

## PPT04 - Fundamentos de Python (UFCD 10793)

Sandra Liliana Meira de Oliveira



# Tomada de Decisão

## Estrutura MATCH...CASE

*O STRUCTURAL pattern matching introduzido versão [3.10](#) do Python*

```
match expression:  
    case pattern_1:  
        ...  
    case pattern_2:  
        ...
```

```
parameter = "um_valor_especifico"  
match parameter:  
    case <primeira_condição>:  
        <primeiro_bloco_de_instruções>  
    case <segunda_condição>:  
        <segundo_bloco_de_instruções>  
  
    ...  
  
    case <ultima_condição>:  
        <ultimo_bloco_de_instruções>  
    case _:  
        <bloco_de_instrucoes_por defeito>
```

```
status = 404  
  
match status:  
    case 400:  
        print("Bad request")  
    case 404:  
        print("Not found")  
    case 418:  
        print("I'm a teapot")  
    case _:  
        print("Something's wrong with the Internet")
```

# Tomada de Decisão

## Estrutura MATCH...CASE

- O valor do “parameter” é avaliado;
- Se o valor de efetuar correspondência com o primeiro caso (case), é executado o conjunto de instruções imediatamente a seguir e a execução pára.
- Caso contrário passa para o “case” seguinte e assim sucessivamente.
- Se não encontrar nenhuma correspondência o bloco de instruções executado é o que segue “case \_:”

```
parameter = "um_valor_especifico"
match parameter:
    case <primeira_condição>:
        <primeiro_bloco_de_instruções>
    case <segunda_condição>:
        <segundo_bloco_de_instruções>

    ...

    case <ultima_condição>:
        <ultimo_bloco_de_instruções>
    case _:
        <bloco_de_instrucoes_por defeito>
```

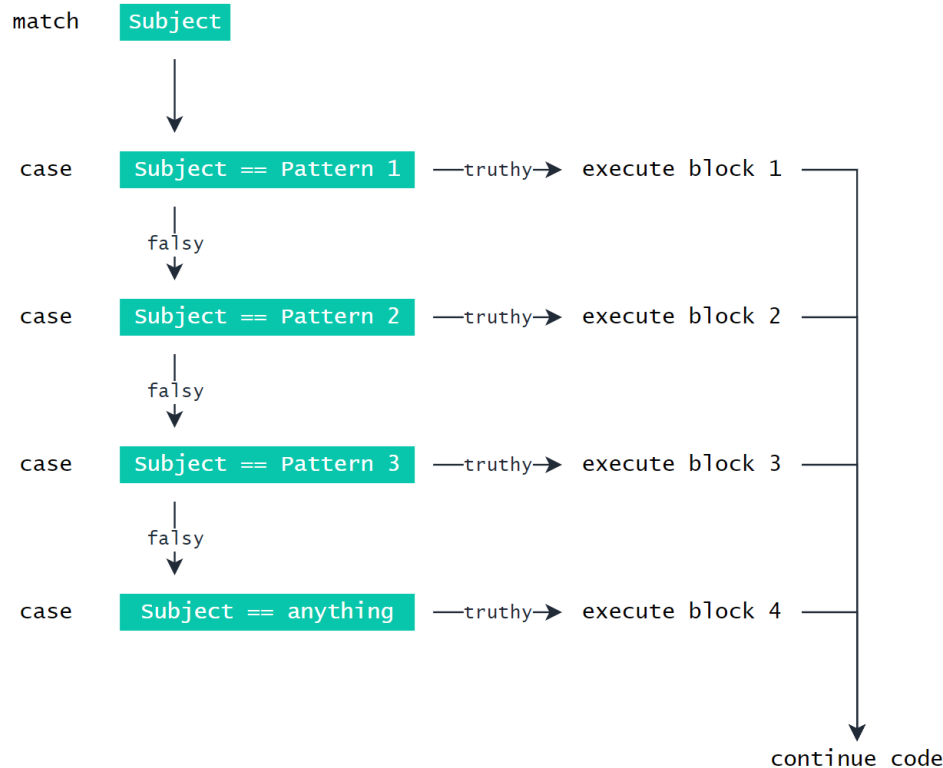
```
match <valor_a_avaliar>:
    case <primeira_condição>:
        ...
    case <segundo_condição>:
        ...

    ...

    case <ultima_condição>:
        ...
    case _:
        <bloco_de_instrucoes_por defeito>
```

# Tomada de Decisão

## Estrutura MATCH...CASE



```
nota=int(input("Qual a nota do aluno:"))
match nota:
    case 1:
        print("Mau!")
    case 2:
        print("Insuficiente!")
    case 3:
        print("Suficiente!")
    case 4:
        print("Bom!")
    case 5:
        print("Muito Bom!")
    case _: #valor por defeito
        print("Nota inválida!")
```

Código mais limpo com o  
Match ... Case

# Tomada de Decisão

## Estrutura MATCH...CASE

<https://peps.python.org/pep-0622/>

<https://peps.python.org/pep-0635/>

<https://python-course.eu/python-tutorial/structural-pattern-matching.php>

# Exercício 1

## Estrutura MATCH...CASE

Escreve um programa que receba o nome de um produto e o seu preço, e retorne o preço total considerando os descontos seguintes:

1. Se o produto for um smartphone, será aplicado um desconto de 10%.
2. Se o produto for um tablet, será aplicado um desconto de 15%.
3. Se o produto for um laptop, será aplicado um desconto de 20%.
4. Para qualquer outro produto, não haverá desconto.

Utilize a estrutura **match...case** para determinar o desconto a ser aplicado.

# Exercício 1 - Resolução

## Estrutura MATCH...CASE

```
produto=input("produto [smartphone,tablet,laptop,outros]:\n--->")
preco =float(input("preço:\n--->"))

match produto:
    case "smartphone":
        desconto = 0.1
    case "tablet":
        desconto= 0.15
    case "laptop":
        desconto = 0.2
    case _:
        desconto = 0.0

preco_total = preco - (preco * desconto)

print(f"O preço total do produto é:", preco_total)
```

# Exercício 2

## Estrutura MATCH...CASE

Escreva um programa que solicite ao utilizador dois números inteiros a operação matemática a ser realizada (+, -, \* e /).

Utilize a estrutura match...case para executar a operação escolhida e devolver o resultado.



# Exercício 2 - Resolução

## Estrutura MATCH...CASE

```
# Solicita ao utilizador os dois números
num1 = int(input("Digite o primeiro número: "))
num2 = int(input("Digite o segundo número: "))

# Solicita ao utilizador a operação desejada
operacao = input("Escolha a operação (+, -, *, /): ")

# Realiza a operação escolhida utilizando a estrutura match...case
match operacao:
    case "+":
        resultado = num1 + num2
    case "-":
        resultado = num1 - num2
    case "*":
        resultado = num1 * num2
    case "/":
        resultado = num1 // num2
    case _:
        resultado = "Operação inválida!"

# Devolve o resultado da operação ou a mensagem de operação inválida
print("Resultado:", resultado)
```

# Estruturas de Repetição

# Ciclos

- A implementação de um problema leva muitas vezes a uma análise de uma bloco de informação. A referida análise, implica a repetição de um conjunto de tarefas para cada elemento da informação.
- As instruções da linguagem que permitem implementar estes raciocínios, são denominadas de **ciclos**.

# Estrutura While

- A instrução **While** (também designada por **ciclo While**), executa uma instrução ou bloco de instruções enquanto uma determinada condição for verdadeira.
- A sua sintaxe é:

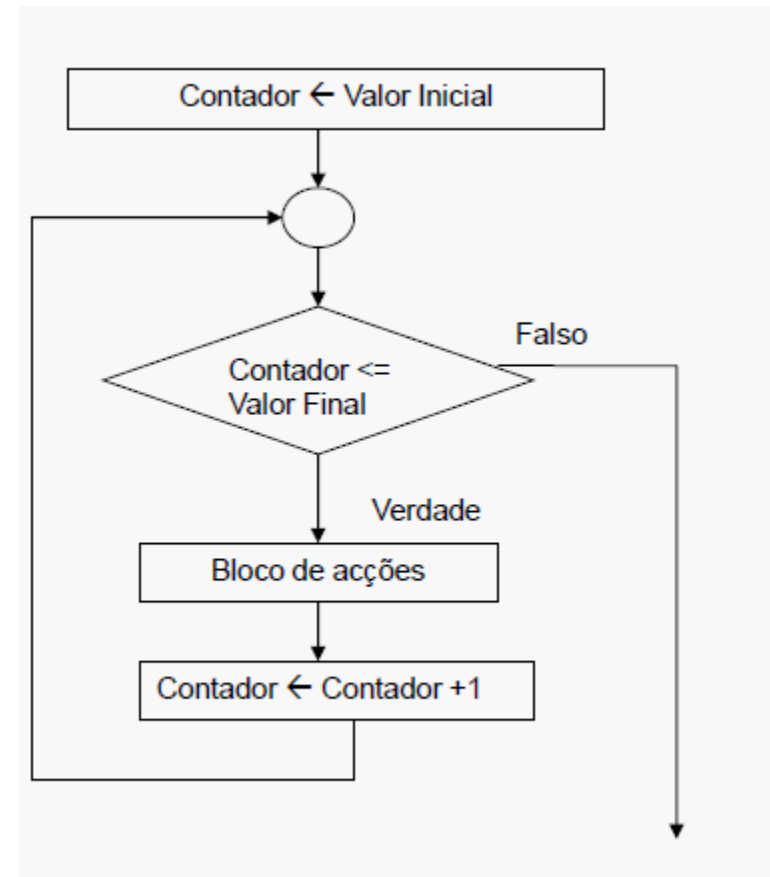
**While <condição>:**  
**instruções**

Os dois pontos ( :) são significativos e obrigatórios. Separam o cabeçalho da instrução composta do corpo .

A linha após os dois pontos deve ser recuada (indentação).

# Estrutura While

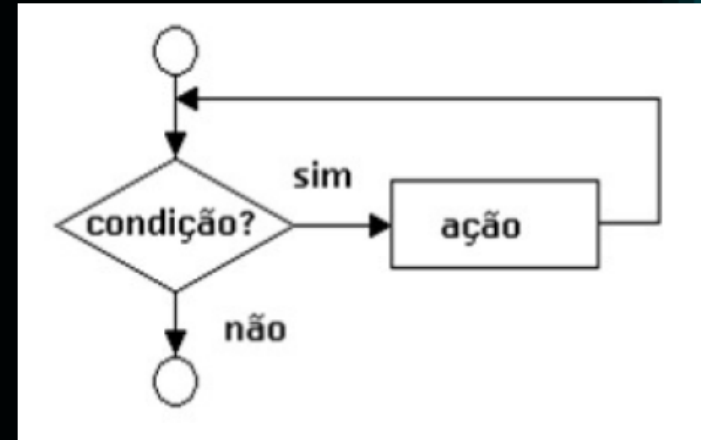
- O funcionamento deste ciclo passa pelos seguintes pontos:
  - Avaliação da condição;
  - Se o resultado da avaliação for Falso, o ciclo termina e o programa continua na instrução imediatamente a seguir ao while;
  - Se o resultado da avaliação for Verdade, é executada a instrução (ou bloco de instruções) associado ao while;
  - Volta-se ao primeiro ponto



# Estrutura While

Permite executar um conjunto de instruções enquanto a condição for verdadeira

```
i=1
while i<6:
    print(i)
    i+=1 #i=i+1
else:print("i é maior ou igual a 6!")
```



A instrução else, permite executar um bloco de instruções uma vez, quando a condição não for mais verdadeira.

No código anterior i é a nossa variável de controlo (determina o número de vezes que o ciclo é executado - contador).

**Tem, obrigatoriamente que ser incrementada em cada ciclo, senão entramos em ciclo infinito**

Quando i=6 o bloco dentro do while deixa de ser executado e é executado o código imediatamente a seguir ao else.

**A instrução else: não é obrigatória.** Nesta situação após execução do ciclo while, o programa continua em instruções seguintes existentes.

# Estrutura While – Break and Continue

```
i = 1
while i < 6:
    print(i)
    if (i == 3):
        break
    i += 1
```



Resultado: 1  
2  
3

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```



Resultado: 1  
2  
4  
5  
6

Com a instrução break, podemos parar o ciclo mesmo se a condição while for verdadeira

Com a instrução continue, podemos parar a iteração atual e continuar na seguinte

# Exercício1 - While

Escreve um programa que calcule a soma dos 10 primeiros números inteiros positivos e devolve o resultado para o ecã



# Exercício1 – While - Resolução

```
i=1
soma=0
while(i<=10):
    soma+=i
    i+=1

print(soma)
```

# Exercício2 - While

Escreve um programa, em que utilizes a instrução while, que devolve a tabuada do 8.

# Exercício2 – While - Resolução

```
i=1
while(i<=10):
    print("8 * %d = %d"%(i,8*i))
    i=i+1
```

# Estrutura For

Processa cada item numa sequência, portanto, é usado com os tipos de dados de sequência do Python - **strings, listas e tuplos**.

Cada item, por sua vez, é (re) atribuído à variável do loop, e o corpo do loop é executado.

```
for <variavel_a_percorrer> in SEQUENCIA:  
    Instruções
```

# Estrutura For

Exemplo: Ciclo que itere um determinado número de vezes ou que itere numa determinada sequência de números.

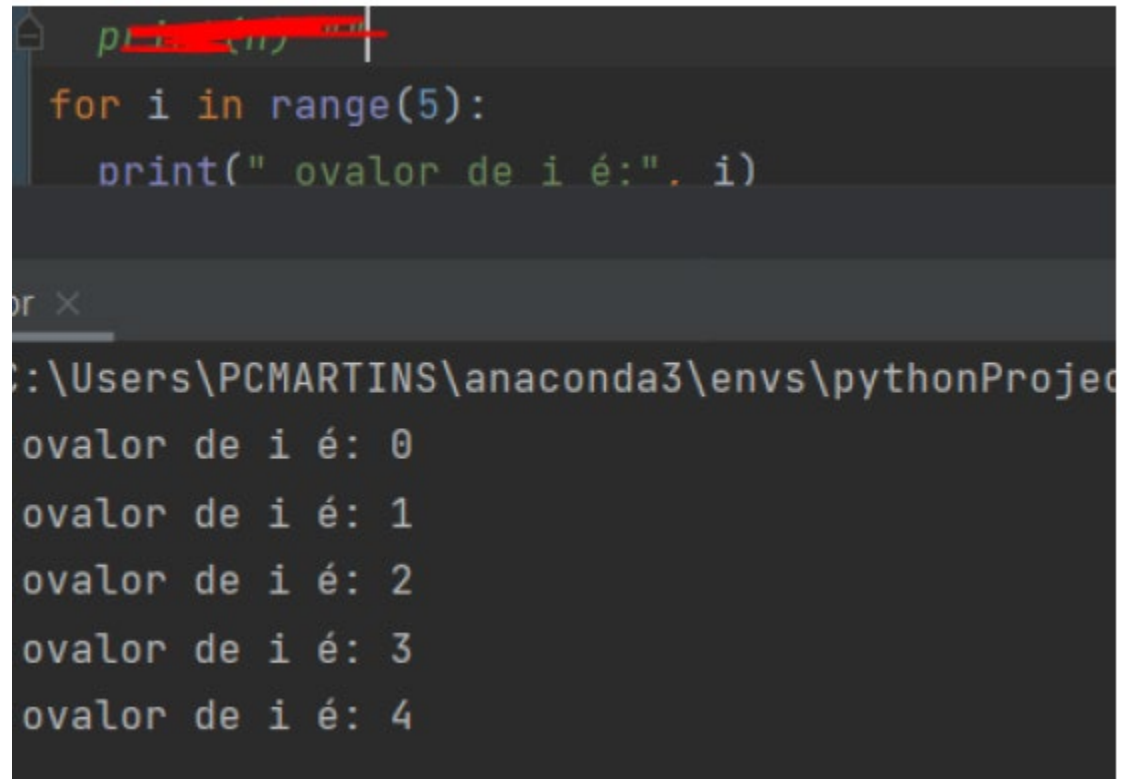
Função `range()` - `range(start, stop, step)`

Parameter	Description
<i>start</i>	Optional. An integer number specifying at which position to start. Default is 0
<i>stop</i>	Required. An integer number specifying at which position to stop (not included).
<i>step</i>	Optional. An integer number specifying the incrementation. Default is 1

```
for i in range(5):  
    print('o valor de i é:', i)
```

# Estrutura For

```
for i in range(5):  
    print('o valor de i é:', i)
```



The screenshot shows a Python IDE with a dark theme. At the top, a line of code `print(i)` is partially visible and crossed out with a red line. Below it, a `for` loop is defined: `for i in range(5):` followed by an indented `print(" ovalor de i é:". i)`. The output window at the bottom displays the results of the loop, showing five lines of text: "ovalor de i é: 0", "ovalor de i é: 1", "ovalor de i é: 2", "ovalor de i é: 3", and "ovalor de i é: 4".

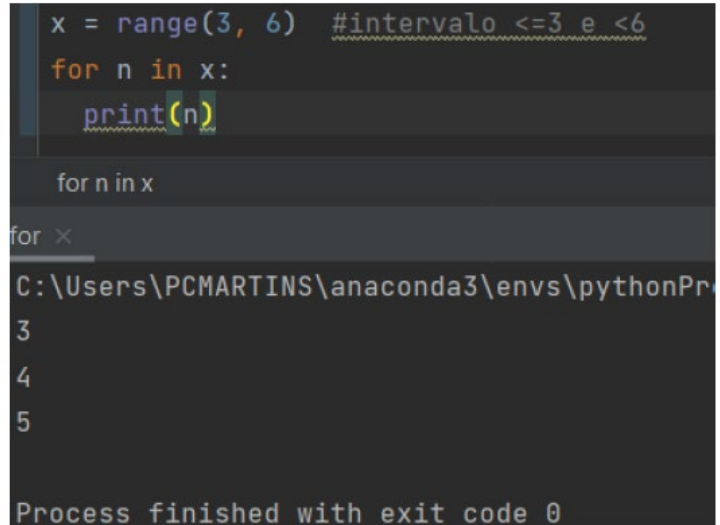
```
print(i)  
for i in range(5):  
    print(" ovalor de i é:". i)  
  
C:\Users\PC MARTINS\anaconda3\envs\pythonProje  
ovalor de i é: 0  
ovalor de i é: 1  
ovalor de i é: 2  
ovalor de i é: 3  
ovalor de i é: 4
```

# Estrutura For

- Exemplo

Crie uma sequência de números de 3 a 5 e imprima cada item na sequência

```
x = range(3, 6)
for n in x:
    print(n)
```



```
x = range(3, 6) #intervalo <=3 e <6
for n in x:
    print(n)

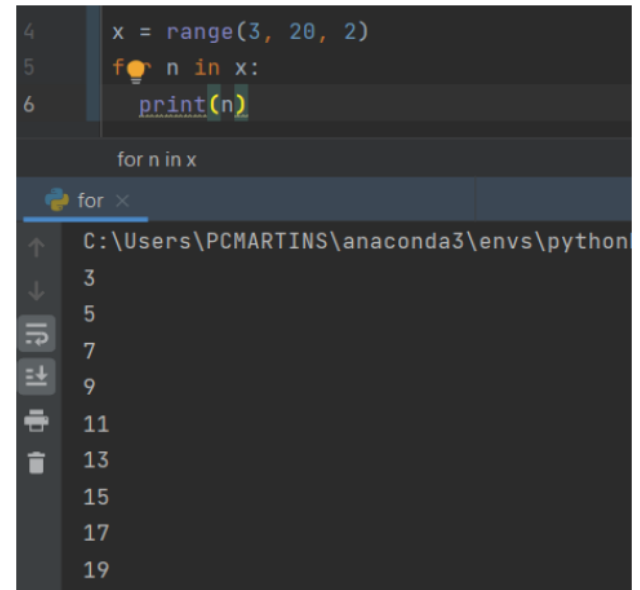
for n in x
for x
C:\Users\PC MARTINS\anaconda3\envs\pythonPr
3
4
5

Process finished with exit code 0
```

# Estrutura FOR

Crie uma sequência de números de 3 a 19, mas incremente em 2 em vez de 1:

```
x = range(3, 20, 2)
for n in x:
    print(n)
```



The screenshot shows a Python IDE with a dark theme. The editor has three lines of code: `x = range(3, 20, 2)`, `for n in x:`, and `print(n)`. Below the editor, a console window displays the output of the program, which is a list of numbers from 3 to 19 in increments of 2: 3, 5, 7, 9, 11, 13, 15, 17, 19. The console window has a title bar that says "for n in x" and a path "C:\Users\PC MARTINS\anaconda3\envs\python".

```
x = range(3, 20, 2)
for n in x:
    print(n)
```

for n in x

C:\Users\PC MARTINS\anaconda3\envs\python

3  
5  
7  
9  
11  
13  
15  
17  
19



# Estrutura For

Permite executar um conjunto de instruções várias vezes:

```
#Qual o resultado?  
for x in range(6):  
    print(x)
```

O 6 corresponde ao número de iterações, neste caso de 0 até 5

A função `range()` controla o número de ciclos com base no valor dentro da função, iniciando em 0 (por padrão), e incrementando 1 (por padrão).

```
for x in range(6):  
    print(x,end=" ")  
else:  
    print("O ciclo terminou!")
```

Neste tipo de estrutura também pode ser utilizado o `else`

# Estrutura FOR – Função Range

```
for x in range(2, 6):  
    print(x)
```

Imprime os valores de 2 a 5

```
for x in range(2, 30, 3):  
    print(x)
```

Imprime os valores de 2 a 29 em saltos de 3

# FOR – Break and Continue

```
for x in range(6):  
    if x == 3: break  
    print(x)  
else:  
    print("O ciclo terminou!")
```

- Os comandos break e continue também podem ser utilizados no for.
- Não esquecer que o break interrompe o ciclo e por isso o que está dentro do else nunca é executado se o break for executado.

# FOR dentro de FOR – Nested Loops

## FOR DENTRO DE FOR

Todos os ciclos do for interno são executados a cada iteração do for externo

```
for x in range(4):  
    print("C1",end=" ")  
    for y in range(3):  
        print("C2",end=" ")
```

# Nested LOOP – Ciclo FOR

## Nested For loop

```
for i in range(1, 11):  
    for j in range(1, 11):  
        print(i*j, end=" ")  
    print('')
```

The diagram illustrates the structure of a nested for loop. A green bracket on the left groups the two for loops under the label "Outer Loop". A red bracket on the left groups the inner for loop and its body under the label "Inner loop". A blue arrow points from the `print(i*j, end=" ")` statement to the label "Body of inner loop". A purple bracket on the right groups the inner loop's body and the final `print('')` statement under the label "Body of Outer loop".

# Nested LOOP – Ciclo FOR

```
for i in range(1, 11): #outer loop
    #nested loop
    # iterações de 1 to 10
    for j in range(1, 11):
        # print multiplication
        multiplicacao = i * j
        print("%d*%d=%d\n"%(i,j,multiplicacao))
print("concluido")
```

1. Neste programa, o ciclo for externo itera números de 1 a 10.
2. Na primeira iteração do ciclo aninhado(nested), o número é 1. Na próxima, é 2. e assim por diante até 10.
3. Em seguida, para cada iteração do loop externo, o loop interno será executado dez vezes.
4. Em cada iteração de um loop interno, calculamos a multiplicação de dois números.

Exercício 1 – reproduzir o código acima 😊

## Exercício2 - For

Escreve um programa que calcule a soma dos 10 primeiros números inteiros positivos e devolve o resultado para o ecã

# Exercício2 – For - Resolução

```
soma=0
for x in range(1, 11):
    soma+=x

print(soma)
```



# Exercício3 - For

Escreve um programa que devolve todos os números entre 100 e 200 (exclusive), divisíveis por 3

# Exercício3 – For - Resolução

```
for x in range(100, 200):  
    div=x%3  
    if div==0:  
        print(x)
```