Manipulação de variável do tipo string (str):

Strings são variáveis que tem seus valores representados entre aspas, que podem ser: duplas = “” ou simples = ‘’

Exercícios: digite scripts abaixo no IDLE e rode-os no Terminal de seu sistema operacional:

f) Concatenando variáveis**:**

```
instituicao = 'Instituto '
esfera = 'Federal'
print( instituicao + esfera )
```

g) Fatiamento String:

```
instituicao = 'Instituto federal de Rondônia '

print(f'\nApresentando os 10 primeiros caracteres da string.. \n')

print(instituicao[:10])
```

Pode-se suprimir o zero do início

h) Fatiando String – apresentando um recorte

```
instituicao = 'Instituto federal de Rondônia '

print(f'\nApresentando dados do 5º ao 15º caracter da string.. \n')

print(instituicao[5:25])
```

i) Fatiando String – apresentando um recorte pulando e caracteres

```
instituicao = 'Instituto federal de Rondônia '

print(f'\nApresentando dados do índice 0º ao 20º, pulando de 3 em 3 caracteres \n')

print(instituicao[0:20:3])
```

j) Fatiando Strings - substituindo o primeiro caractere:

```
nome = 'Astrogildo'

nome = 'O' + nome[1:]

print(f'\nO novo nome é: {nome} \n')
```

k) Fatiando Strings – invertendo os caracteres da string

```
disciplina = 'Algoritmo e lógica de programação'

print(f'\nA string invertida é: {disciplina[: : -1]} \n')
```

l) Fatiando Strings – substituindo caracteres

```
palavra = 'paralelepípedo'

palavra = palavra.replace('l', 'r')

print(f'\nA palavra com a letra l trocada pela letra r =  {palavra} \n')
```

m) Fatiando Strings – recebe dados a partir do índice 3 e finaliza 5 índices antes do final

```
palavra = 'reconstitucionalissimamentemente'

palavra = palavra[2 : -5]

print(f'\nA palavra que recebeu dados a partir do índice 3 e até o índice 5 =  {palavra} \n')
```

n) – Verificando o caractere que inicia a palavra - startswith()

```
palavra ='Desenvolvimento'

print(f'\nA palavra incia com a letra D? {palavra.startswith("D")}')

print(f'\nA palavra incia com a letra J? {palavra.startswith("J")}')
```

o) Verificando o caractere do final da palavra - endswith()

```
palavra ='Inovação'

print(f'\nA palavra termina com a letra m? {palavra.endswith("m")}')

print(f'\nA palavra termina com a letra J? {palavra.endswith("o")}')
```

p) Verificando o caractere do início e final da palavra – startswith() e endswith()

```
palavra ='Oportunidade'

print(f'\nA palavra inicia com a letra o? {palavra.startswith("o")} - E termina com a letra e? {palavra.endswith("e")}')
```

Comparando strings (str):

Se perguntarmos ao Python se string ‘a’ é maior que a string ‘X’, a resposta será True. Para entendermos essas relações precisamos conhecer um pouco sobre a tabela ASC.

A tabela ASC é uma convenção que define o valor decimal e binário de cada caractere, esse código pode ser obtido informando um caractere, ou através de um código obter um caractere. Normalmente esses códigos são utilizados quando estamos definindo funções das teclas do teclado.

q) ASC = obtendo o código da tecla ou letra do código

```
print('a' > 'X')

print(chr(97))
print(chr(88))

print()

print(ord('a'))
print(ord('X'))
```

```
=====
True
a
X

97
88
```

ESTRUTURA DE CONTROLE - LAÇOS DE REPETIÇÃO FOR():

O Python possui também o laço de repetição for()

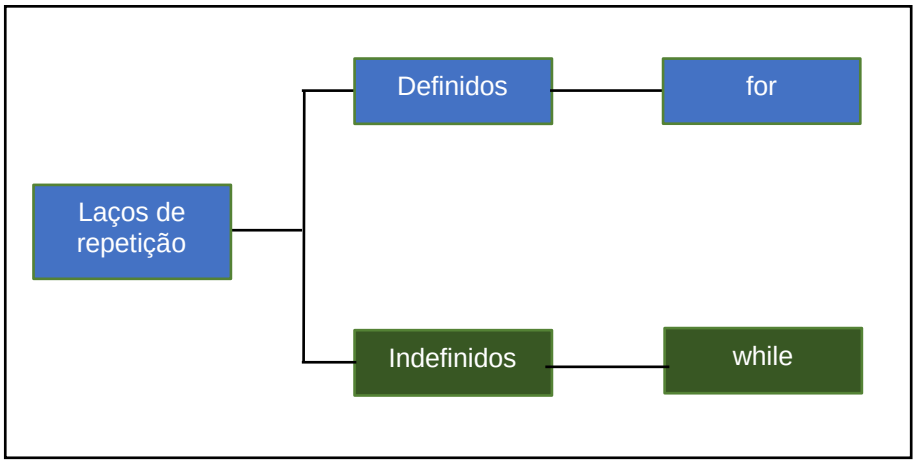

Figura: estruturas de repetição.

LAÇO DE REPETIÇÃO DEFINIDO - comando: for()

- É usado quando podemos calcular exatamente quantas vezes um bloco de comando precisa ser executado.

r) Usando o for para apresentar os valores de uma variável iterável.

```
ifro = 'Ariquemes'
for letra in ifro:
    print(letra)
```

```
= RESTART:
A
r
i
q
u
e
m
e
s
```

- Interpretando a estrutura acima: *para cada letra dentro do iterável* ***ifro*** *imprima essa letra.*

s) Usando o for() + enumerate() para apresentar o índice e valor de cada do índice da variável iterável.

```
ifro = 'Campus'
for indice_letra in enumerate(ifro):
    print(indice_letra)
```

```
= RESTART:
(0, 'C')
(1, 'a')
(2, 'm')
(3, 'p')
(4, 'u')
(5, 's')
```

- Interpretando a estrutura acima: *para cada letra dentro do iterável* ***campus*** *imprima o índice e a letra.*

t) Usando o for() + enumerate para apresentar apenas o índice de cada letra de uma variável iterável - imprimindo apenas os índices do iterador

```
ifro = 'Campus '
for indice_letra in enumerate(ifro):
    print(indice_letra[0])
```

```
= RESTART:
0
1
2
3
4
5
6
```

u) Usando o for + enumerate apresentar somente o valor do iterável. Imprimindo (acessando) apenas os valores do iterável.

```
ifro = 'Campus'
for indice_letra in enumerate(ifro):
    print(indice_letra[1])
```

```
= RESTART:
C
a
m
p
u
s
```

Iteradores: for: implementação de métodos numéricos com a função range().

Em Python, o comando for é utilizado para iterar ao longo de uma sequência. Quando se trata da implementação de métodos numéricos usando o **for**, é necessário avançar ao longo de uma **quantidade finita** de elementos de um contador e para tanto normalmente utilizamos a função range().

A **função range()** é utilizadas todas as vezes que for preciso gerar uma série de números índices num intervalo determinado, possibilitando seu uso nos mais diversos tipos de funções e não só na estrutura de repetição for.

Na função range() o intervalo de índices começa em (*start*) e termina em (*stop*), e para determinar a distância entre os índices inteiros gerados usamos o comando (*step*).

range ([*start*], [*stop*, *step*])

As variáveis que serão utilizadas como parâmetros, contêm colchetes junto ao nome dos parâmetros e servem para informar se o intervalo será aberto ou fechado.

] x [

Sempre que tivermos os colchetes se opondo a variável numérica, o Python irá entender que o número está contido entre os colchetes e possui um intervalo aberto. Dessa maneira, podemos entender que todos os números contidos no intervalo estão inclusos menos os dois extremos. Exemplo: *num script onde x =0 e y = 4, o intervalo seria 1,2,3.*

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

Nesse exemplo as variáveis numéricas x e y não estão contidas no intervalo.

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

Nesse exemplo as variáveis numéricas x e y não estão contidas no intervalo.

Seguindo esse raciocínio e tendo como base a estrutura de controle do comando range: `range ([start], [stop, step])` podemos concluir que o intervalo [start] está fechado, em [stop, o intervalo está aberto, e em, step] o intervalo é fechado.

Sendo assim, ao definirmos um intervalo de índices entre 0 e 10, os números apresentados serão 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, pois em Python os índices são numerados a partir do índice zero. *Exemplo* = range(0, 10, 2)

```
>>> range(0, 10, 2)
range(0, 10, 2)
>>> type(range(0, 10, 2))
<class 'range'>
>>> list(range(0, 10, 2))
[0, 2, 4, 6, 8]
```

- 1ª linha de comando, o Python retorna o objeto do tipo range que é **usado para fazer iterações**;

- 2ª linha de comando, o Python mostra a classe;

- 3ª linha, o comando list() faz um **cast**, de forma que os elementos que estão dentro objeto *range()* sejam apresentados na tela como lista.

v) - apresentar 10 índices numéricos:

Desenvolva um script que mostra na tela os números de 0 a 9, usando o laço de repetição **for().** Lembre-se todo índice no Python começa no 0.

```
for x in range(10):
    print(x)
```

```
= RESTART:
0
1
2
3
4
5
6
7
8
9
```

Neste cenário, a cada ciclo a variável i recebeu um dos índices de range()

Em um outro cenário onde desejamos que sejam apresentados 10 índices numéricos ao contrário do número 10 até o número 0:

x) apresentar 10 índices numéricos ao contrário

```
for i in range(10,0, -1):
    print(i)
```

```
= RESTART:
10
9
8
7
6
5
4
3
2
1
```

Estruturas de repetição For() e a instrução BREAK: normalmente é aplicado quando dentro de uma estrutura de repetição encontramos um estrutura condicional if() que é utilizada para a **interrupção de toda estrutura de repetição**, principalmente quando o objetivo de ter criado a estrutura de repetição ou ciclo foi atendido.

Utilizando a instrução break para interromper a estrutura de repetição:

w) - break para interromper a estrutura de repetição:

```
for i in range(10):
    if i == 5:
        break
    else:
        print(i)
```

```
= RESTART:
0
1
2
3
4
```

Estruturas de repetição for(), range, else e break

Se necessário é possível executar a condicional ao final da estrutura de repetição for, com o objetivo de executar alguma instrução ou bloco de código ao final do loop.

Exemplo 1: usando for, range e else para imprimir i enquanto este for menor ou igual a 3

```
>>> for i in range(3):
...     if i == 5:
...         print(f'O range ultrapassou os limites...')
...     print(f'O valor de i é: {i}')
... else:
...     print(f'For executado com sucesso, valor de i é: {i}')
...
O valor de i é: 0
O valor de i é: 1
O valor de i é: 2
For executado com sucesso, valor de i é: 2
>>>
```

Exemplo 2: usando for, range, else if e break para imprimir i enquanto este for menor ou igual a 3 (regra de negócio)

```
>>> for i in range(1000):
...     if i == 3:
...         print(f'O range atingiu a regra de negócio...')
...         break
...     print(f'O valor de i é: {i}')
... else:
...     print(f'For executado com sucesso, o valor de i é: {i}')
...
```

```
O valor de i é: 0
O valor de i é: 1
O valor de i é: 2
O range atingiu a regra de negócio...
>>>
```

Estruturas de repetição e a instrução CONTINUE:

Normalmente é aplicado quando dentro de uma estrutura de repetição encontramos uma estrutura condicional if() que utilizada para a **interrupção de um ciclo em execução.**

Podemos entender que a instrução continue pula o ciclo atual, mas não termina a execução da estrutura de repetição.

Exemplo: em um cenário, onde precisamos imprimir ou realizar algum cálculo somente com números ímpares.

```
for i in range(20):
    if (i%2 ==0):
            continue
    print(i)
```

```
= RESTART:
1
3
5
7
9
11
13
15
17
19
```

REFERÊNCIA:

Mueller, John Paul. **Programação Funcional Para Leigos**. Alta Books. Edição do Kindle, 2020.

LIMA, Guilherme. 2021. **Python: A origem do nome.** Disponível em: https://www.alura.com.br/artigos/python-origem-do-nome

F. V. C. Santos, Rafael. **Python: Guia prático do básico ao avançado** (Cientista de dados Livro 2) (p. 1). rafaelfvcs. 2ª Edição - Kindle, 2020.

MENEZES, Nilo Ney Coutinho. **Introdução à programação com Python: Algoritmos e lógica de programação para iniciantes**. Editora: Novatec - 3ª edição, 2019 - São Paulo - SP.

Piva Junior, Dilermando. Algoritmos e programação de computadores / Piva Jr ... [et al.]. - 2. ed. - Rio de Janeiro: Elsevier, 2019. 528 - ISBN 9788535292480