Comandos Iterativos

CIÊNCIA DA COMPUTAÇÃO

ATITUS EDUCAÇÃO

- Aula 05 -
Pensamento Computacional

Prof. Me. Lucas R. C. Pessutto

ATITUS EDUCAÇÃO

Revisão

1. Construa um algoritmo que será um oráculo, ele será usado para determinar um animal escolhido através de respostas dadas pelo usuário. Algumas das perguntas possíveis são:
    - O animal é da América do Sul?
    - O animal é do Brasil?

    Observações:
    - Novas perguntas podem ser criadas.
    - Todas as perguntas devem ser respondidas com “S” (para sim) ou “N” (para não).
    - Caso alguma pergunta seja “N” (não) o programa deve encerrar as perguntas relacionadas aquela classificação, as classificações estão definidas por cores.
    - Caso não seja determinado nenhum animal, através da interação com o usuário, o programa deve informar “animal inexistente”.
    - O programa deverá utilizar os comandos: SE – SENAO – FIM_SE, ficando a critério da dupla onde utilizá-los.

Revisão

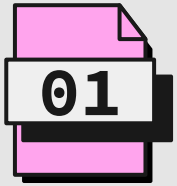
América do Norte

- Canadá
  - Mamífero
    - Roedor
      - Esquilo
    - Carnívoro
      - Coiotes
  - Ave
    - Ganso
- EUA
  - Mamífero
    - Herbívoro
      - Cervo
    - Carnívoro
      - Urso
  - Ave
    - Águia

América do Sul

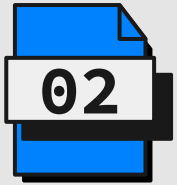
- Brasil
  - Ave
    - Papagaio
  - Mamífero
    - Onça
  - Herbívoro
    - Cavalo
- Argentina
  - Mamífero
    - Puma
  - Ave
    - Não voadora
      - Ema
    - Voadora
      - Condor

Na aula de hoje...



Algoritmos Iterativos

Motivação para uso da Iteração

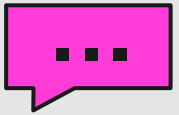


Comando while

Sintaxe
Usos de repetição: Validação de dados e repetição de cálculos

Comando for

Geração de Sequências
Sintaxe



Exercício

Faça um programa que leia a média de um aluno e, baseado no valor obtido, informe se o aluno está aprovado ou reprovado.

Considerar como aprovado o aluno que obtiver média ≥ 7.

```
media = float(input("Informe a média do aluno: "))

if media >= 7:
    print("Aprovado")
else:
    print("Reprovado")
```

Exercício – parte 2

Faça um programa que leia a média de um aluno e, baseado no valor obtido, informe se o aluno está aprovado ou reprovado.

Considerar como aprovado o aluno que obtiver média ≥ 7.

Supor uma turma de 5 alunos.

Exercício - parte 2

```python
media = float(input("Média do aluno 1: "))
if media >= 7:
    print("Aprovado")
else:
    print("Reprovado")

media = float(input("Média do aluno 2: "))
if media >= 7:
    print("Aprovado")
else:
    print("Reprovado")

media = float(input("Média do aluno 3: "))
if media >= 7:
    print("Aprovado")
else:
    print("Reprovado")
```

```python
media = float(input("Média do aluno 4: "))
if media >= 7:
    print("Aprovado")
else:
    print("Reprovado")


media = float(input("Média do aluno 5: "))
if media >= 7:
    print("Aprovado")
else:
    print("Reprovado")
```

Exercício – parte 3

Faça um programa que leia a média de um aluno e, baseado no valor obtido, informe se o aluno está aprovado ou reprovado.

Considerar como aprovado o aluno que obtiver média ≥ 7.

Supor uma turma de 50 alunos.

Comandos Iterativos (loops)

Repita 50 vezes:

```
media = float(input("Média do aluno: "))
if media >= 7:
    print("Aprovado")
else:
    print("Reprovado")
```

Comandos Iterativos (loops)

* **Objetivo**: repetir comando (ou blocos de comandos) um número finito de vezes

* Controle da Repetição:
    → Teste: repete enquanto uma condição for verdadeira

    → Contagem: executa uma repetição um número determinado de vezes

Comandos Iterativos

WHILE

```
while <expressão lógica>:
    <comandos indentados>
```

FOR

```
for <variável> in <iterável>:
    <comandos indentados>
```

Comando Iterativo while

Enquanto chover, ficarei em casa

Enquanto o ônibus não chega, leio o livro

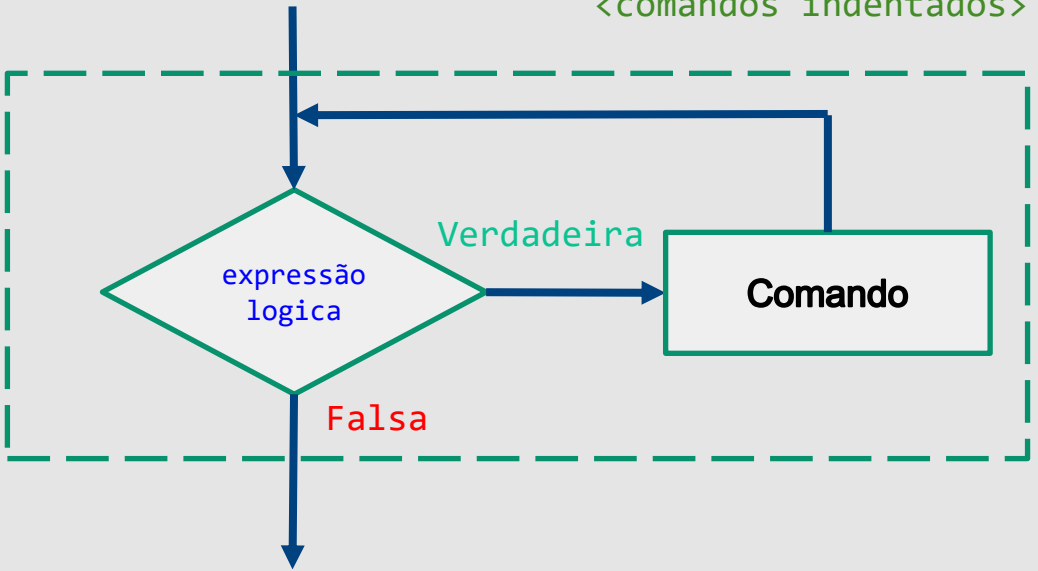
Comando Iterativo while

Linguagem Algorítmica

```
enquanto <expressão_logica> faca
    <comando>
fimenquanto
```

Linguagem Python

```
while <expressão lógica>:
    <comandos indentados>
```

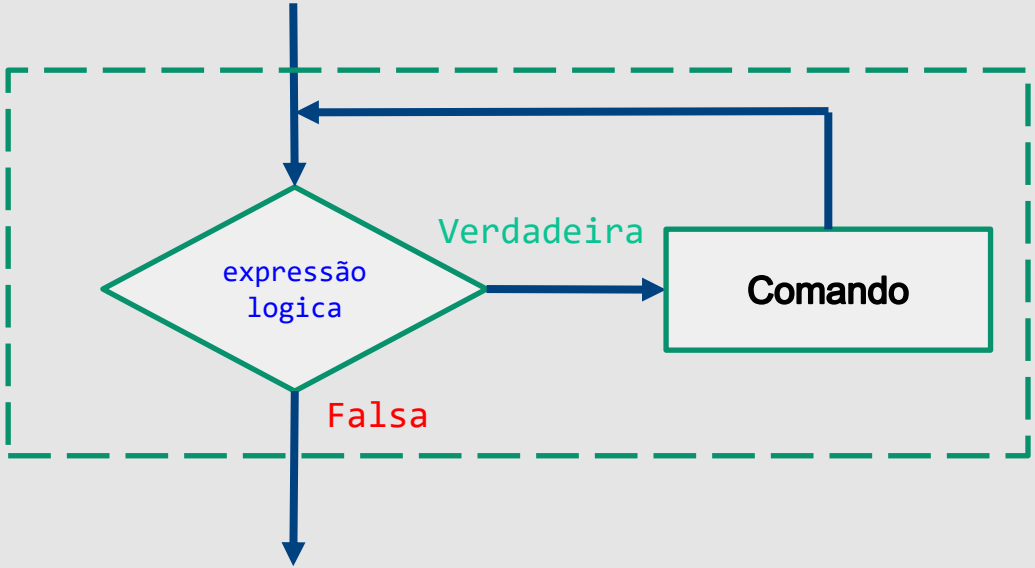


Comando Iterativo while

* Uma repetição executa várias vezes os comandos dentro do caminho verdadeiro

* A quantidade de execuções depende de uma condição de parada

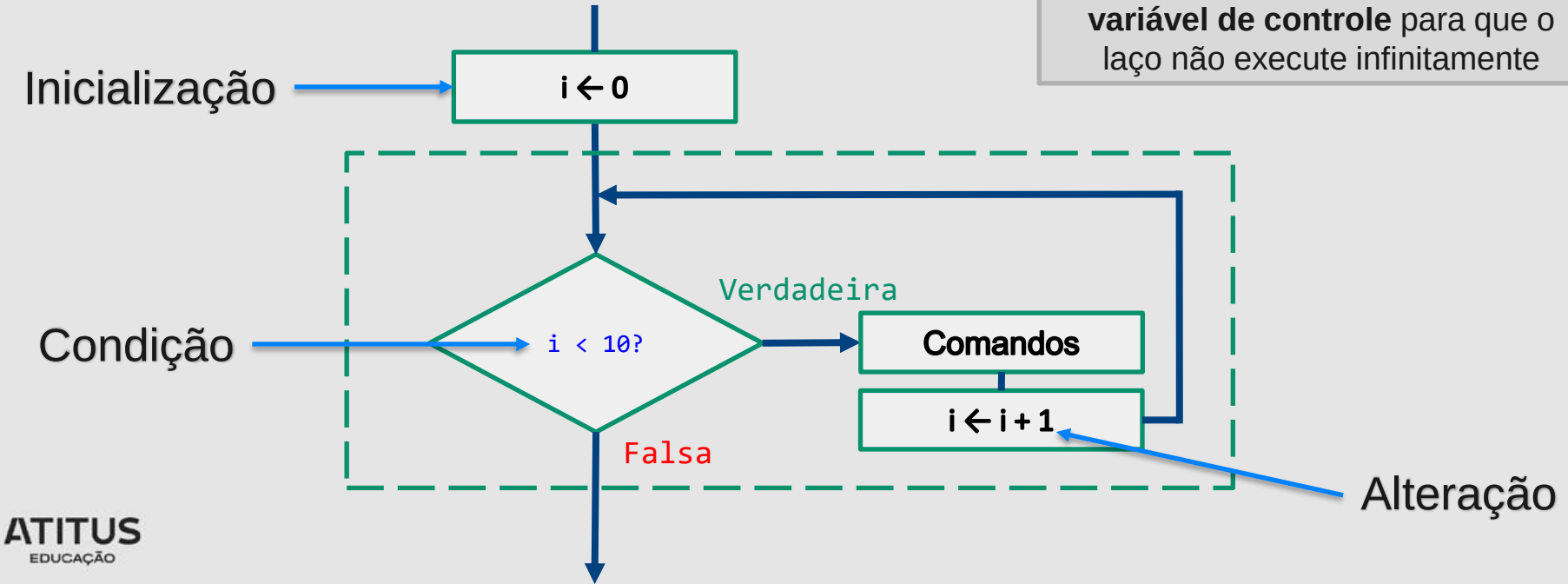
* Quando a condição de parada for avaliada como falsa o algoritmo sai do laço



Variável de Controle

* Exemplo: executar uma sequência de comandos 10 vezes

Algum dos comandos no caminho **verdadeiro** precisa alterar a **variável de controle** para que o laço não execute infinitamente



Comando Iterativo while

A **expressão lógica** aqui segue as mesma regras utilizadas no comando if

```
while <expressão lógica>:
    <comandos indentados>
```

Os comandos dentro do bloco indentado serão repetidos **enquanto** a expressão lógica resultar em **Verdadeiro**

Comando Iterativo while

```
while <expressão lógica>:
    <comandos indentados>
```

1. A condição definida em <expressão lógica> é avaliada
    a. Se verdadeira:
        I. **Comandos que estão em <comandos indentados> são executados**
        II. Retorna ao passo 1.
    b. Se falsa:
        a. Encerra o comando while

Exemplo 1: Contando de 1 a 10

```
i = 1
while i <= 10:
    print(f"Iteração: {i}")
    i = i + 1
```

```
Iteração: 1
Iteração: 2
Iteração: 3
Iteração: 4
Iteração: 5
Iteração: 6
Iteração: 7
Iteração: 8
Iteração: 9
Iteração: 10
```

Os comandos print e incremento da variável i são repetidos 10 vezes, até a variável i assumir o valor 11

Praticando com while

```python
a = 1
while a < 5:
    a += 1
print(a)
```

```
5
```

```python
a = 1
while a <= 5:
    print(a)
    a += 1
print(a)
```

```
1
2
3
4
5
6
```

```python
a = 0
while a < 5:
    a += 1
    print(a)
print(a)
```

```
1
2
3
4
5
5
```

```python
a = 6
b = 0
while (a - 2) > (b + 1):
    print(f"{a} - {b}")
    a -= 1
print(a)
```

```
6 - 0
5 - 0
4 - 0
3
```

Retomando o Exercício Inicial

Faça um programa que leia a média de um aluno e, baseado no valor obtido, informe se o aluno está aprovado ou reprovado.

Considerar como aprovado o aluno que obtiver média ≥ 7.

Supor uma turma de 5 alunos.

```
i = 0
while i < 5:
    media = float(input(f"Média do aluno {i + 1}: "))
    if media >= 7:
        print("Aprovado")
    else:
        print("Reprovado")
    i += 1
```

Problema 01: Fatorial

Enunciado de um problema:

Escreva um programa para calcular o fatorial de um número informado pelo usuário

Análise do Problema

O fatorial de um número n é dado pela seguinte equação matemática:

$$n! = n \times (n - 1) \times (n - 2) \times \cdots \times 1$$

Problema 01: Fatorial

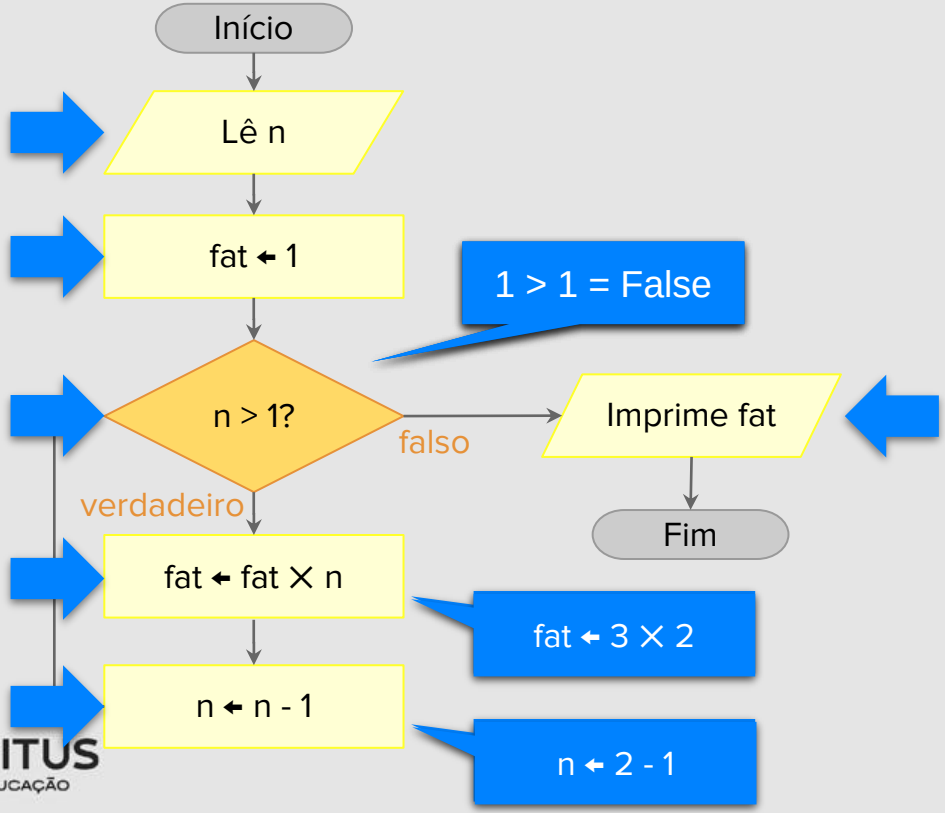
Análise do Problema

A quantidade de termos da equação depende do valor de n

Exemplo: O fatorial de 4 é dado por:

$$4! = 4 \times 3 \times 2 \times 1 = 24$$

Problema 01: Fatorial



Teste de mesa: seja n = 3 o valor fornecido ao computador

n	fat
~~3~~ ~~2~~ 1	~~1~~ ~~3~~ 6

Problema 01: Fatorial

Implementação do Programa

```python
n = int(input("Informe um número positivo: "))

fat = 1
while n > 1:
    fat = fat * n
    n = n - 1

print(f"FAT = {fat}")
```

* Pense em outras soluções possíveis para o problema do fatorial.

Verificação de erros com while

* Podemos utilizar um comando while para validar os dados digitados pelo usuário

* Por exemplo, no programa do fatorial, podemos repetir a leitura dos dados caso o usuário digitar um valor negativo

```
n = int(input("Informe um número positivo: "))

while n < 0:
    n = int(input("VALOR INVÁLIDO! Informe um número positivo: "))

fat = 1
while n > 1:
    fat = fat * n
    n = n - 1
print(f"FAT = {fat}")
```

```
Informe um número positivo: -2
VALOR INVÁLIDO! Informe um número positivo: -5
VALOR INVÁLIDO! Informe um número positivo: -7
VALOR INVÁLIDO! Informe um número positivo: -123
VALOR INVÁLIDO! Informe um número positivo: 8
FAT = 40320
```

Repetir operação com while

* Podemos utilizar um comando while para obter uma confirmação, podendo calcular diversas vezes o fatorial

```
resposta = "sim"
while resposta == "sim":
    n = int(input("Informe um número positivo: "))
    while n < 0:
        n = int(input("VALOR

    fat = 1
    while n > 1:
        fat = fat * n
        n = n - 1

    print(f"FAT = {fat}")

    resposta = input("Calcular o fatorial de outro número [sim] ou [nao]? ")
```

```
Informe um número positivo: 7
FAT = 5040
Calcular o fatorial de outro número [sim] ou [nao]? sim
Informe um número positivo: 19
FAT = 121645100408832000
Calcular o fatorial de outro número [sim] ou [nao]? sim
Informe um número positivo: 6
FAT = 720
Calcular o fatorial de outro número [sim] ou [nao]? nao
```

Comando Iterativo para

```
para <variavel> de <inicio> ate <fim> faca
    <comandos indentados>
fimpara
```

Sequências em Python

Frequentemente queremos repetir um conjunto de comandos *N* vezes de tal forma que a variável de controle do laço assume os valores de uma **sequência de inteiros**

```
i = 1
while i <= 10:
    print(f"Iteração: {i}")
    i = i + 1
```

Um erro muito comum nesse tipo de laço é esquecer de inicializar ou de incrementar a **variável de controle**

Sequências em Python

* Podemos criar sequências imutáveis em Python utilizando o comando `range`

```
seq = range(stop)
```

Aberto à direita

* A variável `seq` recebe uma sequência de inteiros no intervalo $[0, stop)$

* O parâmetro stop deve ser um número inteiro (maior que zero?)

Sequências em Python

* Exemplos:

```
seq1 = list(range(10))
print(seq1)                 # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

seq2 = list(range(1))
print(seq2)                 # [0]

seq3 = list(range(0))
print(seq3)                 # []

seq4 = list(range(-10))
print(seq4)                 # []
```

Repetição com for

* Repetições com `while` são adequadas em situações nas quais o número de iterações é desconhecido
  - Ex: repetir até que o usuário digite zero

* Por outro lado, o comando `for` tem uso indicado nas situações em que o número de iterações é conhecido a priori
  - Ex: ler 20 notas de alunos

Repetição com for

A **variável** especificada aqui assume cada um dos valores do objeto **iterável**

Uma **sequência** é um exemplo de **iterável**, mas existem outros, como listas, conjuntos, dicionários.

```
for <variável> in <iterável>:
    <comandos indentados>
```

Os comandos do bloco indentado serão repetidos de acordo com a **quantidade** de elementos no objeto iterável

É uma construção similar a do "***para todo***" na matemática quando usado para definir relações de pertinência em conjuntos:

$$\forall x \in \{1, 2, \ldots, n\} \ldots$$

Repetição com for

* A variável i assume cada um dos 10 valores [0, 1, 2, ..., 9] de uma sequência gerada com range

* A cada repetição um valor diferente de `i` será impresso na tela

* Na comparação com o while podemos ver que o controle da repetição fica simplificado (menos linhas, menos chance de erros)

```python
# imprime sequência com for
for i in range(10):
    print(f"Valor de i: {i}")
```

```python
# imprime sequência com while
i = 0
while i < 10:
    print(f"Valor de i: {i}")
    i = i + 1
```

Exercício Inicial com for

Faça um programa que leia a média de um aluno e, baseado no valor obtido, informe se o aluno está aprovado ou reprovado.

Considerar como aprovado o aluno que obtiver média ≥ 7.

Supor uma turma de 5 alunos.

```
for i in range(5):
    media = float(input(f"Média do aluno {i + 1}: "))
    if media >= 7:
        print("Aprovado")
    else:
        print("Reprovado")
```

Um pouco mais sobre sequências

* Podemos criar sequências mais elaboradas usando o mesmo comando

```
seq = range(start, stop[, step])
```

* A variável `seq` recebe uma sequência de inteiros no intervalo $[start, stop)$
* Os parâmetros `start`, `stop` e `step` (opcional) devem ser inteiros
* Os valores em `seq` são equiespaçados por `step`
* O parâmetro `step` assume o valor 1 caso não seja especificado

Referência para a função range: https://docs.python.org/pt-br/3/library/stdtypes.html#range

Sequências em Python

```python
seq1 = list(range(1, 10))
print(seq1)                     # [1, 2, 3, 4, 5, 6, 7, 8, 9]

seq2 = list(range(1, 11))
print(seq2)                     # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

seq3 = list(range(0, 12, 2))
print(seq3)                     # [0, 2, 4, 6, 8, 10]

seq4 = list(range(-5, 0))
print(seq4)                     # [-5, -4, -3, -2, -1]

seq5 = list(range(0, -6, -1))
print(seq5)                     # [0, -1, -2, -3, -4, -5]
```

Problema 02: Pares num Intervalo

Enunciado de um problema:

Escreva um programa que imprima todos os números pares dentro de um intervalo

Análise do Problema

Início = 5 e fim = 9	6 – 8
Início = 2 e fim = 6	2 – 4 – 6
Início = 1 e fim = 8	2 – 4 – 6 – 8
Início = 2 e fim = 5	2 – 4

Problema 02: Pares num Intervalo

Implementação do Programa

```
inicio, fim = input("Informe o início e o fim do intervalo: ").split(" ")
inicio, fim = int(inicio), int(fim)

for i in range(inicio, fim + 1):
    if i % 2 == 0:
        print(f"{i} ", end="")
```

Não muda de linha depois de imprimir

Problema 02: Pares num Intervalo

Implementação do Programa

O que você acha dessa solução?

```
inicio, fim = input("Informe o início e o fim do intervalo: ").split(" ")
inicio, fim = int(inicio), int(fim)

for i in range(inicio, fim + 1, 2):
    print(f"{i} ", end="")
```

Problema 02: Pares num Intervalo

Implementação do Programa

```
inicio, fim = input("Informe o início e o fim do intervalo: ").split(" ")
inicio, fim = int(inicio), int(fim)

if inicio % 2 != 0: # Garante que o primeiro valor do intervalo é par
    inicio += 1

for i in range(inicio, fim + 1, 2):
    print(f"{i} ", end="")
```

Problema 01: Fatorial

Enunciado de um problema:

Escreva um programa para calcular o fatorial de um número informado pelo usuário

Análise do Problema

O fatorial de um número n é dado pela seguinte equação matemática:

$$n! = n \times (n - 1) \times (n - 2) \times \cdots \times 1$$

Problema 01: Fatorial

Implementação do Programa

O que você acha dessa solução?

```
n = int(input("Informe um número positivo: "))

fat = 1
for i in range(1, n + 1):
    fat = fat * n

print(f"FAT = {fat}")
```

Problema 01: Fatorial

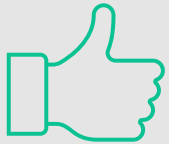
Implementação do Programa

O que você acha dessa solução?

```
n = int(input("Informe um número positivo: "))

fat = 1
for i in range(1, n + 1):
    fat = fat * i

print(f"FAT = {fat}")
```



Problema 03: Fibonacci

Enunciado de um problema:

Faça um algoritmo e o programa em Python correspondente que leia um valor inteiro e informe o termo equivalente da Série de Fibonacci.

Problema 03: Fibonacci

Análise do Problema

Definição da Série de Fibonacci:

$$fib(pos) = \begin{cases} 0, & se\ pos = 0 \\ 1, & se\ pos = 1 \\ fib(pos-2) + fib(pos-1), & se\ pos > 1 \end{cases}$$

Descrita por Leonardo de Pisa em 1202. Nessa sequência, todo o termo a partir do segundo corresponde à soma dos dois termos imediatamente anteriores.

0 1 1 2 3 5 8 13 21 ...

https://pt.wikipedia.org/wiki/Sequ%C3%AAncia_de_Fibonacci

Problema 03: Fibonacci

Descrição dos Dados e Algoritmos

1. Precisamos “lembrar” os valores Fibonacci de pos-1 e pos-2
   - ✓ Criamos duas variáveis para isso

```
t_2 = 0
t_1 = 1
t   = t_1 + t_2, para n > 2
(atualizar os valores de t_1 e t_2)
```

Problema 03: Fibonacci

```python
pos = int(input("Entre com o termo desejado: "))

if pos < 0:
    print("A posição não pode ser negativa!")
else:
    if pos == 0:
        fib = 0
    elif pos == 1:
        fib = 1
    else:
        t_2 = 0
        t_1 = 1
        for i in range(2, pos + 1):
            fib = t_2 + t_1
            t_2 = t_1
            t_1 = fib
    print(f"O termo {pos} da série de Fibonacci é {fib}")
```

Resumo

* Repetições tanto com `while` quanto com `for` permitem que se execute um conjunto de comandos várias vezes

* No `while` o controle do laço deve ser feito explicitamente pelo programador (condição de parada, variável de controle, etc.)

* No `for` esse controle fica mais implícito dependendo da definição do objeto iterável (sequência, lista, etc.)

* Quando se sabe exatamente quantas repetições serão necessárias, em geral é preferível aplicar o comando `for`

* Quando essa quantidade de repetições é incerta, o ideal é utilizar o comando `while`