Exercícios resolvidos com **for , while** e **listas**

Este documento apresenta uma série de exercícios resolvidos que demonstram a utilização dos comandos de repetição **for** , **while** e **listas**.

EXERCÍCIO 1: Escreva um programa que receba a nota de 10 alunos como entrada e que imprima qual delas é a maior e qual é a média da turma.

RESOLUÇÃO:

```
MAX = 10   # número de valores que serão digitados

print('Programa Maior Nota')
print('==================')
print('Digite ', MAX, ' Notas:')

maior_nota = 0 # não preciso do -math.inf, pois não há nota menor que 0
soma = 0

for i in range(MAX):
    print('Nota', i+1, ':')
    nota = float(input())
    soma += nota
    if (nota > maior_nota): maior_nota = nota

print(f'A maior nota digitada foi {maior_nota}')
print(f'A média da turma é{soma / MAX: .2f}')
```

Observe que neste programa utilizamos a variável MAX (de valor 10) bem no início do programa para definir o limite superior do `range()` e, por consequência, o número de iterações do `for`. Desta maneira fica muito simples mudar o programa para que ele passe a aceitar como entrada 30 notas ou 50 notas ou qualquer outro valor (basta mexer no valor de MAX, especificado na linha 1).

EXERCÍCIO 2: Utilizando o comando for, construir um programa Python que leia um número inteiro positivo n como entrada e calcule e exiba na tela o valor de n! (ou seja, o fatorial de n). Obs: se o programa receber um valor negativo como entrada, informar ao usuário que é impossível calcular o fatorial.

RESOLUÇÃO:

```
# lê n
print('CALCULO DO FATORIAL')
print('*******************')
n = int(input('Digite n (inteiro positivo): '))

# calcula n!
fatorial = 1
for i in range(2, n+1):
    fatorial = fatorial * i

# imprime o resultado
if n  >= 0:
    print(f'{n}! = {fatorial}')
else:
    print('Não é possível calcular o fatorial de um número negativo')
```

Neste exemplo, um laço montado com "i" variando entre 2 e "n" (número digitado pelo usuário) é utilizado para o cálculo do fatorial. Para isso, bastou fazer `range(2, n+1)`. Se o usuário digitar o valor 0 será gerado (sequência de 1 a 0) `range(1,1)` e se digitar 1 será `range(1,2)`, ambas impossíveis de serem geradas, o que fará com que o programa não entre no `for` e mantenha o fatorial com o valor 1.

EXERCÍCIO 3: Faça um programa que receba como entrada uma palavra e um número *n* e que imprima na tela essa palavra *n* vezes conforme mostrado abaixo:

```
Digite uma palavra: ENCE
Quantas vezes vai repetir: 7
ENCE
 ENCE
  ENCE
   ENCE
    ENCE
     ENCE
      ENCE
```

RESOLUÇÃO:

```
# apenas vai
palavra = input('Digite uma palavra: ')
n = int(input('Quantas vezes vai repetir: '))

for k in range(n): print(k * ' ' + palavra)
```

Neste exemplo, o operador de repetição de strings * (asterisco) foi usado para repetir k vezes espaço em branco em cada linha (`k * ' '`)

EXERCÍCIO 4: Modifique o programa anterior para fazer a palavra ou caractere digitado "ir e voltar", formando uma seta, conforme mostrado a seguir. No primeiro exemplo *n* = 7 e no segundo *n* = 8. Veja que quando *n* é ímpar a "seta" fica com "bico único" e quando é ímpar fica com "bico duplo"

```
Digite uma palavra ou caractere: #
Quantas vezes vai repetir: 7
#
 #
  #
   #
  #
 #
#
```

```
Digite uma palavra ou caractere: #
Quantas vezes vai repetir: 8
#
 #
  #
   #
   #
  #
 #
#
```

RESOLUÇÃO:

```
# vai e volta
palavra = input('Digite uma palavra: ')
n = int(input('Quantas vezes vai repetir: '))

# ida
for k in range(n // 2): print(k * ' ' + palavra)

# se n for ímpar tem que imprimir o "bico"
if n % 2 == 1: print((k + 1)  * ' ' + palavra)

# volta
for k in reversed(range(n//2)): print(k * ' ' + palavra)
```

EXERCÍCIO 5: **O número 3025 possui a seguinte característica:**

30 + 25 = 55
552 = 3025

Crie um programa que pesquise e imprima todos os números de quatro algarismos que apresentem tais características.

RESOLUÇÃO:

```
for n in range(1000, 10000):
    a = n // 100 # dois primeiros dígitos
    b = n % 100  # dois últimos dígitos

    if (a + b)**2 == n: print(n)
```

```
>>>
2025
3025
980
```

EXERCÍCIO 6: **Faça um programa que imprima a tabuada de 1 a 10**

RESOLUÇÃO:

```
for i in range(1,11):
    print('TABUADA DE', i)
    print('============')
    for j in range(1,11):
        print(i,'x',j,'=',i*j)
    print()
```

Neste exemplo, é preciso usar laços aninhados para multiplicar cada i (variando de 1 a 10) por cada j (variando também de 1 a 10).

EXERCÍCIO 7: **Faça um programa que registra a quantidade de votos para eliminação no BBB de 3 candidatos : Akiko, Dimitrov e Rakesh. Os votos devem ser registrados até que se digite -1. No final indique quem foi eliminado. Caso haja empate (duplo ou triplo) para mais votado, considere que todos os candidatos empatados serão eliminados.**

RESOLUÇÃO:

```
a = d = r = 0    # votos em Akiko, Dimitrov e Rakesh
voto_atual = 99 # usado para registrar voto atual e mantém loop
maior = 0       # registrará qual é a maior quantidade de votos
while (voto_atual != -1): # recebe os votos
    print('Voto\n====')
    print('1-Akiko\n2-Dimitrov\n3-Rakesh\n-1-Encerrar')
    voto_atual = int(input())
    if voto_atual == 1:
        a += 1
        if a >= maior: maior = a
    elif voto_atual == 2:
        d += 1
        if d >= maior: maior = d
    elif voto_atual == 3:
        r += 1
        if r >= maior: maior = r

# diz quem saiu
if a == maior: print('Akiko eliminado -', maior, 'votos')
if d == maior: print('Dimitrov eliminado -', maior, 'votos')
if r == maior: print('Rakesh eliminado -', maior, 'votos')
```

Neste exercício, não é possível usar `for`, pois não sabemos quando -1 será digitado. O segredo deste programa para tratar os empates, é manter o maior número de votos na variável "maior". Ao final do laço, bastará ver quais candidatos têm número de votos igual ao valor de "maior".

EXERCÍCIO 8: **Faça um programa que divida sucessivamente por 2 um número real digitado pelo usuário e imprima o resultado da divisão enquanto esse número for maior que 0.1**

RESOLUÇÃO:

```
N = float(input('Digite N: '))

while N / 2 > 0.1:
    print(N, '/ 2 =',N/2)
    N = N / 2
```

É mais simples resolver esse exercício com while, pois assim não precisamos nos preocupar em tentar calcular de antemão o número de divisões que serão realizadas.

EXERCÍCIO 9: **Considere uma lista em memória com dez números reais. Faça um programa que calcule e imprima:**

- **A média dos valores.**
- **A quantidade de números negativos**
- **A soma dos números positivos dessa lista.**

RESOLUÇÃO:

```
# lista usada para exemplo
# (mas o programa tem que funcionar para qualquer lista com 10 números)
l = [7.5, 6.5, 5.0, 3.0, 0.5, 9.2, 7.8, 9.1, 7.9, -1.5]

# -------- aqui começa o que foi pedido no enunciado --------------------

media = sum(l) / len(l)  # para a média, não preciso de loop

qtd_neg = 0
soma_pos = 0

for numero in l:
    if numero < 0: # negativo, atualizo quantidade
        qtd_neg += 1
    else:          # positivo, atualizo soma
        soma_pos += numero

print(f'média: {media:.2f}')
print(f'total de negativos: {qtd_neg}')
print(f'soma dos positivos: {soma_pos}')
```

EXERCÍCIO 10: **Dada uma lista com 3 elementos, faça um programa que utilize as funções max(), min() e sum() para dizer qual é o menor, o maior e o do meio.**

RESOLUÇÃO:

```
# lista usada para exemplo
# (mas o programa tem que funcionar para qualquer lista com 3 números)
w = [50, 30, 40]

# -------- aqui começa o que foi pedido no enunciado --------------------

menor = min(w)
maior = max(w)
do_meio = sum(w) - min(w) - max(w)

print(menor, do_meio, maior)
```

EXERCÍCIO 11: **Faça um programa que imprima uma lista com 7 elementos antes e depois dos elementos de índice 5, 4 e 0 serem removidos (nesta ordem)**

RESOLUÇÃO:

```
# lista usada para exemplo
# (mas o programa tem que funcionar para qualquer lista de 7 elementos)
letras = ['A', 'B', 'C', 'D', 'E', 'F', 'G']

# -------- aqui começa o que foi pedido no enunciado --------------------

letras.pop(5)
letras.pop(4)
letras.pop(0)

print(letras)
```

EXERCÍCIO 12: **Crie um programa que altere uma lista *w* de *n* elementos da seguinte forma:**

- **Os elementos de índice par deverão ser multiplicados por 3**
- **Os elementos de índice ímpar deverão ser divididos por 2**

RESOLUÇÃO:

```
# lista usada para exemplo
# (mas o programa tem que funcionar para qualquer lista de qualquer tamanho!!!)
w = [3, 8, 4, 2]

# -------- aqui começa o que foi pedido no enunciado --------------------

n = len(w) # obtém n, que é o tamanho de w

for i in range(n):   # loop nos índices
    if i % 2 == 0:   # índice par, multiplico valor da posição i por 3
        w[i] *= 3
    else:            # índice ímpar, divido valor da posição i por 2
        w[i] /= 2

print(w)
```

EXERCÍCIO 13: **Dada a lista mista acom n elementos, crie uma nova lista bonde os elementos do tipo string tenham sido substituídos pelo valor -999.**
Exemplo: ["abcd", "acme", 1000, "xyz", 45.50, -1] → b = [-999, -999, 1000, -999, 45.50, -1]

RESOLUÇÃO:

```
# lista usada para exemplo
# (mas o programa tem que funcionar para qualquer lista de qualquer tamanho!!!)
a = ["abcd", "acme", 1000, "xyz", 45.50, -1]

# -------- aqui começa o que foi pedido no enunciado --------------------

b = []

for elemento in a:
    if type(elemento) == str: # a função type retorna o tipo de um valor
        b.append(-999)
    else:
        b.append(elemento)

print(a); print(b)
```

EXERCÍCIO 14: **Criar um programa que armazene seis nomes em uma lista. Depois solicite um número entre 0 e 5 e imprima o nome da pessoa que está armazenado no índice cujo valor é igual a esse número. Caso o usuário digite um valor fora da faixa entre 0 e 5, imprima uma mensagem de erro na tela.**

RESOLUÇÃO:

```
MAX_NOMES = 6

lst_nomes = []


for x in range(MAX_NOMES):
    print("digite o nome ", x, ": ")
    nome = input()
    lst_nomes.append(nome)


print('Digite um numero entre 0 e ', MAX_NOMES - 1, ': ')
n = int(input())

if (n > 0) and (n < MAX_NOMES):
     print(lst_nomes[n])
else:
     print('Número inválido')
```

Neste exemplo, declaramos a variável MAX_NOMES com o valor 6 (número de elementos que serão inseridos na lista) no início do programa e a utilizamos em diferentes linhas do código. Conforme apresentado no capítulo anterior, esse é um recurso bastante prático que podemos utilizar em programas, para que seja possível alterar apenas uma linha de código para fazer com que, automaticamente, o programa leia um número maior ou menor de dados via teclado.

EXERCÍCIO 15: Faça um programa que leia duas listas de números reais "a" e "b", cada uma com 10 elementos. A partir destas duas listas, gere um terceira chamada "c" da seguinte forma: o primeiro elemento de "c" deve ser obtido a partir a soma do primeiro elemento de "a" com o último (décimo) de "b". O segundo elemento de "c" deve representar a soma do segundo elemento de "a" com o penúltimo de "b", e assim por diante.

RESOLUÇÃO:

```
TAM = 10

#parte 1(a) - lê os dados da lista "a"
a = []
for i in range(TAM):
    print('a[', i, ']: ');
    valor = float(input())
    a.append(valor)

#parte 1(b) - lê os dados da lista "b"
b = []
for i in range(TAM):
    print('b[', i, ']: ');
    valor = float(input())
    b.append(valor)

#parte 3 - gera "c"
c = []
for i in range(TAM):
    c.append(a[i] + b[TAM - (i+1)])

print(c) #parte 4 - imprime "c"
```