

Disciplina	Descrição deste documento
Extra Curricular Algoritmos com implementação em Python 3	Parte 3 Comando de Repetição

Comando de Repetição

É muito frequente que para implementar uma determinada lógica um programador precise repetir um trecho de programa um certo número de vezes. Isto pode ser realizado com o uso do comando de repetição *while*. Com ele é possível repetir um conjunto de comandos enquanto uma condição especificada for verdadeira. Estes trechos repetitivos de código também são conhecidos como laços ou – em inglês – como *loops*.

O comando *while* em Python tem a seguinte construção básica que pode ser interpretada como: enquanto a condição for verdadeira execute o conjunto de comandos.

```
while {condição}:  
    {conjunto de comandos}
```

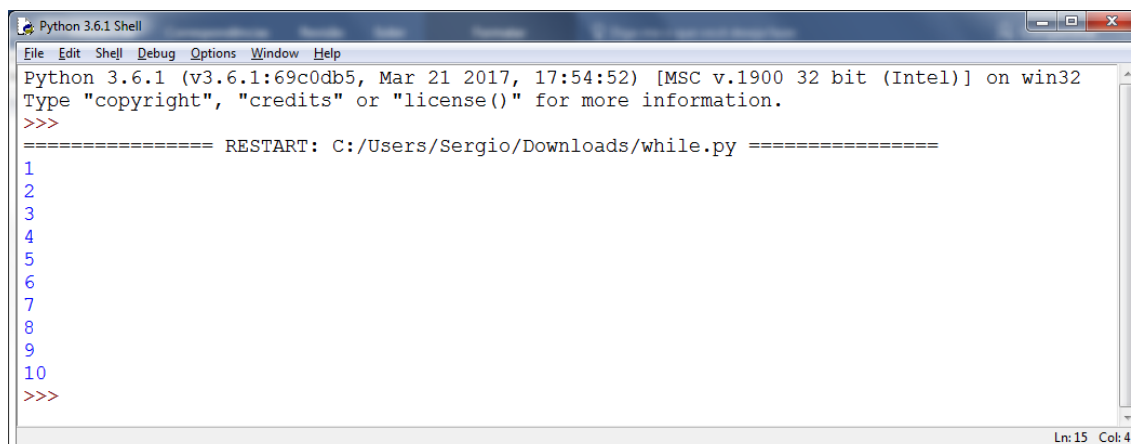
A condição segue exatamente as mesmas regras utilizadas nas condições já vistas quando foi tratado do comando *if - else*. Quanto ao conjunto de comandos subordinados ao *while*, podem ser quaisquer comandos válidos em Python, em qualquer quantidade e extensão. Assim como no comando *if - else*, a indentação é importante pois define a relação de subordinação entre o cabeçalho do comando e seu conjunto subordinado.

Para exemplificar a implementação de um laço tem-se o código abaixo no qual se quer exibir na tela todos os números inteiros entre 1 e 10, sendo um valor em cada linha.

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

①
②
③
④

Neste exemplo, na linha ① é definida uma variável inteira identificada por *Cont* e inicializada com o valor 1. Em seguida – linha ② – tem-se o comando *while* construído com a condição *Cont <= 10*. A avaliação dessa condição resulta em *True* (verdadeiro) de modo que o conjunto de comandos subordinado, constituído pelas linhas ③ e ④, é executado uma primeira vez. Com isso, o valor inicial de *Cont* é exibido e 1 é somado a *Cont*, que passará a ser 2. Após a execução da linha ④ o programa retorna para a linha ② e a condição é avaliada e novamente resultará *True*, pois *Cont* é menor que 10. Isto fará com que o *print* e a soma de 1 em *Cont* sejam executados uma segunda vez. Com isso *Cont* passará a conter o valor 3 e o programa seguirá sucessivamente. Ao final dez linhas terão sido exibidas na tela contendo os valores de 1 a 10.



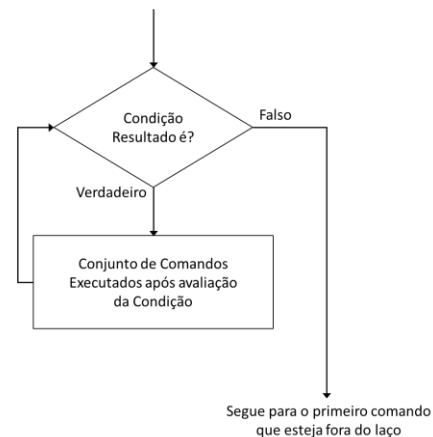
```
Python 3.6.1 Shell  
File Edit Shell Debug Options Window Help  
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/Sergio/Downloads/while.py =====  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
>>>
```

Lógica de funcionamento do laço *while* em Python

No exemplo anterior foi mostrado como usar o comando *while* em Python. Trata-se de um comando de laço no qual o teste da condição é feito no início do laço. O desenho ao lado ilustra esta situação, na qual a avaliação da condição é feita antes de se executar o conjunto de comando subordinado.

Este conceito é importante porque sempre que a condição for previamente falsa o conjunto subordinado não será executado nenhuma vez, uma vez que a avaliação da condição é feita antes.

Todo laço para ser implementado requer quatro elementos: inicialização, condição, iteração e o corpo. Os três primeiros dizem respeito à construção e controle do laço. A inicialização constitui-se de todo código necessário para determinar a situação inicial do laço. A condição é uma expressão lógica (simples ou composta) cujo resultado é avaliado em falso ou verdadeiro que determina se o laço termina ou prossegue, respectivamente. A iteração é todo comando (pode ser um ou mais de um) que modifica as variáveis envolvidas na condição, a cada execução do laço. Por fim, o corpo do laço são os comandos que devem ser executados repetidas vezes.



Informação Relevante: como é fora do Python?

Nada impede que exista um comando de laço no qual a avaliação da condição seja feita após a execução do conjunto de comandos subordinado.

Em outras linguagens de programação, como C e Java, costuma existir um segundo tipo de comando de laço, no caso é o *do-while*, no qual a avaliação da condição é feita **APÓS** a execução dos comandos subordinados. Isso implica que o conjunto de comandos subordinado sempre será executado pelo menos uma vez. Em Python não existe essa opção.

No exemplo da página anterior, a inicialização está na linha ①, a condição é a linha ② e a iteração é a linha ④. O corpo do laço é a linha ③. Como se trata de um laço bem simples para implementar cada elemento foi necessária apenas uma linha cada um.

Primeiro Exercício resolvido e comentado

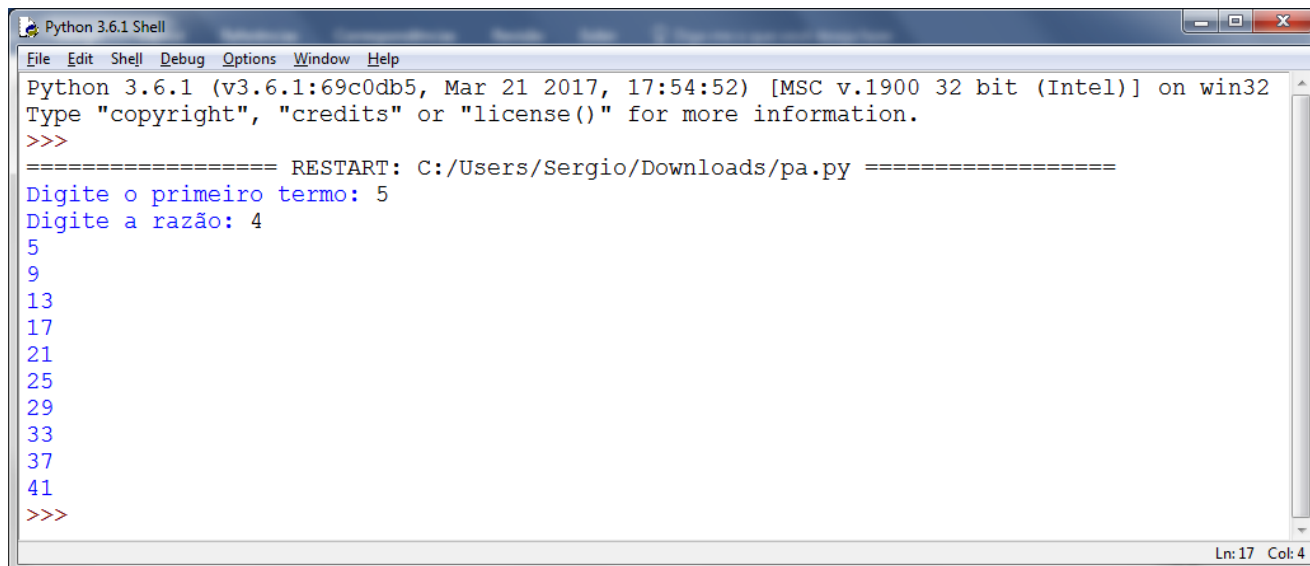
Escreva um programa que mostre na tela os 10 primeiros termos de uma progressão aritmética (PA) com primeiro termo P e razão R. Os dois números P e R são inteiros e devem ser lidos do teclado.

```
P = int(input("Digite o primeiro termo: ")) ①
R = int(input("Digite a razão: "))           ②
Cont = 0                                     ③
while Cont < 10:                             ④
    print(P)                                 ⑤
    P = P + R                               ⑥
    Cont = Cont + 1                          ⑦
```

Nas linhas ① e ② é feita a leitura dos dados de entrada. Na linha ③ é feita a inicialização do laço, atribuindo-se 0 à variável Cont. Na linha ④ está condição de continuidade do laço que foi construída como `Cont < 10`. E a iteração é formada pela linha ⑦ na qual se soma 1 à variável Cont. O corpo do laço é formado pelas linhas ⑤ e ⑥, nas quais é exibido o conteúdo da variável P e calculado o próximo termo a ser exibido, somando R à variável P e armazenando o resultado na própria variável P, preparando assim a variável P para a próxima iteração. A primeira vez que o laço é executado será mostrado o primeiro termo da PA, contido em P. Ao somar R em P, esta passa a conter o segundo termo. Quando o laço for executado pela segunda vez será mostrado o segundo termo e calculado o terceiro. Ao mesmo tempo 1 é somado em Cont a cada repetição, de modo que o mesmo aumentará sempre até atingir o valor 10. Quando isso ocorrer a condição da linha ④ será avaliada em False e o laço terminará.

Neste exemplo a variável Cont foi inicializada com 0 e a condição foi escrita como `Cont < 10`. No exemplo da página anterior a variável Cont foi iniciada com 1 e a condição escrita como `Cont <= 10`. São duas formas diferentes de implementar o mesmo laço. E a escolha entre uma ou outra depende apenas da forma que o programador achar mais apropriada.

Ao executar o programa acima para um caso de teste em que $P = 5$ e $R = 3$, tem-se o seguinte resultado:



```
Python 3.6.1 Shell
File Edit Shell Debug Options Window Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Sergio/Downloads/pa.py =====
Digite o primeiro termo: 5
Digite a razão: 4
5
9
13
17
21
25
29
33
37
41
>>>
```

Tarefas adicionais:

1. Altere o programa da PA para, ao invés de 10 elementos, apresentar na tela uma quantidade Q de elementos, onde Q é uma variável inteira lida do teclado;
2. Altere o programa da PA para também calcular e mostrar a somatória de todos os termos;
3. Altere o programa da PA para exibir em cada linha uma frase como abaixo (supondo $P = 5$ e $R = 4$).

```
Termo 1 da PA = 5
Termo 2 da PA = 9
Termo 3 da PA = 13
...
Termo 10 da PA = 41
```

Segundo Exercício resolvido e comentado

Escreva um programa que permaneça em laço enquanto um valor X lido for diferente de zero. Para cada valor de X apresente na tela se o mesmo é par ou ímpar.

```
X = 1                                ①
while X != 0:                        ②
    X = int(input("Digite X: "))      ③
    if X % 2 == 0:                    ④
        print("{} é par".format(X))  ⑤
    else:                             ⑥
        print("{} é ímpar".format(X)) ⑦
```

Nesta solução o controle do laço não é implementado através de um contador que permitirá a repetição um certo número conhecido de vezes. Neste caso não se sabe quantas vezes o laço irá repetir. O que se sabe apenas é que termina quando zero for digitado para X .

Para garantir que o laço seja iniciado, a variável X deve ser criada contendo qualquer valor diferente de 0, o que é feito na linha ①. Esta é a linha de inicialização. A iteração é implementada na linha ③ que altera o valor da variável X . O novo X lido logo no início do laço também é usado no corpo do mesmo, que é constituído pelas linhas ④ a ⑦. O resto da divisão de X por 2 é calculado e comparado com zero. Se o resultado dessa comparação for verdadeiro, então o número é par, caso contrário é ímpar.

Quando zero for digitado, o programa dirá que zero é par e terminará.

Terceiro Exercício resolvido

Escreva um programa que permaneça em laço enquanto um valor X lido for diferente de zero. Totalize (some todos) e conte os valores digitados, exceto o zero, e apresente esses valores na tela. Caso de teste:

Entrada: 6 8 -30 9 -4 8 16 50 15 7 2 -5 0

Saída: Total dos valores digitados = 82

Quantidade de valores = 12

```
Soma = 0
Qtde = 0
X = 1
while X != 0:
    X = int(input("Digite X: "))
    if X != 0:
        Soma = Soma + X
        Qtde = Qtde + 1
        # Este if evita que o zero seja contado
        # Adiciona X na variável Soma
        # Soma 1 na quantidade

print("Total dos valores digitados = {}".format(Soma))
print("Quantidade de valores = {}".format(Qtde))
```

Exercícios propostos

1. Escreva um programa que calcule os N primeiros termos de uma progressão geométrica (PG) com razão R e primeiro termo P fornecidos pelo usuário. Também deve ser calculada e apresentada a soma desses N termos. Caso de teste:

Entrada: N = 8 P = 2 e R = 3

Saída: 2 6 18 54 162 486 1458 4374

2. Escreva um programa que leia valores numéricos inteiros e totalize (totalizar é somar todos os números) separadamente os positivos e os negativos até que o usuário digite 0. Ao final mostre na tela esses dois totais. Caso de teste:

Entrada: 12 -3 5 1 -4 -9 6 0

Saída: Total dos positivos = 24

Total dos negativos = -16

3. Escreva um programa que leia um número inteiro N e em seguida leia N números reais, calculando a soma de todos os valores digitados.
4. Escreva um programa que leia um número inteiro N e em seguida leia N números reais, calculando a soma de todos os valores positivos fornecidos, ignorando os negativos.