



python™

Aula 04 – Laços de Repetição

UNIP - CIÊNCIA DA COMPUTAÇÃO

DISCIPLINA: IPE

PROFESSORES:

CÉLIO E LUCIANA

Laço While (ENQUANTO)

Permite executar um bloco de código ENQUANTO uma determinada condição ainda for VERDADEIRA.

Exemplo (python):

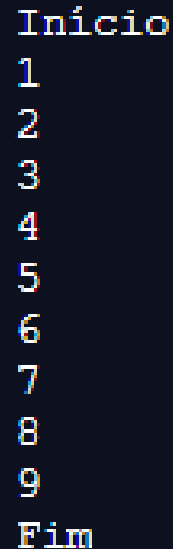
```
x = 1

print('Início')

while x < 10:
    print(x)
    x = x + 1

print('Fim')
```

Resultado:



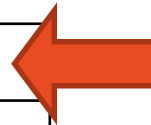
```
Início
1
2
3
4
5
6
7
8
9
Fim
```

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |



Executa a linha 1, criando variável em memória e atribuindo valor

| Variável Memória | Valor |
|------------------|-------|
| x | 7 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |



Executa a linha 2, printando mensagem na tela

| Variável Memória | Valor |
|---------------------|-------|
| x | 7 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |

Executa a linha 3. Nessa linha, já é verificado se a condição do ENQUANTO é verdadeira:
`x < 10???`

Se VERDADEIRA, entra no laço (vai para a linha 4)
Se FALSO, não entra no laço (vai para a linha 6)

| Variável Memória | Valor |
|------------------|-------|
| x | 7 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |

No caso desse programa $x < 10$ ($7 < 10$), ou seja, a condição é VERDADEIRA.

Assim, entra no laço e executamos a linha 4.

| Variável Memória | Valor |
|------------------|-------|
| x | 7 |

| | |
|--------------|---|
| Saída (seq.) | 7 |
|--------------|---|

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |

Executamos a linha 5, atualizando o valor de x em memória.


| Variável Memória | Valor |
|------------------|-------|
| x | 8 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |



Chegamos no final dos comandos que estão dentro do laço.

Assim, voltamos para a linha 3 e verificamos novamente se a condição ainda é verdadeira.

X < 10??? ... ou seja, 8 < 10???

| Variável Memória | Valor |
|---------------------|-------|
| x | 8 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |

No caso, a condição **AINDA** é **VERDADEIRA**.

Executamos novamente a linha 4, que agora irá imprimir o valor 8 (e não mais 7 como anteriormente).

| Variável Memória | Valor |
|------------------|-------|
| x | 8 |
| Saída (seq.) | 7 |
| | 8 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |

Executamos a linha 5, atualizando o valor de x em memória.


| Variável Memória | Valor |
|------------------|-------|
| x | 9 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |



Chegamos NOVAMENTE no final dos comandos que estão dentro do laço.

Assim, voltamos para a linha 3 e verificamos novamente se a condição ainda é verdadeira.

X < 10??? ... ou seja, 9 < 10???

| Variável Memória | Valor |
|---------------------|-------|
| x | 9 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |

Sim, a condição **AINDA** é **VERDADEIRA**.

Vamos executar novamente a linha 4, que agora irá imprimir o valor 9.

| Variável Memória | Valor |
|------------------|-------|
| x | 9 |

| | |
|--------------|---|
| Saída (seq.) | 7 |
| | 8 |
| | 9 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |

Executamos a linha 5, atualizando o valor de x em memória.

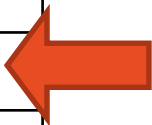
| Variável Memória | Valor |
|------------------|-------|
| x | 10 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |



E adivinha? Pois é, chegamos NOVAMENTE ao final dos comandos que estão dentro do laço.

Vamos voltar para a linha 3 e verificar se a condição ainda é verdadeira.

`X < 10???` ... ou seja, `10 < 10???`

| Variável Memória | Valor |
|---------------------|-------|
| x | 10 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-----------------|
| 1. | x = 7 |
| 2. | print('Início') |
| 3. | while x < 10: |
| 4. | print(x) |
| 5. | x = x + 1 |
| 6. | print('Fim') |

Agora MUDOU!!!! A condição é **FALSA!!!**

X < 10 10 < 10 ← FALSO!!!!

Sendo **FALSO**, agora iremos **SAIR** do laço!!!!


| Variável Memória | Valor |
|------------------|-------|
| x | 10 |
| Saída (seq.) | 7 |
| | 8 |
| | 9 |

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |



Por FIM, vamos executar a linha 6 e imprimir no console a última mensagem ('FIM').


Depois o programa termina pois não temos mais linhas para executar.

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |



Agora a pergunta que não quer calar:

Qual era mesmo o último valor de X (na memória) após a execução do laço????

- a) 8
- b) 9
- c) 10
- d) 11

While – Sequência de execução

Quando usamos o Enquanto, precisamos tomar alguns cuidados. O 1º ponto de atenção é TER CERTEZA que entrará no enquanto.

Código:

| | |
|----|-------------------------------------|
| 1. | <code>x = 7</code> |
| 2. | <code>print('Início')</code> |
| 3. | <code>while x < 10:</code> |
| 4. | <code> print(x)</code> |
| 5. | <code> x = x + 1</code> |
| 6. | <code>print('Fim')</code> |
| 7. | <code>print(x) #imprimirá 10</code> |

Agora a pergunta que não quer calar:

Qual era mesmo o último valor de X (na memória) após a execução do laço????

- a) 8
- b) 9
- c) 10
- d) 11

| Variável Memória | Valor |
|------------------|-------|
| x | 10 |

Cuidados com Laço While

Exemplo 1 – NÃO ENTRARÁ NO LAÇO (condição FALSA já na primeira verificação):

```
x = 10
```

```
print('Início')
```

```
while x < 10:  
    print(x)  
    x = x + 1
```

```
print('Fim')
```



Comandos BREAK, CONTINUE e PASS

Comando `break`

Interrompe a execução do laço que o envolve.

```
x = 1

while x < 10:

    print(x)

    if x == 5:
        break

    x = x + 1
```



```
1
2
3
4
5
█
```

Comandos BREAK, CONTINUE e PASS

Comando `continue`

Força a próxima iteração do laço que o envolve, ou seja, vai direto para a próxima execução do mesmo ignorando a atual.

```
x = 1

while x < 10:

    x = x + 1

    if x <= 5:
        continue

    print(x)
```



```
6
7
8
9
10
➤ □
```

Comandos BREAK, CONTINUE e PASS

Comando `pass`

Comando vazio, quando queremos que nada seja feito.

Pode ser utilizado para forçar o processamento de um laço (mais utilizado no FOR) ou prepara-lo para futuras manutenções.

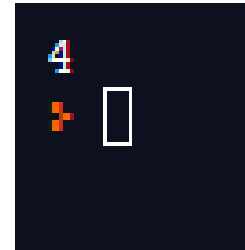
```
x = 1
while x < 10:
    if x <= 5:
        # Logo Logo será implementado
        # algo aqui
        pass
    else:
        print('print else')
    print(x)
    x = x + 1
```

```
1
2
3
4
5
print else
6
print else
7
print else
8
print else
9
:
```

Comandos BREAK, CONTINUE e PASS

Comando `pass` – exemplo com FOR

```
x = 0  
  
for x in [1, 6, 3, 1, 4]:  
    pass  
  
print(x)
```



Laço *for* (PARA) – Como é em algoritmos e em algumas linguagens (ex. Java)

Na disciplina de Algoritmos vimos que o laço PARA (*for*) serve para repetir um bloco de comandos usando uma variável de controle.

Exemplo em Portugol:

```
Para i de 1 ate 10 passo 1 faça  
    escreva (“Variável: “ + i)  
fimpara
```

Em algumas linguagens o código é bastante semelhante, observe o mesmo exemplo escrito em Java:

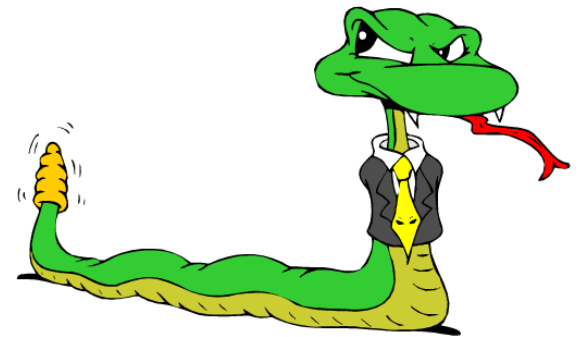
```
for (i = 1; i <= 10; i = i + 1) {  
    System.out.println(“Variável: “ + i);  
}
```

Laço *for* no Python

Entretanto, esse comando no Python é “diferentão”.

No slide anterior vimos que a variável de controle é incrementada até chegar em um valor (Portugol, Java, C++, C# etc.)

No [python](#) a variável de controle irá iterar sobre uma sequencia de itens/valores que iremos definir.





Laço *for* no Python

- Exemplo FOR (algoritmos/Portugol):

```
Para i de 1 ate 10 faça  
    escreva (“Variável: “ + i)  
fimpara
```

- Exemplos do mesmo laço em **python**:

```
for i in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:  
    print('Variável: ', i)
```

ou

```
for i in range(1, 11):  
    print('Variável: ', i)
```

```
Variável: 1  
Variável: 2  
Variável: 3  
Variável: 4  
Variável: 5  
Variável: 6  
Variável: 7  
Variável: 8  
Variável: 9  
Variável: 10  
> █
```

Criação de Sequências – Definindo Manualmente

- No python existem vários tipos de sequencias (listas, tuplas, dicionários, etc.). Mas **NÃO É HORA** de vermos todos eles agora.
- Nesse momento, vamos definir manualmente uma sequencia no formato de uma listagem simples. *Repare* também nos resultados:

```
for i in [2, 5, 3, 4]:  
    print(i)
```

```
2  
5  
3  
4  
➤
```

```
for c in ['p', 'y', 't', 'h', 'o', 'n']:  
    print(c)
```

```
P  
y  
t  
h  
o  
n
```

Gerando Sequências – range()

- Podemos gerar sequencias com o método range()

Exemplos:

Se vou utilizar a sequencia apenas p/ o *FOR*, coloco ele diretamente:

```
for i in range(1, 11):  
    print('Variável: ', i)
```

Se vou utilizar a sequencia depois p/ outros processos, coloco numa variável. É comum a conversão do range para uma lista:

```
X = list(range(1, 11))  
for i in X:  
    print('Variável: ', i)  
print(sum(X))
```

Criação de Sequências – Parâmetros do método range()

Existe três formas de se chamar o método range(), no qual diferencia-se apenas a quantidade de parâmetros:

range(**fim**) ← nesse formato, o 1º elem. será sempre zero (0)

range(**inicio**, **fim**)

range(**inicio**, **fim**, **passo**)

Inicio – 1º elemento da sequencia

Fim – 1º elemento QUE NÃO ENTRA na sequencia

Passo – Define o passo (pode ser um número negativo)



Exercícios - Quais sequencias serão geradas???

1. `range(7)` ???
2. `range(2, 6)` ???
3. `range(11, 5)` ???
4. `range(11, 30, 3)` ???
5. `range(10, 1, -2)` ???



Respostas - Quais sequencias serão geradas???

1. `range(7) ???` `[0, 1, 2, 3, 4, 5, 6]`
2. `range(2, 6) ???` `[2, 3, 4, 5]`
3. `range(11, 5) ???` `[]` *Sequencia vazia!!!*
4. `range(11, 30, 3) ???` `[11, 14, 17, 20, 23, 26, 29]`
5. `range(10, 1, -2) ???` `[10, 8, 6, 4, 2]`

Compreensão de listas (*func. avançada*)

- Uma funcionalidade bastante útil e poderosa do python.
- Gera uma nova lista aplicando uma função (ou cálculo) para cada elemento da lista original:

Sintaxe: [`açãoUsandoAvariavel` for `variavel` in `sequencia`]

Exemplos:

```
[meuNumero for meuNumero in range(5)]
```

```
[0, 1, 2, 3, 4]
```

```
[meuNumero*3 for meuNumero in range(5)]
```

```
[0, 3, 6, 9, 12]
```

```
[print(meuNumero) for meuNumero in range(2)]
```

```
0
```

```
1
```

```
□
```