2º BIMESTRE – PROGRAMAÇÃO ORIENTADA A OBJETOS - PYTHON

Sumário

26 – MANIPULANDO CADEIA DE CARACTERES

Manipulação de caracteres em programação é um recurso bastante necessário e útil. Vejam abaixo:

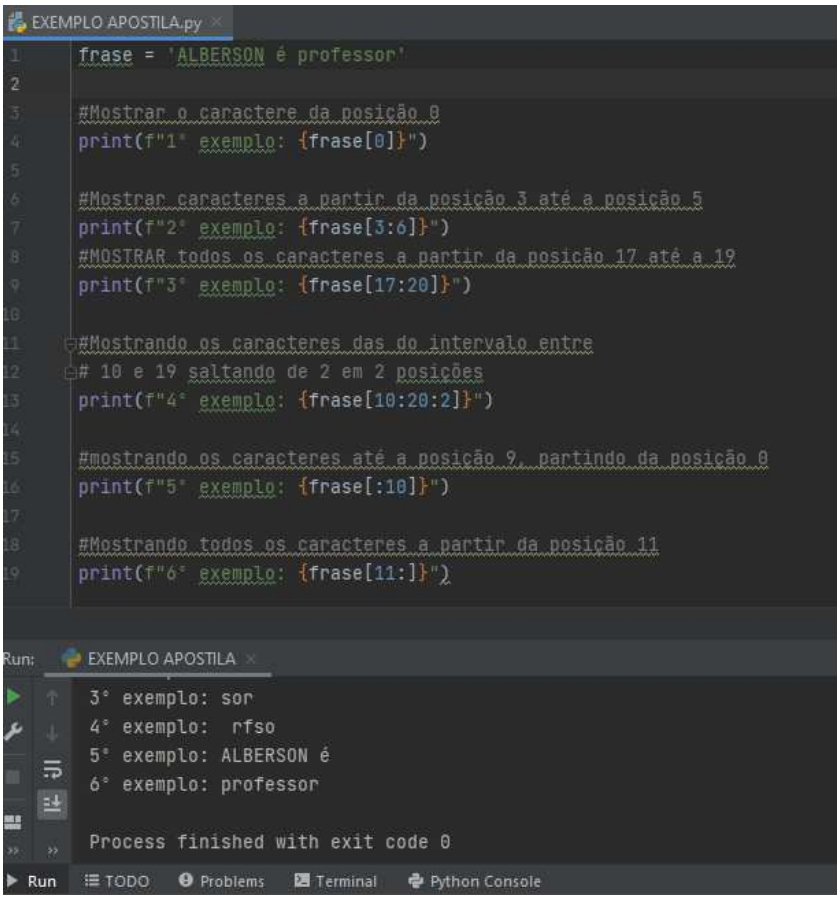
Com a linha acima, estamos atribuindo à variável ***frase*** o conteúdo ***'ALBERSON é professor'***. Entretanto, podemos considerar também em PYTHON, que esta variável funciona automaticamente como um vetor. Assim sendo, podemos imaginar que esta variável armazena também a frase da seguinte forma:

frase

A	L	B	E	R	S	O	N		é		p	r	o	f	E	s	s	o	R
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

26.1 - FATIAMENTO DE STRINGS

Observem abaixo as formas de fatiamento de string que podemos usar em python.



VALE DESTACAR QUE OS EXEMPLOS ACIMA ESTÃO MOSTRANDO PARTES DA STRING ARMAZENADA NA VARIÁVEL. ASSIM SENDO, O CONTEÚDO DA VARIÁVEL CONTINUA ARMAZENANDO A SEQUÊNCIA INICIAL DE CARACTERES.

Vamos ver outras utilidades:

```
EXEMPLO APOSTILA.py
frase = 'ALBERSON é professor'

#mostrar os caracteres a partir da posição 7, pulando de 3 em 3 até o final da string
print(f'A PARTIR DA POSIÇÃO 7, MOSTRAR O RESTANTE DA FRASE PULANDO DE 3 EM 3 POSIÇÕES = {frase[7::3]}')

#mostrando todos os caracteres da string com estrutura for
for x in range(0, len(frase)):
    print(f'frase[{x}] = {frase[x]}')
```

Teremos então os seguintes dados impressos:

```
Run: EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberso
A PARTIR DA POSIÇÃO 7, MOSTRAR O RESTANTE DA FRASE PULANDO DE 3 EM 3 POSIÇÕES = N osr
frase[0] = A
frase[1] = L
frase[2] = B
frase[3] = E
frase[4] = R
frase[5] = S
frase[6] = O
frase[7] = N
frase[8] =
frase[9] = é
frase[10] =
frase[11] = p
frase[12] = r
frase[13] = o
frase[14] = f
frase[15] = e
frase[16] = s
frase[17] = s
frase[18] = o
frase[19] = r

Process finished with exit code 0
```

VALE DESTACAR NA ESTRUTURA DE REPETIÇÃO for, MOSTRADA ACIMA, QUE ESTÁ SENDO O MÉTODO len, QUE CONTA QUANTOS CARACTERES A STRING POSSUI.

EM OUTRAS LINGUAGEM DEVERÍAMOS USAR -1 A FRENTE DO len(frase)-1 PARA MOSTRARMOS ATÉ O CARACTER DA POSIÇÃO 19.

26.2 – ANÁLISE DE STRINGS

Analisar strings é simplesmente usar métodos normalmente necessários quando estamos manipulando strings, tais como exemplo:

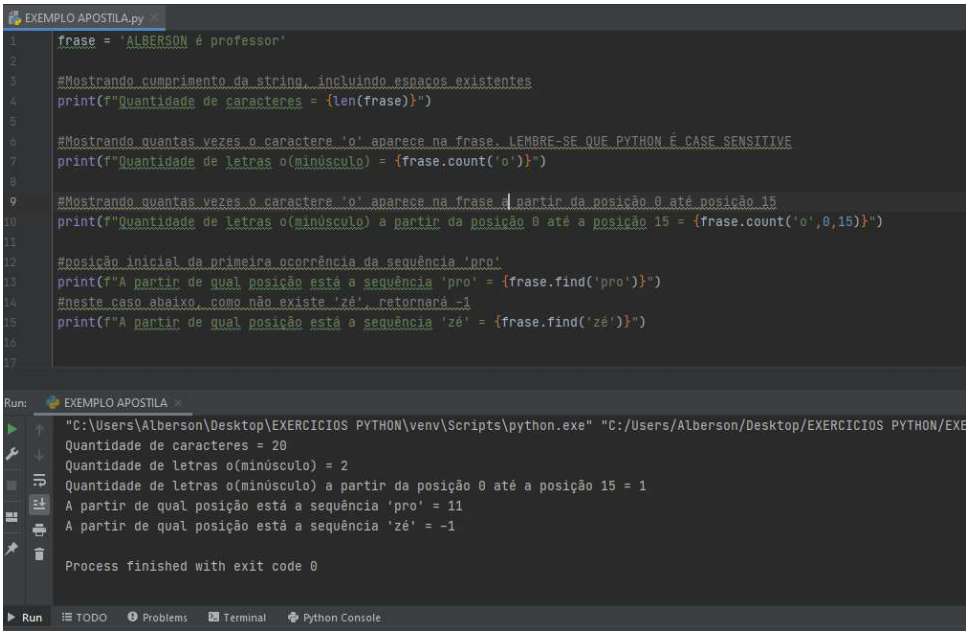
- Saber quantos caracteres tem a string
- Letra inicial da string
- Letra que a string termina
- Primeira palavra de uma frase
- Etc...

Vamos a alguns exemplos, para ilustrar estes tipos de necessidades comumente utilizadas por programadores experientes, considerando a frase armazenada na variável ***frase***:

frase:
"ALBERSON é professor"

De acordo com o mencionado até então, podemos usá-la como vetor:
frase:

A	L	B	E	R	S	O	N		é		p	r	o	f	e	s	s	o	r
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

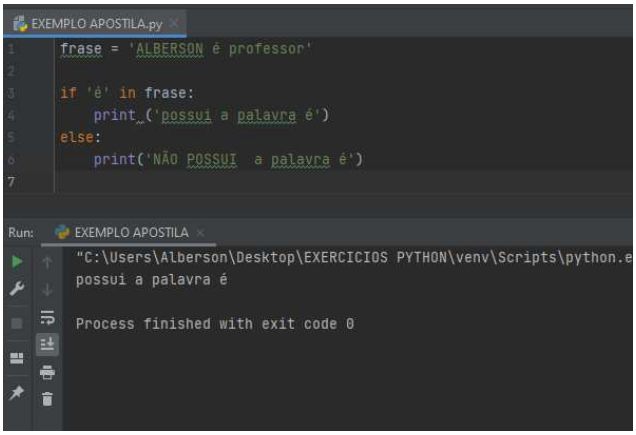


CURIOSIDADE !!!!
PODEMOS TAMBÉM USAR A SEGUINTE LINHA DE COMANDO:

```
print('Quantidade de letras o ={}'.format(frase.upper().count('O')))
```

Desta forma vamos verificar quantas letras 'O' temos na frase, independente se em maiúsculo ou minúsculo, visto que toda frase será transformada em maiúscula, para fazermos a análise.

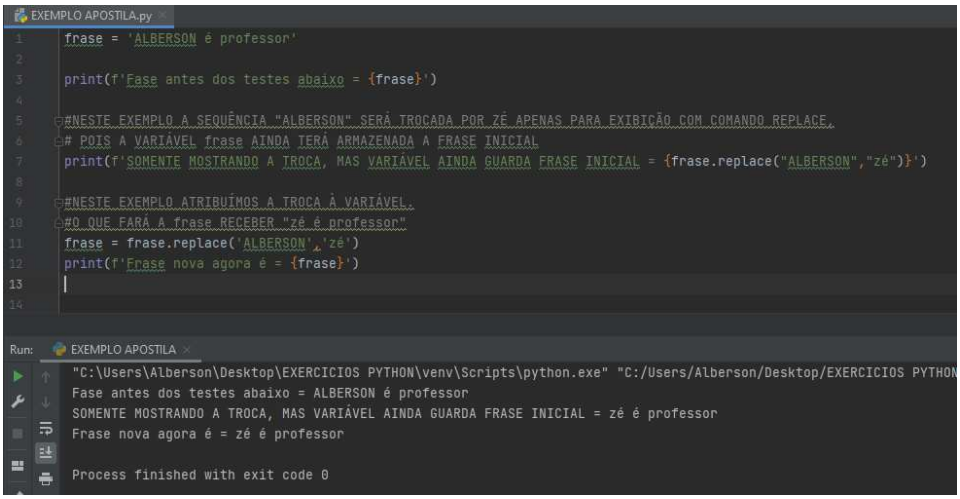
No teste a seguir, estamos usando o operador ***in*** que verifica se na variável ***frase*** existe a letra "é" (com acento), em qualquer ponto da string.



26.3 – TRANSFORMAÇÕES DE STRINGS

As transformações dizem respeito a alterações dos dados armazenados em variáveis do tipo caracteres.

Vamos a alguns exemplos que deixam mais claro o entendimento sobre este assunto.



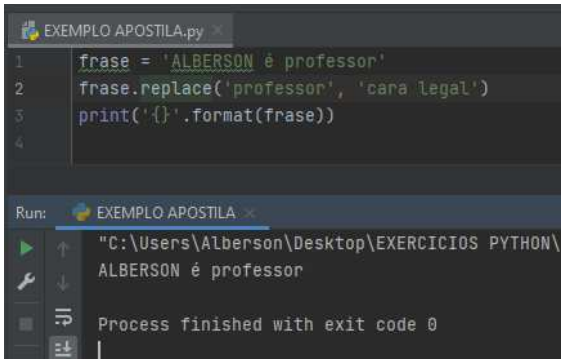
Alguns métodos foram tratados anteriormente nesta apostila. Só para relembrar:

MÉTODOS USADOS ACIMA:	SIGNIFICADO, OU RETORNO:
isnumeric()	Verifica se o objeto guarda um valor numérico (int ou float)
isalnum()	Verifica se o objeto guarda um valor alfanumérico (números e letras)
isalpha()	Verifica se o objeto guarda um valor alfabético (somente letras)
islower()	Verifica se o objeto está guardando somente letras minúsculas
isupper()	Verifica se o objeto está guardando somente letras maiúsculas
isspace()	Verifica se o objeto está guardando somente sequência de espaços
Istitle()	Verifica se o objeto está guardando letras maiúsculas e minúsculas, ou seja, se está **capitalizada**
strip()	Retira espaços do início e fim da sequência de string (das extremidades e não entre palavras da frase)
rstrip()	Retira os espaços em branco no fim da string
lstrip()	Retira os espaços em branco no início da string

VALE RESSALTAR QUE:

OS MÉTODOS ACIMA RETORNAM DADOS DE ACORDO COM SIGNIFICADO DO RETORNO, ASSIM NÃO PODEM SER ESCRITOS EM LINHAS ISOLADAS DE COMANDO

Por que no teste a seguir, a frase é 'Alberson é professor' e não 'alberson é cara legal' ?



26.4 – SEPARAÇÃO DE STRINGS

Este recurso é bem interessante e será usado em exemplos, exercícios e capítulos posteriores desta apostila.

Vamos a alguns exemplos, considerando ainda a frase que usamos anteriormente:

frase:

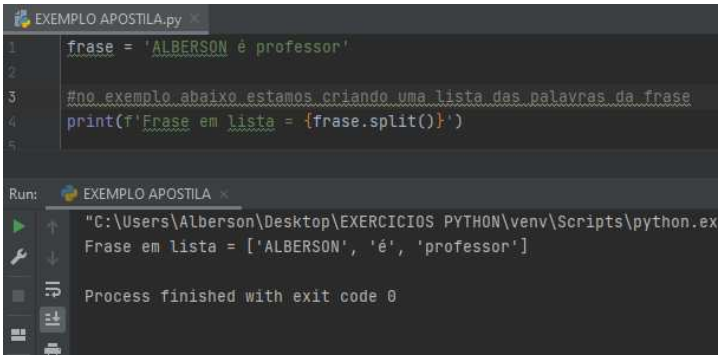
"ALBERSON é professor"

Ou então, usaremos como vetor:

frase:

A	L	B	E	R	S	O	N		é		p	r	o	f	e	s	s	o	r
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Vamos exibir a frase criando uma LISTA DE PALAVRAS contidas na mesma, veja:



Frase em lista pode ser enxergada como se segue:

A	L	B	E	R	S	O	N
0	1	2	3	4	5	6	7

é
0

P	r	O	f	e	s	s	o	R
0	1	2	3	4	5	6	7	8

Observe que os índices de posições são reorganizados para cada palavra quando a frase foi separada com SPLIT().

Outro teste importante que podemos fazer, é guardar esta lista de palavras separadas em uma variável, e usá-la como um sub vetor, vejamos como imprimir cada palavra da frase:

```
EXEMPLO APOSTILA.py
frase = 'ALBERSON é professor'

#no exemplo abaixo estamos APENAS MOSTRANDO uma lista das palavras da frase
print(f'Frase em lista = {frase.split()}')

#abaixo estamos ATRIBUINDO a uma variável a lista de palavras criadas
x = frase.split()
#nas linhas abaixo estamos mostrando cada palavra da lista separadamente
print(f'1ª palavra da frase= {x[0]}')
print(f'2ª palavra da frase= {x[1]}')
print(f'3ª palavra da frase= {x[2]}')

Run: EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Use
Frase em lista = ['ALBERSON', 'é', 'professor']
1ª palavra da frase= ALBERSON
2ª palavra da frase= é
3ª palavra da frase= professor

Process finished with exit code 0
```

X:

A	L	B	E	R	S	O	N
0							

é
1

P	r	O	f	e	s	s	o	R
2								

Reparem que em X os índices referem-se às palavras da frase.

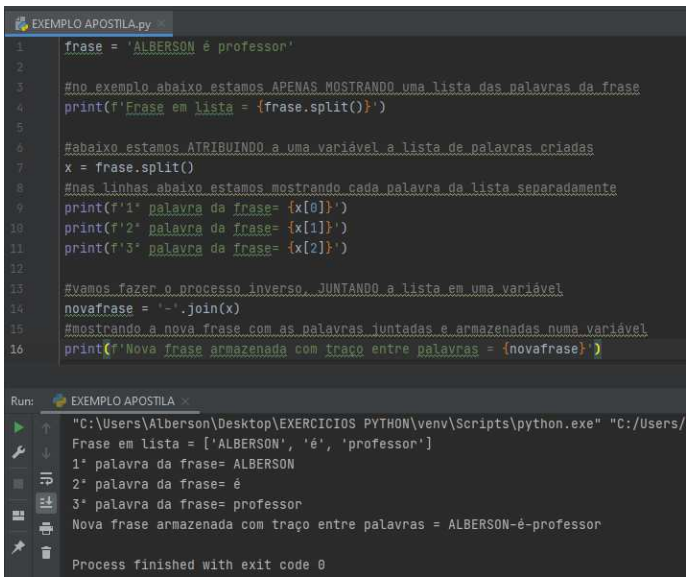
Podemos imprimir caracteres específicos de uma palavra contida na lista, veja:

```
EXEMPLO APOSTILA.py
frase = 'ALBERSON é professor'
x= frase.split()
print('{}'.format(x[2][1]))

Run: EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIO
r

Process finished with exit code 0
```

No caso acima estamos imprimindo o SEGUNDO caractere da TERCEIRA palavra da lista x

26.5 – JUNÇÃO DE LISTAS DE STRINGS

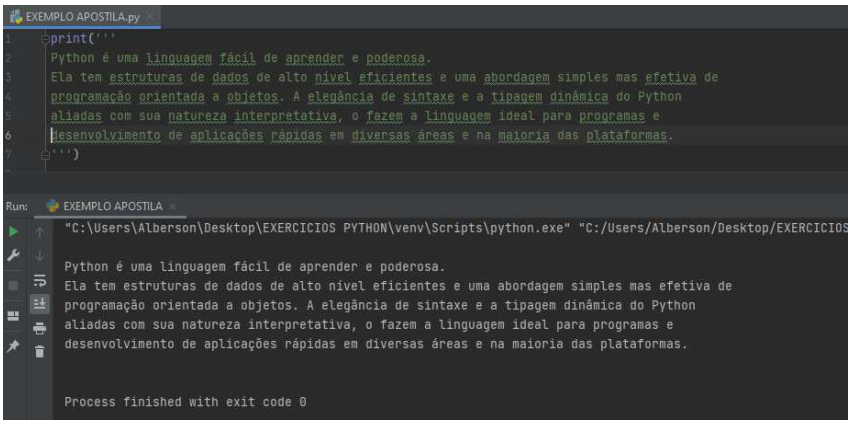
Ao contrário do split(), podemos usar join() para juntar uma lista. Veja o exemplo abaixo:



Vale dizer que se deixar dentro das aspas um espaços, teríamos uma frase com espaços entre as palavras juntadas.

26.6 – IMPRESSÃO DE TEXTOS EXTENSOS COM print (‘‘‘.....’’’)

Podemos usar o print para imprimir textos extensos, usando os limitadores ‘‘‘.....’’’, sem necessidade de criarmos um print() para cada linha do texto. Veja exemplo abaixo:



No exemplo acima reparem que escrevemos na tela várias linhas, porém sem usarmos vários print. Isso foi possível porque escrevemos o texto entre ‘‘‘ <texto> ’’’ num só comando print.

27-DESAFIOS GERAIS - STRINGS:

1) Fazer um programa em python para:
    a. Mostrar todas as letras de uma frase em letras maiúsculas
    b. Todas as letras da frase em letras minúsculas
    c. Informar quantas letras tem na frase
    d. quantas letras tem na primeira palavra da frase
2) Fazer um programa para solicitar ao usuário digitar um número entre 0 e 9999. Em seguida, mostre quantos milhares, centenas, dezenas e unidades tem o número digitado.
3) Fazer um programa python para verificar se o nome de uma cidade informada pelo usuário inicia ou não com a palavra “São”.
4) Fazer um programa python para verificar se no nome de uma pessoa existe “Silva”. Mostre somente true ou false.
5) Fazer um programa em python que solicite a digitação de um nome qualquer. Pede-se:
    a. Mostrar quantas vezes a letra “A” ou “a” aparece
    b. Em que posição a letra “A” aparece a primeira vez
    c. Em que posição a letra “A” aparece a última vez
6) Fazer um programa em python para mostrar o primeiro e o último nome de pessoa digitado pelo usuário, independente da quantidade de palavras do nome digitado.

28 – VARIÁVEIS COMPOSTAS

São variáveis capazes de armazenar vários dados na memória ao mesmo tempo. Em python temos três tipos de variáveis composta que serão abordadas desse ponto para frente, a saber:

- TUPLAS ( )
- LISTAS [ ]
- DICIONÁRIOS { }

28.1 – VARIÁVEIS COMPOSTAS – TUPLAS

As tuplas fazem lembrar vetores.

As TUPLAS SÃO IMUTÁVEIS, ou seja, durante a execução do programa não teremos como alterar um dado nela armazenada. Conseguiremos somente redefinir a tupla

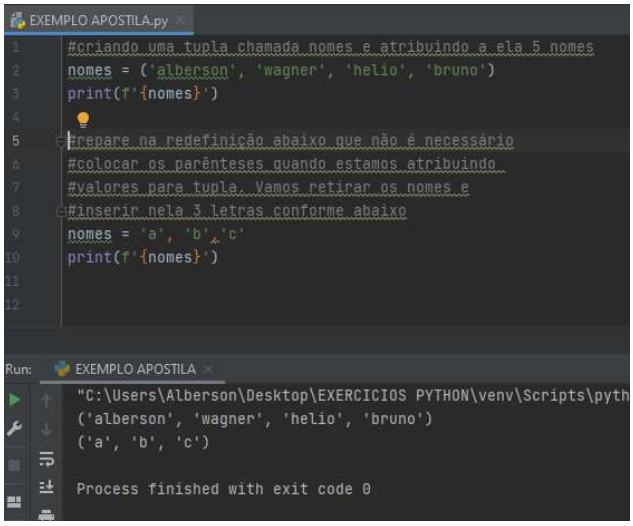
Vejamos como criar uma tupla e acessar os dados nela armazenados:

nomes:

	a	l	b	e	r	s	o	n	,	w	a	g	n	e	r	,	h	e	l	i	o	,	b	r	u	n	o
-4										-3							-2						-1					
0										1							2						3					

Para os próximos exemplos consideremos a tupla ***'nomes'*** mostrada acima. Veja que tuplas possuem índices negativos e positivos que podem ser usados para referenciarmos suas posições. Vamos aos exemplos:

Exemplo 1:



REPARE NO CÓDIGO:

a) Inicialmente criamos a tupla com 5 nomes e depois só conseguimos redefinir a tupla.
b) **Você não conseguirá trocar somente um item da tupla**, trabalhando nela como se fosse um vetor. Terá que reescrever todos os valores da tupla, alterando os dados que desejar.
c) **AO CRIAR, OU REDEFINIR UMA TUPLA, não é necessário envolver os conteúdos nos parênteses, mas vamos adotar o costume de usá-los.**

Exemplo 2:
O uso do print para exibição do conteúdo da tupla já foi abordado em manipulação de strings. Em tuplas o acesso aos dados é feito da mesma forma, ou seja, podemos acessar os itens usando os índices de posições dos dados armazenados, veja:

```
EXEMPLO APOSTILA.py
#criando uma tupla chamada nomes e atribuindo a ela 5 nomes
nomes = ('alberson', 'wagner', 'helio', 'bruno')

#Imprimir todos a partir do índice 0
print(f'{nomes[0:]}')

#Imprimir todos a partir do índice 0, indicando índice final e salto de posição
#repare que indico abaixo como índice final o valor 4, apesar de a mesma não existir
print(f'{nomes[0:4:1]}')

#imprimir os itens 0, 1 e 2, POIS O 3 NÃO SERÁ EXIBIDO
print(f'{nomes[0:3]}')

#imprimindo todos itens que estão em posições de índices pares na tupla
print(f'{nomes[0::2]}')

#imprimindo todos itens que estão em posições de índices impares na tupla
print(f'{nomes[1::2]}')

Run: EXEMPLO APOSTILA
('alberson', 'wagner', 'helio', 'bruno')
('alberson', 'wagner', 'helio', 'bruno')
('alberson', 'wagner', 'helio')
('alberson', 'helio')
('wagner', 'bruno')

Process finished with exit code 0
```

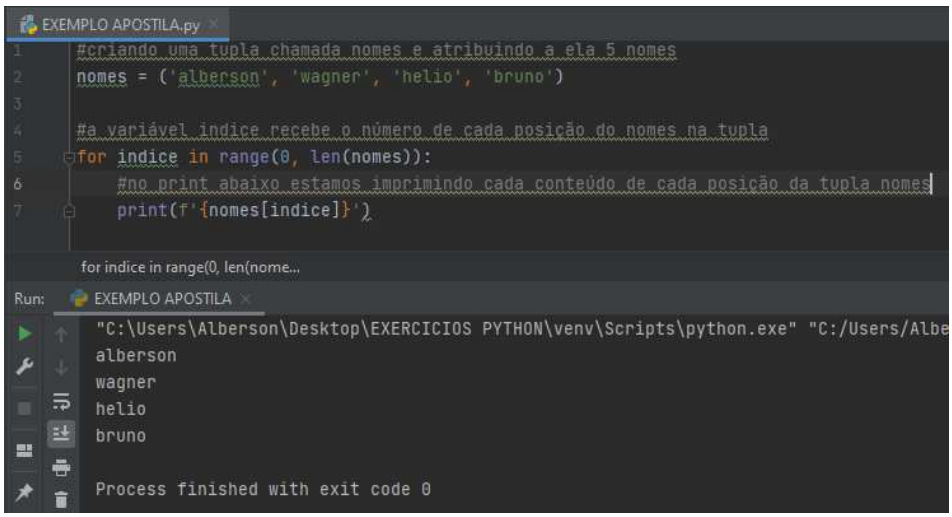
Exemplo 3:
Podemos usar estruturas de repetições para exibir os conteúdos da tupla, veja:

```
EXEMPLO APOSTILA.py
#criando uma tupla chamada nomes e atribuindo a ela 5 nomes
nomes = ('alberson', 'wagner', 'helio', 'bruno')

#a variável conteúdo recebe os conteúdos da tupla
for conteudo in nomes:
    #no print abaixo estamos imprimindo cada conteúdo da tupla nomes
    print(f'{conteudo}')

Run: EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe"
alberson
wagner
helio
bruno

Process finished with exit code 0
```

Repare neste programa!!!!

a) A variável ***conteúdo*** não é iniciada, ela recebe cada nome e será usada no comando print para exibir o nome daquele passo do ***for***

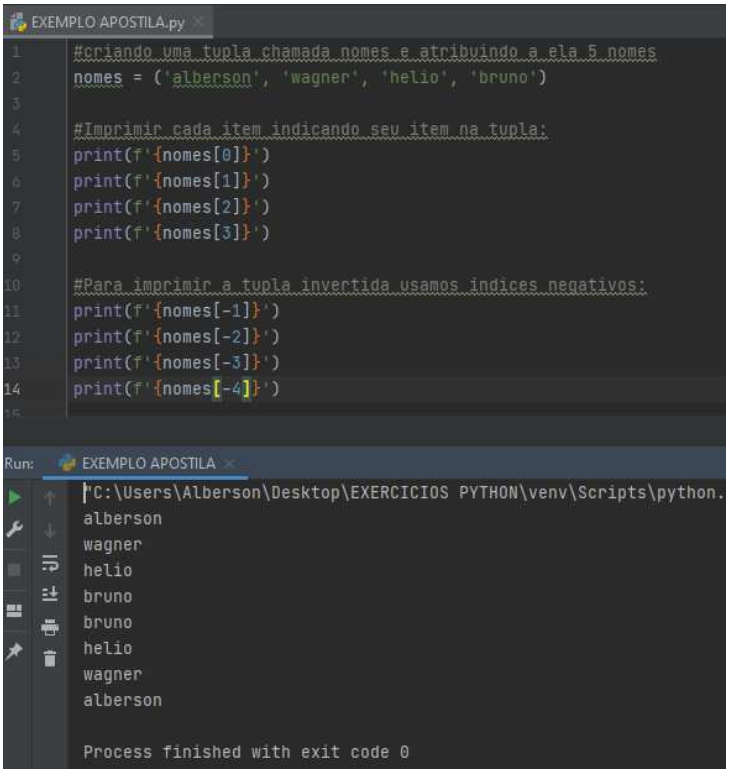
Exemplo 4:
Outro exemplo do uso de ***for*** para imprimir os itens da tupla:



Analisando o programa !!!!

a) Veja que o resultado impresso é exatamente igual ao anterior, porém neste caso temos que referenciar o nome da tupla e o índice da posição que estamos imprimindo os dados.

Exemplo 5:



Importante neste programa!!

a) Quando indicamos índice -1 para impressão de conteúdo da tupla, vamos imprimir a última posição da mesma e assim sucessivamente.

Exemplo 6:
Para impressão de intervalos, usando índices negativos e os dois pontos ":" na sintaxe de impressão, devemos fazê-lo como a referência de índices positivos, porém indicando o sinal de menos "-", veja abaixo:

```
EXEMPLO APOSTILA.py
1  #criando uma tupla chamada nomes e atribuindo a ela 5 nomes
2  nomes = ('alberson', 'wagner', 'helio', 'bruno')
3
4  #Para imprimir toda a tupla invertida, podemos usar os '::'
5  #repare porém, que devemos usar o passo negativo, veja:
6  print(f'{nomes[-1::-1]}')
7
8  # imprimindo intervalos de nomes armazenados na tupla,
9  # neste caso, o último item é o -1
10 print(f'{nomes[-2:-4:-1]}')
11
12 #imprimindo todos itens em ordem invertida, numa só linha de print:
13 print(f'{nomes[-1:-5:-1]}')
14
15 #repare nos dois exemplos abaixo, o primeiro não mostrará nada, pois pelo fato
16 #do índice ser negativo, é necessário o uso do -1 no passo, como feito a seguir
17 print(f'{nomes[:-5:]}')
18 print(f'{nomes[:-5:-1]}')

Run: EXEMPLO APOSTILA
('bruno', 'helio', 'wagner', 'alberson')
('helio', 'wagner')
('bruno', 'helio', 'wagner', 'alberson')
()
('bruno', 'helio', 'wagner', 'alberson')

Process finished with exit code 0
```

Repare nestes exemplos:

a) No print da linha 13 vai imprimir a partir do último item da tupla, ou seja o (-1) até o item (-4) que é o primeiro, **porém, para o item -4 ser impresso há necessidade de indicarmos a posição -5 (inexistente, mas necessária para impressão neste caso)**

b) Os prints das linhas 17 e 18, indicamos um índice de final de impressão negativo, portanto, o python entenderá que partiremos da primeira posição negativa, no caso -1, é vai até a -5. **Veja que isso só funciona com indicação do passo negativo.**

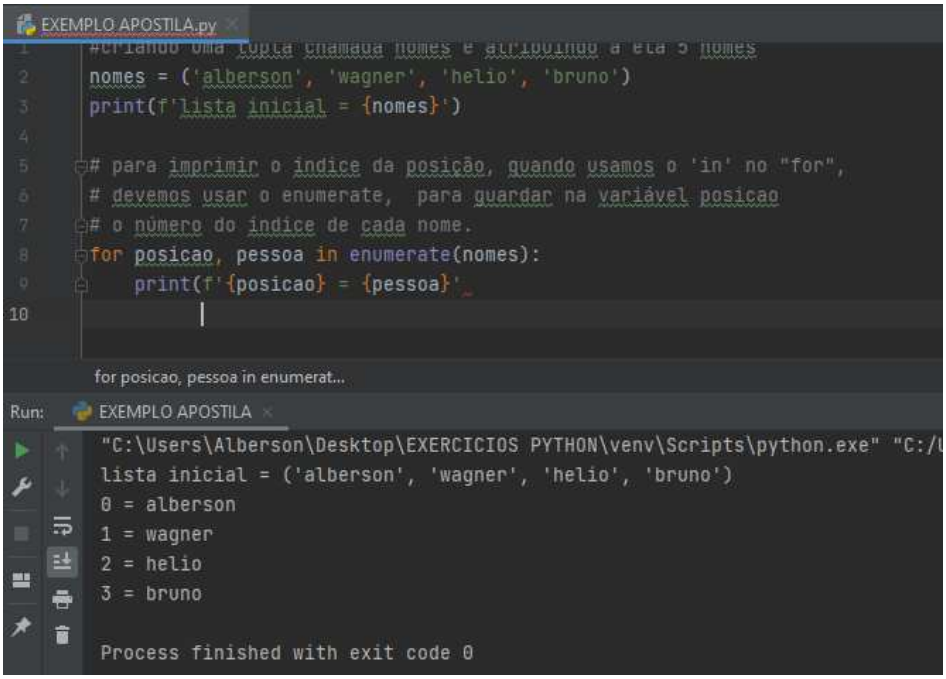
Exemplo 7:
Como mencionado, as tuplas são IMUTÁVEIS. Veja a seguir que se tentarmos alterar um item de uma posição específica dará erro. Só poderemos redefinir a lista como um todo:

```
EXEMPLO APOSTILA.py
1  #criando uma tupla chamada nomes e atribuindo a ela 5 nomes
2  nomes = ('alberson', 'wagner', 'helio', 'bruno')
3  print(f'lista inicial = {nomes}')
4
5  #a linha abaixo redefine a tupla, substituindo os nomes anteriores
6  nomes = ('ze', 'joao', 'maria')
7  print(f'lista redefinida = {nomes}')
8
9  #a linha abaixo não funcionará, dará erro
10 nomes[0] = 'joaozinho'

Run: EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/EX
lista inicial = ('alberson', 'wagner', 'helio', 'bruno')
lista redefinida = ('ze', 'joao', 'maria')
Traceback (most recent call last):
  File "C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\EXEMPLO APOSTILA.py", line 10, in <module>
    nomes[0] = 'joaozinho'
TypeError: 'tuple' object does not support item assignment

Process finished with exit code 1
```

Repare:

a) O comando de atribuição da linha 10, onde tentamos substituir zé por Joãozinho da erro, pois tuplas SÃO IMUTÁVEIS

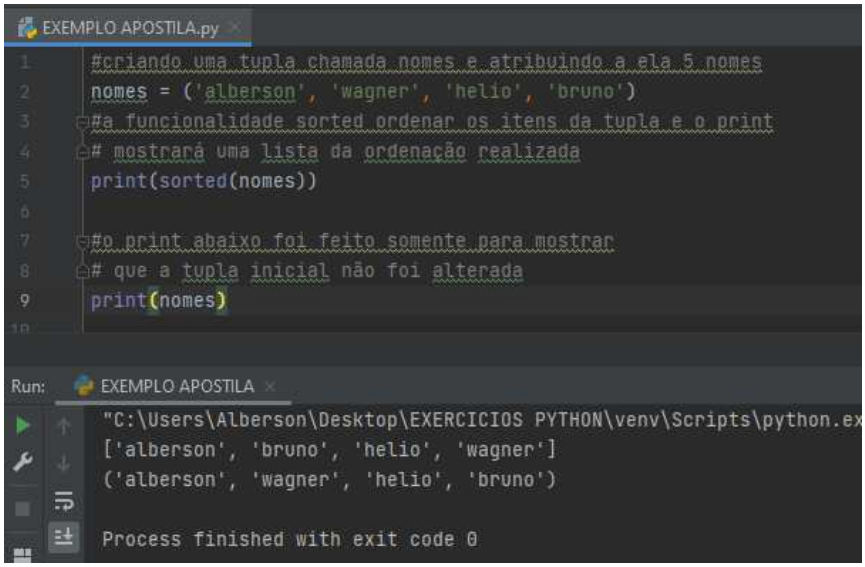
Exemplo 8:



Neste exemplo, repare !!

a) O comando ***for*** está sendo controlado por duas variáveis, porém isto foi feito somente para buscarmos o índice da posição que armazena o dado da tupla.

Exemplo 9:

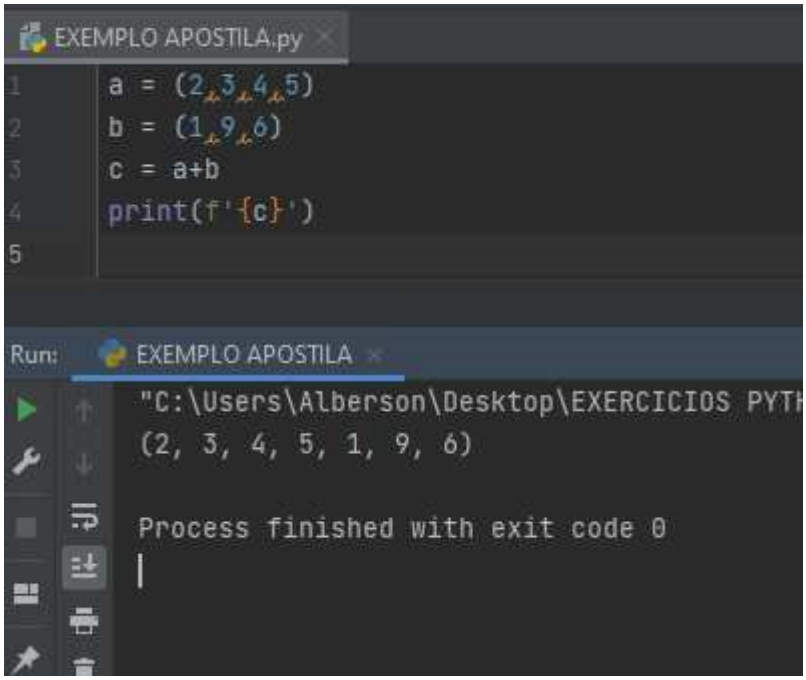


Veja:

a) No exemplo acima estamos criando uma lista ordenada dos nomes da tupla e exibindo na tela para o usuário

Agora vamos exemplificar uso de tuplas que armazenam dados numéricos vejamos exemplos:

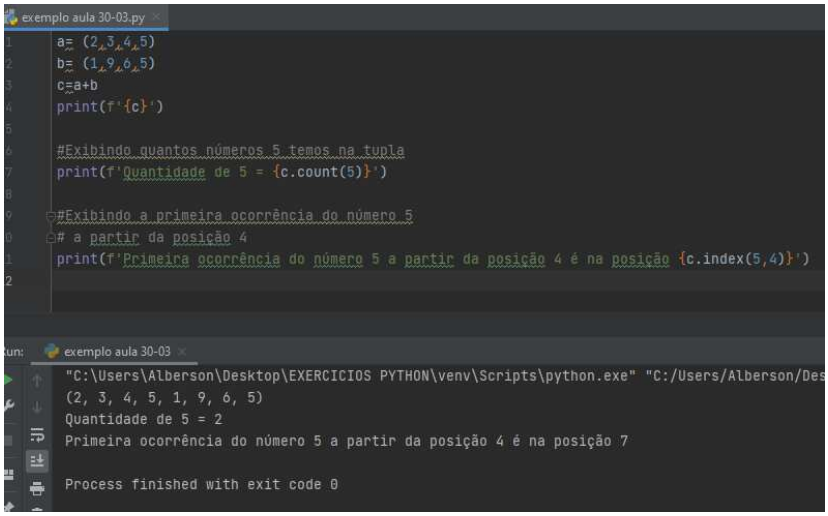
Exemplo 10: UNINDO DADOS DE DUAS OU MAIS TUPLAS:



Explicando:

a) As tuplas ***a*** e ***b*** foram criadas e a tupla ***c*** armazenará a UNIÃO de ***a*** e ***b***
b) Depois é exibida a tupla resultante da UNIÃO, ou seja, imprimimos a tupla ***c***
c) CASO QUEIRAMOS JUNTAR ***b+a*** A ORDEM DOS ITENS DA TUPLA SERIA PRIMEIRAMENTE OS ITENS DE ***b*** E DEPOIS OS ITENS DE ***a***

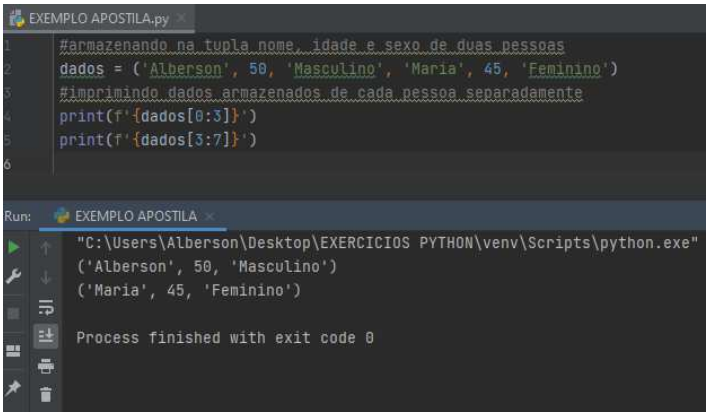
EXEMPLO 11:



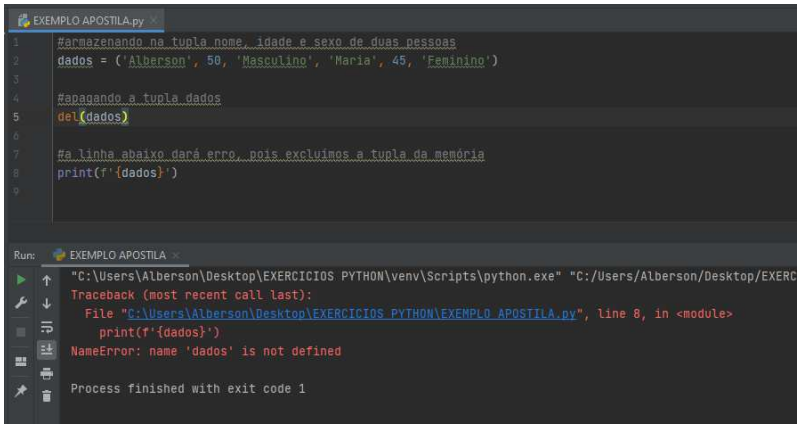
Observe neste exemplo:

a) Usamos os métodos ***count()*** e ***index()***, para verificar quantas ocorrências de um item existem e em que posição estão na tupla, **respectivamente**.

Exemplo 12: UMA TUPLA PODE ARMAZENAR DADOS DE DIFERENTES TIPOS. ISSO NÃO ERA POSSÍVEL EM VETORES, LEMBRA? Vejamos exemplos:



Exemplo 13: PODEMOS EXCLUIR A TUPLA DA MEMÓRIA COM O COMANDO del(<nomedatupla>), veja:



REPARE NO ERRO!!!

a) O ERRO OCASIONADO NA LINHA 8 FOI PELO MOTIVO DE TER EXCLUÍDO A TUPLA DADOS DA MEMÓRIA.
b) Não será possível apagar um item da tupla, pois conforme havia mencionado a tupla é IMUTÁVEL. Portanto, um comando del(dados[0]), por exemplo, daria erro e o programa iria travar neste comando.

28.1.1-DESAFIOS GERAIS - TUPLAS:

1) Guardar numa tupla números entre 0 e 10 em extenso. Solicite ao usuário para digitar um número pelo teclado e mostre o número equivalente por extenso. Exija do usuário a digitação do número no intervalo definido.
2) Crie uma tupla com classificação de um campeonato de futebol com 10 times. Em seguida mostre:
    a. Os 5 primeiros colocados
    b. Os 4 últimos da colocados
    c. A listagem dos times em ordem alfabética
    d. Em que posição está um time informado pelo usuário
3) Crie uma tupla com 5 números aleatórios. Logo depois mostre qual o maior e o menor número armazenado. Mostre também a tupla com os números que foram sorteados
4) Crie uma tupla com 5 números digitados pelo usuário um a um. Depois mostre:
    a. Quantas vezes aparece o número 10.
    b. Em que posição tem o número 3. Pode ser que não apareça, cuide disso!
    c. Quais são os pares armazenados na tupla
5) Criar uma tupla contendo produtos e seus respectivos preços (tudo numa só tupla). Logo depois imprima para o usuário uma lista exatamente formatada como no exemplo abaixo:

```
----------------------------------------
          LISTAGEM DE PREÇOS
----------------------------------------
Lápis.....................R$   1.75
Borracha..................R$   2.00
Caderno...................R$  15.90
Estojo....................R$  25.00
Transferidor..............R$   4.20
Compasso..................R$   9.99
Mochila...................R$ 120.32
Canetas...................R$  22.30
Livro.....................R$  34.90
----------------------------------------
```

6) Crie uma tupla contendo 5 palavras quaisquer. Em seguida, mostre somente as vogais de cada palavra armazenada, veja exemplo:

```
Na palavra APRENDER temos a e e
Na palavra PROGRAMAR temos o a a
Na palavra LINGUAGEM temos i u a e
Na palavra PYTHON temos o
Na palavra CURSO temos u o
```

28.2 – VARIÁVEIS COMPOSTAS - LISTAS

As listas se diferem das TUPLAS porque aceitam alterações de itens, durante a execução do programa, ou seja, podemos atribuir e ler novos dados para diferentes posições já existentes. Antes de exemplificarmos o uso de listas, vale dizer que **devem ser criadas entre colchetes**, diferentemente das **tuplas que são criadas dentro de parênteses**.

Vamos considerar uma lista com seguintes dados:

nomes:

	a	l	b	e	r	s	o	n	,	w	a	g	n	e	r	,	h	e	l	i	o	,	b	r	u	n	O
positivo	0			4				7		8			11			14			17			20			23			26
negativo	-8			-4				-1		-8			-5			-2			-5			-2			-5			-2

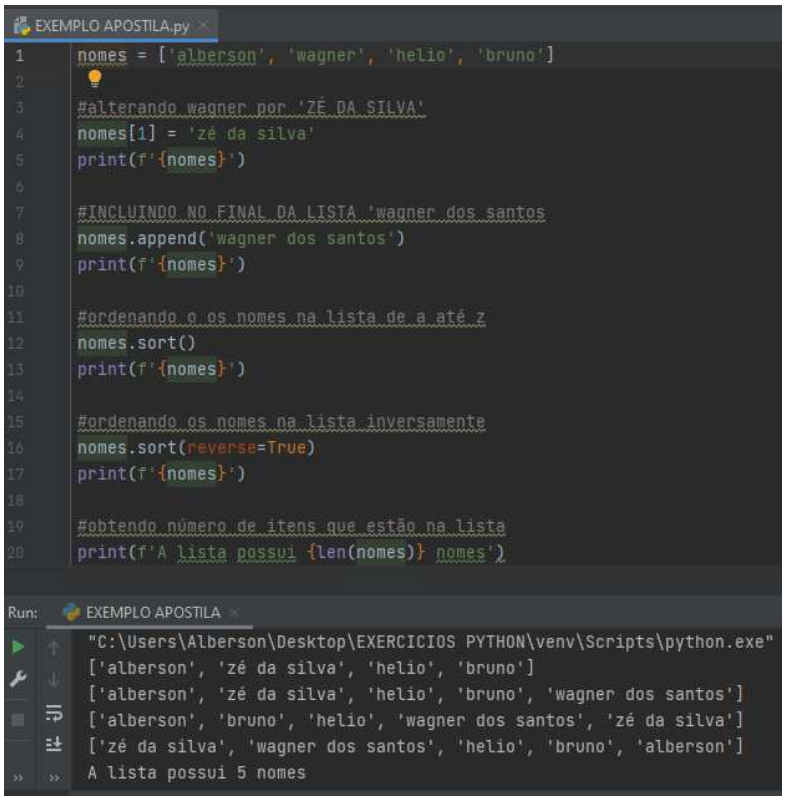
Para criá-la basta digitarmos:

```
nomes = ['alberson', 'wagner', 'helio', 'bruno']
```

Repare que a criamos quase que da mesma forma que as listas, substituindo apenas os parênteses por colchetes.

As listas apresentam algumas funcionalidades e métodos a mais que as tuplas, que veremos em exemplos a seguir.

Exemplo 1: Vou exemplificar abaixo uso de alguns métodos e funcionalidades de Listas



Observações importantes, mostrados neste programa:

a) Funcionalidade append() = usado para inserir itens no FINAL DA LISTA
b) Método sort() = usador para colocar a lista em ordem do menor para o maior valor
c) Método sort(reverse = True) = usado para colocar a lista em ordem do maior para o menor (ou seja, inversa)
d) Funcionalidade len() = usada para verificar quantos itens existem na lista
e) Na linha 4, estamos alterando o conteúdo inicial ‘wagner’ para ‘zé da silva’, o que não é possível fazer em tuplas.

Exemplo 2:

```
EXEMPLO APOSTILA.py
1  nomes = ['alberson', 'wagner', 'helio', 'bruno']
2
3  #alterando wagner por 'ZÉ DA SILVA'
4  nomes[1] = 'zé da silva'
5  print(f'{nomes}')
6
7  #na linha abaixo estamos inserindo um outro nome na
8  # posição 2, os demais itens serão reordenados
9  nomes.insert(2, 'joão oliveira')
10 print(f'{nomes}')
11
12 #removendo último nome da lista
13 nomes.pop()
14 print(f'{nomes}')
15
16 #removendo um item específico da lista. Após remoção
17 #a lista será reordenada
18 nomes.pop(1)
19 print(f'{nomes}')

Run: EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/U
['alberson', 'zé da silva', 'helio', 'bruno']
['alberson', 'zé da silva', 'joão oliveira', 'helio', 'bruno']
['alberson', 'zé da silva', 'joão oliveira', 'helio']
['alberson', 'joão oliveira', 'helio']
```

Observações importantes:

a) Funcionalidade insert(): usada para inserir um item numa posição informada da lista
b) Método pop(): usado para inserir o último item da lista
c) Método pop(<índice>): Apagará o item específico da lista, como na linha 18

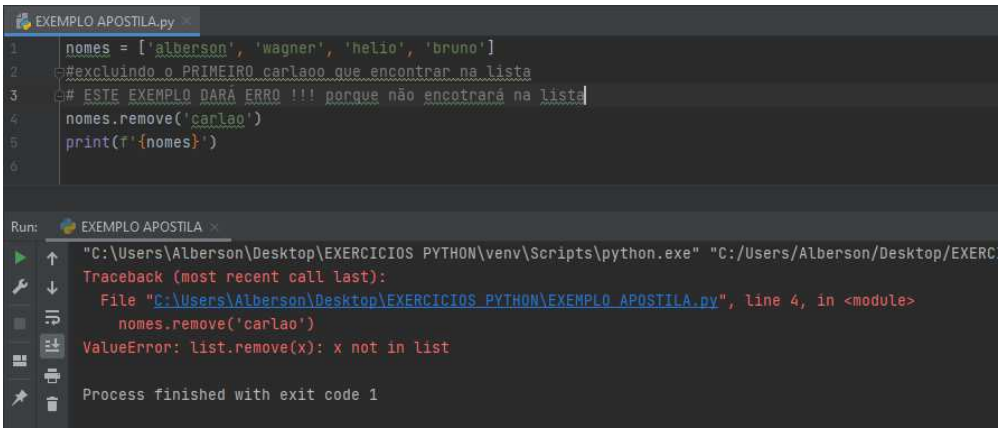
Exemplos 3:

```
EXEMPLO APOSTILA.py
1  nomes = ['alberson', 'wagner', 'helio', 'bruno']
2  #excluindo o PRIMEIRO helio que encontrar na lista
3  nomes.remove('helio')
4  print(f'{nomes}')
5
6  #para mostrar dos itens 0 e 1
7  print(f'{nomes[0:2:]}')
8
9  #para mostrar do item 1 para frente
10 print(f'{nomes[1::]}')
11
12 #para mostrar a partir da primeira posição saltando de 2 em 2
13 print(f'{nomes[::2]}')
14
15 #para mostrar a partir da segunda posição saltando de 2 em 2
16 print(f'{nomes[1::2]}')

Run: EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python
['alberson', 'wagner', 'bruno']
['alberson', 'wagner']
['wagner', 'bruno']
['alberson', 'bruno']
['wagner']

Process finished with exit code 0
```

Observações IMPORTANTES!!!

a) A funcionalidade remove(<texto_item>): Remove a PRIMEIRA ocorrência do conteúdo indicado para remoção.
b) Os demais exemplos de print demonstram que podemos usar como manipuladores de caracteres, porém trabalhando com itens da lista. Fizemos isso com string, com tuplas e agora vemos que também é possível usar em listas.

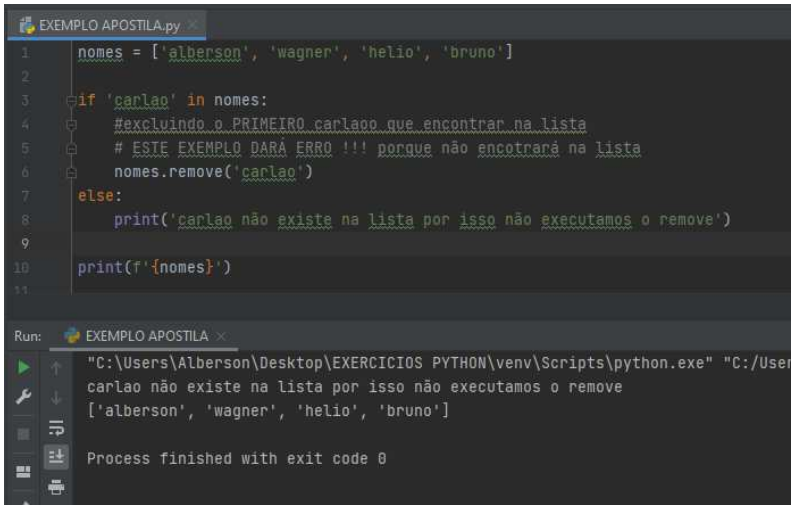
Exemplo 4:



IMPORTANTE INFORMAR QUE:

a) Quando tentamos usar o remove, indicando nome que não existe vai dar erro, conforme exemplo mostrado.

Exemplo 5: Solução para não dar erro ao usar a funcionalidade remove:



Analisemos a solução:

a) Neste caso, no comando ***if*** estou testando se o item 'carlao' existe na lista nomes. Se existir permitirá a exclusão com o remove, caso contrário, exibe mensagem conforme mostrada no exemplo.
b) Ao sair da estrutura de condição mostra os nomes da lista
c) NESTE EXEMPLO O USO DO OPERADOR *in* É MUITO IMPORTANTE NO COMANDO *if*

Exemplo 6:

```
EXEMPLO APOSTILA.py
#as duas próximas linhas mostram como criar uma lista vazia
valores1 = []
valores2 = list()
# atribuindo itens para lista valores1 pelo uso do append
valores1.append(9)
valores1.append(3)
valores1.append(1)

#mostrando lista valores1
print(f'lista valores1 {valores1}')

#Usando repetição para inserir 3 números informados
#pelo usuário na lista valores2
for x in range(1,4,1):
    valores2.append(int(input(f'Digite um numero para lista 2 item {x}')))

#Mostrando lista valores2, criada pelo usuário
print(f'lista valores2 {valores2}')
```

```
Run: EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Us
lista valores1 [9, 3, 1]
Digite um numero para lista 2 item 1 2
Digite um numero para lista 2 item 2 5
Digite um numero para lista 2 item 3 7
lista valores2 [2, 5, 7]

Process finished with exit code 0
```

Importante sobe este exemplo:

a) Podemos iniciar uma lista vazia com ***[]*** ou com método ***list()***, como exemplificado
b) Usamos funcionalidade ***append***() para o PROGRAMADOR inserir itens na lista ***valores1***
c) Com estrutura de repetições o usuário poderá entrar com 3 valores inteiros para lista ***valores2***

Exemplo7:

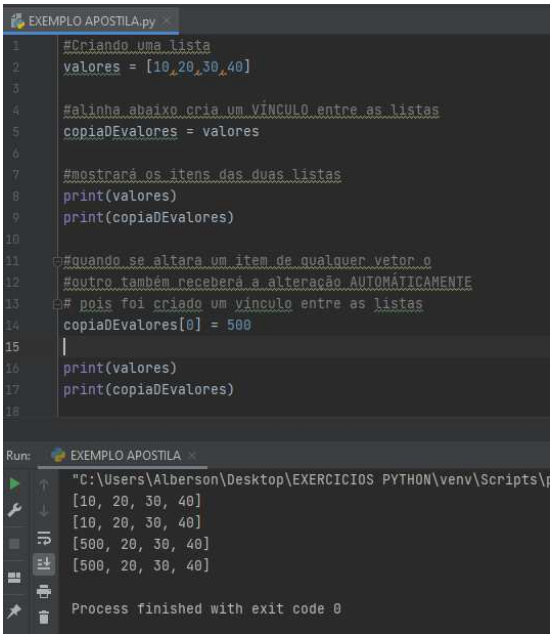
```
EXEMPLO APOSTILA.py
#Criando uma lista vazia chamada valores
valores = list()

#Usando repetição para inserir 3 números informados
#pelo usuário na lista valores2
for x in range(1,4,1):
    valores.append(int(input(f'Digite um numero para lista 2 item {x}: ')))

#mostrando cada valor armazenado pelo usuário e o indice onde foram inseridos
for indice, v in enumerate(valores):
        print(f' valor {v}, foi armazenado na posição {indice}')
```

```
Run: EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alb
Digite um numero para lista 2 item 1: 10
Digite um numero para lista 2 item 2: 20
Digite um numero para lista 2 item 3: 30
 valor 10, foi armazenado na posição 0
 valor 20, foi armazenado na posição 1
 valor 30, foi armazenado na posição 2

Process finished with exit code 0
```

Neste exemplo:

a) Estamos imprimindo os valores digitados pelo usuário numa frase dentro de uma estrutura de repetição usando ***enumerate()***

Exemplo 8: CRIANDO LISTA COM VÍNCULO (LIGAÇÃO ENTRE AS LISTAS)

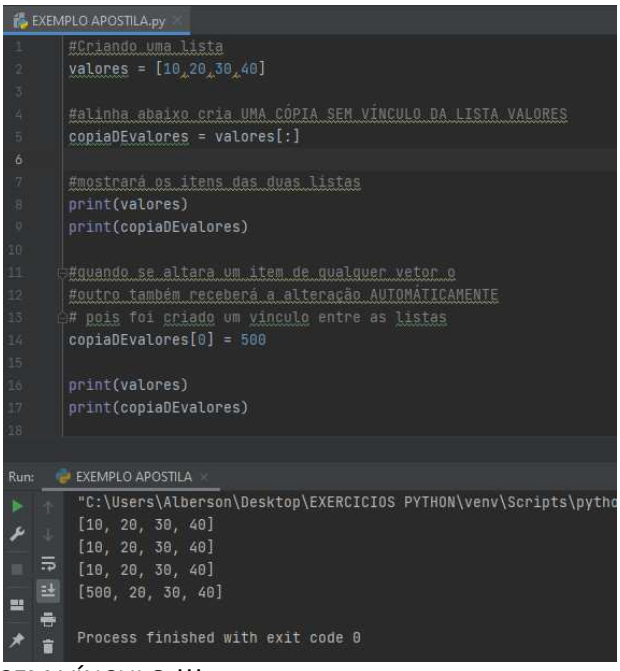


IMPORTANTE SOBRE O EXEMPLO:

a) **Na linha 5, deste exemplo, estamos criando uma *LIGAÇÃO* da lista *valores* com a lista copiaDEvalores e entre elas EXISTIRÁ SEMPRE UM VÍNCULO.**

b) **Perceba o que é o vínculo, quando alteramos um dado de uma das listas, como feito na linha 14, pois quando alteramos *o item 0 da lista copiaDEvalores*, o *item 0 da lista valores* será alterado automaticamente.**

EXEMPLO 9: CRIANDO UMA CÓPIA INDEPENDENTE DE UMA LISTA



IMPORTANTE !!!! CÓPIA DA LISTA SEM VÍNCULO !!!

a) Na linha 5 está sendo feita mera cópia independente da lista ***valores***

b) A alteração feita no ***item 0 da lista copiaDEvalores* NÃO CAUSARÁ NENHUMA MUDANÇA NO ITEM 0 DA LISTA *valores***

28.2.1-DESAFIOS GERAIS - LISTAS:

1) Fazer um programa para solicitar ao usuário a digitação de 5 números e armazená-los numa lista. Após armazená-los mostre o Maior e o Menor valores e os seus índices na lista. Caso tenha valores iguais e estes forem o maior e/ou menor números, mostre todas as posições destes na lista criada.
2)Fazer um programa que solicite ao usuário a criação de uma lista com VÁRIOS números DIFERENTES, ou seja, caso o usuário digite um valor já existente peça outro número e não insira este número repetido na lista. Quando o usuário digitar 0, deve ser finalizada a criação da lista.
3) Fazer um programa que solicite 10 números ao usuário. Não permita inclusão de números repetidos e no momento da inclusão do número na lista, o insira-o na ordem crescente e informe ao usuário a posição correta onde o número foi inserido, ou seja, seu local correto dentro da lista. **Não use a funcionalidade sort() em hipótese alguma neste programa.**
4) Fazer um programa para o usuário digitar 8 números inteiros. Inclua estes números numa lista. Crie uma lista só com os pares digitados e outra só com os ímpares digitados. Entretanto os números da primeira lista devem ser incluídos em ordem DECRESCENTE, sem usar sort()

28.3- LISTAS DENTRO DE LISTAS:

É importante dizer que nas listas, assim como em tuplas, podemos ter diferentes tipos de dados armazenados. Além disso, podemos criar listas dentro de listas, ou seja, estruturas dados dentro de novas estruturas de dados que serão criadas.

Observe os exemplos a seguir com atenção:

Exemplo 1:

EXEMPLO APOSTILA.py

```
1  #Criando uma lista chamada dados com nome e idade de pessoas
2  dados = ['alberson', 25]
3
4  #vamos criar abaixo uma nova lista para guardar
5  #cada CONJUNTO de dados numa posição especifica
6  dadoscompletos = list()
7  #vamos adicionar no final da lista dadoscompletos a estrutura dados
8  dadoscompletos.append(dados[:])
9  print(f'lista dados completos = {dadoscompletos}')
10
11 #vamos incluir NOVOS CONJUNTOS DE LISTAS dentro da lista dadoscompletos
12 dadoscompletos.append(['irene', 15])
13 dadoscompletos.append(['josé', 55])
14
15 print(f'lista dados = {dados}')
16 print(f'lista dados completos = {dadoscompletos}')
17 print(f'Item 2 da lista dadoscompletos é {dadoscompletos[2:]}')
18 print(f'dadoscompletos invertida é {dadoscompletos[-1::-1]}')
```

Run: EXEMPLO APOSTILA

```
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/
lista dados completos = [['alberson', 25]]
lista dados = ['alberson', 25]
lista dados completos = [['alberson', 25], ['irene', 15], ['josé', 55]]
Item 2 da lista dadoscompletos é [['josé', 55]]
dadoscompletos invertida é [['josé', 55], ['irene', 15], ['alberson', 25]]

Process finished with exit code 0
```

Explicando programa:

a) Foi criada a lista ***dados*** com somente um nome e uma idade
b) A estrutura da lista ***dados*** FOI INCLUÍDA no final da lista ***dadoscompletos*** na linha 8, que estava vazia por ter sido criada na linha 6;
c) LEMBRE-SE DE QUE QUANDO CRIAR UMA CÓPIA DE UMA LISTA DENTRO DE OUTRA, COMO FEITA NA LINHA 8, DE INDICAR OS ':' DENTRO DO APPEND, PARA NÃO CRIAR VÍNCULO ENTRE AS LISTAS
d) Com a linha 9 foi impressa a lista ***dadoscompletos*** e ao exibi-la, os dados ***'alberson', 25*** estão sendo mostrados envolvidos por quatro colchetes [[ ]]. Isto significa que para incluir novos itens em ***dadoscompletos***, temos que fazê-lo neste formato, ou seja, como feito nas linhas 12 e 13.
e) Para mostrar as listas contidas na lista ***dadoscompletos*** basta referenciarmos o índice do conjunto em questão. Perceba que cada conjunto está envolvido internamente sempre dentro de outros colchetes.

ILUSTRANDO A SITUAÇÃO ACIMA:

dados:

alberson	25
0	1

Dadoscompletos

alberson	25	irene	15	josé	55
0	1	0	1	0	1
0		1		2	

Exemplo 2:

```
EXEMPLO APOSTILA.py
1  #podemos criar desta forma uma nova lista de listas
2  conjuntos = [['juca', 20], ['zé', 30], ['bruna', 15]]
3
4  #exibindo os itens da lista conjuntos, criada anteriormente
5  print(f'conjunto = {conjuntos [:]}')
6
7  #incluindo o conjunto 'leo', 45 no final da lista de listas
8  conjuntos.append(['leo', 45])
9
10 #inserindo  o conjunto 'alberto', 70 na posição 1 que era ocupada pelo zé
11 conjuntos.insert(1, ['alberto', 70])
12
13 #mostrando o resultado final dos comandos acima
14 print(f'conjunto = {conjuntos [:]}')
15

Run:  EXEMPLO APOSTILA
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/De
conjunto = [['juca', 20], ['zé', 30], ['bruna', 15]]
conjunto = [['juca', 20], ['alberto', 70], ['zé', 30], ['bruna', 15], ['leo', 45]]

Process finished with exit code 0
```

Vamos ver o que aconteceu acima:

a) (linha 2) - Criamos uma lista ***conjuntos*** com 3 listas internas
b) (linha 8) – adicionamos ao final da lista ***conjuntos*** o nome ***leo*** e sua idade ***45***
c) (linha 11) – Inserirmos na posição 1 da lista ***conjuntos*** o nome ***alberto*** e sua idade ***70***, empurrando com isso os demais itens para posições subsequentes.
d) (linha 14) – exibe-se o resultado final dos comandos executados.

ATENÇÃO : TODOS OS MÉTODOS E FUNCIONALIDADES ENSINADOS ANTERIOREMENTE PARA LISTAS SIMPLES, FUNCIONARÃO PARA LISTAS DE LISTAS.

ILUSTRANDO O CASO ACIMA:

conjuntos:

juca	20	zé	30	bruna	15
0	1	0	1	0	1
0		1		2	

Após a execução do programa, teremos:

conjuntos:

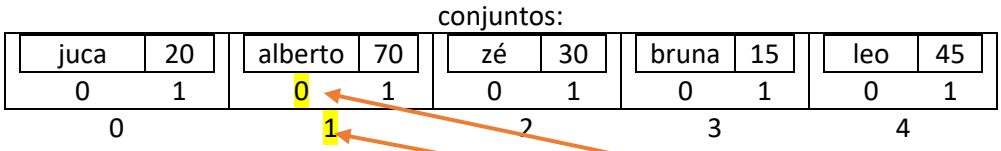
juca	20	alberto	70	zé	30	bruna	15	leo	45
0	1	0	1	0	1	0	1	0	1
0		1		2		3		4	

Exemplo 3:

```
#podemos criar desta forma uma nova lista de listas
conjuntos = [['juca', 20], ['zé', 30], ['bruna', 15]]
#incluindo a lista ['leo', 45] no final da lista de lista conjuntos
conjuntos.append(['leo', 45])
#inserindo  a lista ['alberto', 70] na posição 1 que era ocupada pelo zé em conjuntos
conjuntos.insert(1, ['alberto', 70])

#mostrando a lista toda
print(f'conjunto = {conjuntos}')

#mostrando o nome contindo no primeiro item da lista conjunto
print(f'nome da segunda lista, inserida na lista conjuntos = {conjuntos [1][0]}')
```

```
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/De
conjunto = [['juca', 20], ['alberto', 70], ['zé', 30], ['bruna', 15], ['leo', 45]]
nome da segunda lista, inserida na lista conjuntos = alberto

Process finished with exit code 0
```

Para entender melhor temos os seguintes dados na lista ***conjuntos:***




```
print(f'nome da segunda lista, inserida na lista conjuntos = {conjuntos [1][0]}')
```

No comando print:

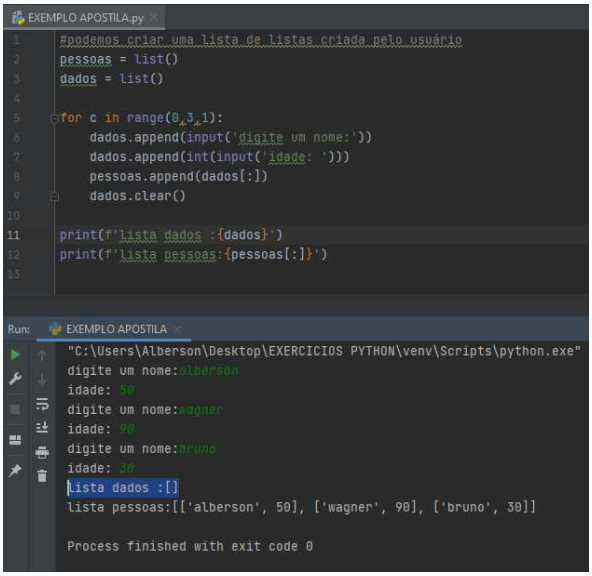
a) (linha 12) a referência indica nos primeiros colchetes qual o item de conjuntos e nos segundos colchetes o dado da posição 0.

Exemplo 4: várias formas de imprimir conteúdos de listas compostas por listas:

```
#podemos criar desta forma uma nova lista de listas
pessoas = [['juca', 20], ['zé', 30], ['bruna', 15]]

#imprimindo cada lista da lista pessoas
for p in pessoas:
    print(f'{p}')

#imprimindo os dados contantes na lista pessoas em frases
for nome, idade in pessoas:
    print(f' nome: {nome} idade: {idade}')

for indice, dados in enumerate(pessoas):
    print(f'pessoa[{indice}] é nome: {dados[0]} idade: {dados[1]}')
```

```
['juca', 20]
['zé', 30]
['bruna', 15]
 nome: juca idade: 20
 nome: zé idade: 30
 nome: bruna idade: 15
pessoa[0] é nome: juca idade: 20
pessoa[1] é nome: zé idade: 30
pessoa[2] é nome: bruna idade: 15

Process finished with exit code 0
```

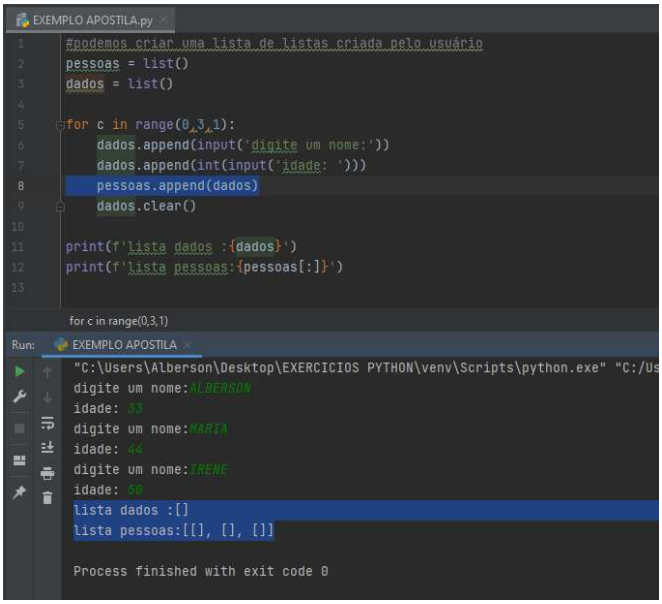
Exemplo 5:



IMPORTANTE OBSERVAR:

a) (linhas 2 e 3) Criadas duas listas vazias
b) (linha 5) Estrutura de repetição para usuário digitar e armazenar na ***lista dados*** o nome e a idade de UMA PESSOA
c) (linha 8) Inserindo o nome e idade armazenados na ***lista dados***, na ***lista pessoas***
d) (linha 9) USAMOS O MÉTODO CLEAR PARA LIMPAR TODOS OS ITENS DA LISTA ***dados.*** Assim sendo, a lista ***dados*** não armazena vários conjuntos de nomes e idades, ou seja, sempre guardará o nome e idade que acabou de ser armazenado e logo após a inclusão na ***lista pessoa*** serão apagados da ***lista dados***.
e) (linha 11) imprime a lista dados vazia
f) (linha12) imprime a ***lista pessoas*** com os dados que foram sendo copiados da ***lista dados***.

Exemplo 6: CUIDADO PARA NÃO FAZER CÓPIA COM VÍNCULO DE UMA LISTA NA OUTRA, VEJA O QUE PODE ACONTECER:



CUIDADO !!!! VEJA O QUE ACONTECEU:

1) NA LINHA 8, PELO FATO DE TERMOS COPIADO A LISTA *dados* COM VÍNCULOS PARA LISTA *pessoas*, TODO MOMENTO QUE LIMPAMOS A LISTA DADOS NA LINHA 9, A LISTA VINCULADA *pessoas* TAMBÉM SERÁ APAGADA.

28.3.1-DESAFIOS GERAIS – LISTAS DE LISTAS:

1) Criar uma lista de listas contendo vários nomes e salários de pessoas. Mostre:
   a) total acumulado de salários digitados
   b) Informe quantas pessoas foram informadas
   c) menor salário informado (mesmo que em duplicidade)

2) Digite 10 números inteiros e os armazene em uma lista na ordem que foram digitados. Crie duas outras listas, uma de números pares e outra de ímpares, ambas em ordem crescente.

3) Crie uma lista que simule armazenamento de números inteiros em uma matriz de dimensão 3x3. Após armazenado mostre a matriz formatada na tela como se apresenta normalmente, ou seja, uma tabela de valores.

4) Melhore o exercício 3 de tal forma que após armazenados os números, seja feita a soma dos valores e seja mostrado os dados contidos na diagonal principal desta matriz.

5) Faça um programa para armazenar em N listas os palpites da mega sena gerados pelo computador. O usuário deve informar quantos jogos quer simular. Lembre-se de que para cada jogo não podem ser gerados números repetidos.

6) Faça um programa que armazene numa lista diversos nomes e matrículas de alunos, bem como suas duas notas tiradas num bimestre. Logo após, o programa deve pedir para que seja mostrada a situação final de cada aluno, ou seja, o usuário informa a matrícula e a média final é mostrada acompanhada do nome do aluno.

28.4- VARIÁVEIS COMPOSTAS - DICIONÁRIOS:

Os dicionários são um tipo de estrutura de dados cujo os índices de cada posição podem ser LITERAIS, ou seja, podemos DAR NOMES a cada posição que armazenará um valor qualquer.

Para criar um dicionário usaremos ***{ }***, OU ENTÃO, ***dict()***.

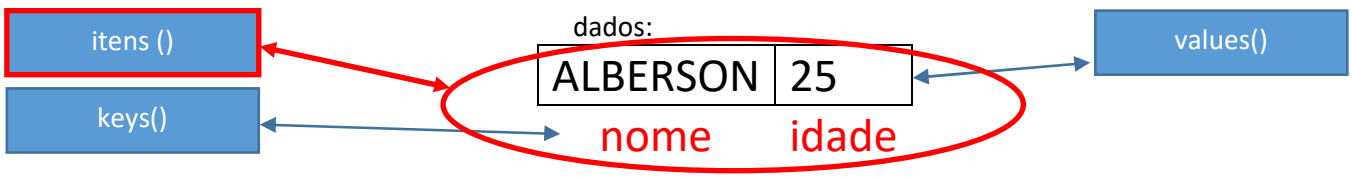
Sintaxe:

<nomevariável> = dict()
Ou
<nomevariável> = { }

Exemplos:

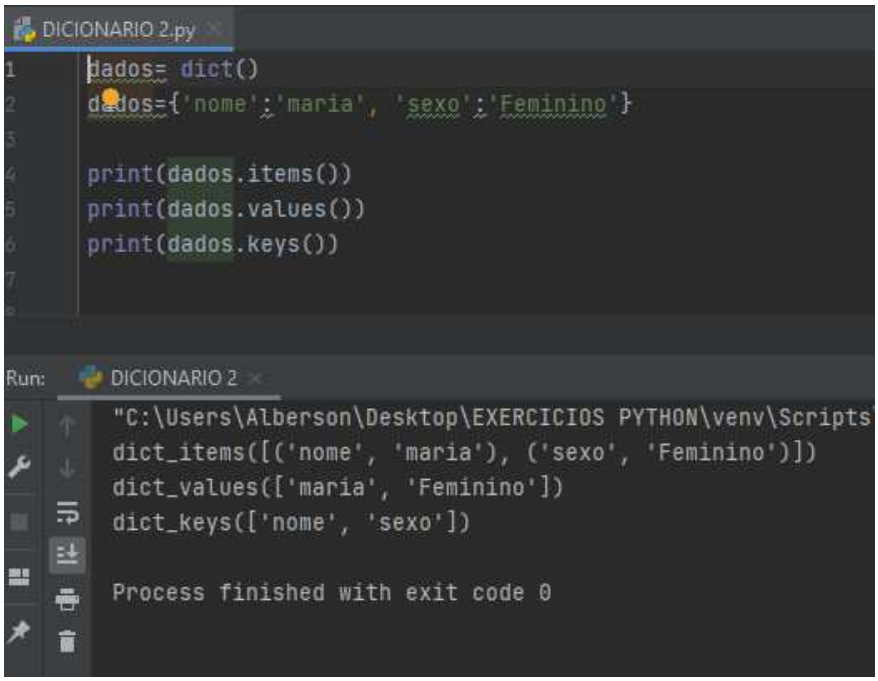
dados = {'nome':'ALBERSON' , 'idade':25}

Neste caso acima criaremos a seguinte estrutura:



PERCEBA QUE OS ÍNDICES DESTES DADOS ARMAZENADOS NA ESTRUTURA ACIMA SÃO: ***nome*** E ***idade*** e não mais índices 0 e 1.

Exemplo 1:



Observe acima como serão impressos os itens (items()) do dicionário, os valores (values()) e também as chaves (Keys()). Veja também, que a impressão do itens foi mostrada em forma de composição de elementos, cada elemento dentro de um parênteses da lista criada.

Exemplo 2: Podemos também optar por imprimir das seguintes formas:

```
DICIONARIO1.py
1  dados = {'nome': 'ALBERSON', 'idade':25,  'peso': 70.5}
2
3  #imprimindo o nome armazenado no indice 'nome':
4  #repare que será  necessário o uso das as duplas dentro do colchetes
5  #senão o print mostrará erro, por conta do uso da aspa simples
6  #já ser usada para delimitar os dados da impressão
7  print(f'{dados["nome"]}')
8  print(f'tem {dados["idade"]} anos e pesa {dados["peso"]} kg')
9
10 #poderiamos usar também somente o nome da estrututa e a referência
11 #da estrutura de dicionário com o indice desejado, veja:
12 print(dados['nome'])
13 print(dados['idade'])
14 print(dados['peso'])
15

Run:  DICIONARIO1
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C
ALBERSON
tem 25 anos e pesa 70.5 kg
ALBERSON
25
70.5
```

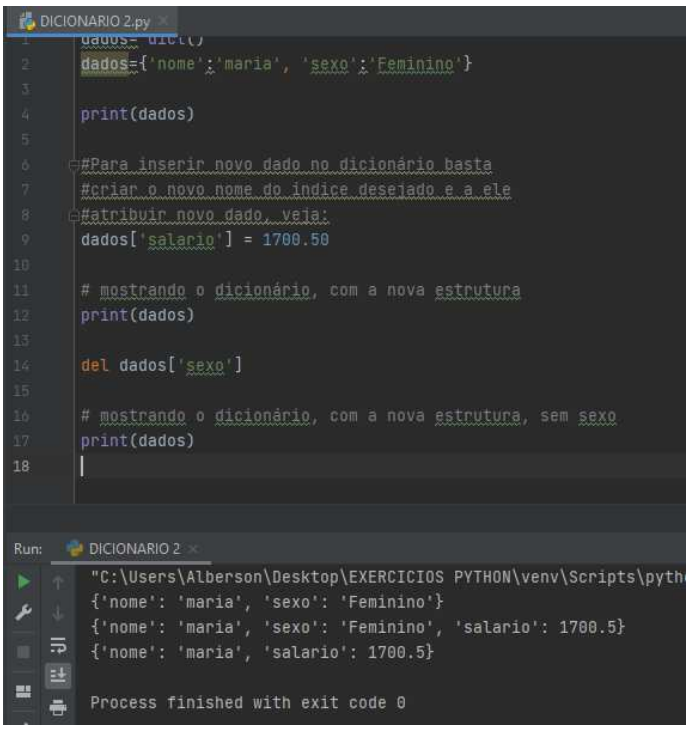
Exemplo 3: ADICIONANDO ELEMENTOS AO DICIONÁRIO:

```
DICIONARIO 2.py
1  dados= dict()
2  dados={'nome':'maria', 'sexo':'Feminino'}
3
4  print(dados)
5
6  #Para inserir novo dado no dicionário basta
7  #criar o novo nome do indice desejado e a ele
8  #atribuir novo dado, veja:
9  dados['salario'] = 1700.50
10
11 # mostrando o dicionário, com a nova estrutura
12 print(dados)
13

Run:  DICIONARIO 2
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\p
{'nome': 'maria', 'sexo': 'Feminino'}
{'nome': 'maria', 'sexo': 'Feminino', 'salario': 1700.5}

Process finished with exit code 0
```

Observem:

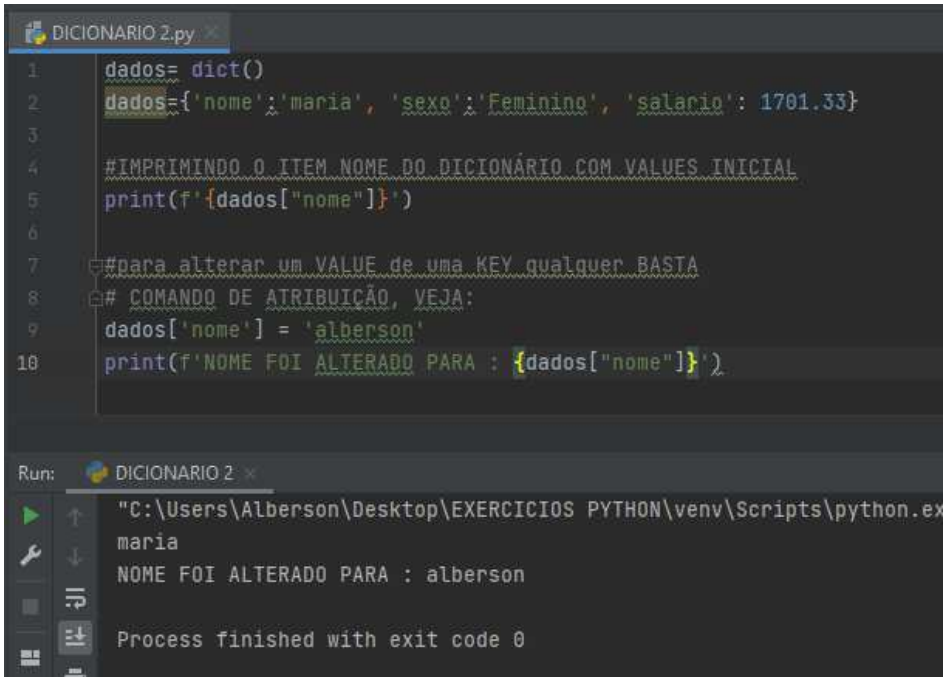
a) Foi criado um dicionário chamado ***dados***
b) Imprimimos primeiramente a estrutura inicial criada
c) Na linha 9, adicionamos o índice ***salario*** que receberá também o valor ***1700.50***
d) Por fim, mostramos a nova estrutura do dicionário criado.

Exemplo 4: REMOVENDO ELEMENTO DO DICIONÁRIO COM O COMANDO ***del***



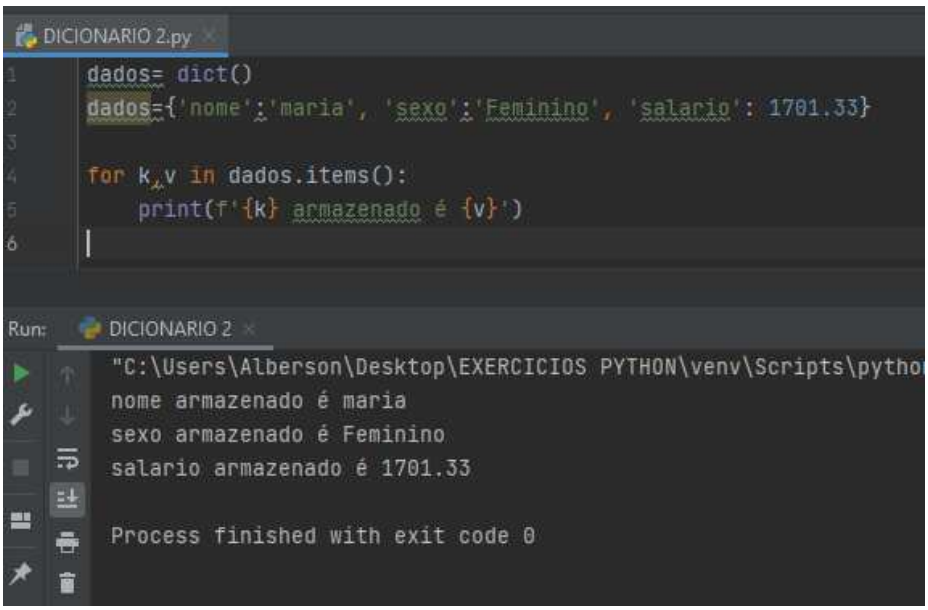
Observe: Neste exemplo o comando escrevemos o comando ***del dados[‘sexo’]*** que exclui este elemento do dicionário bem como o valor nele armazenado.

Exemplo 5: ALTERANDO VALUES DE UM DICIONÁRIO



Neste exemplo, alteramos o nome armazenado incialmente: ‘***maria’*** para ***‘alberson’.***

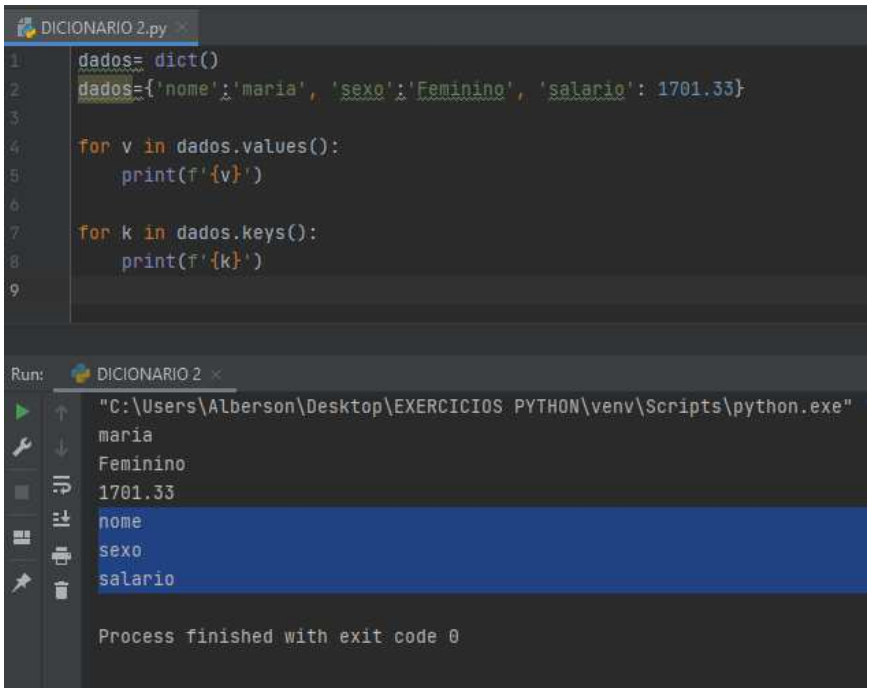
Exemplo 6: Usando estruturas de repetições para percorrer cada elemento da estrutura



O exemplo acima é bastante interessante, repare:

a) Criamos uma estrutura de repetição para percorrer cada chave (Keys()) e cada valor (valeus()) entre os itens(items()) armazenados
b) A variável ***K*** foi usada para armazenar cada nome de chave e a variável ***v*** foi usada para armazenar cada valor armazenado como itens.
c) À medida que o laço é executado são exibidas as frases formatadas
d) <u>PERCEBERAM QUE EM DICIONÁRIO NÃO USAMOS O enumerate() E SIM O items()</u>

Exemplo 7: Podemos percorrer também somente os keys() ou só os values(), veja os dois exemplos no código a seguir:



Explicando melhor:

a) O primeiro laço de repetição percorre os valores do dicionário e os exibe. IMPORTANTE DIZER QUE A VARIÁVEL QUE CONTROLA O LAÇO NÃO PRECISARIA SER CHAMADA DE ***v***
b) O segundo laço de repetição percorre as chaves do dicionário e as exibe. IMPORTANTE DIZER QUE A VARIÁVEL QUE CONTROLA O LAÇO NÃO PRECISARIA SER CHAMADA DE ***k***

ATENÇÃO !!!!

PODEMOS DESENVOLVER PROGRAMAS JUNTANDO LISTAS, TUPLAS E DICIONÁRIOS

Vamos observar atentamente a estrutura a seguir e montar vários DICIONÁRIOS dentro de uma LISTA. PARA EXEMPLIFICAR, criarei um a lista de clientes contendo vários dicionários com nome, idade e sexo:

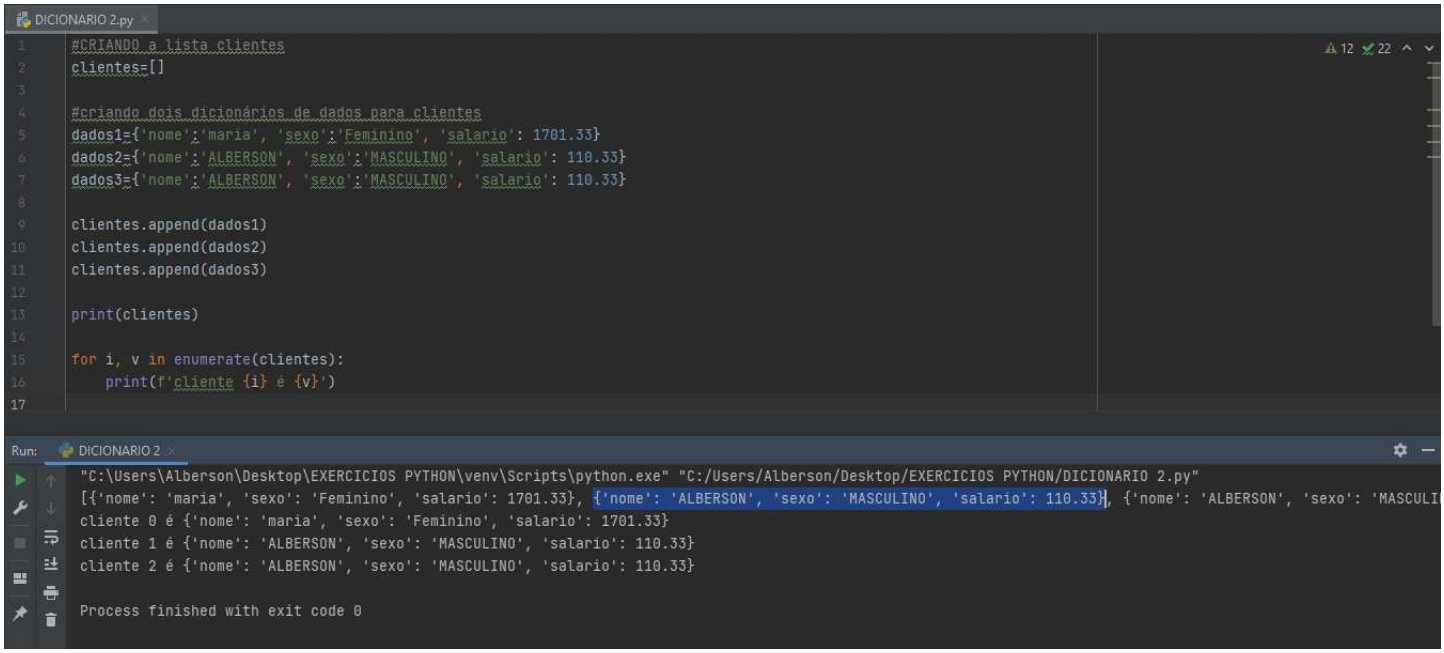
CLIENTES

MARIA	FEMININO	1701,35
nome	SEXO	SALARIO
0		

ALBERSON	MASCULINO	110,33
nome	SEXO	SALARIO
1		

ALBERSON	MASCULINO	110,33
nome	SEXO	SALARIO
2		

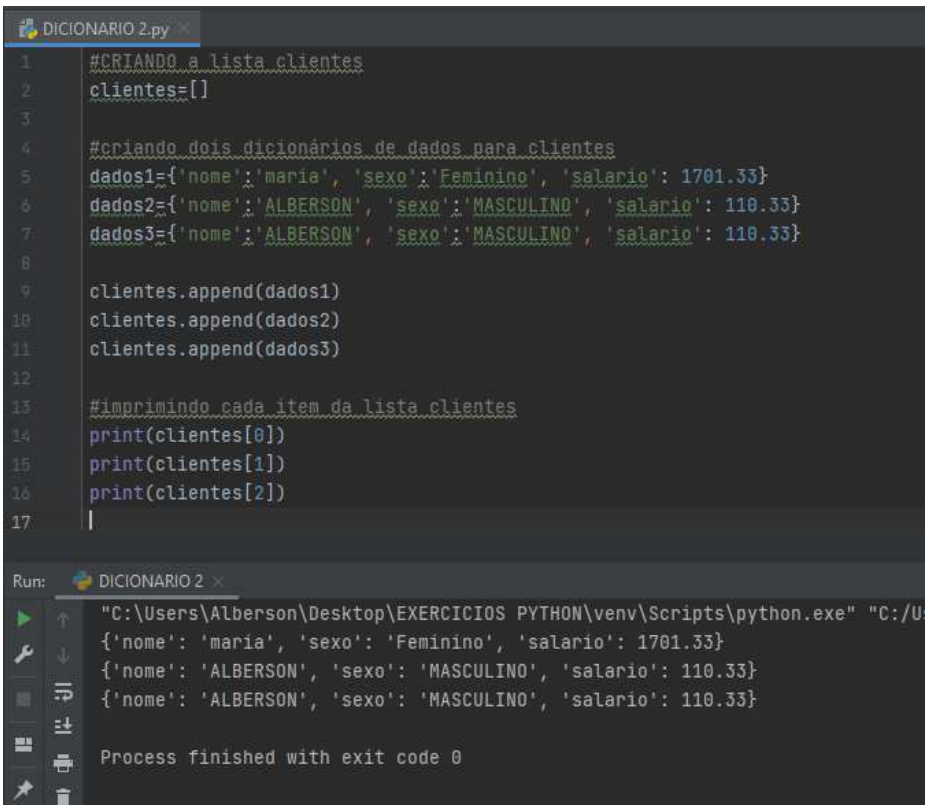
Exemplo 8: PARA JUNTAR VÁRIOS DICIONÁRIOS EM UMA LISTA, FAREMOS O SEGUINTE:



Neste exemplo:

a) Usamos o ***append()*** para adicionar no os dados dos 3 dicionários em uma lista.
b) Após a inclusão dos dicionários na lista ***clientes,*** mandei imprimir os itens adicionados linha de comando 13.
c) O laço for, percorre a lista e usa ***i*** para referenciar o índice e ***v*** para imprmir o dado contido na posição da lista, para isso, torna-se necessário o uso do ***enumerate().***

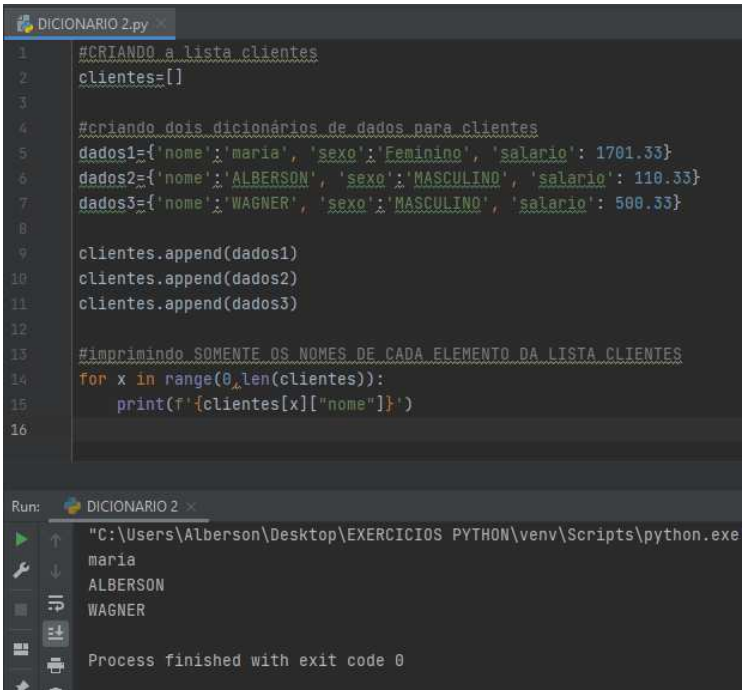
Exemplo 9: Outras formas de imprimir os dicionários contidos na lista:



É importante lembrar:

a) As linhas 14, 15, e 16 deste exemplo estão imprimindo dados contidos em cada posição da LISTA clientes.

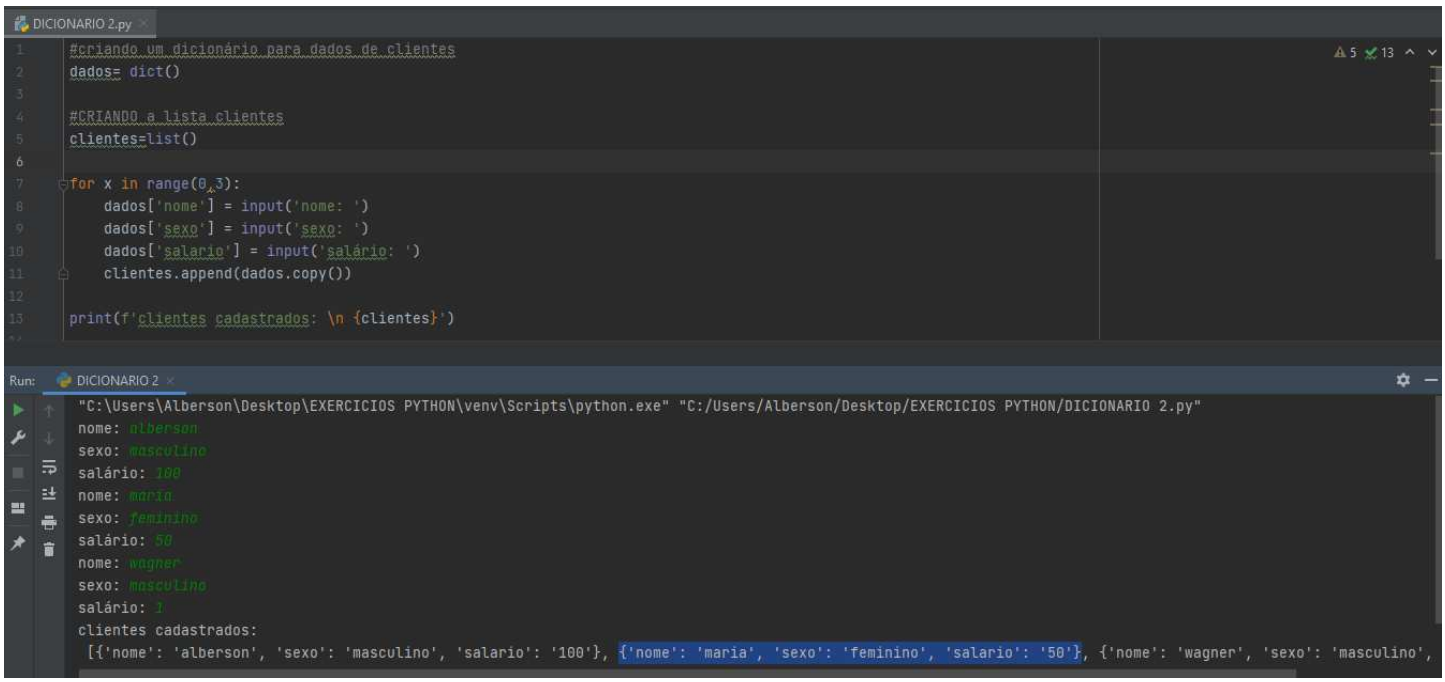
Exemplo 10:



Sobre o comando ***for(),*** acima:

a) Percorrerá toda a lista criada e vai imprimir o ***'nome'*** de cada item ***x***

Exemplo 11:

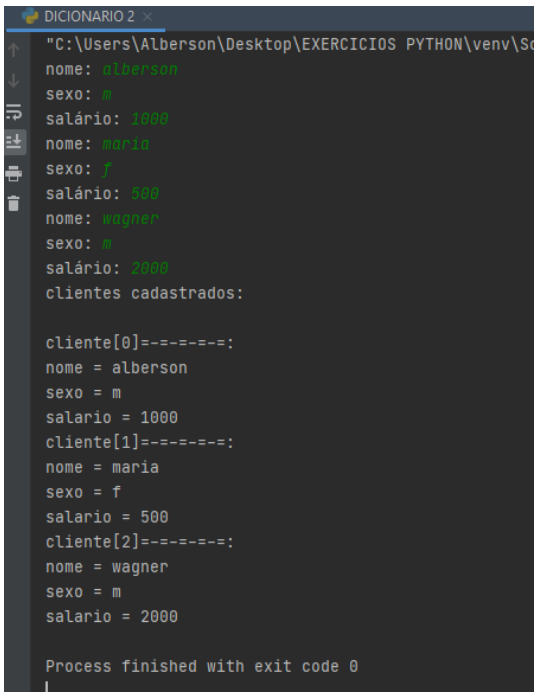


Neste exemplo foi criada uma estrutura de repetição ***for,*** para:

a) entrar com dados digitados pelo usuário no dicionário ***dados***.
b) Depois damos um ***append(),*** dos dados copiados com ***copy()*** , para dentro de clientes. NÃO PODEMOS USAR FATIAMENTO (:) NO LUGAR DO COPY, pois o dicionário não aceita este método.

Exemplo 12: Este próximo exemplo é bem interessante, avalie:

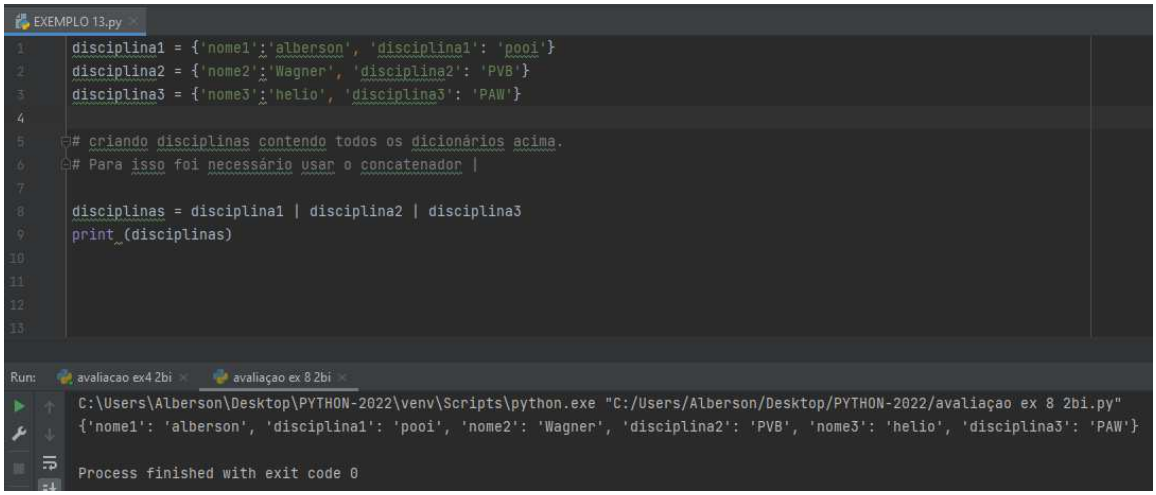
Após a execução teremos:



Repare aqui:

a) **Na linha 13:** laço criado para percorrer os itens da LISTA CLIENTES. Usamos o ***enumarate()*** para pegar o índice numérico da posição da LISTA CLIENTES, referente a cada cliente nela incluído.
b) **Na linha 14:** esta linha imprime a mensagem informando qual a posição do cliente que será impresso, dentro da LISTA CLIENTES
c) **Na linha 15:** Nesta linha percorremos todos dados DOS DICIONÁRIOS INCLUSOS NA LISTA CLIENTES e vamos imprimir os keys() e o values() de cada elemento.

Exemplo 13: Este próximo exemplo É IMPORTANTE, OBSERVE:



Neste exemplo:

a) **Linha 1 até linha 3** – Criei 3 dicionários distintos
b) **Linha 8** – Juntei todos os 3 dicionários criados anteriormente num único chamado disciplinas, usando |
c) **Linha 9** – Exibindo dicionário disciplinas

Atenção: Para juntar dicionários com | (pipeline) os índices devem ser DISTINTOS

28.4.1-DESAFIOS GERAIS – DICIONÁRIOS:

1) Crie um programa para inserir num dicionário o nome e a média de um aluno. Diante da média, insira também no dicionário o resultado do aluno, a saber:
   a. Média de 0 até 5.9 – “aluno em recuperação”
   b. Média maior ou igual a 6.0 – “Aluno na média”

   Após armazenar a situação no dicionário imprima o nome, a média e a situação armazenadas no dicionário
2) Fazer um programa para inserir num dicionário o nome de 4 jogadores e um número sorteado entre 1 e 6, que simulará que o determinado jogador atirou um dado. Após armazenados os 4 nomes e os números dos dados de cada jogador, classifique os jogadores, sabendo que o vencedor foi o que tirou maior número no dado.
   Veja exemplo abaixo:

```
Valores sorteados:
   O jogador1 tirou 5
   O jogador2 tirou 2
   O jogador3 tirou 6
   O jogador4 tirou 1
Ranking dos jogadores:
   1º lugar: jogador3 com 6
   2º lugar: jogador1 com 5
   3º lugar: jogador2 com 2
   4º lugar: jogador4 com 1
```

3) Crie um programa para solicitar a leitura de um nome de um trabalhador, ano do seu nascimento e a data que iniciou sua carreira profissional. Guarde em um dicionário o nome do trabalhador, quantos anos ele tem (diante do ano informado pelo usuário) e quantos anos de trabalho ele tem (diante da data inicial de trabalho). Sabendo que normalmente uma pessoal se aposenta com pelo menos 30 anos de contribuição, mostre o nome do trabalhador, quantos anos faltam para se aposentar e qual será sua idade quando aposentar.
4) Fazer um programa para solicitar o nome de um time de futebol e quantas partidas disputou. Logo depois, o programa deve perguntar ao usuário quantos gols foram feitos pelo time **em cada partida**. Ao finalizar as digitações dos números de gols por partida, armazene num dicionário: nome do time, quantas partidas foram disputadas, as quantidades de gols informadas em cada partida e a SOMA TOTAL DE GOLS DO TIME. No final mostre os dados armazenados de tal forma que o usuário entenda o que está sendo impresso. OBSERVAÇÃO (deve-se ter no dicionário, uma lista com cada quantidade de gols feitos por partida)
   Exemplo de armazenamento e impressão dos resultados:

   {‘time‘: ‘palmeiras’, ‘gols por partida’: [2, 4, 3, 0, 0, 0,1], ‘total de gols:’, 10}

   Imprima da seguinte forma:
   Perceba que o nome do time é palmeiras e disputou 7 partidas, fez os seguintes gols:
   Partida 1 : 2 gols
   Partida 2 : 4 gols
   Partida 3 : 3 gols
   Partida 4 : 0 gols
   Partida 5 : 0 gols
   Partida 6 : 0 gols
   Partida 7 : 1 gols
   Total de gols ===> 10