Linguagem de Programação

05: CONTROLE ITERATIVO – LAÇO INDEFINIDO - WHILE

Nossos **objetivos** nesta aula são:

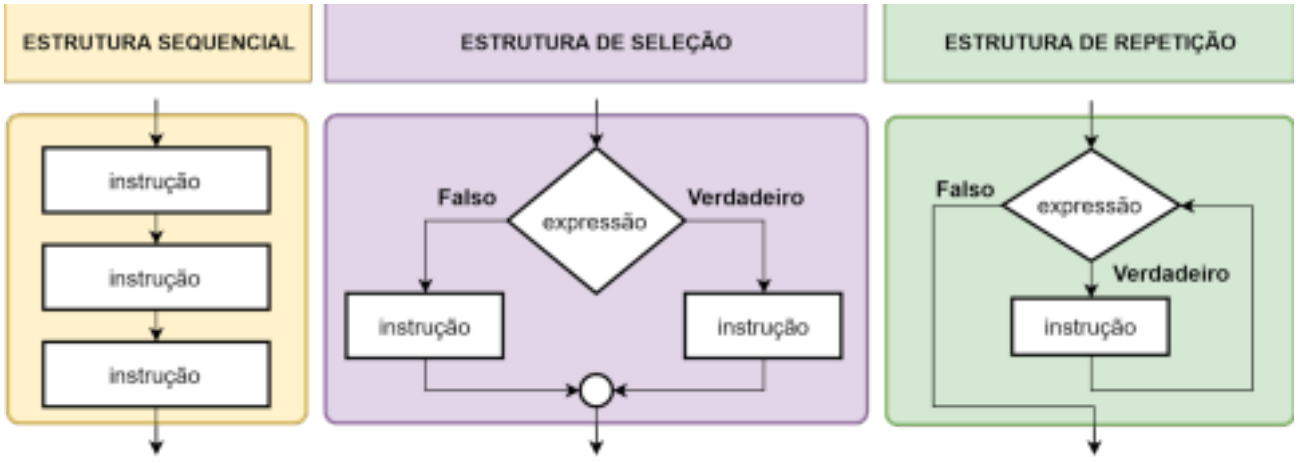
- Definir o conceito de estrutura de controle **iterativo**;
- Entender o conceito de **laços**;
- Executar passo a passo, avaliando os valores de variáveis, um algoritmo que envolve estruturas de seleção e repetição;
- Conceituar variável **contadora** e variável **acumuladora**;
- Explicar como validar dados de entrada usando estrutura de repetição;
- Iniciar o exercício da habilidade de escrever programas que empregam estruturas de repetição, iterativas.

A referência para esta aula são as seções **3.4 (Controle de Seleção)** do **Capítulo 3 (Control Structures)** do livro:

DIERBACH, C. Introduction to Computer Science Using Python: A Computational Problem Solving Focus. 1st Edition, New York: Wiley, 2012.

INTRODUÇÃO

Já vimos duas formas de controlar o fluxo de um programa: estrutura de controle sequencial e estrutura de controle de seleção ou condicional.



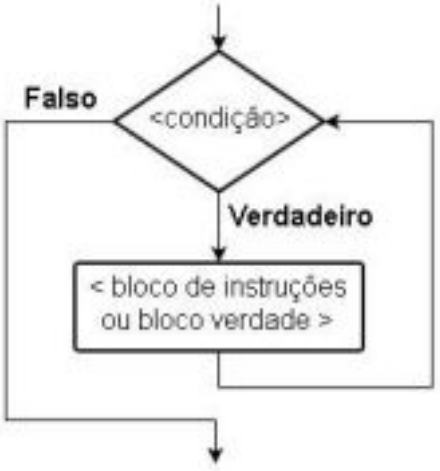
Vamos iniciar as **estruturas de repetição**, também conhecidas como **iterativas** ou de **laço** (loop), que, junto com as demais estruturas vistas até agora, propiciarão a solução de uma gama muito maior de problemas.

As estruturas de repetição permitem que uma ação seja executada **mais de uma vez** sem que tenhamos que executar novamente o programa.

Uma estrutura de **controle iterativo**, estrutura de repetição, laço de repetição ou simplesmente **laço** é uma estrutura de controle de fluxo formada por um conjunto de instruções que são executadas um **número determinado de vezes** ou enquanto uma determinada **condição** for verdadeira.

ESTRUTURA DE REPETIÇÃO

Podemos classificar as estruturas de repetição em 2 tipos: **definidas** e **indefinidas**. Na estrutura de repetição definida sabemos previamente o número de vezes que o bloco será executado, por exemplo, podemos iterar sobre os itens de uma lista, do primeiro ao último. Já na estrutura de repetição indefinida, a iteração é controlada por uma condição, portanto não sabemos a priori quantas vezes o laço será repetido.



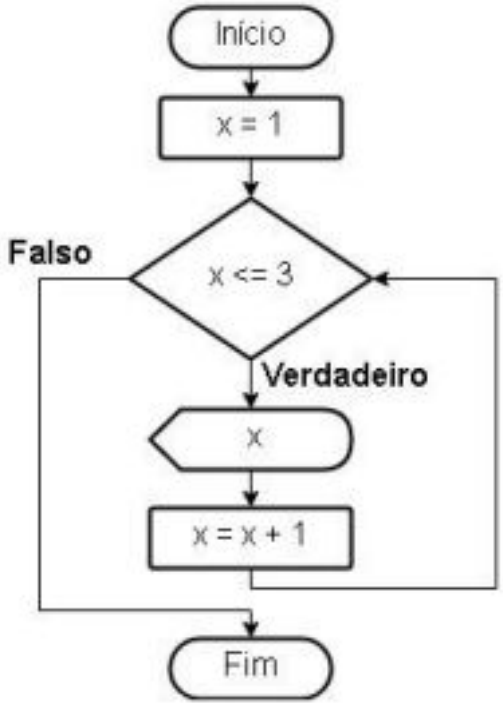
A primeira estrutura que veremos é o laço indefinido, representado em Python pelo comando **while**, que é uma estrutura de controle iterativo que executa repetidamente um bloco de instruções baseado em uma expressão booleana ou condição. A iteração será mantida até que a condição dada se torne falsa.

Esta estrutura permite executar diversas vezes um bloco de instruções, sempre verificando **antes** se a **<condição>** é verdadeira.

A condição é uma expressão booleana, semelhante às expressões que usamos nas estruturas condicionais com if e elif.

No fluxograma dado, A **<condição>** é primeiro avaliada, se for verdadeira, o **<bloco verdade>** é executado e a condição é reavaliada. Quando a **<condição>** for falsa, a iteração termina e a execução continua com a instrução após o laço do while. Em python, a definição desse laço fica:

```
while <condição>:
  <Bloco de instruções ou bloco verdade>
```



2
ibe os números inteiros de 1 a 3.

Teste de Mesa - Teste de Execução

```
   x x <= 3 Tela
 1 1 <= 3 True 1
 2 2 <= 3 True 2
 3 3 <= 3 True 3
4 4 <= 3 False
```

A execução para no momento que o teste da condição avalia para **Falso**

A variável **x** é inicializada com o valor igual a **1** e a condição é avaliada pela primeira vez. Como condição **x <= 3** é verdadeira, a instrução dentro do laço é executada, exibindo o valor de **x** e atualizando o valor de **x** para **2**. O controle de execução retorna para o topo do laço na avaliação da condição novamente. Como **2** é menor ou igual a **3**, o bloco dentro do laço é executado, exibindo o valor de **x** e atualizando **x** para **3**. Retoma-se ao topo da estrutura com o teste da condição, como é verdade, o bloco de instruções é executado novamente pela terceira vez. Nesse ponto, a variável **x** assumiu o valor 4 e a condição **x <= 3** resulta em falso (False), terminando assim a repetição do bloco. Traduzindo para Python:

```
x = 1
while x <= 3:
 print(x)
 x = x + 1
```

É importante observar que, no caso acima descrito, as instruções do laço são executadas 3 vezes, enquanto a condição é avaliada 4 vezes. Isto acontece pois toda vez que a condição é avaliada, o

bloco de instruções ou bloco verdade é executado e retorna-se à verificação da condição, até que a condição seja avaliada para Falso.

Para praticar

1) Escreva um programa para exibir os números de 50 a 100.
2) Escreva um programa para escrever na tela a contagem regressiva do lançamento de um foguete. O programa deve exibir 10, 9, 8, ..., 1,0 e Fogo!

VARIÁVEL CONTADORA

Variável contadora é uma variável utilizada para controlar a contagem do número de execuções do bloco de instruções.

Recebe um valor inicial (geralmente 0 ou 1) e é incrementada em algum ponto do bloco do laço de um valor constante (geralmente 1).

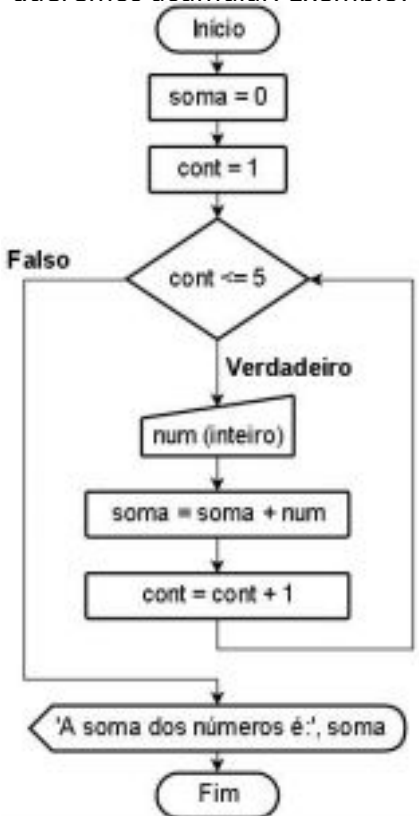
Incrementar uma variável é o mesmo que somar um valor constante a essa variável. O valor da variável pode ser decrementado também, como ocorreu no exercício 2 da página anterior.

Para praticar

1) Escreva um programa para exibir todos os números pares de 0 até o número informado pelo usuário.
2) Escreva um programa para exibir todos os números pares de 0 até o número informado pelo usuário, sem a utilização do if para avaliar se o número é par.

VARIÁVEL ACUMULADORA

Variável acumuladora é uma variável utilizada para controlar a contagem, que recebe um valor inicial (geralmente 0) e é incrementada em algum ponto do bloco do laço com o valor que queremos acumular. Exemplo: Calcular a soma de 5 números digitados pelo usuário.




```
soma = 0
cont = 1

while cont <= 5:
    num = int(input('Digite um inteiro: '))
    soma = soma + num
    cont = cont + 1

print('A soma dos números é:', soma)
```

Teste de Mesa

soma	cont	cont <= 5	num
0	1	True	3
3	2	True	4
7	3	True	2

```
 9 4 True 5
14 5 True 8

22 6 False
(fim do laço)
```

Tela: A soma dos números é: 22

DICA: Execute o código acima no Python Tutor para visualizar a execução do código passo a passo e acompanhar as alterações que estão ocorrendo nos valores das variáveis.

LAÇO INFINITO

Laço infinito é uma estrutura de controle iterativo que nunca termina (ou eventualmente termina com um erro de sistema).

Exemplo: Calcular a soma de 5 números digitados pelo usuário, porém a condição nunca é atualizada, pois a variável contadora não está sendo atualizada.

```
soma = 0
cont = 1

while cont <= 5:
 num = int(input('Digite um número inteiro: '))
 soma = soma + num

print('A soma dos números é:', soma)
```

Nesse caso é preciso interromper manualmente a execução do programa, seja finalizando o processo que está sendo executado via um gerenciador de processos/tarefas ou utilizando a combinação de teclas CTRL + C, que no Python irá lançar a exceção **KeyboardInterrupt** e parar a execução do programa.

Para praticar

Crie os programas a seguir e analise o código para prever o resultado esperado. Em seguida execute-os e compare os resultados com a sua previsão. Lembre-se de conferir a indentação do código ao digitá-lo:

```
# Programa 5a

n = 10
soma = 0
atual = 1

while atual <= n:
    soma = soma + atual

print(soma)
```

```
# Programa 5b

n = 10
soma = 0
atual = 1
```

```
                                    = n - 1
while atual <= n:
 soma = soma + atual  n        print(soma)
```

Algum dos códigos foi capaz de somar os números inteiros de 1 a 10? Você consegue explicar por quê?

Validação de dados de entrada

Consiste em verificar se o valor informado pelo usuário está correto ou não. Anteriormente usamos a estrutura condicional para checar informações de entrada e, em caso de erro, encerramos o programa.

Usando uma estrutura de repetição podemos permitir que o usuário digite o dado enquanto esteja informando de forma incorreta.

Exemplo: Se o usuário precisa digitar um número no intervalo $1 \leq n \leq 50$, podemos usar a estrutura de repetição que recebe o número e, enquanto este número estiver fora do intervalo permitido, pede novamente a que o número seja digitado.

```
n = int(input('Digite um número de 1 a 50:'))

while n < 1 or n > 50:
 n = int(input('Fora do intervalo, digite novamente: '))

print('Número aceito!')
```

Execução:

Tela n n < 1 or n > 50

Digite um número de 1 a 50: 0 True or False => True Digite um número de 1 a 50: 51 False or True => True Digite um número de 1 a 50: 10 False or False => False Número aceito!

Exemplo: Escreva um programa que leia e valide a entrada de um número inteiro e positivo. Usando uma estrutura de repetição podemos permitir que o usuário digite o dado enquanto esteja informando de forma incorreta.

```
n = -1
```

```
while n < 0:
 n = int(input('Digite um número inteiro e positivo: '))

print('Número aceito!')
```


Interrupção do Laço – Flag booleana

O controle do laço while pode ser feito por meio de uma variável booleana, chamada de flag booleana. Dentro do laço verificamos se uma ou mais condições são atendidas através de uma uma estrutura de seleção e alteramos o valor da flag para que o laço seja encerrado. Se necessário podemos combinar mais de uma flag na condição, usando os operadores lógicos.

Reescrevendo o exemplo anterior, da entrada de um número inteiro e positivo, com o uso de uma flag booleana, chegamos ao seguinte fluxograma:

```
valido = False

while not valido:
 n = int(input('Digite um número inteiro e positivo: '))
if n >= 0:
 valido = True

print('Número aceito!')
```

É importante notar que o nome da flag e o valor inicial que lhe é atribuído devem ser coerentes com a situação do problema modelado, isto é, iniciar uma flag cujo nome seja VALIDO com um valor True não faz sentido, pois teríamos que alterá-la para False justamente no momento em que o número fosse validado. Isso significaria que a flag guardaria o valor True enquanto o número ainda não foi validado (ou seja, é INVÁLIDO) e guardaria o valor False quando o número finalmente fosse VÁLIDO.

Para praticar

1) Escreva um programa que leia um número inteiro *n* e escreva um número inteiro *m* que corresponde ao número *n* invertido. Por exemplo, se *n* igual a 1234, a saída será *m* igual a 4321.
2) Escreva um programa que calcule o **MMC (mínimo múltiplo comum)** entre dois números

naturais.

3) Escreva um programa que calcule o **MDC (máximo divisor comum)** entre dois números naturais.

4) Escreva um programa que leia um número inteiro e positivo representando um número binário, determine o seu equivalente decimal.

Exemplo: Dado 10010 a saída será 18, pois $2^4 + 0.\ 2^3 + 0.\ 2^2 + 1.\ 2^1 + 0.\ 2^0 = 18$

5) Escreva um programa que leia um número inteiro e positivo representando um número decimal, determine o seu equivalente binário.

Exemplo: Dado 18 a saída deverá ser 10010.

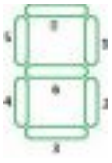
6) A série de Fibonacci é formada pela sequência 1, 1, 2, 3, 5, 8, 13, 21, 34, ... Escreva um programa que apresente a série de Fibonacci até o *n-ésimo* termo (n > 0).

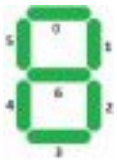
7) Escreva um programa que apresente todos os ímpares de 1 até 99.

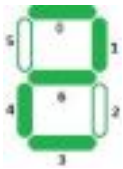
EXERCÍCIO – DISPLAY DE 7 SEGMENTOS

Vamos criar um programa que exiba uma contagem regressiva em um display de 7 segmentos.

Um display de 7 segmentos é um componente digital que possui 7 elementos (segmentos) que podem ser acesos ou apagados para formar números. Veja a figura abaixo. Cada segmento possui um número:

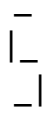

Display com todos os segmentos apagados


Display com todos os segmentos acesos


Display com os segmentos 0,1,3,4 e 6 acesos formando o número 2

Escreva uma função que irá receber o um número inteiro entre 0 e 9 e exibir na tela o número simulando um display de 7 segmentos, em seguida, escreva um programa que use esta função para fazer uma contagem regressiva de 9 a 0. Para deixar a contagem mais realista, utilize a função *sleep*, que pode ser importada do módulo *time*.

```
>>> from time import sleep # importa a função a partir do módulo
>>> sleep(1) # espera 1 segundo
>>> exibe_display(5) # exibe o display do número 5
```



EXERCÍCIOS PROPOSTOS (ESTRUTURA DE REPETIÇÃO)

1) Crie os programas a seguir e analise o código para prever o resultado esperado. Em seguida execute-os e compare os resultados com a sua previsão. Lembre-se de conferir a indentação do código ao digitá-lo:

```
# Programa 5a

n = 10
soma = 0
atual = 1

while atual <= n:
 soma = soma + atual
atual = atual + 1

print(soma)
```

```
# Programa 5b

n = 10
soma = 0
atual = 1

while atual <= n:
 soma = soma + atual
atual = atual + 1

 print(soma)
```

Qual a diferença no resultado da execução de ambos os códigos?

2) Escreva um programa para escrever os 10 primeiros múltiplos de 3.

3) Dada a quantidade de alunos de uma turma, receber a cada passo a nota de 3 provas (um número real entre 0 e 10), calcular a média semestral e mostrar uma mensagem para os alunos aprovados, com média maior ou igual a 6. (Não usar listas)

4) Idem ao exercício anterior, mas exibir ao final da execução, a média geral da turma, considerando todos os alunos, e a média geral apenas dos alunos aprovados. (Não usar listas)

5) Escreva um programa que leia dois números inteiros e calcule a multiplicação entre os

números dados, sem o uso do operador *, mas sim pela soma sucessiva de um deles.

6) Exemplo: 3 x 4 = 3 + 3 + 3 = 4 + 4 + 4 = 12

7) Escreva um programa que leia dois números inteiros e calcule o quociente da divisão entre os números dados, sem o uso do operador /, apenas com operações de soma ou subtração entre eles. Lembre-se de que podemos entender o quociente da divisão de dois números como sendo a quantidade de vezes que podemos retirar o divisor do dividendo. Exemplo: 20 / 4 = 5, uma vez que podemos subtrair o número 4 do número 20 cinco vezes.

8) Escreva um programa que leia dois números inteiros e calcule o resto da divisão, sem o uso do operador %, apenas com operações de soma e subtração entre eles. Ex: 20 % 4 = 0.

9) Faça um programa que, dado x inteiro e n natural, calcule x^n sem utilizar o operador **.

10) Escreva um programa que leia uma sequência de números inteiros, encontre e imprima o maior e o menor número. A entrada de um número negativo indica que sequência terminou.

11) Faça um programa que mostre um menu para o usuário escolher uma operação ou sair do programa. Por exemplo:

```
Soma .......... +
Subtração ..... -
Multiplicação.. *
Divisão........ /
Sair .......... 0
```

O programa deve reconhecer a opção escolhida pelo usuário e solicitar a entrada de dois números caso a opção escolhida seja uma das operações. Depois de fazer a operação e mostrar o resultado o programa deve continuar em execução para que o usuário faça selecione operação e outros números. O programa só terminará caso o usuário digite 0.

12) Uma pesquisa sobre a população de uma determinada região coletou os seguintes dados, referentes a cada habitante, para serem analisados:

- Idade (em anos)
- Sexo (M - masculino, F - feminino)

A fim de indicar o final da entrada, após a sequência de dados dos habitantes, o usuário entrará com o valor -1 para a idade, o que deve ser interpretado pelo programa como fim

de entrada de dados. Após receber todos os dados, exiba a maior idade digitada e percentual de indivíduos do sexo feminino com idade entre 18 e 35 anos, em relação ao total de entrevistados. (Não usar listas)

13) A empresa em que trabalha pretende realizar uma pesquisa sobre a aceitação de um produto, e necessita de um programa que irá coletar os dados a seguir:

- idade – valor numérico indicando o número de anos de vida;
- sexo – "masculino" para homens e "feminino" para mulheres;
- opinião com relação ao produto: 1: péssimo, 2: ruim, 3: regular, 4: bom, 5: ótimo.

E em seguida exibir os seguintes resultados:

- A quantidade de mulheres maiores de 20 anos indicaram o produto como bom;
- A quantidade de mulheres maiores de 30 anos indicaram o produto como ruim;
- A quantidade de homens que indicaram o produto como péssimo;
- O total de homens que participaram da pesquisa.
- O total de mulheres que participaram da pesquisa.

Obs.: O programa deve processar os dados em tempo real, finalizado quando o usuário digitar -1 para a idade. (Não usar listas)