Estruturas de Repetição

Conteudista: Prof. Me. Hugo Batista Fernandes
Revisão Textual: Esp. Jéssica Dante

Objetivo da Unidade:

- Estudar os conceitos e a sintaxe de estruturas de repetição, bem como exemplos de suas aplicações.

Material Teórico

Estruturas de Repetição (*Loop*)

Em programação de computadores, uma estrutura de repetição (*loop*) é uma sequência de instruções que é continuamente repetida até que uma determinada condição seja alcançada. *Loops* estão entre os mais básicos e poderosos conceitos de programação.

Os programadores usam *loops* para percorrer valores, somar números, repetir funções ou instruções de códigos e muitas outras coisas.

Estruturas de repetição (*loop*) são semelhantes às estruturas condicionais (*if, else* etc.), contudo, ao contrário de uma instrução *if*, que apenas avalia uma condição uma única vez, um *loop* será executado várias vezes até que a condição retorne um valor falso.

Loops são suportados por todas as linguagens de programação modernas, embora suas implementações e sintaxe possam ser diferentes. Em Python temos dois tipos de estruturas de repetição: *for* e *while*.

- ***for:*** é um loop que é executado por um número predefinido de vezes;
- ***while:*** é um loop que é repetido enquanto uma expressão for verdadeira.

Leitura

Loop for – Estruturas de Repetição em *Python*

Clique no botão para conferir o conteúdo.

ACESSE

Estrutura de Repetição *for*

Estruturas de repetição do tipo ***for*** são excelentes quando temos um número predefinido de repetições, ou seja, quando temos um número finito de repetições a serem executadas.

A sintaxe para a estrutura *for* é:

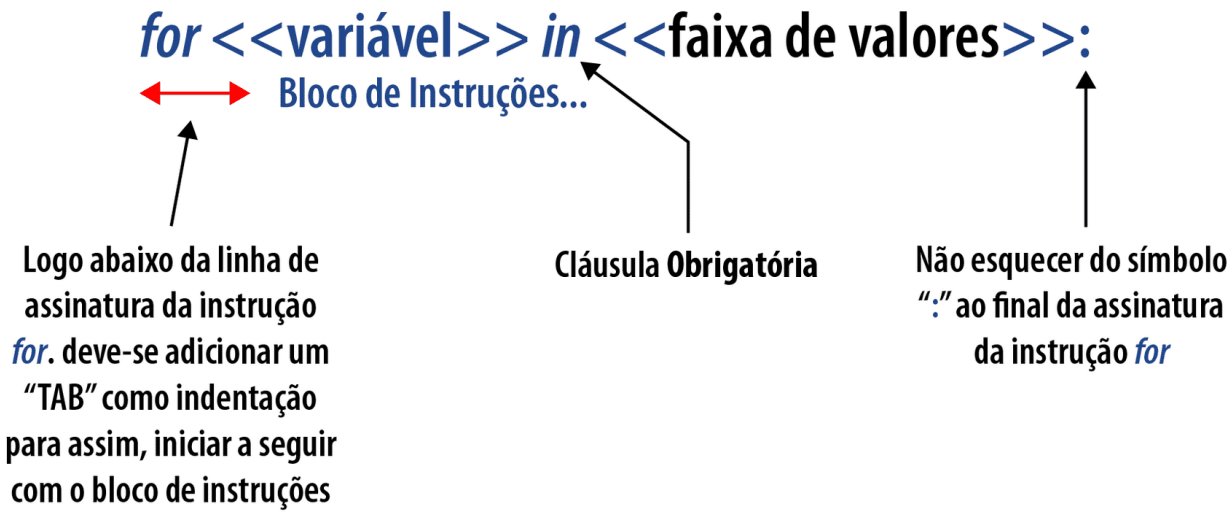

Figura 1

- **Variável**: refere-se à variável de iteração. É por meio dessa variável que identificamos a sequência de repetição durante a execução do *Loop*. Essa variável irá iterar de acordo com a faixa de valores declarada;

- **Faixa de valores**: esse parâmetro corresponde à faixa de valores ou quantidade de repetições que a estrutura irá executar. Esse parâmetro pode ser uma *string*, uma lista, um dicionário, uma tupla ou um objeto que permita iterações.

Por exemplo, imaginemos um cenário onde desejamos que nosso programa faça uma sequência progressiva de cinco (5) adições, acrescentando sempre 3 ao resultado da soma anterior. Em Python temos algo do tipo:

Código:

```
1  soma = 0
2  for i in (0,1,2,3,4):
3      soma = soma +3
4      print(soma)
5
```

Figura 2
Fonte: Acervo do Conteudista

Saída:

Figura 3
Fonte: Acervo do Conteudista

Explicando o Código

- **Linhas 1**: declaramos uma variável com o nome de “soma” e atribuímos o valor “0” para ela;

- **Linha 2**: assinatura da instrução for. Como variável para iteração, declaramos a variável "i", em seguida declaramos a cláusula obrigatória ***in*** e por fim, declaramos a sequência numérica a ser percorrida. Nesse caso, entre 0 e 4;

- **Linha 3**: a cada repetição, a variável "soma" acumulará o valor contido anteriormente mais (+) o valor 3. Dessa forma, ao final de 5 repetições, teremos uma progressão aritmética;

- **Linha 4**: após atualização do valor contido na variável "soma", utilizamos a função *print* para exibir o resultado na tela.

É importante ressaltar que a instrução *for*, no Python, irá percorrer por todos a sequência declarada no parâmetro de faixa de valores, assim, uma *string* com 5 caracteres fará com que nosso for seja executado 5 vezes, em uma sequência com 10 números, o *for* será executado 10 vezes, e assim por diante. Vejamos o exemplo a seguir, onde temos como parâmetro para a faixa de valores, 6 valores aleatórios.

```
1 for i in (2,5,9,56,4,58):
2     print(i)
3
```

Figura 4
Fonte: Acervo do Conteudista

Saída:

Figura 5
Fonte: Acervo do Conteudista

Explicando o Código

- **Linha 1**: assinatura da instrução *for*. Como variável para iteração, declaramos a variável “i”, em seguida declaramos a cláusula obrigatória ***in*** e por fim, declaramos a sequência numérica de seis números (aleatória) a ser percorrida;

- **Linha 2**: a cada repetição, utilizamos a função *print* para exibir na tela o valor contido na variável de iteração “i”. É importante ressaltar que como a faixa de valores possui 6 números, o *for* executará 6 repetições.

Comumente, para uma sequência finita, é utilizado em *Python* a função *range*, uma função que gera uma sequência numérica. A sintaxe dessa função é:

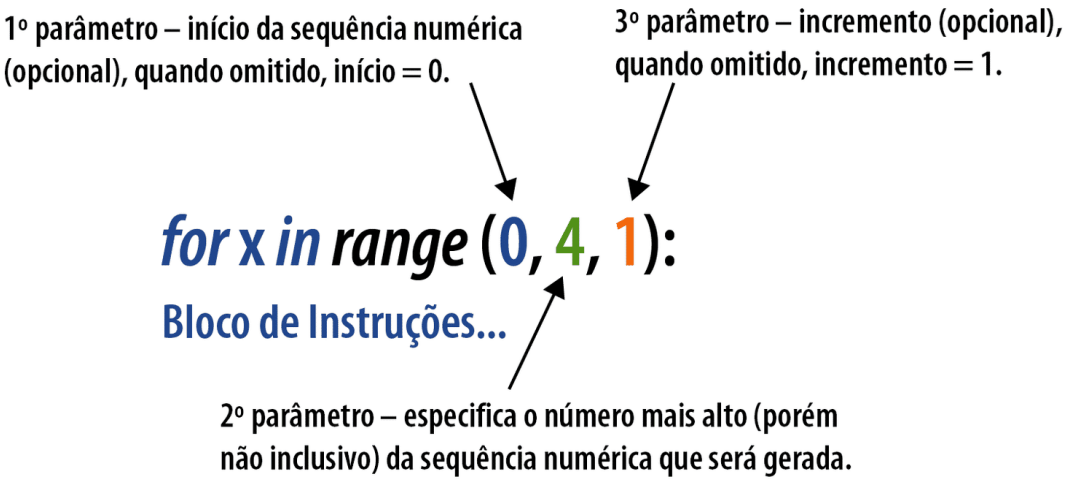

Figura 6

A seguir, temos um código que exibe na tela uma contagem progressiva de 5 números entre 0 e 4.

Código:

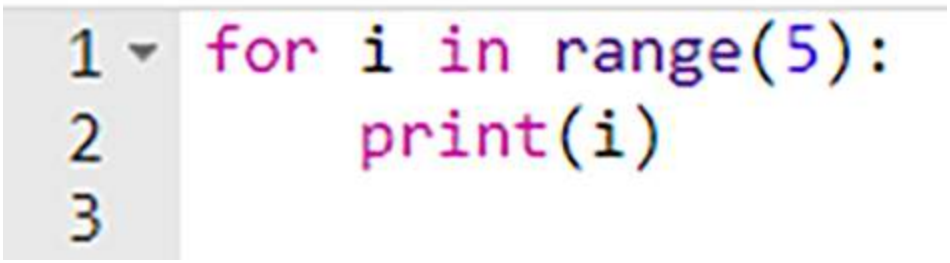

Figura 7
Fonte: Acervo do Conteudista

Saída:

Figura 8
Fonte: Acervo do Conteudista

Explicando o Código

- **Linha 1**: assinatura da instrução *for*. Como variável para iteração, declaramos a variável “i”. Para a função “*range*”, declaramos que o fim da sequência será o número 5, contudo, como explicado, esse número não é incluso na sequência;

- **Linha 2**: a cada repetição, utilizamos a função *print* para exibir na tela o valor contido na variável de iteração “i”. É importante ressaltar que conforme o parâmetro da função “*range*” (5), será criada uma sequência entre 0 e 4 e assim, o *for* executará 5 repetições.

Vejamos outro exemplo, criaremos agora um programa que exibirá uma progressão numérica de zero (0) à quinze (15) tendo como incremento o valor 2. Ou seja, nosso programa irá exibir uma progressão aritmética de razão 2. Para tanto, iremos usar a estrutura ***for*** com a cláusula *range* configurando o parâmetro de incremento para 2.

Codigo:

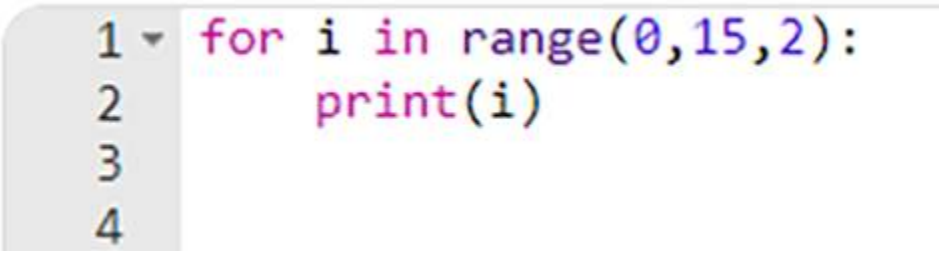

Figura 9
Fonte: Acervo do Conteudista

Saída:

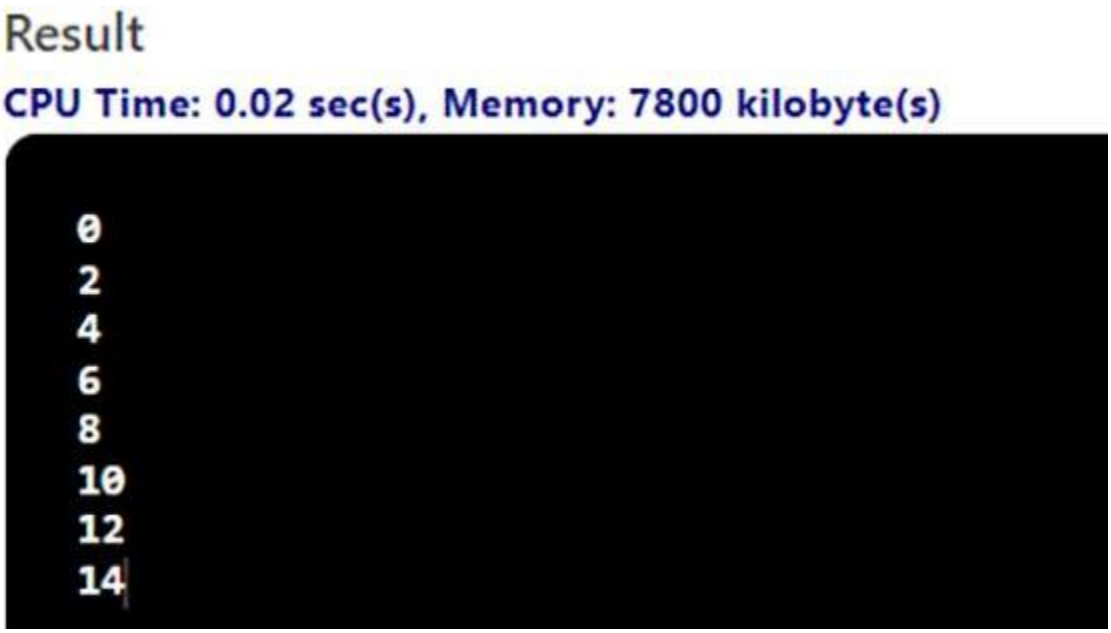

Figura 10
Fonte: Acervo do Conteudista

Explicando o Código

- **Linha 1**: assinatura da instrução *for*. Como variável para iteração, declaramos a variável “i”. Para a função “*range*”, declaramos que o fim da sequência será o número 15 (segundo parâmetro), contudo, como explicado, esse número não é incluso na sequência. Diferente do exemplo anterior, configuramos o incremento (terceiro parâmetro) para o valor 2. Assim, a cada repetição do *for*, o valor de “i” será somado com o valor +2;

- **Linha 2**: a cada repetição, utilizamos a função *print* para exibir na tela o valor contido na variável de iteração “i”. É importante ressaltar que conforme o parâmetro da função “*range*” (15), será criada uma sequência entre 0 e 14, porém, como o incremento está configurado para somar o valor +2 a cada repetição, serão executadas a quantidade de repetições possíveis entre 0 e 14 em uma progressão de múltiplos de 2. Assim, temos 8 repetições, que é a quantidade de números múltiplos de 2 dentro da sequência de 0 a 14.

Leitura

Progressão Aritmética

Clique no botão para conferir o conteúdo.

ACESSE

É importante destacar que dentro de um bloco de instruções em uma estrutura de repetição, podemos, por exemplo, utilizar estruturas de decisão. Vejamos um exemplo. Devemos criar um programa que dado uma sequência numérica progressiva (de 0 a 30), identifique quais números dessa sequência são múltiplos de 3.

Código:

```
1  qtdMultiplos = 0
2  for i in range(1,30):
3      if(i%3==0):
4          qtdMultiplos+=1
5          print(i)
6
7  print("A quantidade de números divisíveis por 3 na sequência de 1 a 30 é",qtdMultiplos )
8
```

Figura 11
Fonte: Acervo do Conteudista

Saída:

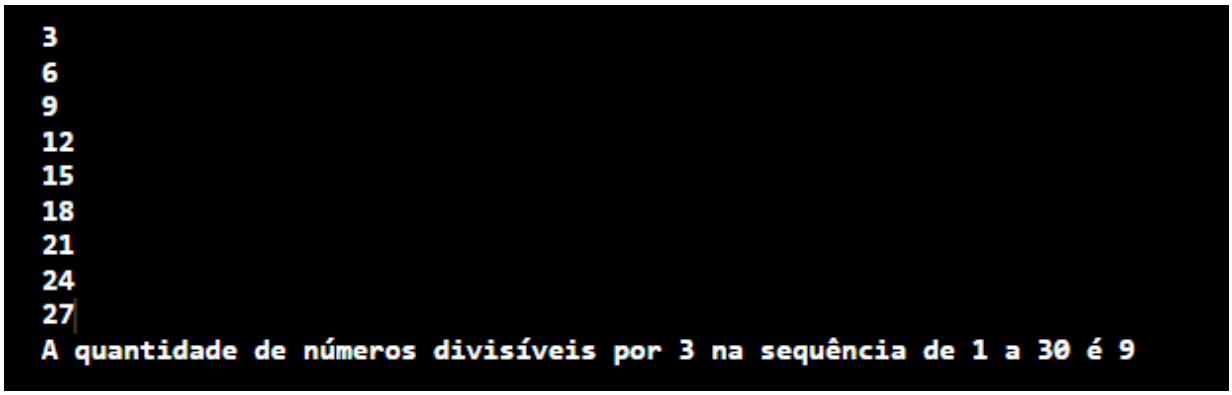

Figura 12
Fonte: Acervo do Conteudista

Explicando o Código

- **Linha 1**: declaramos uma variável com o nome de "qtdMultiplos" e atribuímos o valor "0" para ela;

- **Linha 2**: assinatura da instrução for. Como variável para iteração, declaramos a variável "i", em seguida, por meio da função "*range*", configuramos uma sequência numérica até o número 30 a partir do número 1;

- **Linha 3**: a cada repetição, essa linha é executada, e dessa forma, é testada a condição. Se o número atual da sequência (contido na variável "i") dividido por três, resultar seu **resto** (por isso o operador de módulo) em zero, o número é divisível por 3 e assim, as linhas 4 e 5 serão executadas;

- **Linha 4**: ao ser executada, essa linha adiciona "+1" para o valor atribuído à variável "qtdMultiplos";

- **Linha 5**: utilizamos a função *print* para exibir na tela o valor contido na variável de iteração "i";

- **Linha 7**: ao final do *for*, ou seja, após concluir todas as repetições da estrutura, essa linha será executada. Utilizamos a função *print* para exibir na tela o valor contido na variável "qtdMultiplos".

Por meio da estrutura *for*, podemos, por exemplo, percorrer todos os caracteres contidos em uma variável com valor do tipo *string*. Vejamos um exemplo.

Código:

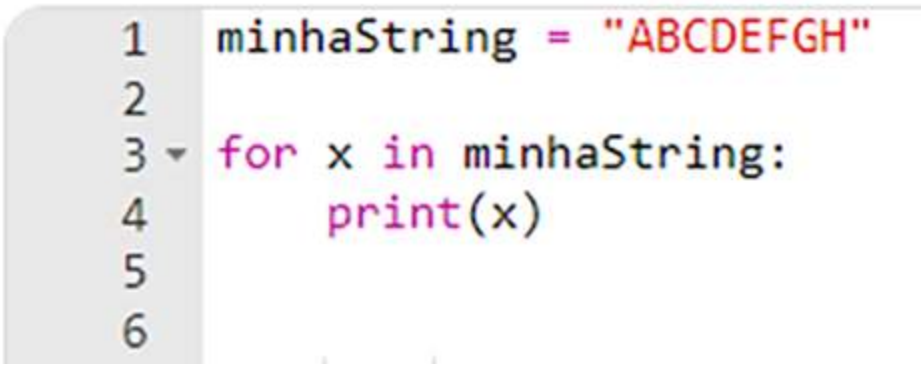

Figura 13
Fonte: Acervo do Conteudista

Saída:

Figura 14
Fonte: Acervo do Conteudista

Explicando o Código

Temos na linha 1 a declaração da variável e a atribuição de um valor do tipo *string*. Na linha 3, temos a assinatura da instrução *for*, como variável para iteração, declaramos a variável "x", em seguida, logo após a cláusula ***in***, descrevemos a variável "minhaString". Por ser um tipo *string*, a estrutura *for* é capaz de percorrer toda a cadeia de caracteres que formam o valor *string* e dessa forma, um a um, cada caractere é acessado e visualizado na tela por meio da função *print*.

Estrutura de Repetição *while*

Estruturas de repetição do tipo *while* são utilizadas quando em nosso código devemos repetir um trecho de código ou instruções. Estruturas de repetição *while* são criadas escrevendo a palavra-chave *while*, seguida de uma condição (como em uma instrução *if*) e, em seguida, do código que deseja executar.

A sintaxe para a estrutura *while* é:

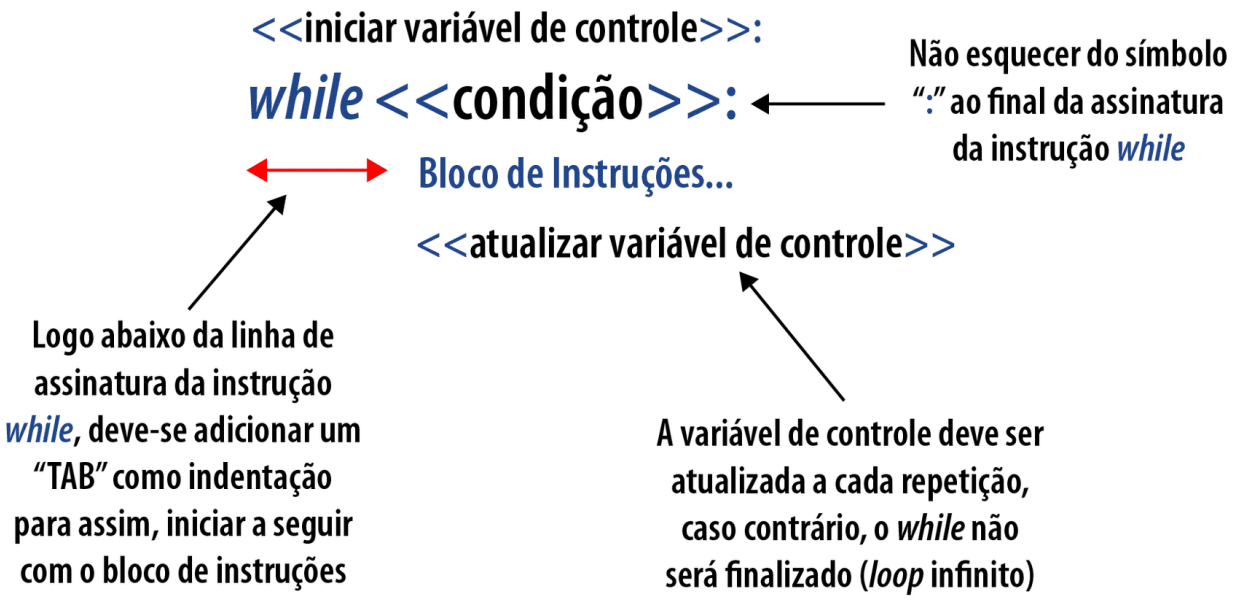

Figura 15
Fonte: Acervo do Conteudista

Para o início de nossos estudos com a estrutura *while*, vamos criar um programa que exibe na tela uma contagem progressiva de 1 a 5. Assim, temos o seguinte código.

Código:

```
1  contador = 0
2  while (contador < 5):
3      contador+=1
4      print(contador)
5
```

Figura 16
Fonte: Acervo do Conteudista

Saída:

Figura 17
Fonte: Acervo do Conteudista

Explicando o Código

- **Linha 1**: declaramos uma variável com o nome de "contador" e atribuímos o valor "0" para ela. Essa será a variável para iteração, será utilizada como **variável de controle**;

- **Linha 2**: assinatura da instrução *while*. Declaramos a condição (contador < 5). Essa condição faz com que a instrução *while*

continue a executar enquanto o valor contido na variável "contador" seja um número menor que 5;

- **Linha 3**: a cada repetição, a variável "contador" acumulará o valor contido anteriormente mais (+) o valor 1. Dessa forma, ao final de 5 repetições, teremos uma progressão aritmética;

- **Linha 4**: após atualização do valor contido na variável "contador", utilizamos a função *print* para exibir o resultado na tela.

Em muitos cenários, o algoritmo exige que se repita um certo trecho de código, contudo sem a informação sobre em qual momento a repetição será finalizada. Desse modo, é escrito um código onde dada uma condição, o processo de repetição é finalizado. Vejamos um exemplo. Temos um cenário onde nosso programa deve solicitar que o usuário digite um nome até que seja digitado o número zero (0). Seguimos com o código em *Python*.

Código:

```
1  nome = ""
2  while True:
3      texto = input("Digite um nome ou '0' para encerrar o programa ")
4
5      if(texto == "0"):
6          print("Programa finalizado")
7          break
8      else:
9          nome = nome + texto +"\n"
10
11 print(nome)
12
```

Figura 18
Fonte: Acervo do Conteudista

Explicando o Código

- **Linha 1**: declaramos uma variável com o nome de "nome" e atribuímos o valor " ", ou seja, vazio. Essa será a variável que irá armazenar os nomes digitados pelo usuário do programa;

- **Linha 2**: assinatura da instrução *while*. Diferente do *while* anterior, não é declarada uma condição com algum operador lógico, apenas a palavra reservada "*True*", dessa forma, forçamos o *while* a continuar executando de forma infinita ou até que alguma ação faça o *while* finalizar;

- **Linha 3**: a cada repetição, é solicitado ao usuário do programa que digite um nome ou o número zero. O valor digitado é atribuído para a variável "texto";

- **Linha 5**: caso o valor contido na variável "texto" seja igual a zero (0), as linhas 6 e 7 serão executadas. Ressaltamos aqui que a instrução "***break***" quando executada em um *while*, executará a parada desse *loop;*

- **Linha 9**: essa linha será executada enquanto o usuário do programa digitar textos diferentes de zero (0). Ao ser executada, essa linha armazenará na variável "nome" o valor digitado adicionando com o valor contido nessa variável anteriormente. Por fim, adiciona um caractere de quebra de linha "\n";

- **Linha 11**: essa linha será executada quando o *while* encerrar, ou seja, quando o usuário digitar zero (0). Utilizamos a função *print* para exibir na tela os valores contidos na variável "nome".

Podemos criar uma outra solução para o problema anterior utilizando uma variável do tipo *Boolean* como condição do *While*, mas diferente do código anterior, essa variável

pode mudar de estado (de verdadeiro para falso), assim, quando diferente de verdadeiro, irá encerrar o programa.

Código:

```
1  nome = ""
2  continuar = True
3  while continuar:
4      nome = nome + input("Digite um nome ") + "\n"
5
6      x = input("Deseja continuar? Digite 'Sim' ou 'Não' ")
7
8      if(x.upper() =="SIM"):
9          continuar = True
10     else:
11         continuar = False
12
13 print(nome)
14
```

Figura 19
Fonte: Acervo do Conteudista

Explicando o Código

- **Linha 1**: declaramos uma variável com o nome de “nome” e atribuímos o valor “ ”, ou seja, vazio. Essa será a variável que irá armazenar os nomes digitados pelo usuário do programa;

- **Linha 2**: declaramos uma variável com o nome de “continuar” e atribuímos o valor “*True*”. Essa será a variável que iremos utilizar como condição para que a estrutura *while* continue a executar repetições;

- **Linha 3**: assinatura da instrução *while*. Utilizamos a variável "continuar" como condição. Enquanto o valor dessa variável for verdadeiro (*True*), o *while* irá executar repetições;

- **Linha 4**: a cada repetição, é solicitado ao usuário do programa que digite um nome. Ao ser executada, essa linha armazenará na variável "nome" o valor digitado juntando com o valor contido nessa variável anteriormente. Por fim, adiciona um caractere de quebra de linha "\n";

- **Linha 6**: é solicitado ao usuário do programa que digite "Sim" ou "Não" para expressar o desejo de continuar a executar o programa;

- **Linha 8**: com a estrutura de decisão, testamos a condição: Se o valor contido na variável "x" for igual à "SIM" (note o texto como um valor em maiúsculo), será atribuído o valor "*True*" para a variável "continuar", caso contrário, será atribuído o valor "*False*". É importante ressaltar que garantimos que o valor contido em "x" seja um texto em maiúsculo utilizando a função "*upper*()". Por meio dessa sequência de instruções mudamos (ou mantemos) o estado da variável "continuar" e dessa forma, controlamos o *while* acerca de sua execução;

- **Linha 13**: essa linha será executada quando o *while* encerrar, ou seja, quando o usuário digitar um valor diferente de "SIM". Utilizamos a função *print* para exibir na tela os valores contidos na variável "nome".

Vídeo

Aula Python: ensinando funções básicas com *strings* com *upper lower*

title

Em alguns cenários é preciso não permitir que alguns trechos de código dentro da estrutura de repetição sejam executados. Porém, garantindo que a estrutura continue sendo executada, assim temos a cláusula *continue*. Dentro de uma estrutura de repetição, ao ser executada, essa cláusula fará com que seja executada a primeira linha da estrutura de repetição, ignorando as demais linhas de códigos seguintes.

Vejamos um exemplo. Criaremos um programa que exibe uma sequência de números entre zero (0) e nove (9), porém, nosso programa não deve exibir os números 1 e 3.

Código:

```
1  contador = 0
2
3  while contador < 10:
4      if(contador == 1 or contador == 3):
5          contador+=1
6          continue
7
8      print(contador)
9      contador+=1
10
```

Figura 20
Fonte: Acervo do Conteudista

Saída:

Result

executed in 0.838 sec(s)

```
0
2
4
5
6
7
8
9
```

Figura 21

Fonte: Acervo do Conteudista

Explicando o Código

- **Linha 1**: declaramos uma variável com o nome de “contador” e atribuímos o valor “0” para ela. Essa será a variável para iteração, será utilizada como **variável de controle;**

- **Linha 2**: assinatura da instrução *while*. Declaramos a condição (contador < 10). Essa condição faz com que a instrução *while* continue a executar enquanto o valor contido na variável “contador” seja um número menor que 10;

- **Linha 4**: por meio da estrutura de decisão, o programa verifica se o valor contido na variável “contador” é igual a “1” ou “3”. Caso verdadeiro, para essas duas situações, serão executadas as linhas 5 e 6;

- **Linha 5**: cada vez que essa linha é executada é adicionando +1 ao valor contido na variável “contador”;

- **Linha 6**: cada vez que essa linha é executada, a cláusula continue executa a primeira linha da estrutura de repetição. Assim, as linhas 8 e 9 não serão executadas;

- **Linha 8**: ao ser executada, essa linha exibe por meio da função *print*, o valor contido na variável “contador”;

- **Linha 9**: é adicionando +1 ao valor contido na variável “contador”.

Vídeo

Repetição com *While* + *Break* e *Continue*

Exercício

Escreva um programa que leia um grupo de valores (não sabemos quantos são) para calcular e visualizar a média desses valores e,

também, determinar e visualizar o maior deles. Utilize uma estrutura de repetição *while* ou *for*.

Importante!

Antes que leia a resposta para o desafio proposto, tente criar sua própria solução. Como sugestão, desenvolva primeiro o algoritmo, pode ser em pseudocódigo ou um fluxograma, o importante aqui é pensar no passo a passo para a resolução do problema, em seguida, implemente seu algoritmo na linguagem Python.

CONTINUE

Resolução

Seguimos com o seguinte código em *Python* para a solução do desafio proposto.

```
1  qtdvalores = int(input("Digite a quantidade de números que serão calculados "))
2  contador = 0
3  valor = 0
4  while contador < qtdvalores:
5      valor += float(input("Digite um valor "))
6      contador+=1
7
8  media = valor/ contador
9  print(media)
10
```

Figura 22
Fonte: Acervo do Conteudista

Leitura

Jdoodle – Online Python 3 IDE

O código anterior pode ser visualizado por meio do *link* a seguir.

Clique no botão para conferir o conteúdo.

ACESSE

Explicando o Código

- **Linha 1**: declaramos uma variável com o nome de "qtdvalores" e utilizamos a instrução ***input*** para

solicitar ao usuário do programa a quantidade de números que serão calculados. Utilizamos a instrução ***int*** para converter o tipo *string* para inteiro;

- **Linha 2**: declaramos uma variável com o nome de "contador" e atribuímos o valor "0" para ela. Essa será a variável para iteração, será utilizada como **variável de controle;**
- **Linha 3**: declaramos uma variável com o nome de "valor" e atribuímos o valor "0" para ela. Iremos utilizar essa variável para armazenar os valores digitados pelo usuário do programa;
- **Linha 4**: assinatura da instrução *while*. A condição para continuar a repetir a instrução *while* é de que o valor contido na variável "contador" seja menor que o valor contido na variável "qtdvalores", enquanto essa proposição for verdadeira, o *while* continuará a executar;
- **Linha 5**: a cada repetição, é solicitado ao usuário do programa que digite um valor. O valor digitado é atribuído para a variável "valor" adicionando (operação aritmética de adição +) com o valor contido anteriormente. Utilizamos a instrução *float* para converter *string* em *float;*
- **Linha 6**: a cada repetição, a variável "contador" acumulará o valor contido anteriormente mais (+) o valor 1;
- **Linha 8**: ao encerrar o *while*, é realizado o cálculo da média. Para isso é realizado a divisão entre o valor contido na variável "valor" e a variável "contador";
- **Linha 9**: utilizamos a função *print* para exibir na tela o valor contido na variável "media".

Leitura

Jdoodle – Online Python 3 IDE

Segue também a solução do desafio utilizando a estruturar *for*.

Clique no botão para conferir o conteúdo.

ACESSE

Em Síntese

Nesta Unidade, estudamos os conceitos e a implementação de estruturas de repetição. É importante que assista a videoaula desta unidade e que leia os livros e materiais complementares indicados nesta Unidade de estudo. É fundamental que além dos estudos em *Python*, busque estudar ou retomar conceitos de desenvolvimento de algoritmos, em especial, o tema desta unidade.

Até a próxima.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Vídeos

Estrutura de Repetição *While* em *Python*

For / In – Estrutura de Repetição em *Python*

Leitura

Python While: Executar Código com Condição Verdadeira

Instrução *while*.

Clique no botão para conferir o conteúdo.

ACESSE

Python For: Usando *Loop* com essa Estrutura de Repetição

Instrução *for*.

Clique no botão para conferir o conteúdo.

ACESSE

Referências

BANIN, S. L. ***Python* 3**: conceitos e aplicações: uma abordagem didática. São Paulo: Érica, 2018. (*e-book*)

PERKOVIC, L. **Introdução à computação usando *Python***: um foco no desenvolvimento de aplicações. São Paulo: LTC Editora, 2016. (*e-book*)

WAZLAWICK, R. S. **Introdução a algoritmos e programação com *Python***: uma abordagem dirigida por testes. São Paulo: LTC Editora, 2017. (*e-book*)