



UNIDADE II

Introdução à Programação Estruturada

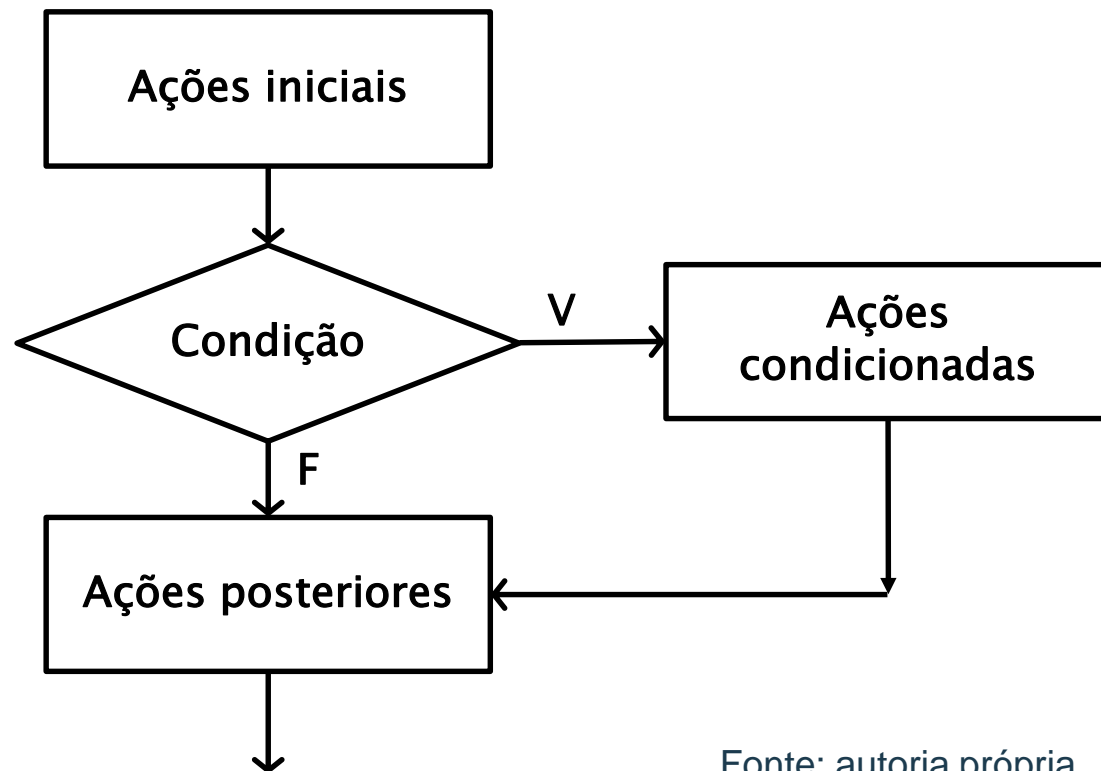
Prof. Me. Ricardo Veras

Estrutura Condicional (*if – elif – else*)

ESTRUTURA CONDICIONAL – IF:

- Esta é uma estrutura que, se determinada condição for **verdadeira**, executa uma ação antes de seguir a execução adiante, de modo que se a condição for falsa, simplesmente, segue a execução adiante sem executar aquela ação;
- Obs.: a condição é uma expressão lógica (uma comparação), cujo valor pode ser: *True* ou *False*.

(V) (F)



Fonte: autoria própria.

Estrutura Condicional (*if* – *elif* – *else*)

ESTRUTURA CONDICIONAL – *IF*:

Sua sintaxe é:

***if* (condição):**

Bloco de comandos da condicional (em Python, um bloco é iniciado com “:”);

Este bloco pode ter tantos comandos quanto necessário;

e é delimitado pelas “linhas indentadas” (afastadas da margem).

Esta linha já não faz parte do bloco da condicional (não está indentada).

Estrutura Condicional (*if – elif – else*)

ESTRUTURA CONDICIONAL – *IF*:

- Neste caso, o bloco delimitado pelas linhas indentadas só será executado se a condição for verdadeira (*True*).

Exemplo:

```
programa.py X
C: > Python > Programas > Geral > programa.py > ...
1  a = int(input("Digite um valor numérico: "))
2  if (a < 10):
3      print("Você digitou um número menor que 10.")
4  print("FIM DO PROGRAMA")
5
```

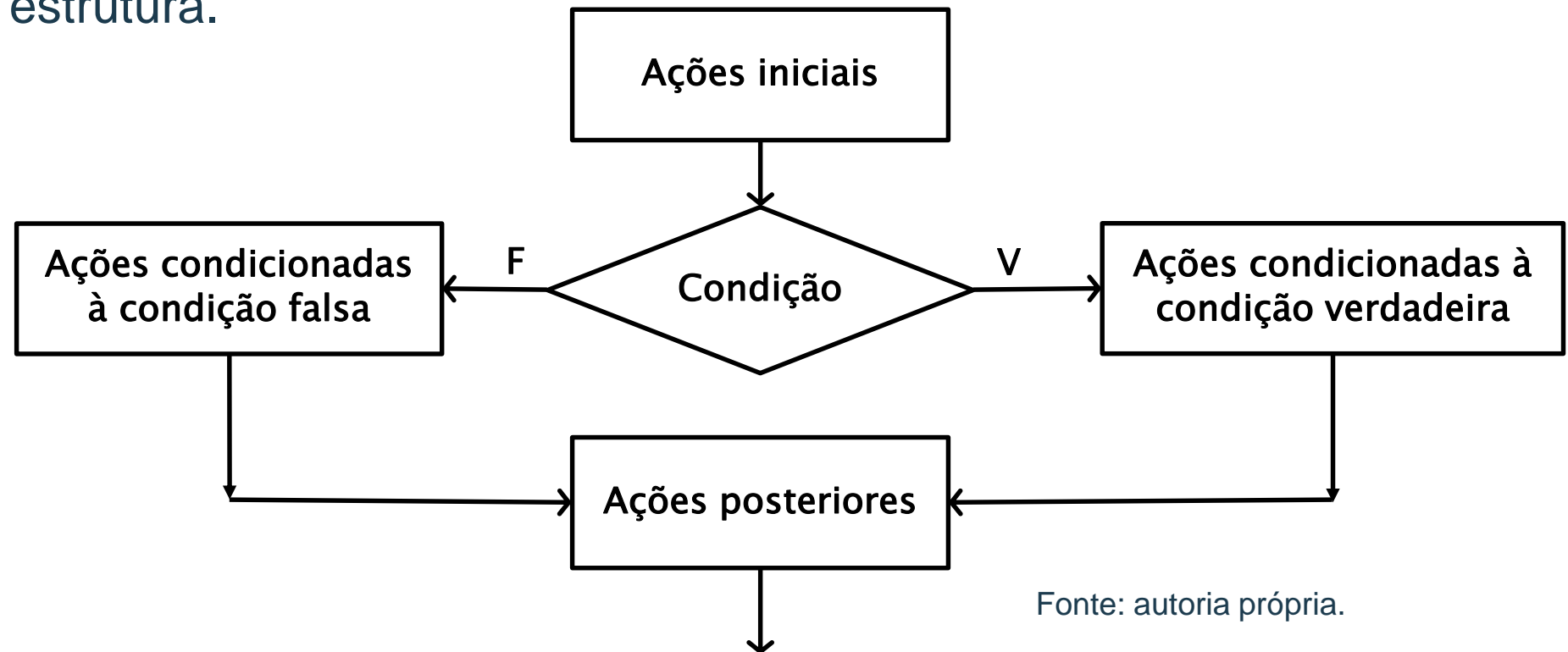
Fonte: autoria própria.

Estrutura Condicional (*if – elif – else*)

ESTRUTURA CONDICIONAL – IF/ELSE:

- Esta é uma estrutura que executará determinada ação se a condição for verdadeira e executará outra ação se aquela mesma condição for falsa, de forma que, depois, segue a execução posterior à estrutura.

Obs.: *Else* \equiv Senão...



Fonte: autoria própria.

Estrutura Condicional (*if* – *elif* – *else*)

ESTRUTURA CONDICIONAL – *IF/ELSE*:

A sintaxe desta estrutura é:

***if* (condição):**

1º bloco de comandos da condicional. Este bloco pode ter tantos comandos;

quanto necessário e somente será executado se a condição for **verdadeira (*True*)**.

***else*:**

2º bloco de comandos da condicional. Este bloco pode ter tantos comandos;

quanto necessário, e somente será executado se a condição for **falsa (*False*)**.

Esta linha não faz parte dos blocos “*if*” ou “*else*”.

Estrutura Condicional (*if – elif – else*)

ESTRUTURA CONDICIONAL – IF/ELSE:

Exemplo:

programa.py X

C: > Python > Programas > Geral > programa.py > ...

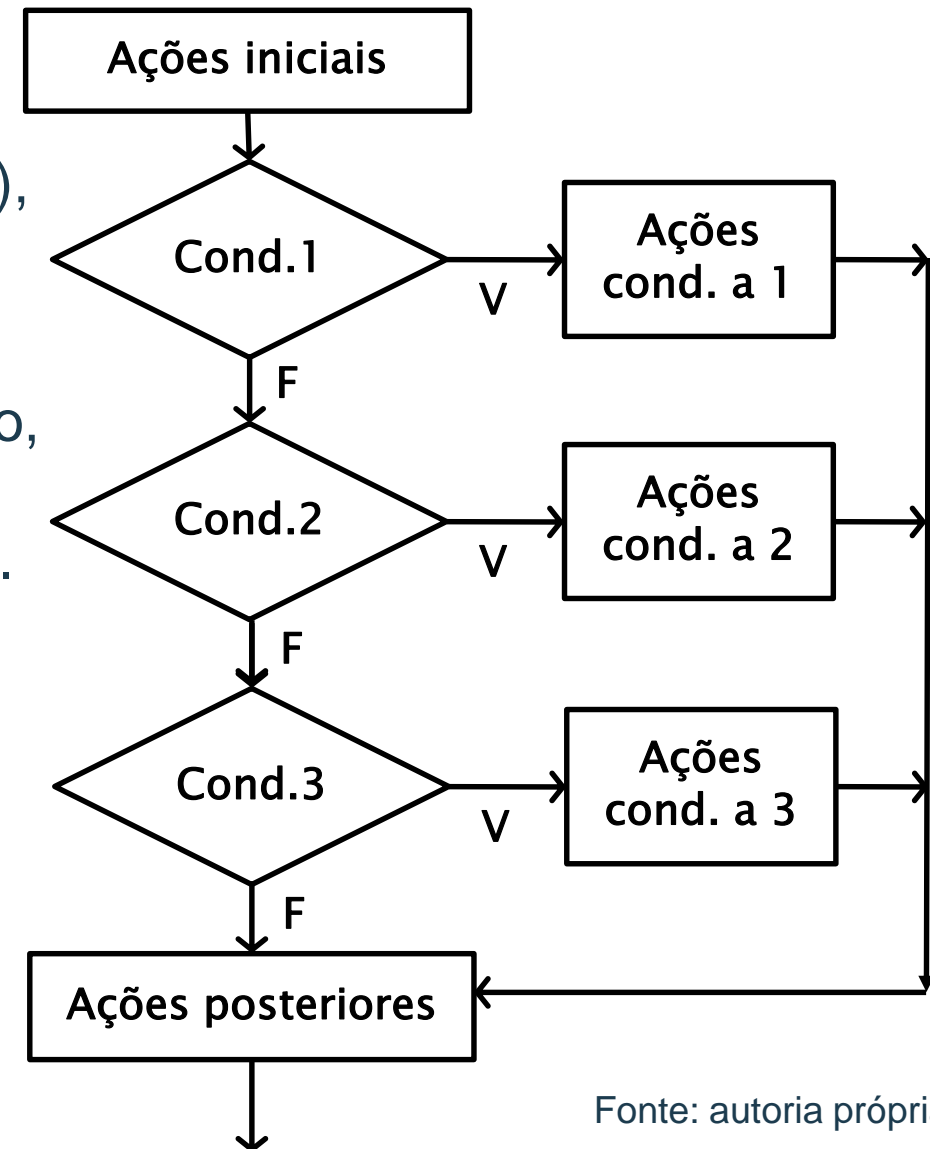
```
1  a = int(input("Digite um valor numérico: "))
2  if (a < 10):
3      print("Você digitou um número menor que 10.")
4  else:
5      print("Você digitou um número que é maior ou igual a 10.")
6  print("FIM DO PROGRAMA")
7  |
```

Fonte: autoria própria.

Estrutura Condicional (*if – elif – else*)

ESTRUTURA CONDICIONAL – IF/ELIF/ELSE:

- Esta estrutura contém várias condições (decisões), cada uma definindo um grupo de ações;
- No entanto, apenas um dos blocos será executado, de forma que será executado o bloco da primeira condição verdadeira e os demais serão ignorados.



Fonte: autoria própria.

Estrutura Condicional (*if – elif – else*)

ESTRUTURA CONDICIONAL – *IF/ELIF/ELSE*:

***if* (condição_1):**

- # 1º bloco de comandos da condicional;
- # Somente será executado se a;
- # condição 1 for **verdadeira (*True*)**.

***elif* (condição_2):**

- # 2º bloco de comandos da condicional;
- # Somente será executado se a;
- # condição 1 for falsa e a 2 **verdadeira**.

...

***else*:**

- # Bloco de comandos do *else*;
- # Somente será executado se;
- # todas as condições anteriores forem **falsas**.

Término da estrutura condicional.

Estrutura Condicional (*if – elif – else*)

ESTRUTURA CONDICIONAL – IF/ELIF/ELSE:

Exemplo:

```
programa.py X
C: > Python > Programas > Geral > programa.py > ...
1  a = int(input("Digite um valor numérico: "))
2  if (a < 10):
3      print("Você digitou um número menor que 10.")
4  elif (a < 100):
5      print("Você digitou um número menor que 100.")
6  elif (a < 1000):
7      print("Você digitou um número menor que 1000.")
8  else:
9      print("Você digitou um número que é maior ou igual a 1000.")
10 print("FIM DO PROGRAMA")
11
```

Fonte: autoria própria.

Interatividade

Imagine que um usuário rode o programa a seguir duas vezes, sendo que ele digita 6 na primeira vez e 8 na segunda vez. O que será impresso na tela da console em cada uma delas?

```
x = int(input("Entre com o valor numérico: "))
if ((x % 2) != 0):
    x += 10
elif ((x % 3) != 0):
    x += 20
else:
    x += 30
print(x)
```

- a) 36 na primeira vez e 28 na segunda vez.
- b) 26 na primeira vez e 38 na segunda vez.
- c) 16 na primeira vez e 38 na segunda vez.
- d) 26 na primeira vez e 18 na segunda vez.
- e) 36 na primeira vez e 38 na segunda vez.

Resposta

Imagine que um usuário rode o programa a seguir duas vezes, sendo que ele digita 6 na primeira vez e 8 na segunda vez. O que será impresso na tela da console em cada uma delas?

```
x = int(input("Entre com o valor numérico: "))
if ((x % 2) != 0):
    x += 10
elif ((x % 3) != 0):
    x += 20
else:
    x += 30
print(x)
```

- a) 36 na primeira vez e 28 na segunda vez.
- b) 26 na primeira vez e 38 na segunda vez.
- c) 16 na primeira vez e 38 na segunda vez.
- d) 26 na primeira vez e 18 na segunda vez.
- e) 36 na primeira vez e 38 na segunda vez.

Estrutura Condicional – Forma curta

FORMA CURTA DA ESTRUTURA CONDICIONAL SIMPLES (IF):

- Existe uma forma curta de se codificar uma decisão simples em uma única linha. Esta forma só pode ser utilizada se houver, **apenas, uma** instrução condicionada à condição;
- *if* <condição>: <instrução>.

Exemplo:

```
programa.py X
C: > Python > Programas > Geral > programa.py > ...
1  a = int(input("Digite o valor de A: "))
2  b = int(input("Digite o valor de B: "))
3  if (a == b): print("O valor de A é igual ao valor de B.")
4  print("FIM DO PROGRAMA")
5
```

Fonte: autoria própria.

Estrutura Condicional – Forma curta

FORMA CURTA DA ESTRUTURA CONDICIONAL COMPOSTA (IF – ELSE):

- Existe uma forma curta de codificar uma decisão composta em uma única linha (conhecida como Operadores Ternários ou Expressão Condicional);
- <instrução_se_verdadeiro> if <condição> else <instrução_se_falso>.
- Obs.: percebam que, neste caso, não utilizamos “dois-pontos” (nem após a condição, nem após o “else”).

```
programa.py X
C: > Python > Programas > Geral > programa.py > ...
1  a = int(input("Digite o valor de A: "))
2  b = int(input("Digite o valor de B: "))
3  print("a é maior") if a > b else print("b é maior")
4  print("FIM DO PROGRAMA")
5  |
```

Fonte: autoria própria.

Estrutura Condicional – Níveis

ESTRUTURA CONDICIONAL ENCADEADA:

- Pode existir qualquer quantidade de encadeamento (níveis) de estruturas condicionais, desde que, a cada nível, seja adicionada uma quantidade equivalente de espaços de indentação.

Exemplo:

```
programa.py X
C: > Python > Programas > Geral > programa.py > ...
1  a = int(input("Digite o valor de A: "))
2  if (a > 10):
3      print("O valor de a é maior que dez")
4      if (a > 100):
5          print("...e também maior que cem")
6          if (a > 1000):
7              print("...e também maior que mil.")
8          else:
9              print("...porém menor que mil.")
10 print("FIM DO PROGRAMA")
11
```

Estrutura Condicional – Comando *pass*

O COMANDO *PASS*:

- A Linguagem Python exige que cada bloco da estrutura condicional “*if*” tenha, **pelo menos, uma instrução**. Por conta disto, dependendo da situação (quando a expressão lógica for mais fácil ou, ainda, quando se quer aguardar para programar em outro momento), cria-se um bloco “vazio” na estrutura. A instrução “*pass*” é uma “instrução vazia”, ou seja, que não executa ação específica alguma.

programa.py X

C: > Python > Programas > Geral > programa.py > ...

```
1  a = int(input("Digite o valor de A: "))
2  b = int(input("Digite o valor de B: "))
3  if b > a:
4      pass
5  else:
6      print("b é menor ou igual a a")
7  print("FIM DO PROGRAMA")
8
```


Estrutura *switch-case*

A ESTRUTURA CONDICIONAL – *SWITCH-CASE*:

- A Linguagem Python não possui a estrutura condicional *SWITCH-CASE*;
- Essa estrutura existe em linguagens como Java, JavaScript, C, C++, C#, entre outras;
- A estrutura do Python que mais se aproxima à estrutura “*switch-case*”, é a estrutura “*if – elif – else*”.

Estrutura *switch-case*

A ESTRUTURA CONDICIONAL – SWITCH-CASE:

- Esta estrutura realiza uma comparação de valores a partir do valor de uma variável. A comparação feita é (unicamente) a de **igualdade**. Nesta estrutura, não é possível se fazer comparações do tipo “maior que”, “menor que”, “diferente de”, entre outras.

Sintaxe (em outras linguagens):

```
switch(variável) {  
    case valor01:  
        <bloco de instruções>  
        break;  
    case valor02:  
        <bloco de instruções>  
        break;  
    default:  
        <bloco de instruções>  
        break;  
}
```

Fonte: autoria própria.

Estrutura Condicional: “if – elif – else” e “switch-case”

Programa feito em “Python”:

```
1  x = int(input("Digite o valor de x: "))
2  if(x == 1):
3      print("O valor de x é igual a 1")
4  elif(x == 2):
5      print("O valor de x é igual a 2")
6  elif(x == 3):
7      print("O valor de x é igual a 3")
8  else:
9      print("O valor de x é maior que 3")
10     print("...ou menor que 1")
```

Programa feito em “Java”:

```
int x = Integer.parseInt(
    JOptionPane.showInputDialog("Digite o valor de x:"));
switch(x) {
    case 1:
        System.out.println("O valor de x é igual a 1");
        break;
    case 2:
        System.out.println("O valor de x é igual a 2");
        break;
    case 3:
        System.out.println("O valor de x é igual a 3");
        break;
    default:
        System.out.println("O valor de x é maior que 3");
        System.out.println("...ou menor que 1");
        break;
}
```

Fontes: autoria própria.

Estrutura *switch-case*

A ESTRUTURA CONDICIONAL – *SWITCH-CASE* (continuação):

- Um bom exemplo de utilização da estrutura *switch-case*, é a geração de Menus de Opção, onde o usuário seleciona, dentre um pequeno número de opções, na forma de números inteiros, qual é a opção de ação que ele pretende realizar;
- Além deste exemplo, pode-se utilizar esta estrutura para os processos automáticos que envolvem a leitura de arquivos externos (geralmente, os arquivos de textos), onde se tem identificadores, neste texto, que remetem o sistema a determinadas ações, dependendo do caractere especial que é lido no arquivo;
 - Observação: uma estrutura “*case*” pode ser substituída por uma estrutura “*if*” em qualquer ocasião. No entanto, nem toda estrutura “*if*” pode ser substituída por uma estrutura “*case*” (como aquelas onde a lógica de comparação é uma lógica que não leva o símbolo de igualdade).

Interatividade

Imagine que tenhamos uma situação em que o programador tenha que fazer uma implementação num programa existente, onde deve-se fazer uma comparação do valor de uma variável com diferentes “**intervalos de valores**” (mais de dois intervalos), como, por exemplo: verificar se o valor está no intervalo entre 5 e 10; ou no intervalo entre 10 e 15; ou no intervalo entre 15 e 20; e, assim por diante, sendo que, dependendo do intervalo em que ele estiver, uma série de comandos (mais de um comando) deverá ser executada. Para que esta situação possa ser programada, qual das estruturas condicionais, a seguir, é a mais indicada a ser utilizada, permitindo o grau de comparação necessária?

- a) Estrutura *if* (condicional simples);
- b) Estrutura *if-else* (condicional composta);
- c) Estrutura *if-elif-else*.
- d) Estrutura *switch-case*.
- e) Estrutura condicional composta, na sua forma curta.

Resposta

Imagine que tenhamos uma situação em que o programador tenha que fazer uma implementação num programa existente, onde deve-se fazer uma comparação do valor de uma variável com diferentes “**intervalos de valores**” (mais de dois intervalos), como, por exemplo: verificar se o valor está no intervalo entre 5 e 10; ou no intervalo entre 10 e 15; ou no intervalo entre 15 e 20; e, assim por diante, sendo que, dependendo do intervalo em que ele estiver, uma série de comandos (mais de um comando) deverá ser executada. Para que esta situação possa ser programada, qual das estruturas condicionais, a seguir, é a mais indicada a ser utilizada, permitindo o grau de comparação necessária?

- a) Estrutura *if* (condicional simples);
- b) Estrutura *if-else* (condicional composta);
- c) Estrutura *if-elif-else*.
- d) Estrutura *switch-case*.
- e) Estrutura condicional composta, na sua forma curta.

Estrutura “*while*” de repetição

- De um modo geral, as estruturas de repetição são aquelas que permitem executar mais de uma vez (repetidas vezes) um mesmo conjunto de códigos (bloco de códigos).

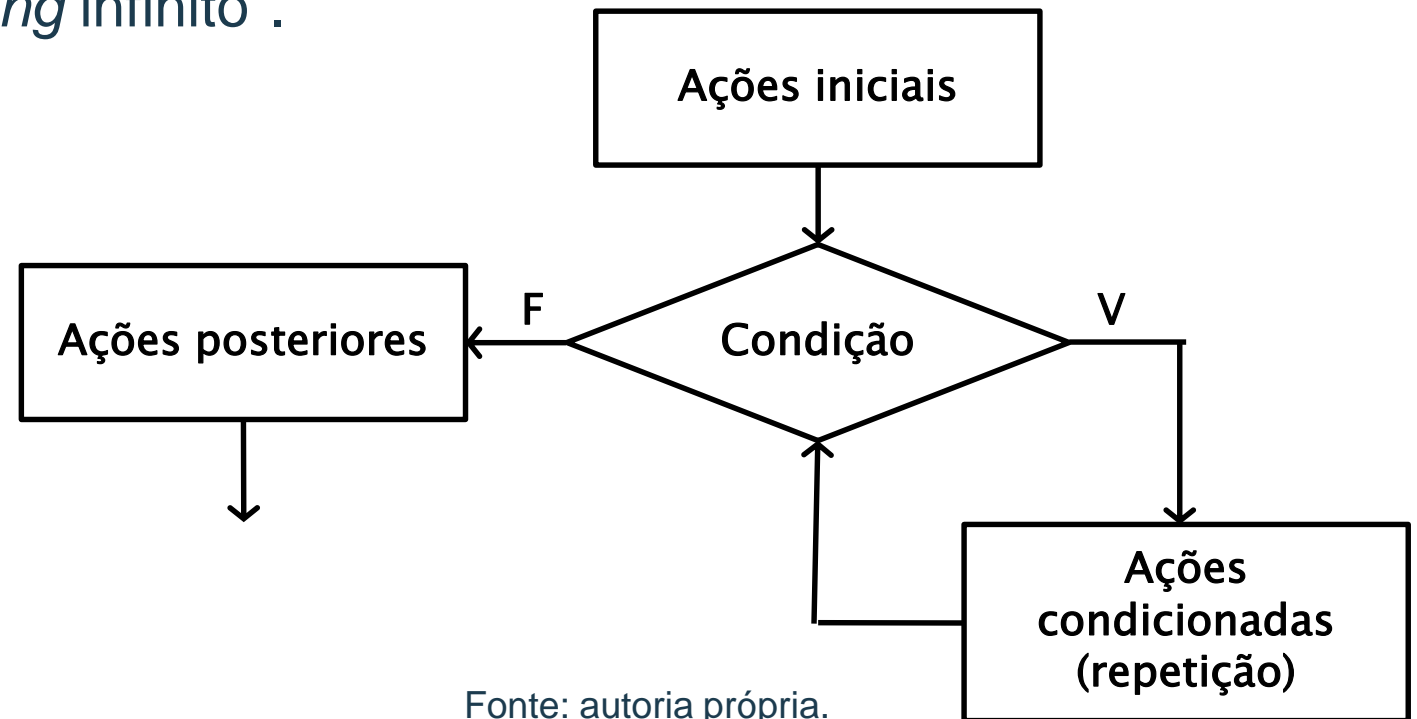
A estrutura “*while*” (enquanto) de repetição, repete um mesmo processamento “enquanto” a sua condição for verdadeira (semelhante à estrutura *if* – da condicional). Sua sintaxe é:

***while* (condição):**

- # Bloco cuja execução é controlada pela condição;
- # Executará repetidas vezes enquanto a condição for verdadeira;
- # Somente deixará de repetir quando a condição se tornar falsa.

Estrutura “*while*” de repetição

- Assim como a estrutura condicional, a estrutura de repetição “*while*” contém um elemento de “decisão”, o qual realiza uma operação lógica de comparação, cuja resposta será, ou verdadeira ou falsa.
- Devemos tomar cuidado para que o elemento de decisão torne-se falso em algum momento, evitando o que chamamos de “*looping* infinito”.



Fonte: autoria própria.

Estrutura “while” de repetição

Exemplo:

- Programa que imprime os números de 0 a 5.

programa.py X

C: > Python > Programas > Geral > programa.py > ...

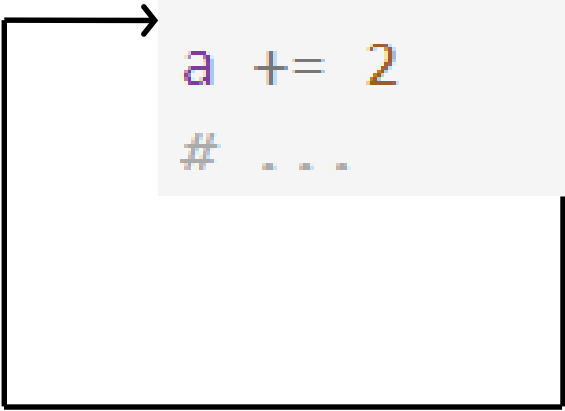
```
1  a = 0
2  while (a <= 5):
3      print(a)
4      a += 1
5  print ("FIM DO PROGRAMA")
6
```

```
0
1
2
3
4
5
FIM DO PROGRAMA
```

Contadores e acumuladores

- A condição de uma estrutura “*while*” utiliza uma “variável de controle”. Quando a quantidade de repetições é conhecida, esta variável de controle pode ser utilizada como uma **variável contadora** (uma variável que adquire um valor segundo uma contagem definida).
- A “**variável contadora**” da estrutura “*while*” deve ter um valor inicial e, ao longo das **iterações**, (das repetições) o seu valor deve se alterar de forma constante, até que permita que a condição de repetição da estrutura “*while*” se torne falsa.

```
a = 1  #valor inicial  
# ...  
a += 2  
# ...
```

A diagram illustrating a loop structure. A black arrow starts from the bottom of the code block, goes down, then right, and finally up to point at the line 'a += 2', indicating a loop back to that line.

Fonte: autoria própria.

Contadores e acumuladores

- As “**variáveis acumuladoras**” são utilizadas para “acumular os valores” de acordo com as operações matemáticas (uma soma, ou uma multiplicação por exemplo).
- Num mesmo programa, a variável acumuladora deve ser uma variável diferente da variável contadora (elas devem possuir nomes diferentes uma da outra).
- A diferença entre uma “variável contadora” e uma “variável acumuladora” é que nas “contadoras” o valor adicionado é constante (fixo), enquanto que nas “acumuladoras” o valor adicionado é variável.
- Exemplo: programa que calcula a soma dos números inteiros do intervalo de 1 a 100.

```
c = 1           # valor inicial da variável contadora "c"
a = 0           # valor inicial da variável acumuladora "a"
while(c <= 100): # repete enquanto "c" for menor ou igual a 100
    a += c       # acrescenta o valor de "c" à acumuladora
    c += 1       # acrescenta o valor 1(um) à contadora
print(a)
```

Interrupções (*break/continue*)

- Sabemos que a estrutura “*while*” verifica a sua condição de repetição, somente, no início de cada iteração (em cada uma das repetições).
- Dependendo da situação, pode ser necessário interromper a repetição antes da verificação seguinte da condição do “*while*”. Esta interrupção pode ser realizada com o comando “***break***” ou com o comando “***continue***”.
- O comando “***break***” interrompe a execução da estrutura de repetição como um todo, saindo da estrutura independentemente do valor atual da condição.
 - O comando “***continue***” interrompe, apenas, aquela iteração (não executando os comandos subsequentes do mesmo bloco da estrutura), e passa a executar uma nova iteração, mas continuando na mesma estrutura de repetição.

Interrupções (*break/continue*)

Exemplo:

O programa a seguir solicita a entrada de vários números reais pelo usuário, até que ele digite o valor 0 (zero), de forma que quando isto ocorrer o programa para de solicitar as entradas de valores, e mostra a quantidade de números digitados pelo usuário e a soma de todos eles:

```
soma = 0 # guardará a soma, acumulando valores digitados
qtd = 0 # guardará a quantidade de valores digitados
while (True):
    n = float(input("Digite um valor numérico real: "))
    if (n == 0):
        break # terminará o looping com n = 0(zero)
    soma += n # "acumulador" dos valores de n
    qtd += 1 # "contador" das quantidades de valores
print ("Quantidade de valores digitados: ", qtd)
print ("Soma dos valores digitados: ", soma)
```

Fonte: autoria própria.

Interrupções (*break/continue*)

Mesmo programa, mas utilizando interrupções diferentes podem gerar saídas diferentes:

```
1  n = 0
2  while (n < 10):
3      n += 1
4      if (n == 5):
5          break
6      print (n)
7  print ("FIM DO PROGRAMA")
```

```
1
2
3
4
FIM DO PROGRAMA
```

```
1  n = 0
2  while (n < 10):
3      n += 1
4      if (n == 5):
5          continue
6      print (n)
7  print ("FIM DO PROGRAMA")
```

```
1
2
3
4
6
7
8
9
10
FIM DO PROGRAMA
```

Fontes: autoria própria.

Interatividade

Analizando o código a seguir, o que será impresso na tela do console?

```
x = 1
ac = 0
while (x < 18):
    if (x % 5 == 0): ac += x
    x += 1
print(ac)
```

- a) 0.
- b) 5.
- c) 18.
- d) 30.
- e) 50.

Resposta

Analizando o código a seguir, o que será impresso na tela do console?

```
x = 1
ac = 0
while (x < 18):
    if (x % 5 == 0): ac += x
    x += 1
print(ac)
```

- a) 0.
- b) 5.
- c) 18.
- d) 30.
- e) 50.

O comando *range*

- O comando “***range***” gera os intervalos de valores (um conjunto simples com todos os números inteiros do intervalo definido).

Sintaxes do comando “***range***”:

- ***range*** (fim);
- ***range*** (início, fim);
- ***range*** (início, fim, passo).

O comando *range*

- Início – o intervalo de um *range* inicia-se ou no valor 0 (zero) (“*range*” com, apenas, 1 parâmetro) ou, exatamente, no valor indicado (termo de início).
- Fim – o intervalo sempre terminará no valor equivalente a “fim – 1”.
- Passo – o passo do *range* define “de quanto em quanto” serão os valores do intervalo (será a contagem).
- Os valores de “início”, “fim” e “passo” são sempre números inteiros.

O comando *range*

Exemplos:

`x = range(15)`

... x adquire todos os valores (ordenados) do intervalo de 0 a 14 (inclusive):

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14.

`x = range(8, 13)`

... x adquire todos os valores (ordenados) do intervalo de 8 a 12 (inclusive):

8, 9, 10, 11, 12.

`x = range(7, 26, 5)`

... gera um conjunto de alguns valores do intervalo de 7 a 22 de modo que inicia em 7, somando-se 5 a cada valor seguinte:

7, 12, 17, 22.

A estrutura “*for*”

- Em Python, a estrutura (ou laço) “*for*” é sempre utilizada a partir de um conjunto de valores (como listas, *ranges*, tuplas, etc. – alguns destes são assuntos que veremos mais adiante).
- Durante a execução de uma estrutura “*for*”, a cada iteração (execução completa de seu bloco) a variável de referência aponta para um elemento do conjunto seguindo a sua sequência (do início ao fim), ou seja, em cada iteração a “variável de referência” adquire cada um dos valores do conjunto.

Sintaxe:

- ***for* <variável_de_referência> *in* <conjunto_de_valores>:
 <bloco_de_códigos>.**
- O bloco de códigos desta estrutura (bloco indentado) contém os comandos que serão executados repetidas vezes.

A estrutura “for”

Exemplos de programas com a estrutura “for”:

```
1  for x in range(20):
2      print("Felicidade")
3  print("FIM DO PROGRAMA")
4
```

- Este programa imprime 20 vezes a palavra “Felicidade”.

```
1  mat = [2, 7, 9, 13, 5, 1, -2]
2  for x in mat:
3      print (x)
4  print ("FIM DO PROGRAMA")
```

- Este outro imprime cada valor da matriz *mat*.

Fontes: autoria própria.

A estrutura “*for*”

Exemplo: utilizando a estrutura “*for*”, para somarmos os números de 1 a 100 basta criarmos o intervalo de 1 a 100 com o comando *range* e gerarmos uma estrutura de repetição que acumula a soma dos valores do intervalo:

```
1  '''
2  Para criar o intervalo de 1 a 100 pode-se
3  utilizar o comando "range" iniciando em 1
4  (já que o início é exato) e terminando
5  em 101 (pois o fim sempre é um antes do
6  valor indicado)
7  '''
8  soma = 0
9  for x in range(1, 101):
10     soma += x
11     print (soma)
12     print ("FIM DO PROGRAMA")
```

Fonte: autoria própria.

- ... obs.: esta soma resulta em 5050.

A estrutura “*for*”

Exemplo: utilizando a mesma estrutura “*for*”, do exemplo anterior, que alteração precisaríamos fazer para somar, apenas, os números “pares” do intervalo de 0 a 100?

A estrutura “for”

Exemplo: utilizando a mesma estrutura “**for**”, do exemplo anterior, que alteração precisaríamos fazer para somar, apenas, os números “pares” do intervalo de 0 a 100?

```
1  '''
2  Para criar o intervalo de 1 a 100 só com
3  números pares, o comando "range" deve iniciar
4  no número 2 (já que o início é exato),
5  terminar em 101, e "pular" de 2 em 2 valores.
6  '''
7  soma = 0
8  for x in range(2, 101, 2):
9      soma += x
10 print (soma)
11 print ("FIM DO PROGRAMA")
```

- ... obs.: esta soma resulta em 2550.

Fonte: autoria própria.

Interatividade

Analizando o código a seguir, o que será impresso na tela do console?

```
for x in range(0, 100, 10):  
    if ((x > 20) and (x <= 70)):  
        continue  
    print(x)
```

- a) 1, 11, 71, 81, 91.
- b) 0, 10, 20, 80, 90.
- c) 10, 20, 80, 90, 100.
- d) 30, 40, 50, 60, 70.
- e) 0, 100, 10.

Resposta

Analizando o código a seguir, o que será impresso na tela do console?

```
for x in range(0, 100, 10):  
    if ((x > 20) and (x <= 70)):  
        continue  
    print(x)
```

- a) 1, 11, 71, 81, 91.
- b) 0, 10, 20, 80, 90.**
- c) 10, 20, 80, 90, 100.
- d) 30, 40, 50, 60, 70.
- e) 0, 100, 10.

Referências

- Imagens das telas do VS *Code*: autoria própria.

ATÉ A PRÓXIMA!