# LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

# LÓGICA DE PROGRAMAÇÃO

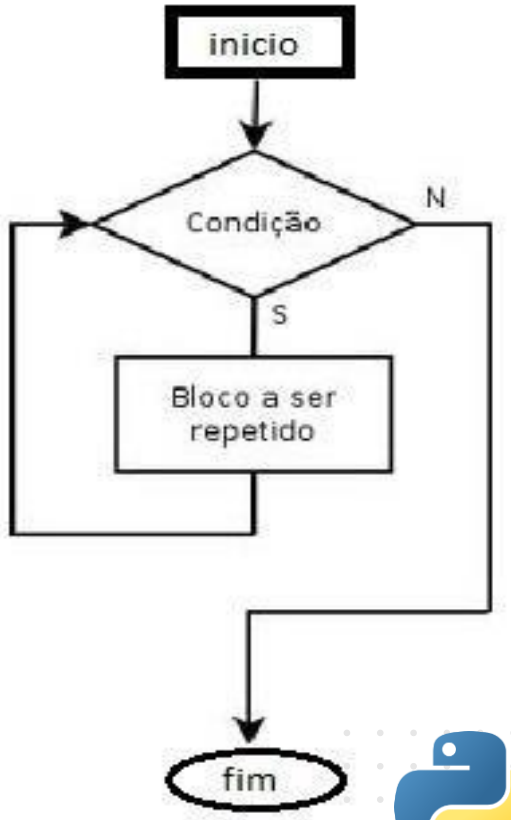
LINGUAGEM DE PROGRAMAÇÃO

**Estrutura de repetição.**

A estrutura de repetição é um recurso das linguagens de programação responsável por executar um bloco de código repetidas vezes enquanto determinada condição é atendida.

No Python, possuímos dois tipos de estruturas de repetição: **for e while.**

Usamos estruturas de repetições para que o sistema fique "**preso**" em um loop até que uma determinada condição de parada seja alcançada.



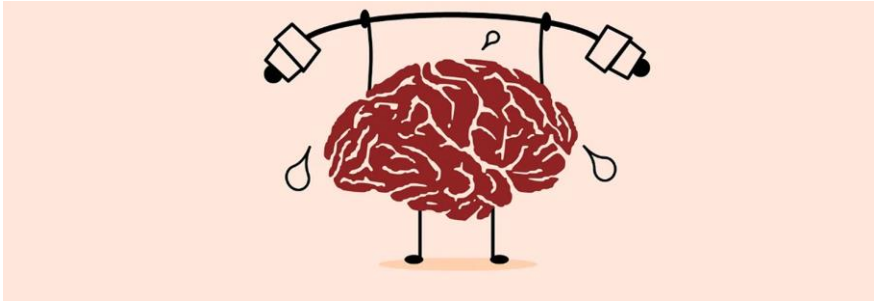
# LÓGICA DE PROGRAMAÇÃO

LOOPS FOR

**for**

A aritmética é muito boa, mas não constitui um programa muito empolgante. Portanto, vamos aprender a utilizar e realizar testes com ***loops,*** que significa dizer ao ***Python*** para **realizar uma tarefa várias vezes, em vez de apenas uma vez**.

O comando ***for in*** tem duas partes, depois da palavra ***for*** deve haver um nome de variável. A essa variável será atribuído um novo valor cada vez que o ***loop for***, for executado.



# LÓGICA DE PROGRAMAÇÃO

## LOOPS FOR

O comando **for** normalmente é utilizado quando você quer repetir um bloco de código um número fixo de vezes.

**for** elemento **in range** (inicio, fim, salto):

**Elemento** é a variável que vai receber um valor da sequência a cada iteração. A cada iteração ela aponta para um valor da sequência.

**Início**: É o valor inicial da sequência. O loop começará a iterar a partir desse valor.

**Fim**: É o valor final da sequência. O loop continuará iterando até alcançar esse valor, mas não incluirá esse valor na iteração.

**Salto**: É o incremento ou decremento entre os valores da sequência. Esse valor determina a diferença entre os elementos consecutivos na sequência.

# LÓGICA DE PROGRAMAÇÃO

## LOOPS FOR

O comando in range em Python é utilizado para criar uma sequência de números em um intervalo específico. A função range é comumente usada em loops, como o for, para iterar sobre uma série de valores.

```
#estrutura de repetição for
for i in range(1, 6, 2):
        print(i)


        1
        3
        5
```

```
#estrutura de repetição for
for i in range(1, 6):
        print(i)


        1
        2
        3
        4...5
```

# LÓGICA DE PROGRAMAÇÃO

## LOOPS FOR

O comando **for** em Python também pode ser usado para iterar sobre os caracteres de uma string. A sintaxe é semelhante à utilizada em listas. Aqui está um exemplo simples:

```
#estrutura de repetição for
for caractere in "Python":
        print(caractere)


        P
        y
        t
        h
        o...n
```

# LÓGICA DE PROGRAMAÇÃO

## LOOPS FOR

O comando **for** em Python também pode ser usado para iterar sobre os caracteres de uma string. A sintaxe é semelhante à utilizada em listas. Aqui está um exemplo simples:

```
#estrutura de repetição for
letras = ['L','U','C','A','S']
for letra in letras:
  print(letra)
```

```
L
U
C
A
S
```

# LÓGICA DE PROGRAMAÇÃO

## LOOPS FOR

O comando **for** em Python é usado para iterar sobre uma sequência (como uma lista) ou outros objetos iteráveis. A sintaxe básica do comando **for** em listas é a seguinte:

```
#estrutura de repetição
#for

for elemento in lista
        print(elemento)
```

```
#estrutura de repetição for
seq = [3, 57, 67, 22, 8]
for n2 in seq:
        print(n2)

        3
        57
        67...22...8
```

# LÓGICA DE PROGRAMAÇÃO

## LOOPS FOR

Podemos usar **else** o para executar algum código após o término do loop.

Para interromper um loop podemos utilizar o **break (parar o loop)**

```
#estrutura de repetição
#for

for numero in [0, 18, 56, 77, 95]:
        print(numero)
else:
        print("Acabou")
```

```
#estrutura de repetição
#for

for numero in range(1000000):
        print(numero)
        if numero == 4:
                break
print("Até mais")
```

# LÓGICA DE PROGRAMAÇÃO

## LOOPS FOR

Para passar para a próxima iteração, podemos utilizar o **continue**:

```
#estrutura de repetição
#for


for x in [1, 10, 20, 30, 40, 50]:
        if x == 30:
                continue
        print(x)
```

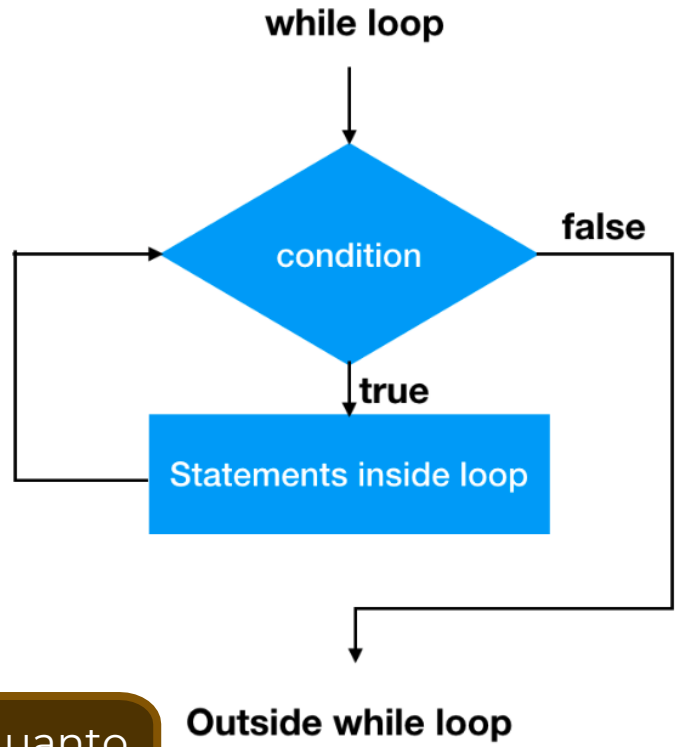
# LÓGICA DE PROGRAMAÇÃO

LOOPS WHILE

**while**

O **while** normalmente é utilizado quando você quer repetir um bloco de código enquanto uma expressão for verdadeira.

O **while** usado indevidamente pode fazer com que o sistema entre em um loop infinito.




while **condição**:

bloco de código

O bloco de código é executando enquanto a condição for verdadeira ou o loop for interrompido com um break.

# LÓGICA DE PROGRAMAÇÃO

## LOOPS WHILE

**while** - exemplo:

```
#estrutura de repetição
#while

x = 1

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

# LÓGICA DE PROGRAMAÇÃO

## LOOPS WHILE

**while** - O **else** é executado quando a condição se torna falsa:

```
#estrutura de repetição
#while


x = 0
while x < 5:
        print(x)
        x = x + 1
else:
        print("Acabou")
```

# LÓGICA DE PROGRAMAÇÃO

## LOOPS WHILE

O **continue** é usado para pular o restante do código no bloco do loop e continuar com a próxima iteração. Quando o continue é encontrado, o controle do programa retorna à condição do loop, ignorando o código restante dentro do bloco do loop

```
#estrutura de repetição
#while


numero = 0
while numero < 5:
        numero += 1
        if numero == 3:
                print("Vamos pular a iteração para",
numero)
                continue
        print("Número:", numero)
print("Fim do loop")
```

# LÓGICA DE PROGRAMAÇÃO

## LOOPS WHILE

O **break** é utilizado em Python para sair imediatamente de um loop quando uma determinada condição é atendida. Em poucas palavras, você deve usar break quando deseja interromper a execução do loop antes de atingir sua condição de término normal, seja devido a uma condição específica ou a um ponto específico no código.

```
#estrutura de repetição
#while

while True:
        resposta = input("Deseja sair? (s/n): ")
        if resposta.lower() == 's':
                print("Saindo do loop.")
                break
        print("Continuando o loop...")
```

# LÓGICA DE PROGRAMAÇÃO

## LOOPS WHILE

Quando parar de escrever o código vai parar de rodar, no entanto, é **MUITO IMPORTANTE** tomar cuidado com a estrutura **while**, pois ela, diferente do for, **não tem um fim definido**.

Isso quer dizer que você pode entrar em um **loop infinito** dependendo da maneira que escreve essa estrutura.

```
#estrutura de repetição
#while


nome = input("Insira um nome: ")
while nome:
    nome = input("Insira um nome: ")


---
x = 0
while x < 10:
    print(x)
```

# VAMOS PRATICAR.

EXEMPLOS E ATIVIDADES DE ESTRUTURAS DE REPETIÇÃO