# Estruturas de Repetição ¶

### Semana 08 | O laço `enquanto`

# Introdução

O laço `enquanto` é uma estrutura de repetição, assim como sua irmão, o laço `para`. A diferença reside que a condição de entrada e saída é controlada por uma condição, estabelecida na declaração do laço. A repetição ocorre enquanto esta condição for verdadeira.

```
while (<condição>):
    comandos
```

Por exemplo, suponha que se queira imprimir $10$ vezes uma mesma mensagem sem graça no console.

In [5]:

```
n = 0
while (n < 10): # Condição é declarada aqui
    print(n+1, "Olá mundo")
    n += 1
```

```
1 Olá mundo
2 Olá mundo
3 Olá mundo
4 Olá mundo
5 Olá mundo
6 Olá mundo
7 Olá mundo
8 Olá mundo
9 Olá mundo
10 Olá mundo
```

#### Uso do `break`

In [8]:

```
n = 0
while (n < 100): # Condição é declarada aqui
    print(n+1, "Olá mundo")
    if (n >= 9):
        break
    n += 1
```

```
1 Olá mundo
2 Olá mundo
3 Olá mundo
4 Olá mundo
5 Olá mundo
6 Olá mundo
7 Olá mundo
8 Olá mundo
9 Olá mundo
10 Olá mundo
```

Uma condição é uma expressão que pode retornar `True` ou `False`.

In [ ]:

```
n = 5

print(n == 5) # Exemplo 1 de condição

print(n > 5) # Exemplo 2 de condição
```

Existem algumas características que tornam mais apropriado o uso do laço `enquanto`. Por exemplo, imagine que se queira controlar a execução de um código enquanto o usuário assim o desejar.

In [10]:

```
quer_sair = False
while (not quer_sair):
    print("Olá, bom dia.")
    resp = input("Pressione qualquer tecla para continuar. 0 para sair: ")

    # Uns comentários para ajudar a leitura
    #if (resp == "0"):
    #    quer_sair = True

    quer_sair = True if resp == "0" else False
print("Tenha um bom dia")
```

```
Olá, bom dia.
Pressione qualquer tecla para continuar. 0 para sair: 0
Tenha um bom dia
```

**Animação**

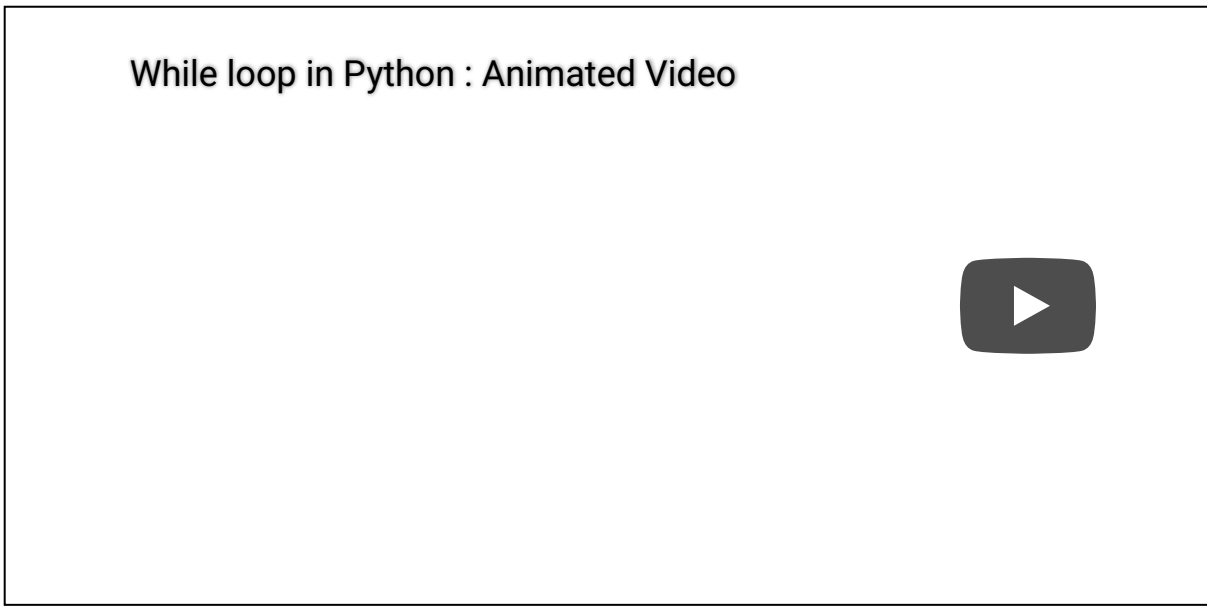
Uma animação simples da execução de um bloco `while` em Python.

In [2]:

```
from IPython.display import YouTubeVideo

YouTubeVideo('bOX6hzsvU4U', width=1024, height=300)
```

Out[2]:




# Exercícios

1) Escreva um laço `while` que some todos os números de $0$ a $100$ (inclusive).

In [ ]:

```
# Seu código aqui
```

2) Usando um laço `while` e um `if` , itere a lista a seguir e verifique se o número $100$ existe na lista. Em caso positivo, imprima o índice da ocorrência. Ex: `o número 100 está no índice 12` .

In [14]:

```
lst = [10, 99, 98, 85, 45, 59, 65, 66, 76, 12, 35, 13, 100, 80, 95]

# Seu código aqui
```

3) Usando um laço `while` , a função `pop()` para remover elementos da lista e a função `len()` para verificar o tamanho da lista, escreva um programa que esvazie uma lista de qualquer tamanho um elemento por vez.

In [27]:

```
# Pratique o que acontece com uma lista quando aplicamos um (ou mais) pop()

lista = [1,2,3,4]

print("Impressão 1", lista)
lista.pop()
print("Impressão 2", lista)
lista.pop()
print("Impressão 3", lista)
lista.pop()
print("Impressão 4", lista)
lista.pop()
print("Impressão 5", lista)
lista.pop() # <= pop de lista varia (pop from empty list)
```

```
Impressão 1 [1, 2, 3, 4]
Impressão 2 [1, 2, 3]
Impressão 3 [1, 2]
Impressão 4 [1]
Impressão 5 []

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call
 last)
<ipython-input-27-4c2649ed3cf5> in <module>
     12 lista.pop()
     13 print("Impressão 5", lista)
---> 14 lista.pop() # <= pop de lista varia (pop from empty list)

IndexError: pop from empty list
```

In [ ]:

```
lst = [10, 99, 98, 85, 45, 59, 65, 66, 76, 12, 35, 13, 100, 80, 95]

# Seu código aqui
```

4) Utilizando um laço `while` e a função `append()`, escreva um programa copie os elementos da lista 1 para a lista 2 um elemento por vez.

In [23]:

```
# Pratique a função append()

lista = []
print("Impressão 1", lista)
lista.append(5)
print("Impressão 2", lista)
lista.append(3)
print("Impressão 3", lista)
lista.append(7)
print("Impressão 4", lista)
```

```
Impressão 1 []
Impressão 2 [5]
Impressão 3 [5, 3]
Impressão 4 [5, 3, 7]
```

In [ ]:

```
lst1 = [10, 99, 98, 85, 45, 59, 65, 66, 76, 12, 35, 13, 100, 80, 95]
lst2 = []

# Seu código aqui
```

5) Utilizando um laço `while` , escreva um programa que imprima os elementos da lista de trás para frente (o último elemento da lista primeiro).

In [24]:

```
# Pratique o acesso a uma posição da lista

lista = [1,2,3,4,5]
print(lista[0])
print(lista[1])
print(lista[2])
print(lista[3])
print(lista[4])
print(lista[5]) # <= índice fora dos limites da lista (list index out of range)
```

```
1
2
3
4
5

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call
 last)
<ipython-input-24-a74610e21893> in <module>
      7 print(lista[3])
      8 print(lista[4])
----> 9 print(lista[5])

IndexError: list index out of range
```

In [ ]:

```
lst = [10, 99, 98, 85, 45, 59, 65, 66, 76, 12, 35, 13, 100, 80, 95]

# Seu código aqui
```

6) Utilizando um laço `while` , popule a `lst2` com cada um de seus elementos formados pela tupla (índice, valor) da `lst1` .

In [26]:

```
lst1 = [10, 99, 98, 85, 45, 59, 65, 66, 76, 12, 35, 13, 100, 80, 95]
"""
A lst2 deverá ficar assim:
[(1, 10), (2, 99), (3, 98), (4, 45) ...]
"""
lst2 = []
```

7) Utilizando `while` , um `if` e um comando `break` escreva um programa que verifique se um número é primo.

In [ ]:

```
# Seu código aqui
```

8) Utilizando o `while` e o operador módulo `%` , escreva um programa que conte os dígitos de um número inteiro qualquer.

In [15]:

```
# Seu código aqui
```

**DESAFIOS**

9) Utilizando o `while` , determinar o máximo divisor comum entre dois números inteiros positivos.

In [ ]:

```
# Seu código aqui
```

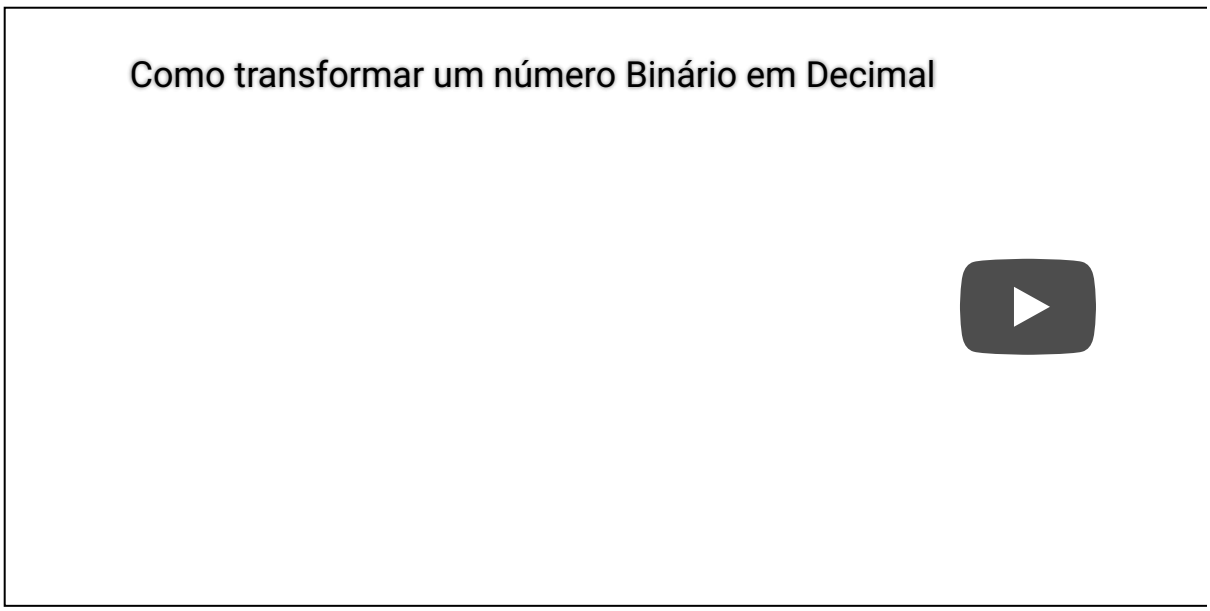
10) Dado um número binário inteiro positivo, determinar sua conversão para decimal.

Assista este vídeo de cerca de dois minutos para entender as regras de transformação de binário para decimal.

In [3]:

```
YouTubeVideo('zToihF2FE9I', width=1024, height=300)
```

Out[3]:




```
valor em decimal      1*2^3 + 0*2^2 + 1*2^1 + 0*2^0  =  10
número binário ->       1       0       1       0
```

In [ ]:

```
# Seu código aqui
```