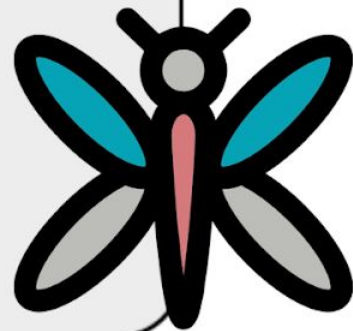


**Projeto "Libélulas":  
Treinamento para  
OBI Modalidade  
Programação**



Núcleo de Estudos e Pesquisa em Tecnologia da Informação



**INSTITUTO FEDERAL**  
Goiano

Campus  
Ceres



**Vai começar...**

*Boas-vindas!*





*Programação **esportiva** e divertida!*



**Ganhe pontos  
com a lista e  
desafios!**



```
0101010101010101000010100101010101010101010100001010111011101000111000100101010101010101010101010100  
0101010101010101000010100101010101010101010100001010111010101010101010101010000101001010101010101010
```



**Agora é a vez de vocês  
resolverem o desafio para casa!**

*Aula 05*

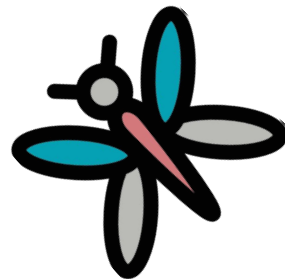
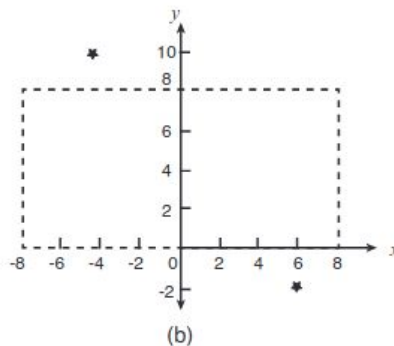
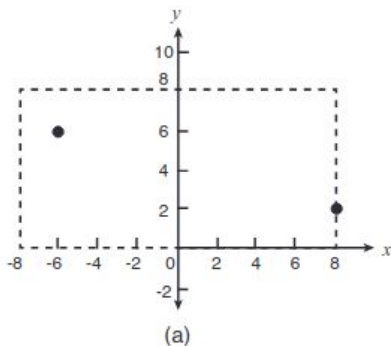




 **INSTITUTO FEDERAL**  
Goiano  
Campus Ceres

Com a crescente popularidade dos campeonatos de “Beach Tennis”, uma empresa está desenvolvendo um aplicativo para celular para ser usado por juízes. O objetivo é que, depois de configurar o aplicativo, um “juiz de vídeo” possa usar a câmera do celular para determinar se o impacto da bola com o piso foi dentro ou fora do campo de jogo.

O campo de jogo é um retângulo de dimensões 16m x 8m. A coordenada  $(0,0)$  é a posição do juiz, como mostrado na figura (a) abaixo. A figura (a) também mostra duas marcações de bolas dentro do campo de jogo (círculos pretos), nas coordenadas  $(-6,6)$  e  $(8,2)$ . Note que uma bola em cima da linha é considerada dentro do campo de jogo.



```
010101010101010100001010010101010101010101010000101011101110100011100010010101010101010101010101010100
010101010101010100001010010101010101010101010000101011101010101010101010101000010100101010101010101010
```



- $-100 \leq X \leq 100$
- $-100 < Y < 100$

[illegible]





# A sua vez de resolver!

```
x = int(input())  
y = int(input())  
  
if x < -8 or x > 8 or y < 0 or y > 8:  
    print('N')  
else:  
    print('S')|
```



```
010101010101010100001010010101010101010101010000101011101110100011100010010101010101010101010100  
010101010101010100001010010101010101010101010000101011101110100011100010010101010101010101010100
```



# **Estruturas de repetição**

## **Parte 01**

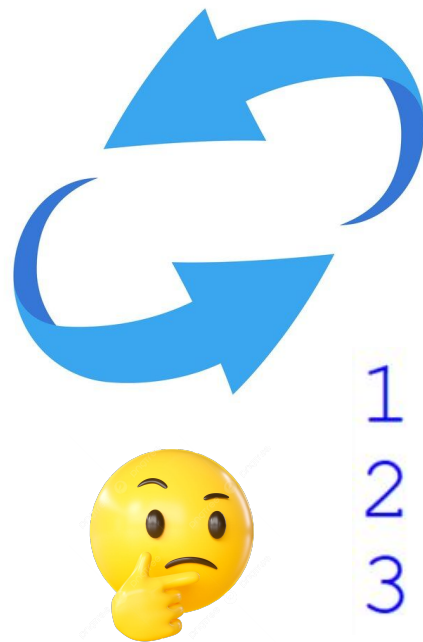
*Aula 05*





# Executando repetições

- **Repetições** representam a base de vários programas, elas são utilizadas para executar a **mesma** parte de um programa **várias vezes**, normalmente **dependendo de** uma **condição**;
- É **possível** imaginar que para **imprimir** três números, começando de **1 a 3**, bastaria usar apenas **vários prints**?! O que acham?



01010101010101010000101001010101010101010101000010101110111010001110001001010101010101010101010100  
01010101010101010000101001010101010101010101000010101110111010001110001001010101010101010101010100



# Repetindo, repetindo e repetindo!

- Para solucionar a repetição anterior, **pode utilizar formas simples** para esse tipo de **problema**, sendo:

```
print(1)  
print(2)  
print(3)
```

- No entanto**, se o objetivo fosse escrever 200 números, a solução **não seria tão agradável**, pois seriam **200 linhas de código**!



```
0101010101010101000010100101010101010101010100001010111011101000111000100101010101010101010101010100  
0101010101010101000010100101010101010101010100001010111011101000111000100101010101010101010101010100
```



# A estrutura FOR

*Aula 05*





# A estrutura de repetição FOR

- Com intuito de resolver **problemas** de **repetição**, surgem as **estruturas de repetição**, como “**FOR**”;
- A estrutura FOR pode ser representada assim, em Python;

■ `for i in range():`



Ao inicializar a repetição esse valor começa em zero (0)



A função range é responsável por definir as condições de parada da função





# Compreendendo o FOR!

- Utilizando **FOR** para contar de 0 a 9, pode ser escrito da seguinte **forma**:

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



Aqui é definido que a repetição do código deve acontecer 10 vezes!



A variável “i” começa em zero, realizado a contagem de 0 a 9





# Manipulando a função range

- A função **range** normalmente gera **valores a partir de 0**, sendo um valor a cada iteração, logo, ao especificarmos apenas 10, estamos informado onde parar;
- Por isso, ao mudar de **10 para 20**, deve imprimir **0 a 19**;
- Com a **mesma função range**, é **possível manipular** a geração **(iteração) dos valores**, definindo início, fim e a forma de geração (de quanto em quanto).

`range()`



```
010101010101010100001010010101010101010101010000101011101110100011100010010101010101010101010100
010101010101010100001010010101010101010101010000101011101110100011100010010101010101010101010100
```





# Manipulando a função range

- Dentro dos parâmetros da função, é possível **manipular**:

```
for i in range(2,10):  
    print(i)
```

Define o número onde a sequência termina, mas não é incluído

Define o valor que deve começar a exibição da variável

```
>> 2, 3, 4, 5, 6, 7, 8, 9
```



# Manipulando a função range

- E também podemos definir **intervalos...**
- Portanto, a **sintaxe básica** é:

`range(início, fim, passo)`

{ **início (opcional)**: número a partir do qual a sequência começa (padrão é 0)  
**fim (obrigatório)**: número **onde a sequência termina**, mas **não é incluído**  
**passo (opcional)**: intervalo entre os números (padrão é 1)

0101010101010101000010100101010101010101010100001010111011101000111000100101010101010101010101010100  
0101010101010101000010100101010101010101010100001010111011101000111000100101010101010101010101010100



# A estrutura WHILE

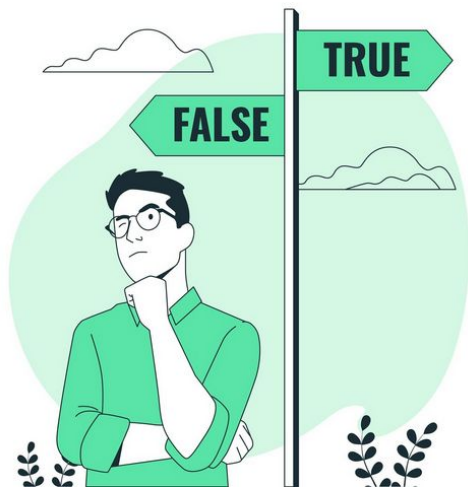
*Aula 05*





# A estrutura WHILE

- Outra estrutura de repetição em Python é o **WHILE**, que repete o bloco **ENQUANTO** a condição for **verdadeira**;
- Seu formato é apresentado em que a **condição** se trata de uma **expressão lógica**;
- O **bloco** representa as **linhas** de programa a **repetir enquanto** o resultado da condição for **verdadeiro**.



```
01010101010101010000101001010101010101010101000010101110111010001110001001010101010101010101010100
01010101010101010000101001010101010101010101000010101110111010001110001001010101010101010101010100
```



# A estrutura WHILE

- Seu formato é apresentado a seguir:

- **while** <condição>: ➡

**bloco**



Pode ter operações  
que modificasse o  
valor da variável.  
Ex:  $x = x + 1$

Pode repetir até uma  
variável ser igual a 10.  
Ex:  $x == 10$



0101010101010101000010100101010101010101010101010000101011101110100011100010010101010101010101010100  
0101010101010101000010100101010101010101010101010000101011101110100011100010010101010101010101010100



# Exemplo com WHILE!

- Para escrever **três números** utilizando o **while**, pode ser **codificado** como a seguir;

`x = 1` → Inicialização da variável

`while x <= 3:` →

Condição de parada da repetição, enquanto X for menor ou igual a três, vai repetir

`print(x)`

`x = x + 1` →

Operação para aumentar o valor a cada repetição





# O desafio da aula!

*Aula 05*





# Exercício 1059 - Beecrowd

beecrowd | 1059



## Números Pares

Adaptado por Neilor Tonin, URI  Brasil

**Timelimit: 1**

Faça um programa que mostre os números pares entre 1 e 100, inclusive.

### Entrada

Neste problema extremamente simples de repetição não há entrada

### Saída

Imprima todos os números pares entre 1 e 100, inclusive se for o caso, um em cada linha.

#### Exemplo de Entrada

#### Exemplo de Saída

2  
4  
6  
...  
100

```
010101010101010100001010010101010101010101010000101011101110100011100010010101010101010101010100
010101010101010100001010010101010101010101010000101011101110100011100010010101010101010101010100
```





## Exercício 1059 - Beecrowd

```
par = 2  
  
while(par <= 100):  
    print(par)  
    par += 2
```



010101010101010100001010010101010101010101010000101011101110100011100010010101010101010101010100  
010101010101010100001010010101010101010101010000101011101110100011100010010101010101010101010100



# Exercício 1059 - Beecrowd

```
par = 2
```

```
while
```

```
pr
```

```
par + 2
```

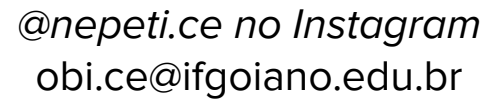
**#Desafio**  
O mesmo exercício  
poderia ser escrito  
usando FOR? Como  
resolveríamos?



```
0101010101010101000010100101010101010101010100001010111011101000111000100101010101010101010101010100
0101010101010101000010100101010101010101010100001010111011101000111000100101010101010101010101010100
```



# Dúvidas e perguntas?!





TREINANDO EM CASA  
**Lista 04 no Beecrowd :)**

**Dúvidas no fórum  
do moodle!**





# Material Licenciado



**Este documento está licenciado sob uma licença Creative Commons CC BY-NC-SA 4.0.**

Texto da licença: <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.pt-br>

*Esta licença permite que outros remixem, adaptem e criem a partir do seu trabalho para fins não comerciais, desde que atribuam a você o devido crédito e que licenciem as novas criações sob termos idênticos.*



**Atribuição** — Você deve dar o crédito apropriado, prover um link para a licença e indicar se mudanças foram feitas.



**NãoComercial** — Você não pode usar o material para fins comerciais.



**Compartilhalgual** — Se você remixar, transformar, ou criar a partir do material, tem de distribuir as suas contribuições sob a mesma licença que o original.



01010100  
01001001

**NEPeTI**

01010101010101010000101001010101010101010101000010101110111010001110001001010101010101010101010100  
01010101010101010000101001010101010101010101000010101110111010001110001001010101010101010101010100