

**FimPara**

**Fim**

# Estrutura de Repetição **CONDICIONAL**

- São usadas quando não sabemos à partida o número de vezes que queremos executar o bloco de comandos;
    - A quantidade de vezes depende de uma condição (**expressão lógica**) controlada internamente.
  - **Enquanto ... faça** – teste no início;
  - **Repita ... até** – teste no final
-

# Estrutura Enquanto ... Faça

## Sintaxe

```
Enquanto <expressão lógica> faça
```

```
    <bloco de comandos>
```

```
FimEnquanto
```

- Repete o bloco de comandos enquanto uma determinada condição – expressada através da condição lógica - for satisfeita.
  - A expressão lógica é verificada logo na entrada do ciclo e a cada repetição.
  - Se for falsa, a execução continua logo após o **FimEnquanto**
-

# Estrutura Enquanto ... Faça

*Exercício: Contar de 1 até 10*

Preparação: a variável *i* irá controlar a repetição e inicia a contagem. Devemos iniciá-la!

```
Algoritmo Conta
Variáveis
    i: inteiro
Início
    i <- 1
    Enquanto i <= 10 faça
        Escreva(i)
        i <- i + 1
    FimEnquanto
Fim
```

# Estrutura Enquanto ... Faça

## *Exercício:*

*Somar números introduzidos pelo utilizador, até que seja introduzido o zero.*

---

# Estrutura **Repita ... Até**

## Sintaxe

### **Repita**

<bloco de comandos>

**Até** <expressão lógica>

- Repete o bloco de comandos até que determinada condição – expressada através da condição lógica - seja satisfeita.
  - A expressão lógica é verificada no final de cada repetição.
  - Se for verdadeira, a execução continua logo após o **Até**.
-

# Estrutura **Repita ... Até**

***Exercício:***

*Escreva o algoritmo de contagem até 10 usando esta estrutura.*

---



# Estrutura Enquanto x Repita

## Enquanto

- É necessário teste desde o início do ciclo;
- O bloco de comando só é executado se a condição for verdadeira logo no início;
- A execução termina quando a expressão lógica for falsa.

## Repita

- Usam-se quando não é necessário teste no início;
  - O bloco de comandos é executado pelo menos 1 vez;
  - A execução termina quando a expressão lógica for verdadeira.
-

# Intervenções no Ciclo

- Algumas linguagens de programação têm comandos para forçar o retorno ou interrupção do ciclo:
    - Comando `volte` – retorno ao início do ciclo;
    - Comando `interrompa` – faz com que a execução do ciclo pare.
-

# Intervenções no Ciclo

**Enquanto** <expressão lógica> **faça**

<bloco de comandos>

**se** <condição> **então**

**interrompa**

**FimSe**

<bloco de comandos>

**FimEnquanto**

<comando>

...

# Estruturas Condicionais

- Permitem tomar decisões em função de condições estabelecidas permitindo ações **alternativas**.
  - Iremos explorar a estrutura condicional
    - **Se**
-

# Estruturas Condicionais SE

- São estruturas de controlo;
  - Executam um ou mais comandos se a condição testada for *verdadeira*;
  - Em alguns casos, executam um ou mais comandos de a condição for *falsa*.
  - Podem ser simples ou compostas.
-

# Estruturas Condicionais SE Compostas

- Se a condição for **verdadeira**, o bloco de comandos 1 é executado.
- **Caso contrário**, o bloco de comandos 2 é executado.

```
Se <condição> então  
    <bloco de comandos 1>  
Senão  
    <bloco de comandos 2>  
FimSe
```

---

# Estruturas Condicionais SE Compostas

- <condição> pode ser uma variável booleana, uma constante (V ou F) ou uma expressão de valor lógico.

```
Se <condição> então  
    <bloco de comandos 1>  
Senão  
    <bloco de comandos 2>  
FimSe
```

---

# Estrutura Condicionais SE

*Exercício: Solicitar um número ao utilizador e verificar se o número introduzido é maior que 10*

**Algoritmo** MaiorQue10

**Variáveis**

num: real

**Início**

Escreva ("Introduza um número:")

Leia(num)

**Se** num >10 **então**

Escreva ("O número é maior que 10")

**Senão**

Escreva ("O número é menor ou igual que 10")

**FimSe**

**Fim**



# Estrutura Condicionais SE

## *Exercício:*

*Calcule a média de dois testes e indique se o aluno está ou não aprovado (nota de aprovação  $\geq 9.5$ )*

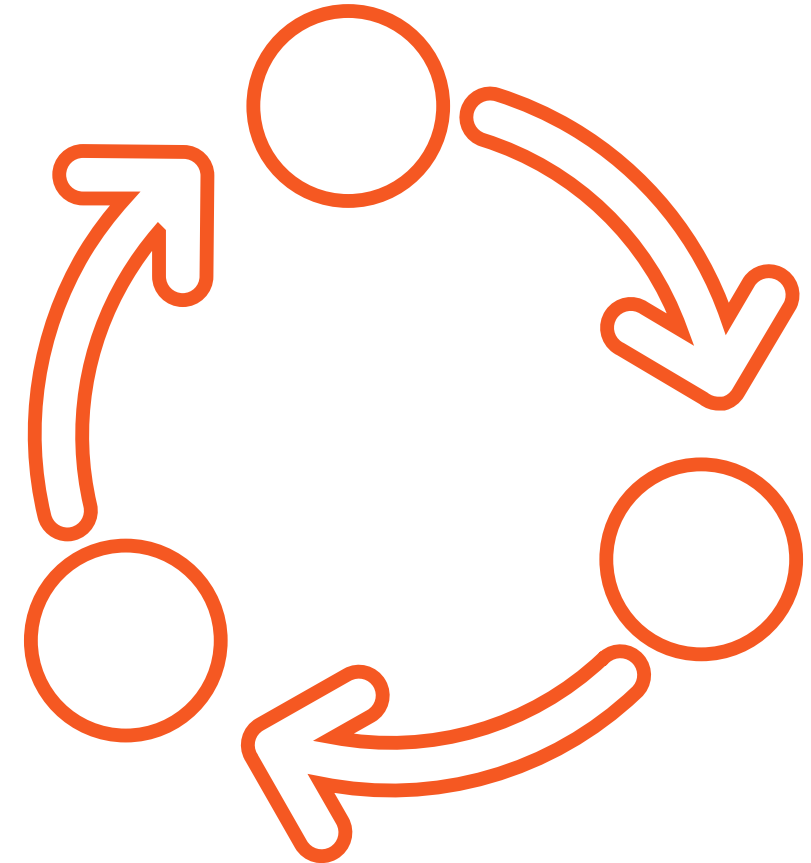
---

# Estruturas Condicionais SE Encadeadas

```
Se <condição 1> então  
    <bloco de comandos 1>  
Senão se <condição 2> então  
    <bloco de comandos 2>  
Senão se <condição 3> então  
    <bloco de comandos 3>  
...  
Senão se <condição N> então  
    <bloco de comandos N>  
FimSe
```

# Combinação de Estruturas

- Podemos combinar as estruturas estudadas:
  - Condicional dentro de condicional (já apresentado);
  - Condicional dentro de ciclo;
  - Ciclo dentro de condicional;
  - Ciclo dentro de outro ciclo (estruturas de repetição encadeadas);
  - Etc.



# Combinação de Estruturas

*Exercício: Listar os números de 1 a 100, destacando os múltiplos de 10.*

**Algoritmo** 0 a 10 destacando múltiplos 10

**Variáveis**

i: inteiro

**Início**

**Para** i **de** 1 **a** 100 **faça**

Escreva(i)

**Se** i **mod** 10 = 0 **então**

Escreva (" (múltiplo de 10)")

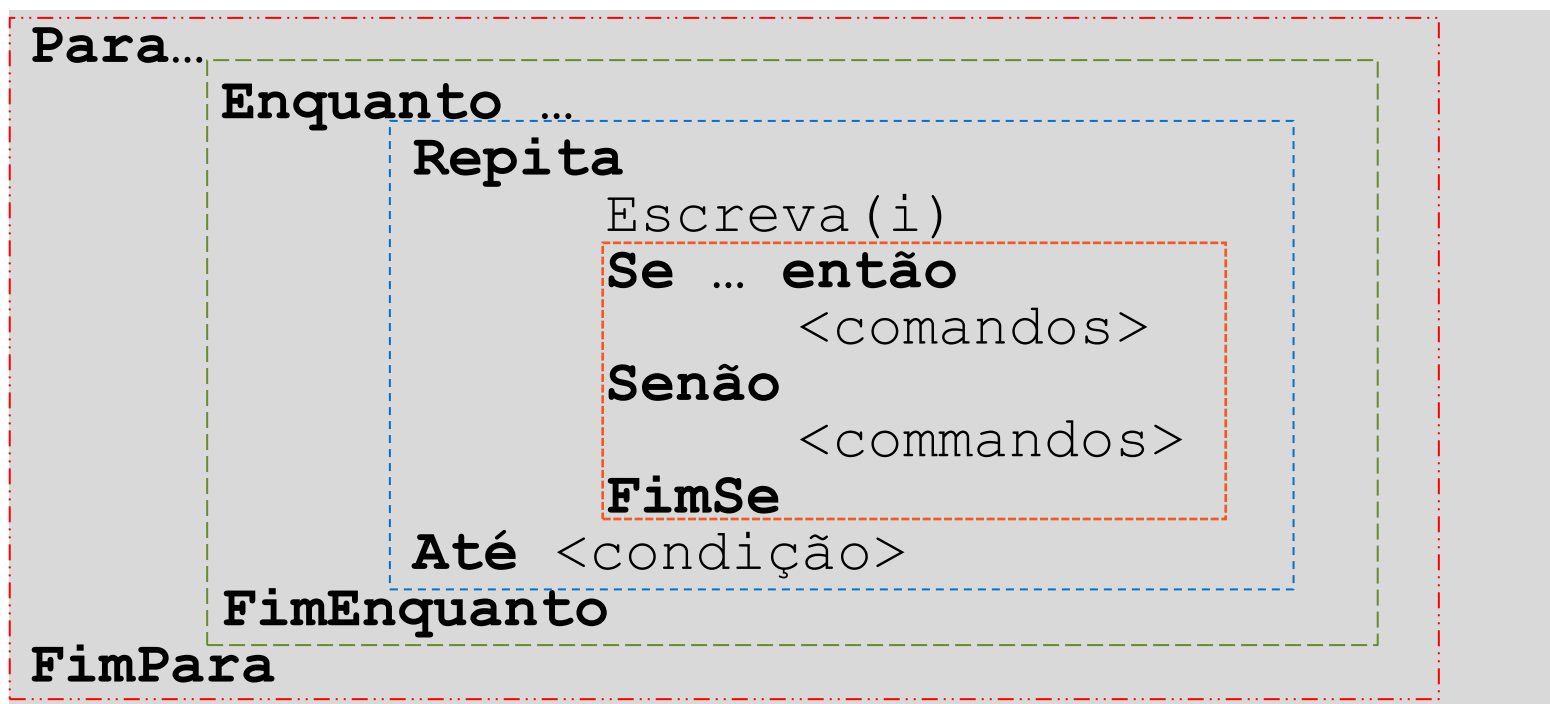
**FimSe**

**FimPara**

**Fim**

# Combinação de Estruturas

Ao combinar estruturas, estas não se podem “cruzar”! Devem estar sempre contidas.



# Testes de algoritmos

- Testar, 'depurar', *debugging* um programa ou algoritmo consiste em analisar os resultados do mesmo passo a passo, linha por linha, de forma a identificar possíveis erros.
- Os ambientes de desenvolvimento disponibilizam ferramentas específicas para tal.
- Para o fazer manualmente, utilizam-se frequentemente **tabelas de teste**.

```
mirror_mod = modifier_ob.  
# Add mirror object to mirror  
mirror_mod.mirror_object
```

```
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.name))  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly one mirror")
```

```
--- OPERATOR CLASSES ---
```

```
types.Operator):  
    "X mirror to the selected  
    object.mirror_mirror_x"  
    "Mirror X"
```

```
context):  
    context.active_object is not None
```

# Testes de algoritmos

Tabela de teste para uma possível execução do algoritmo da média de duas notas.

Lin	nota 1	nota 2	média	Comentário
1	?	?	?	Solicita nota 1
2	8	?	?	Solicita nota 2
3	8	7	?	Calcula Média
4	8	7	8	Mostra média
5	8	7	8	Fim

---

# Testes de algoritmos

- Na **tabelas de teste** temos:
    - Uma coluna para o número da linha do algoritmo;
    - Uma coluna para cada variável (de entrada e saída) do algoritmo;
    - Uma coluna para comentários.
  - É útil para analisar o funcionamento de um algoritmo pequeno ou um excerto de um algoritmo maior.
-



# Exercícios Propostos

1. Solicitar os nomes e notas dos alunos de uma dada turma. Calcular a média final da turma e apresentar o aluno com melhor e pior nota. O final da introdução deve ocorrer com o utilizador introduzir “FIM” como nome de aluno.
  2. Listar os números primos entre dois números introduzidos pelo utilizador.
  3. Listar os números de Fibonacci de 1 até um número introduzido pelo utilizador.
-

# Pensamento Computacional



2022/23

TPC: exercícios propostos aula 2

---