



Curso de Python

27-01-2021

<https://github.com/tsbressan/CursoPythonUnisinos>

Estrutura de controle e Estrutura de Repetição

- **Estrutura de repetição (loop ou laço):**

Dois comandos utilizados para estrutura de repetição: for e while

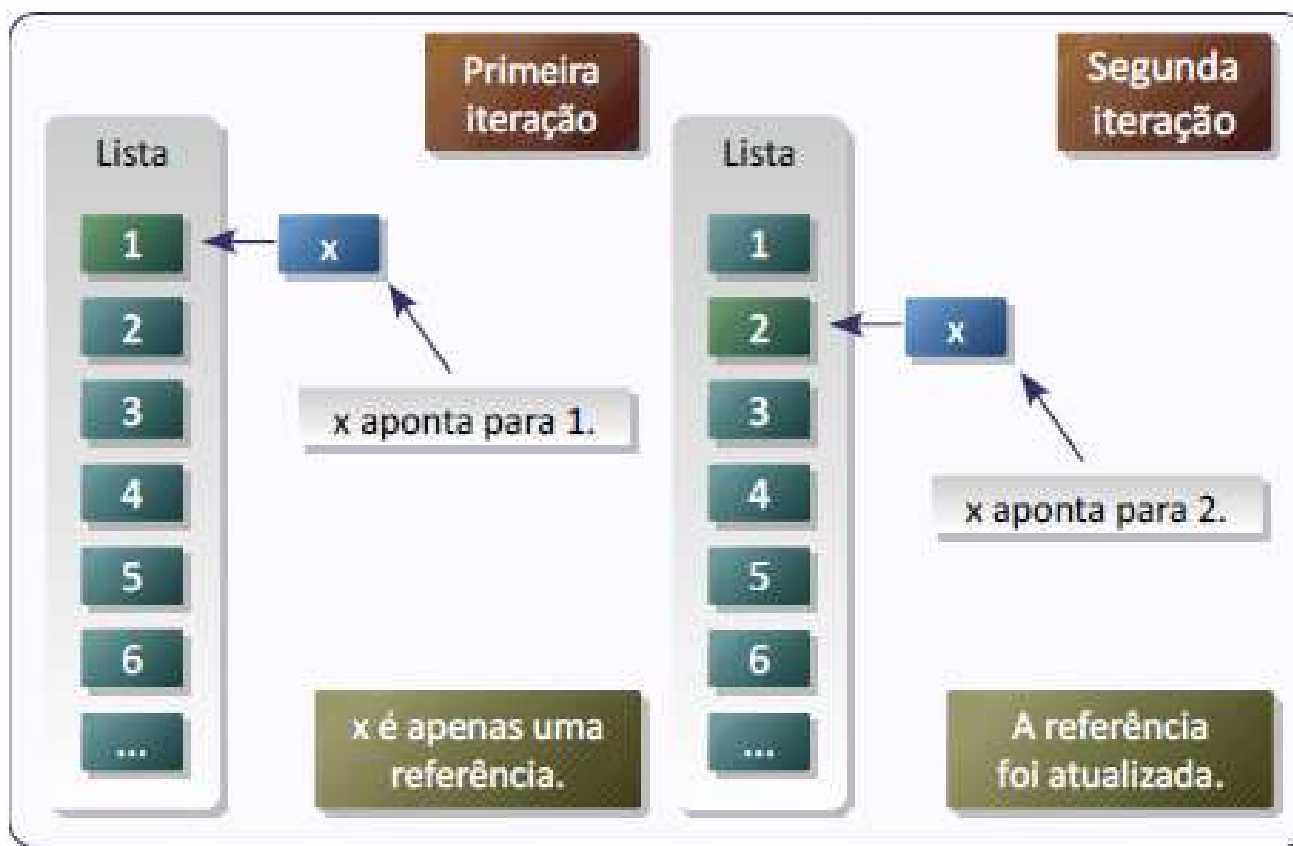
--- Utilizada quando queremos que um bloco de código seja executado várias vezes----

O for é usado quando se quer iterar sobre um bloco de código um número determinado de vezes.

O while é usando quando queremos que o bloco de código seja repetido até que uma condição seja satisfeita, neste caso, utilizando uma expressão booleana (true ou false)

Estrutura de controle e Estrutura de Repetição

- **Estrutura de repetição (loop ou laço):**



Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **for**

```
# Aqui repetimos o print 3 vezes
```

```
for n in list(range(0, 3)):
```

```
    print("Número: " , n)
```

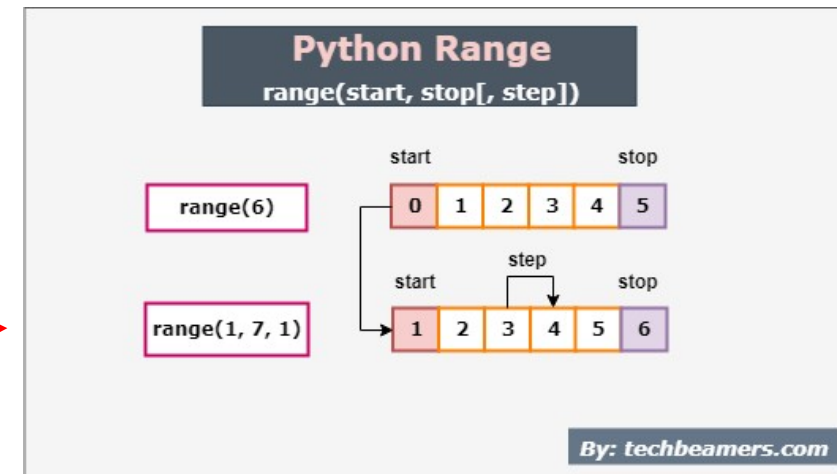
```
#####
```

```
# Para iterar(todos os elementos) sobre um vetor
```

```
v = [1, 2, 3, 4, 10]
```

```
for numero in v:
```

```
    print(numero ** 2)
```



Estrutura de controle e Estrutura de Repetição

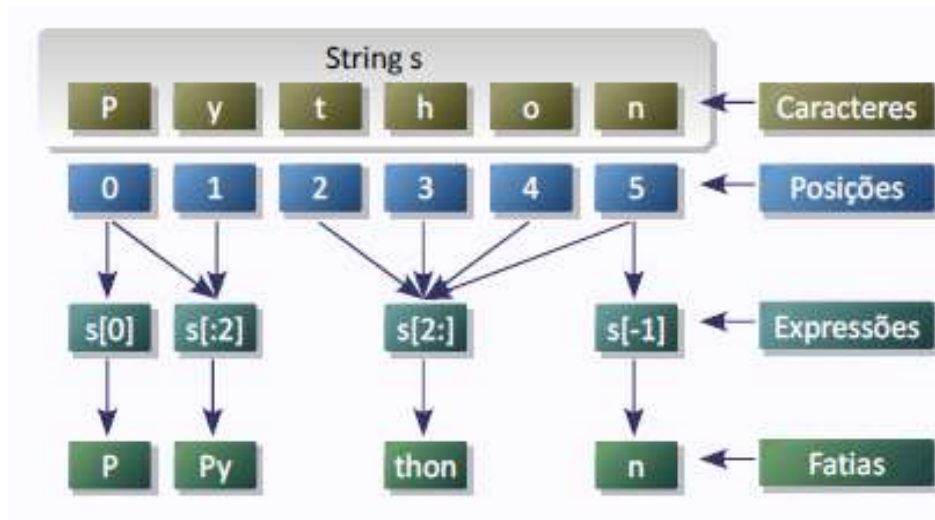
- Estrutura de repetição: **for**

Muito utilizado em strings (já usamos)

```
palavra = "unisinos"
```

```
for letra in palavra:
```

```
    print(letra)
```



IMPORTANTE: para auxiliar as estruturas de repetição, inclui-se dois comandos importantes: break e continue. break utilizado para sair da execução (loop). continue finalizar a execução daquele trecho e inicia o próximo (dentro do mesmo loop)

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **for**

```
for n in range(0,3):  
    string_digitada = input("Digite uma palavra: ")  
    if string_digitada.lower() == "quit":  
        print("Finalizou a execução!")  
        break  
    if len(string_digitada) <= 3:  
        print("Palavra muito pequena")  
        continue  
    if len(string_digitada) > 3:  
        print("Palavra digitada está correta..")
```

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **for**

Exercícios práticos de for

- 1) Mostrar todos os valores de 1 até 10
- 2) Mostrar os valores pares de 1 até 10
- 3) Encontrar o maior e o menor valor do intervalo de 1 até 10
- 4) Imprimir na tela somente os valores armazenados num vetor, onde o índice do vetor é par.

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

O while é usado quando queremos que o bloco de código seja repetido até que uma condição seja satisfeita, neste caso, utilizando uma expressão booleana (true ou false)

while <condição_lógica>:

←→ #linhas de código

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

Exemplo:

Inicia-se o n em 0, e repetimos o print até que seu valor seja maior ou igual a 3:

```
n = 0
```

```
while n <= 3:  #condição for verdadeira – teste lógico
```

```
    print(n)
```

```
    n += 1
```

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

```
n=0
```

```
n = int (input("Digite um número inteiro (-1 para sair ou quit após execução)"))
```

```
while n != -1:
```

```
    string_digitada = input("Digite uma palavra: ")
```

```
    if string_digitada.lower() == "quit":
```

```
        print("Finalizou a execução!")
```

```
        break
```

```
    if len(string_digitada) <= 3:
```

```
        print("Palavra muito pequena")
```

```
        continue
```

```
    if len(string_digitada) > 3:
```

```
        print("Palavra digitada está correta..")
```

```
print ("código finalizado")
```

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

Exemplo prático com vetor:

```
seq = []  
i = 0  
while i < 5:  
    novo_elemento = i  
    seq.append( novo_elemento )  
    i = i + 1  
  
print (seq)
```

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

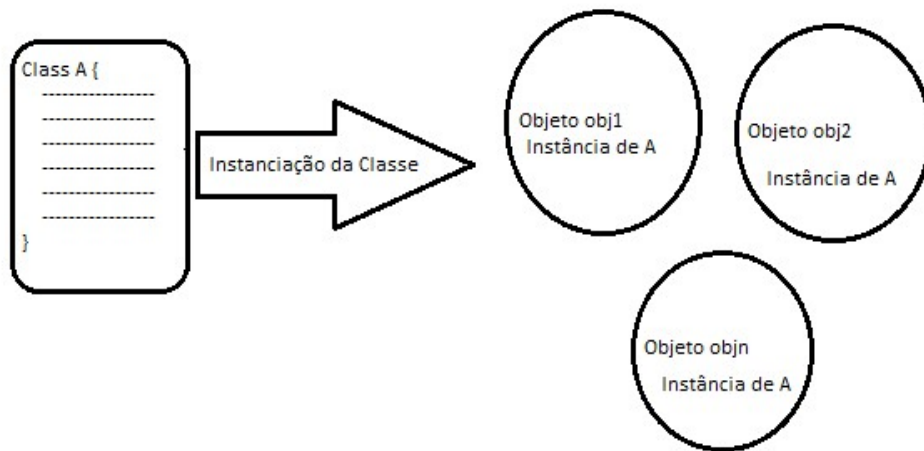
Exercícios práticos de while:

- 1) Mostrar todos os valores de 1 até 10
- 2) Mostrar todos os valores de 10 até 1
- 3) Digitar 3 números inteiros, armazenar em vetor, realizar a soma e média.
- 4) Utilizando vetor, receber 3 números float.

Trabalhando com classes e métodos (funções) no Python

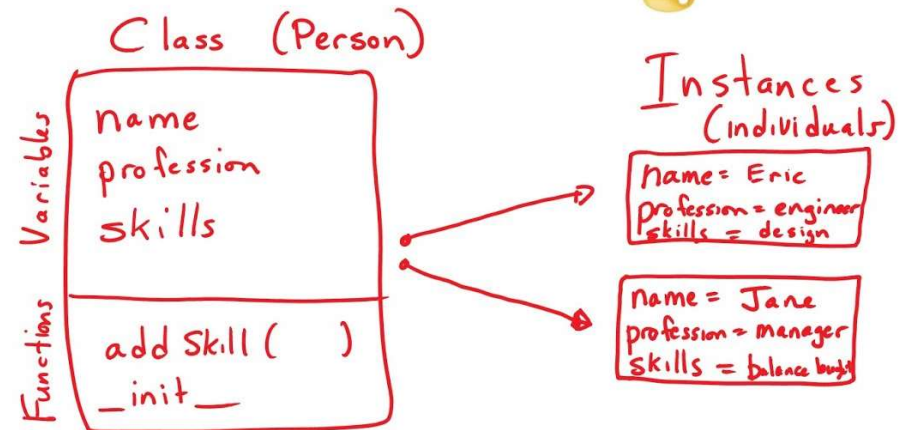
- Classes e Funções são blocos de códigos que realizem determinada tarefa programada, que normalmente precisam ser usadas várias vezes (ou seja, são reutilizadas).
- Uma classe associa dados (atributos) e operações (métodos) numa só estrutura. Um objeto é uma instância de uma classe. Ou seja, uma representação da classe. Por exemplo, Regis é uma instância de uma classe chamada Pessoa, mas a Pessoa é a classe que o representa de uma forma genérica. Se você criar um outro objeto chamado Fabio, esse objeto também será uma instancia da classe Pessoa.
- Tudo isso é chamado em programação de orientação a objetos.
- Objetivo principal é a REUTILIZAÇÃO DE CÓDIGO

Trabalhando com classes e métodos (funções) no Python



<https://i.stack.imgur.com/9aGX2.png>

Introduction to Python Classes



<https://www.youtube.com/watch?v=LwOg0b0ZwCM>

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções):

```
def main ():
```

```
    #linha de código1
```

```
    #linha de código2
```

```
    #linha de código3
```

```
>> main()
```

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções):

Palavra reservada **def** realiza a definição da função

main é o nome da função (pode ser alterado)

Entre () define o(s) parâmetro(s) do método. Pode ser vazio.

: ao final indica que a próxima linha deve ser indentada.

Para executar a função, basta chamar o nome (**main()**), se existir, passar os parâmetros dentro dos parênteses.

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções):

```
def teste (nome):
```

```
    print("Meu nome é: ", nome)
```

```
>> teste("Thiago")
```

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções): Mais de um parâmetro

```
def teste (nome, idade):
```

```
    print("Meu nome é: ", nome, " Idade: ", idade)
```

```
>> teste("Thiago", 28)
```

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções): podem receber valores e realizar cálculos.

```
def calcula_media (soma, qtd):  
    media = soma/qtd  
    print(" A média é: ", round(media, 2))  
  
num1= float(input("Digite o valor do 1º número: "))  
num2= float(input("Digite o valor do 2º número: "))  
soma = num1+num2  
calcula_media(soma, 2)
```

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções): usando vetor

```
def calcular_media(Numeros):  
    soma = 0  
    for n in list(range(0, len(Numeros))):  
        soma = soma + Numeros[n]  
  
    media = soma / len(Numeros)  
    print("A média é: ", media)
```

```
Numeros = []  
num = int(input("Quantos números gostaria de digitar?"))
```

```
for n in range(0, num):  
    Numeros.append(eval(input("Digite o número : ")))
```

```
calcular_media(Numeros)
```

Trabalhando com classes e métodos (funções) no Python

- CLASSES

```
class NomeClasse:
```

```
    def metodo(self, args):  
        pass
```

Exemplo:

```
class NomeClasse :
```

```
    def metodo(self, nome):  
        print ("Nome: ", nome)
```

NomeClasse – nome da classe (não tem espaço)

def método () – nome do método da classe: nome_da_classe

#self – palavra reservada obrigatória

args – argumentos ou parâmetros do método

#pass – significa que o método tem conteúdo vazio.

Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**

- **Import <nome_biblioteca>**

importa todo o módulo especificado

Import <nome_biblioteca> from <pacote>

importa apenas o pacote especificado

Ao tentar *importar* um módulo que não existe, um erro será reportado:
ImportError na tela

Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**

- **Exemplo:**














```
import math      Biblioteca
```

```
print (math.sqrt(0))
```

```
print (math.sqrt(4))
```

```
print (math.sqrt(8))
```

Github data Python 2018

Library Name	Type	Commits	Contributors	Releases	Watch	Star	Fork	Commits/ Contributors	Commits/ Releases	Star/ Contributors
 matplotlib	Visualization	25 747	725	70	498	7 292	398	36	368	10
 Bokeh	Visualization	16 983	294	58	363	7 615	2 000	58	293	26
 plotly	Visualization	2 906	48	8	198	3 444	850	61	363	72
Seaborn	Visualization	2 044	83	13	205	4 856	752	25	157	59
<i>pydot</i>	Visualization	169	12	12	17	193	80	14	14	16
 learn	Machine learning	22 753	1 084	86	2 114	28 098	14 005	21	265	26
XGBoost LightGBM CatBoost	Machine learning	3277	280	9	868	11 991	5 425	12	364	43
		1083	79	14	363	5 488	1 467	14	77	69
		1509	61	20	157	2 780	369	25	75	46
 eli5	Machine learning	922	6	22	39	672	89	154	42	112
 SciPy	Data wrangling	19 150	608	99	301	4 447	2 318	31	193	7
 NumPy	Data wrangling	17 911	641	136	390	7 215	2 766	28	132	11
 pandas	Data wrangling	17 144	1 165	93	858	14 294	5 788	15	184	12
 statsmodels	Statistics	10 067	153	21	234	2 868	1 240	66	479	19
 TensorFlow	Deep learning	33 339	1 469	58	7 968	99 664	62 952	23	575	68
PYTORCH	Deep learning	11 306	635	16	816	15 512	3 483	18	707	24
 Keras	Deep learning	4 539	671	41	1 673	29 444	10 964	7	1111	44
dist-keras elephas spark-deep-learning	Distributed deep learning	1125	5	7	41	431	106	225	161	86
		170	13	5	97	913	189	13	34	70
		67	11	3	116	920	206	6	22	84
 Natural Language Toolkit	NLP	13 041	236	24	467	6 405	1 804	55	543	27
spaCy	NLP	8 623	215	56	425	9 258	1 446	40	154	43
gensim	NLP	3 603	273	52	415	6 995	2 689	13	69	26
 Scrapy	Data scraping	6 625	281	81	1 723	27 277	6 469	24	82	97

Last reviewed: 13.02.2018

Created by ActiveWizards

- Matplotlib
- Scipy
- NumPy
- pandas