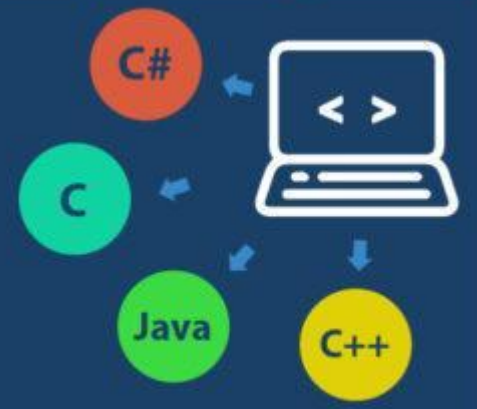
Linguagem de Programação

Professora: Letícia Pieper Jandt

Análise e Desenvolvimento de Sistemas

60 Horas

Estrutura de repetição

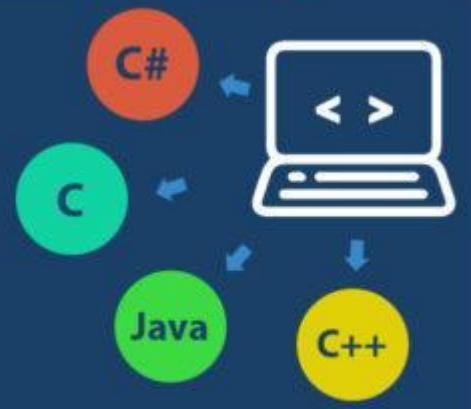


Como em situações do mundo real, em um programa, pode ser necessário repetir um trecho diversas vezes
até que uma determinada condição seja satisfeita.

Em programação, para casos como esse, são usadas estruturas conhecidas como *iteração, repetição, laço*
ou loop.

Python implementa duas estruturas de repetição:
while e for.

While



O comportamento da estrutura de repetição while é muito simples e a sua ideia é a seguinte:

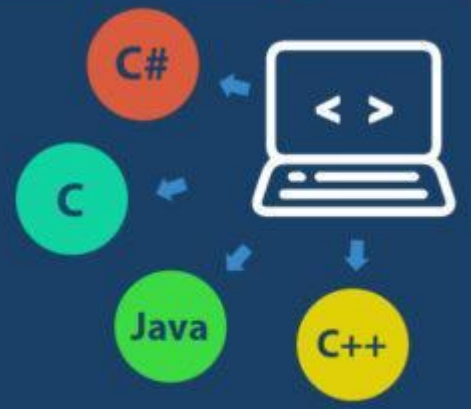
enquanto uma condição for verdadeira, uma instrução que está dentro do laço deverá ser executada.

while <condição (ou um conjunto delas) for verdadeira>:

    #Instruções a serem executadas

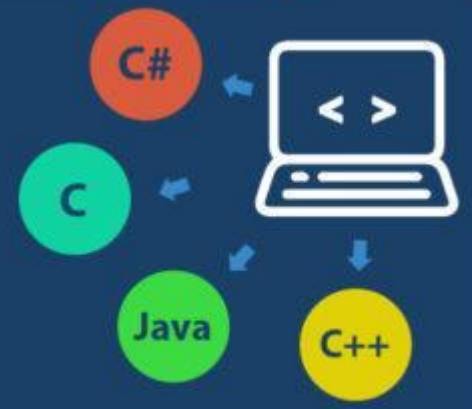
#Instruções a serem executadas após o encerramento do loop

While



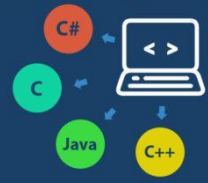

```
cont = 1

while cont <= 10:

    print("Olá!")

    cont += 1 #o mesmo que cont = cont + 1

print("FIM")
```

While



Temos que evitar erros que causam um problema chamado de “loop infinito”.

Esse termo é bastante sugestivo e representa a situação em que um determinado programa entra em um laço e não consegue “escapar”.



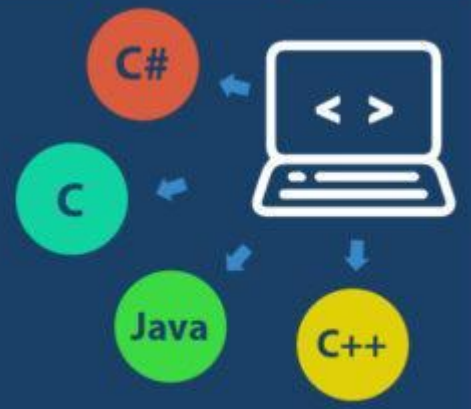
Praticando

Faça um programa, utilizando while, que permita o usuário fazer contas de adição enquanto quiser.

```python
continua = True

while (continua):
    print("Trabalhando com adições!")
    s1 = float(input("Insira o primeiro número: "))
    s2 = float(input("Insira o segundo número: "))

    print(f"A soma de {s1} + {s2} é: {s1+s2}")

    continua = int(input("Quer continuar somando? 1 pra sim 0 para não: "))

print("Fim do if")
```

UNIFASIPE
CENTRO UNIVERSITÁRIO

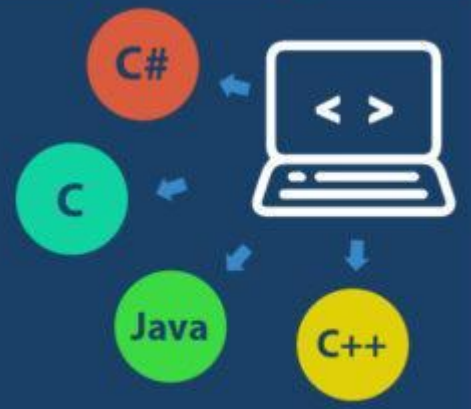
While



E se precisarmos somar números finitos.

```
#Primeira alternativa
print(1+2+3+4+5+6+7+8+9+10)
#Segunda alternativa
soma = 1+2+3+4+5+6+7+8+9+10
print(soma)
```

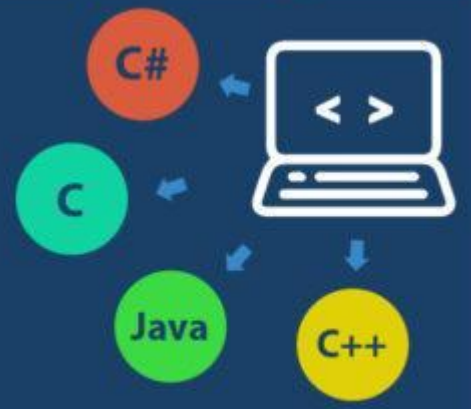
While



E se precisarmos somar números finitos.

```
#Primeira alternativa
print(1+2+3+4+5+6+7+8+9+10)
#Segunda alternativa
soma = 1+2+3+4+5+6+7+8+9+10
print(soma)
```

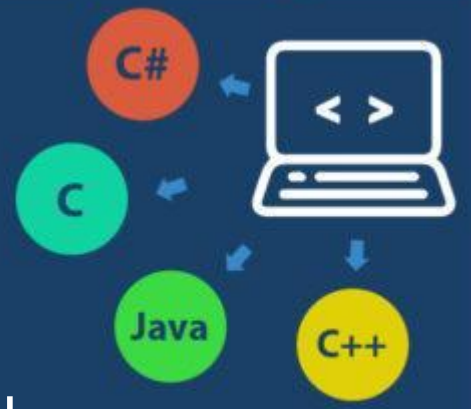
While



Podemos fazer isso melhor:

```
1   soma = 0
2   termo = 1
3   while termo <= 10:
4       print(f"Valor de termo: {termo} - valor de soma: {soma}")
5       soma += termo
6       termo += 1
7   print(soma)
```

FUNÇÃO RANGE



A função range() gera uma sequência de números dentro de uma faixa especificada.

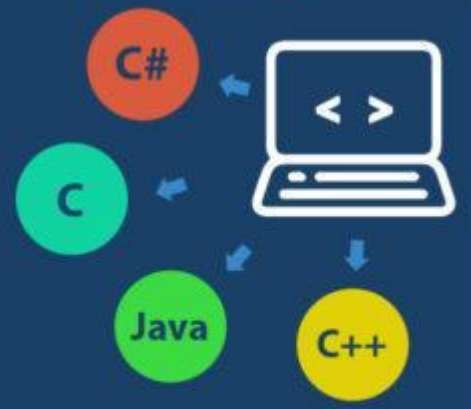
range(<quantidade_de_números_a_serem_gerados>)

range(<início_da_faixa>, <fim_da_faixa>[,<incremento>])

Na primeira maneira, a função range() recebe um parâmetro o qual indica a quantidade N de números a serem gerados, onde N deve ser maior que 0.

UNIFASIPE
CENTRO UNIVERSITÁRIO

FUNÇÃO RANGE



Exemplo 1:

```
print(list(range(3))) #Saída: [0, 1, 2]
```

O primeiro número da sequência gerada é o 0 e o último é o 2.

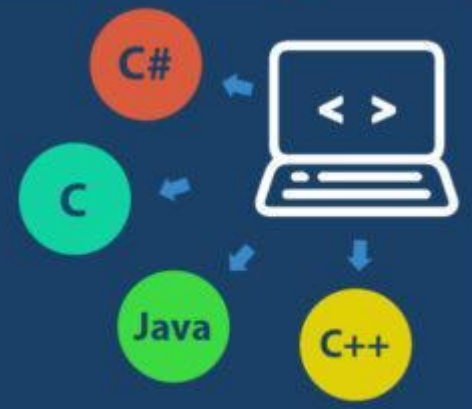
Portanto, a instrução range(N) gera uma lista com N números, onde o primeiro número é 0 e o último é N – 1(ou seja 3 - 1).

FUNÇÃO RANGE

A segunda maneira de executar a função range() possui três parâmetros distintos:

<início_da_faixa>, <fim_da_faixa> e <incremento> (opcional).

FUNÇÃO RANGE

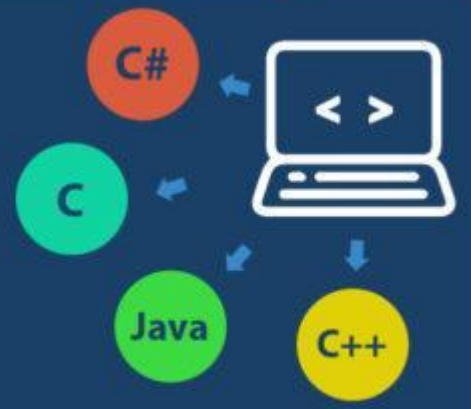


<início_da_faixa> determina o primeiro número que deve ser gerado na faixa.

<fim_da_faixa> está associado ao limite da sequência.

<incremento> indica a razão (positiva ou negativa) da soma e, quando não especificado, seu valor padrão é 1.

FUNÇÃO RANGE

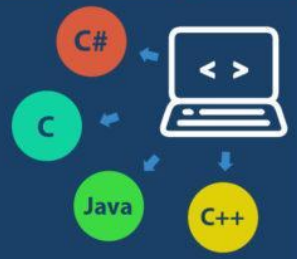


Quando <incremento> for positivo, o último número da sequência corresponde a <fim_da_faixa> – <incremento>.

No caso de <incremento> ser negativo, o último número da sequência é calculado da seguinte forma: <fim_da_faixa> + <incremento> * (–1).

Calma...

FUNÇÃO RANGE




```
1   print(list(range(10, 16)) )#Incremento = 1: [10, 11, 12, 13, 14, 15]
2
3   print(list(range(10, 16, 2)) )#Incremento = 2: [10, 12, 14]
4
5   print(list(range(16, 10, -1)))#Incremento = -1: [16, 15, 14, 13, 12, 11]
6
7   print(list(range(16, 10, -2)))#Incremento = -2: [16, 14, 12]
```

Perceba que quando <incremento> é positivo <início_da_faixa> é menor que <fim_da_faixa>.
No entanto, quando <incremento> é negativo, <início_da_faixa> é maior que <fim_da_faixa>.

UNIFASIPE
CENTRO UNIVERSITÁRIO

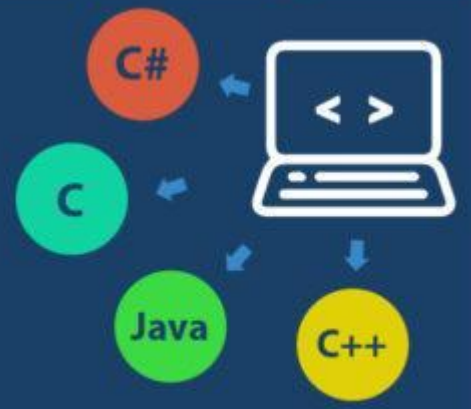
Qual das duas estruturas de repetição é mais adequada? while ou for.

Costumamos dizer que depende da situação.

Em geral, quando a quantidade de iterações é *indeterminada*, a estrutura while é uma boa alternativa.

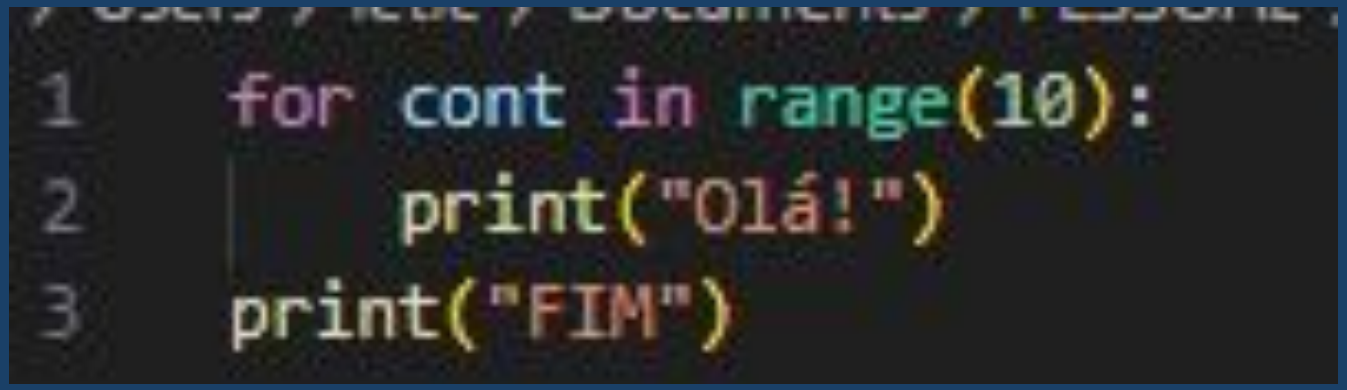
Por outro lado, quando o número de iterações é *definido*, a estrutura for é bastante adequada.

ESTRUTURA FOR




```
for <variável> in range([maneira_1|maneira_2]):
    #Instruções a serem executadas
#Instruções a serem executadas após o fim do loop
```

ESTRUTURA FOR



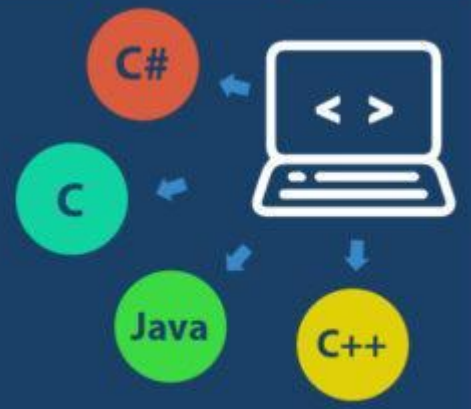
A função range() gera uma sequência de números de acordo com o conjunto de parâmetros informados.
Nesse caso o primeiro número da sequência será 0 e o último 9, totalizando 10 números.
A cada iteração, o valor gerado por range() será atribuído à variável cont, a linha 2, exibirá a mensagem Olá! em cada uma das 10 iterações.
Por fim, a linha 3 será executada ao término do laço.

ESTRUTURA FOR

Exemplo:

```
1    soma = 0
2    for termo in range(1, 11):
3        print(f"Valor de soma é: {soma} e valor de termo é: {termo}")
4        soma += termo
5    print(soma)
```

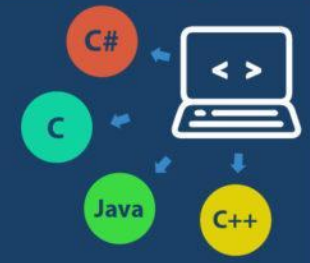
COMANDO BREAK



O papel do comando break: quebrar a execução de uma estrutura de repetição,

isto é,

forçar a saída do fluxo do programa de dentro do laço.



COMANDO BREAK

```
1   print("*** Operação de divisão ***")
2   while True:
3       n1 = int(input("Informe o primeiro número: "))
4       n2 = int(input("Informe o segundo número: "))
5       if n2 == 0:
6           print("Divisor não pode ser 0.\nPrograma será encerrado...")
7           break
8       print(f"{n1} / {n2} = {(n1/n2):.2f}")
9   print("*** Fim da operação de divisão ***")
```

A substring "\n" indica que o cursor deve pular para a próxima linha e, portanto, a mensagem Programa será encerrado... será exibida na linha seguinte.

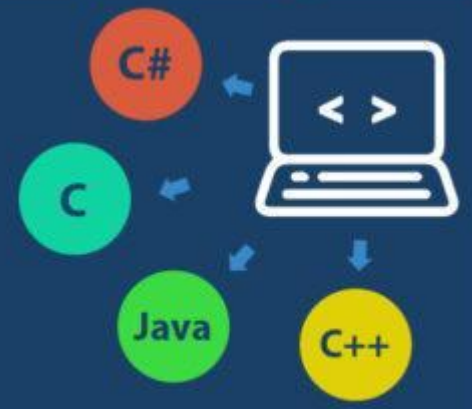
UNIFASIPE
CENTRO UNIVERSITÁRIO

Exercitando

Crie um programa no qual o usuário informe um número inteiro positivo N e armazene-o em uma variável. Em seguida, o usuário deve digitar N números.

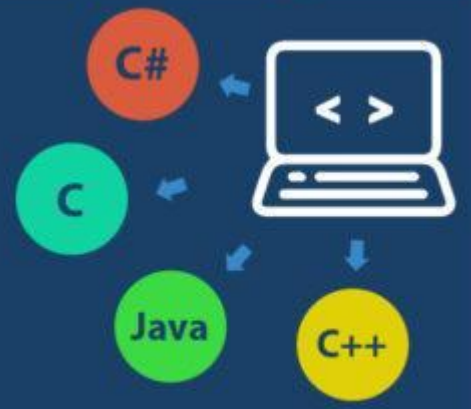
Ao fim, o programa deve imprimir a média aritmética dos N números digitados.

Exercitando




```python
N = int(input("Digite a quantidade de números a informar: "))
soma = 0
for cont in range(N):
    num = float(input("Digite um número: "))
    soma += num
    media = soma / N
print(f"Média = {media:.2f}")
```

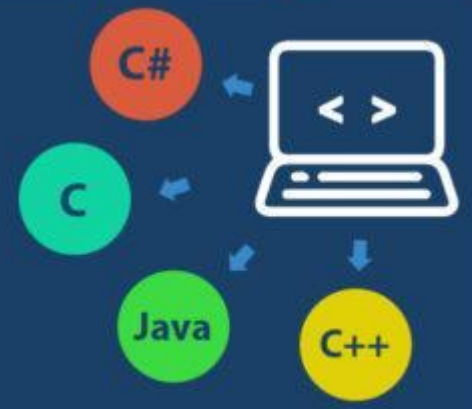
Exercitando



Para construir o programa a seguir, considere que os usuários só informarão números inteiros positivos.

Crie um programa que receba 5 números digitados e, ao fim, exibir quantos números são pares.

Exercitando

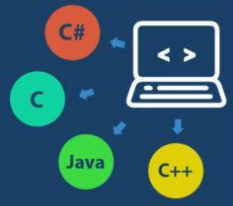



```python
pares = 0
for cont in range(5):
    num = int(input("Digite um número: "))
    if num % 2 == 0:
        pares += 1
print(f"Quantidade de números pares digitados: {pares}")
```

EXERCÍCIOS

1. Crie um programa no qual o usuário informe 2 números inteiros: a e b. Para que o programa continue sua execução, verifique se a < b. Se sim, calcule a soma dos números inteiros no intervalo [a, b]. Caso contrário, informe uma mensagem de erro.

2. Um professor de Matemática deseja construir um programa para gerar uma Progressão Aritmética (PA). Para isso, devem ser informados 3 argumentos: a) primeiro termo, b) quantidade de termos e c) razão.



EXERCÍCIOS

3. Faça um algoritmo para receber um número qualquer e imprimir na tela se o número é par ou ímpar, positivo ou negativo.

4. Faça um algoritmo que receba um número inteiro e imprima na tela o seu antecessor e o seu sucessor.

5. Faça um algoritmo que leia o ano em que uma pessoa nasceu, imprima na tela quantos anos, meses e dias essa pessoa ja viveu. Leve em

consideração o ano com 365 dias e o mês com 30 dias.
(Ex: 5 anos, 2 meses e 15 dias de vida)