**material\_de\_referencia.py**

Versão 3.0 – Fase Final do InterFatecs 2025

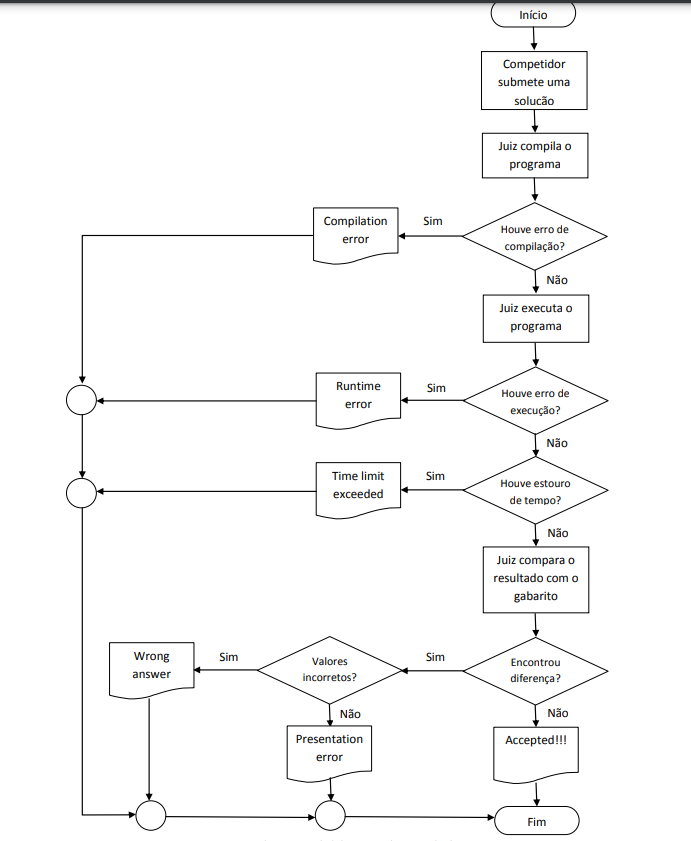
**Código Trifásico – FATEC Itapira**

Alexandre Alcindo  
Gabriel Almir  
Rodrigo Polastro

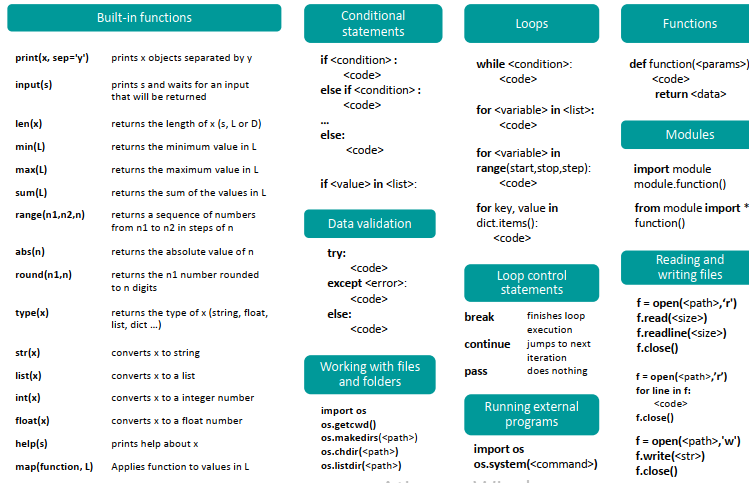
Técnico: Prof. Júnior Gonçalves

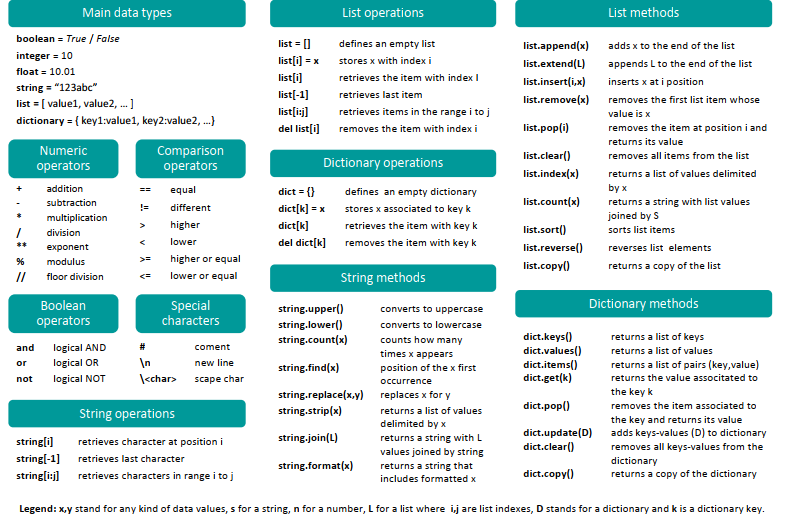
Agosto/2025

**Fluxo de Correção no Interfatecs**

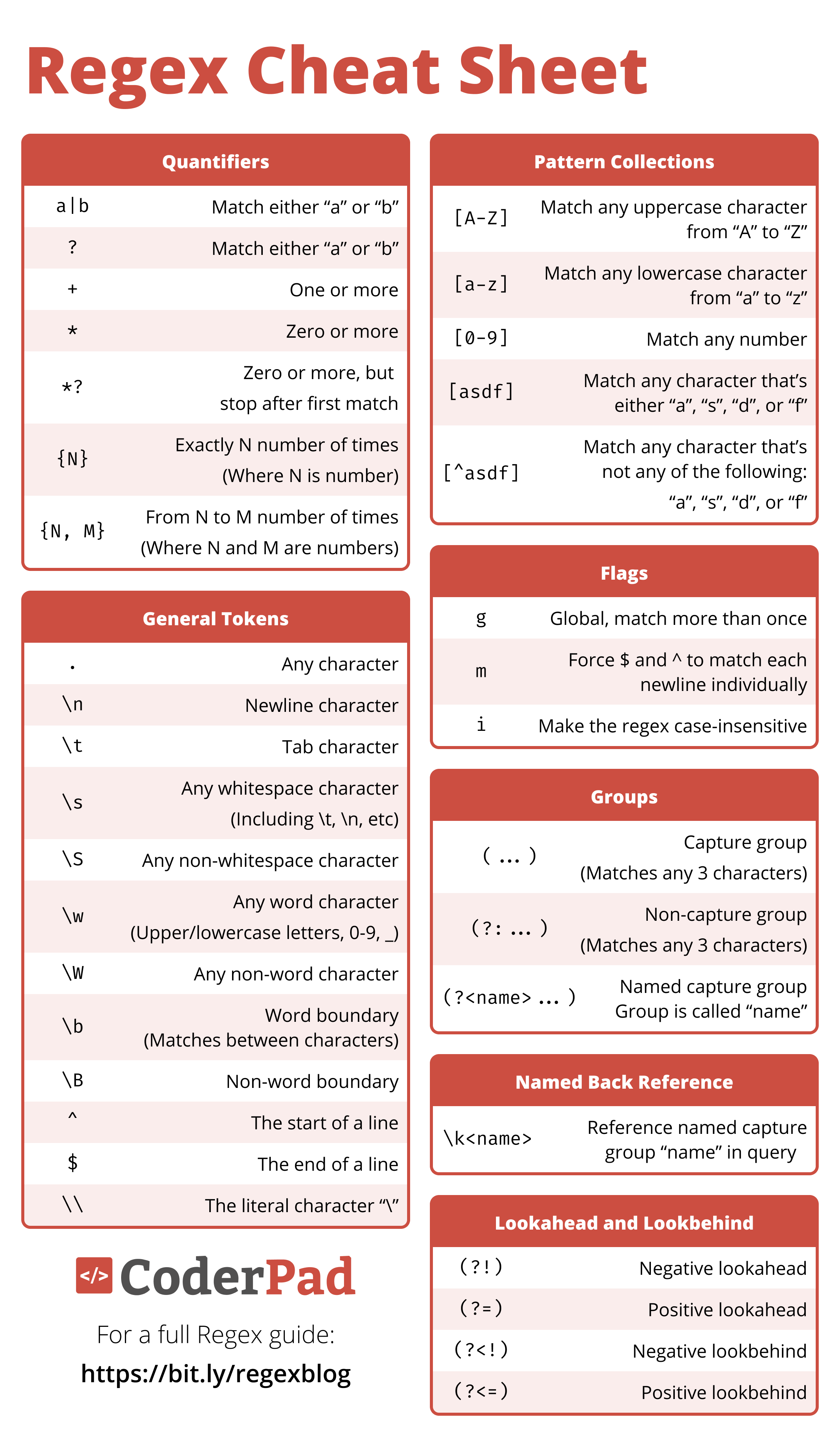


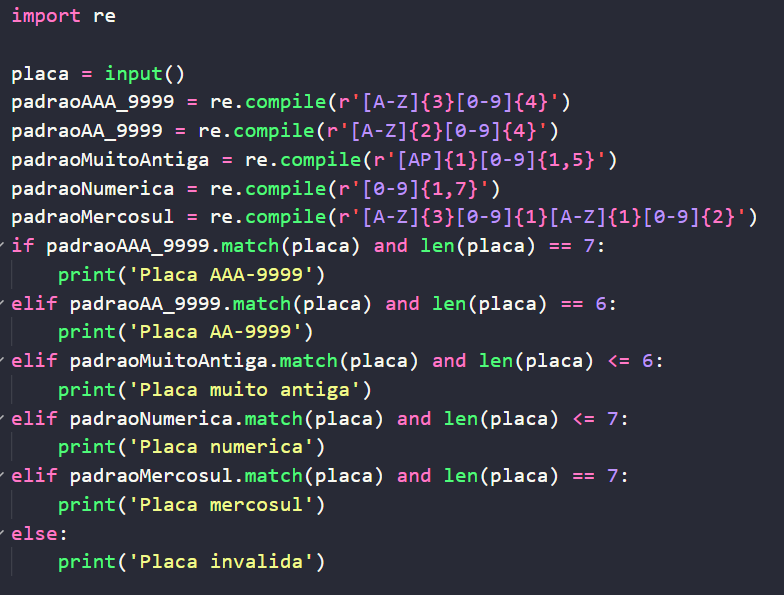
**Referência Básica Python**



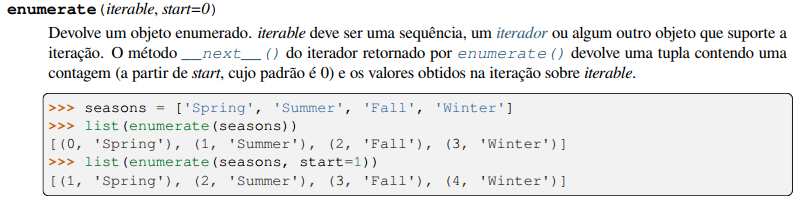
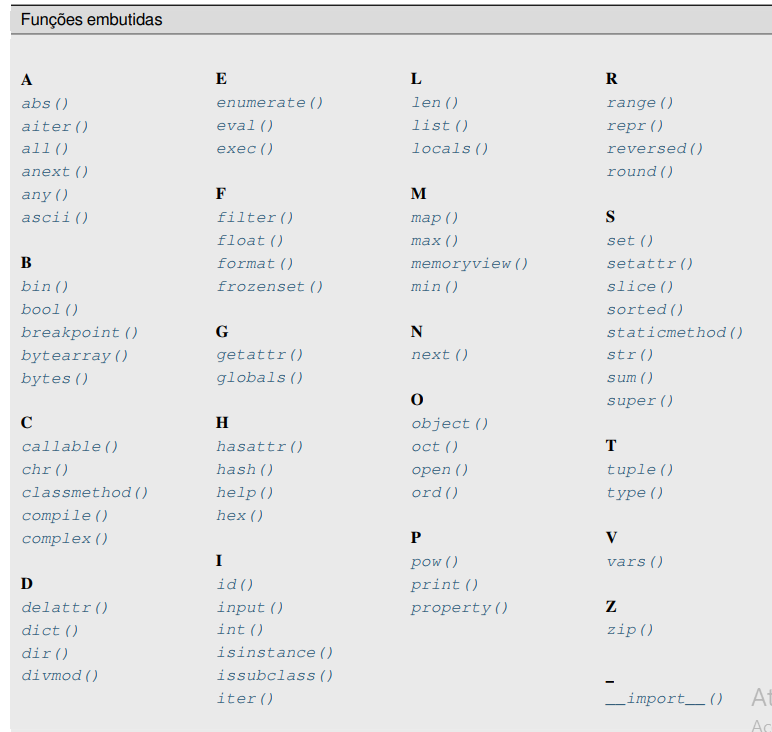


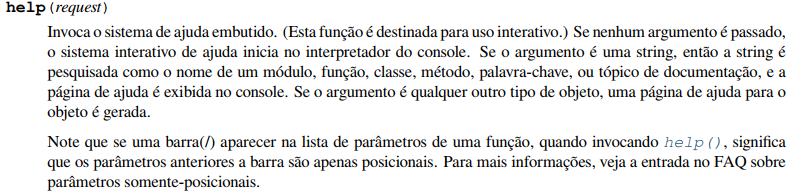
**Regex – Referência Rápida**

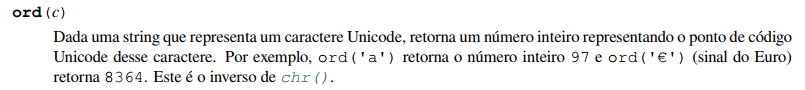


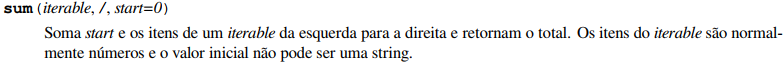
**Exemplo de Validação de strings com Regex**

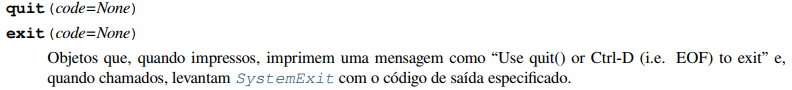
**Funções Embutidas**











**filter**(função,iterável) = volta um ITERATOR EXPRESSION **(não uma lista)** com todos os elementos em “iterável” que retornarem True na função

OBS: a função tem que ser do tipo de retornar verdadeiro ou falso, e iterável pode ser uma lista , tupla, ou qualquer coisa que consiga se aplicar na função e seja iterável) [iterável = poder pegar 1 por 1 basicamente]

**map** (função, iteráveis) -> faz a mesma coisa do que o filter, porém pode a função pode realizar qualquer operação, não precisando ser booleana

**int(str, base) -** conversão entre diferentes bases

**Uma imagem contendo Texto

O conteúdo gerado por IA pode estar incorreto.**

**Extra:** Preencher f-string com caracteres

Tela preta com letras brancas

O conteúdo gerado por IA pode estar incorreto.

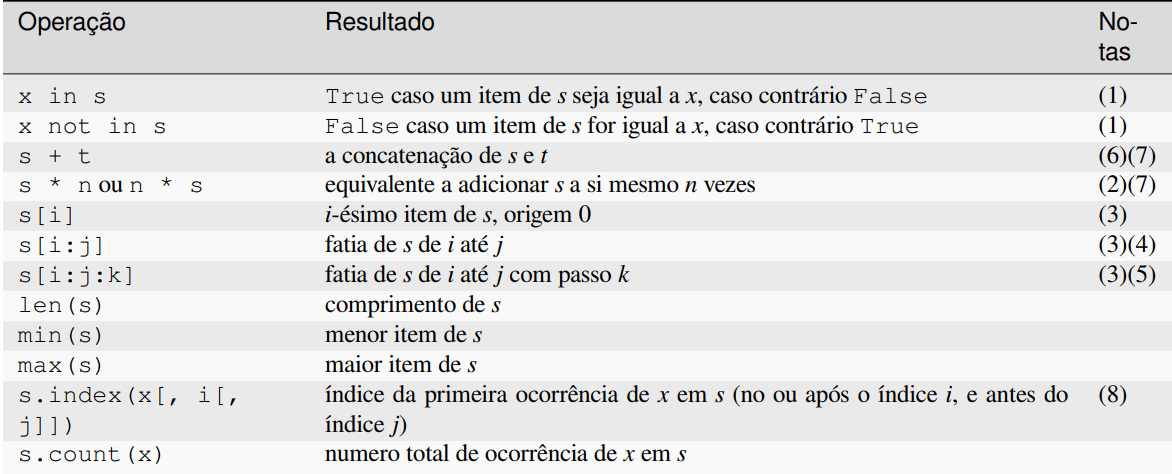
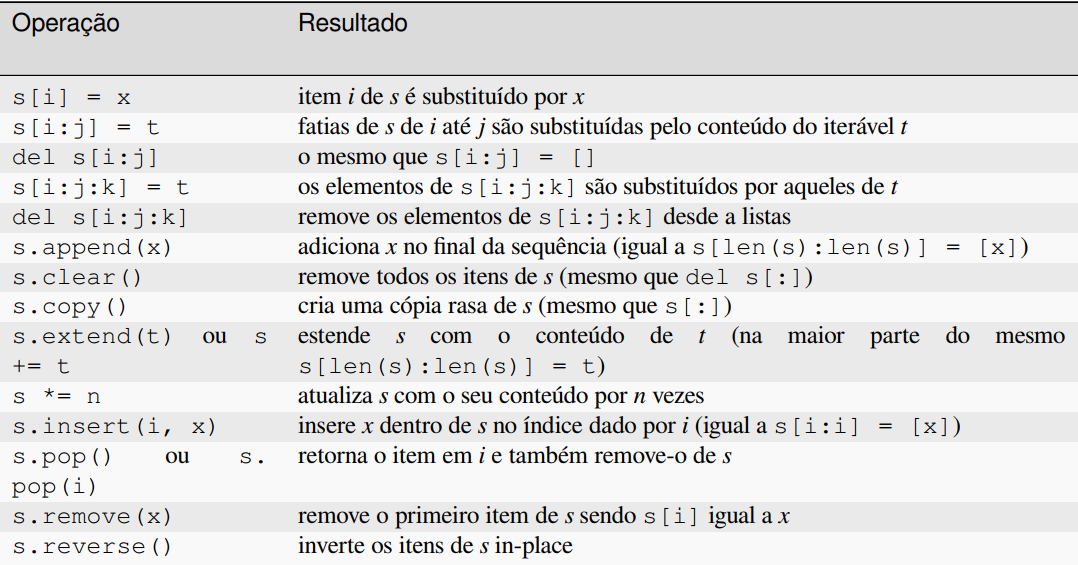
**Extra:** Converter base de número no momento de printar

**Interface gráfica do usuário, Aplicativo

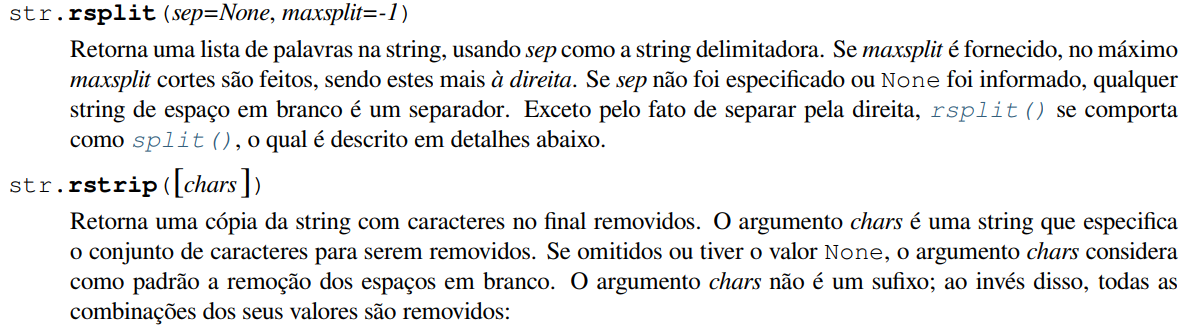
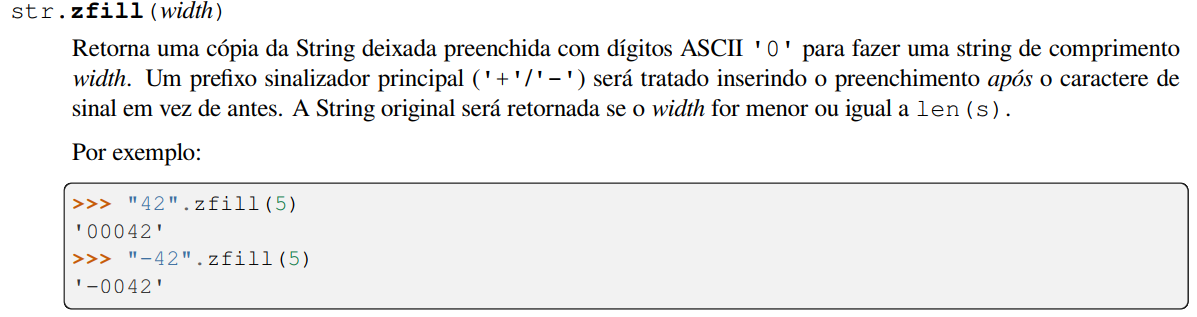
O conteúdo gerado por IA pode estar incorreto.**

**Miscelânea de Funções Úteis**

**Operações em Listas**



**Operações em Strings**



**Rsplit** como **Rstrip** (R = Right) tmb tem o L para Left -> Lsplit e Lstrip (tudo minúsculo, está em maiúsculo aqui apenas para melhor compreensão)

**str.startswith** e **str.endswith** (prefix[start:end]): retorna True se começar ou terminar com seu parâmetro e falso se não

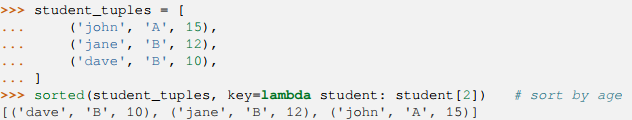
**str.swapcase()** troca maiúsculas por minúsculas e vice versa -> TeStE -> tEsTe

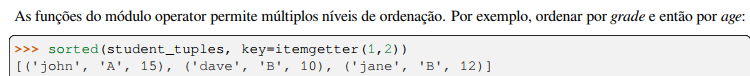
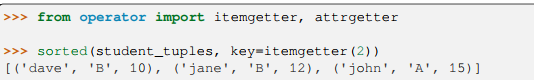
**str.ljust**(length, char) e **str.rjust**(length, char) – Alinha a string à esquerda ou direita utilizando o char informado

**Extra: Ferramenta “Sorted”**

**Sorted()** -> é igual o .sort() de list porém **RETORNA UMA NOVA LISTA** e funciona com qualquer iterável. Tal como o sort(), também possui tem parâmetro **key** que especifica o que deverá ser considerado para a ordenação.

EXEMPLOS:

* Organizando com base em ‘casefold’ (a>Z) 
* Organizado por idade onde esse lambda apenas faz uma ‘referenciação’, esse ‘student’ é como se fosse o ‘as’, está apenas dando um parametro para cada item da lista ‘student\_tuples’, e está chamando cada item o index 2 (terceiro item da tupla para organizar por idade)
* Para ordenar usando mais de um dado, passe os parâmetros da ordenação em uma tupla:  
  
* Organizar usando biblioteca itemgetter (para indexes) pode ser uma maneira de aumentar a eficiência



* Inverter usando parâmetro reverse (tanto sort quanto sorted)

**Bibliotecas**

**Biblioteca Collections**  
**Counter**(string) -> volta um dicionário com quantas vezes aquela letra apareceu

* **Ex**: Counter(‘Teste’) -> volta um dicionário {‘T’:1,’e’:2,’s’:1,’t’:1}

**Counter().most\_common**(inteiro) -> volta a quantidade de números que você colocar em inteiros os mais comuns.

* **Ex:** Counter(‘Teste’).most\_common(2) -> volta uma lista de tuplas dos que mais apareceram -> [(‘e’,2),(‘T’,1)] #obs> como todos valores depois do “e” só apareceram 1 vez, volta o que aparece primeiro
* **Extra**:> você pode usar index para pegar apenas as letras ou as quantidades que apareceram

**defaultdict(funcao) –** Dicionário que utiliza o valor padrão retornado pelo parâmetro “funcao” para as chaves que ainda não foram inicializadas. Evita ter que verificar se uma chave já foi inicializada antes de acessá-la.

**Biblioteca Itertools**

**product(lista1,lista2)** -> volta todas combinações de lista possíveis -> ex l1 = 1,2 e l2 = 3,4, product retorna -> (1,3),(1,4),(2,3),(2,4) #>para exibir colocar list(product(l1,l2))  
a = [1,2,3,4]

**combinations(a,2)** -> todas combinações com 2 valores de a, não repete, ou seja, 3,1 == 1,3, então não é uma combinação e sim uma variação de uma mesma combinação

**permutations(a,2)** -> a mesma coisa que combinations porém sem a restrição de variação de uma combinação

**accumulate(a)** -> o numero atual do index recebe ele + anterior, a saída de a ficaria = [1,3,6,10] onde 3 = a[0]+a[1]///, e ac[2] = 6 pq é == a a[2]+a[1]+a[0]

**Biblioteca Copy**

deepCopy = copiar objetos complexos (objetos com objetos dentro ou arrays com múltiplas dimensões)

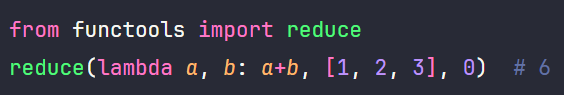
**Biblioteca String**

Listas de caracteres úteis

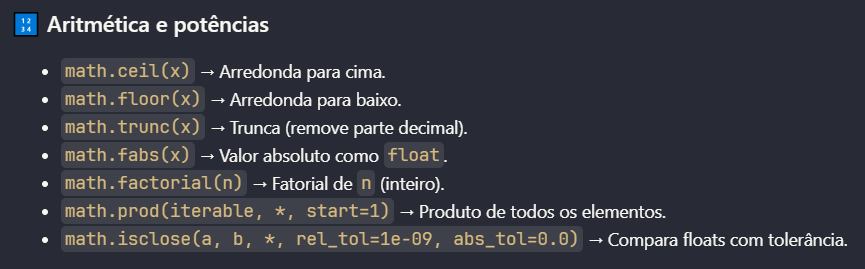
Texto

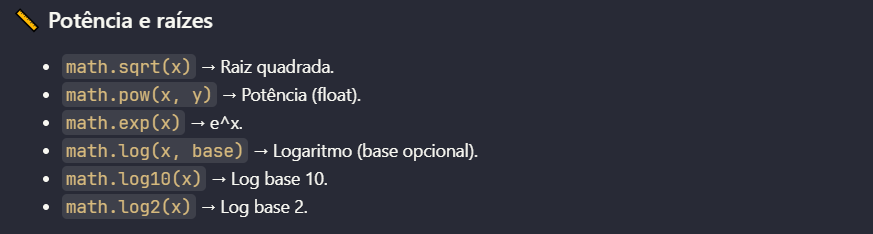
O conteúdo gerado por IA pode estar incorreto.

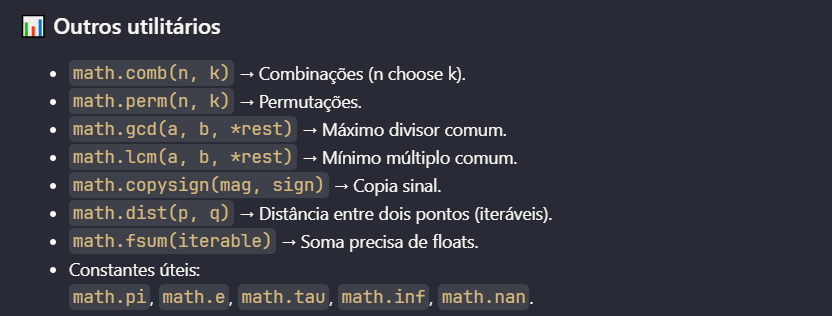
**Biblioteca Functools**

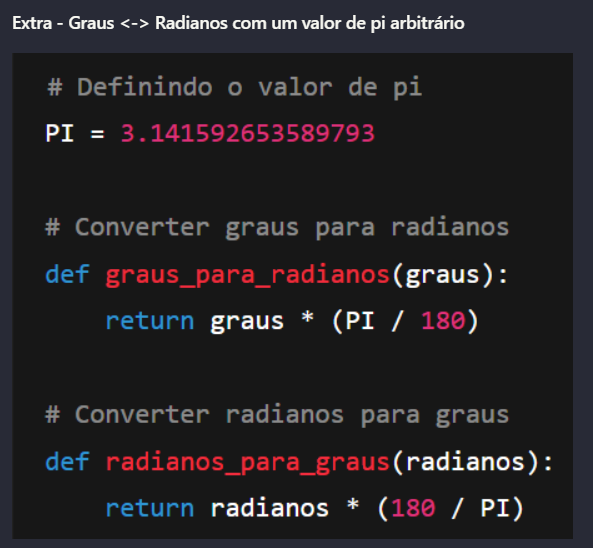


**Biblioteca Math**



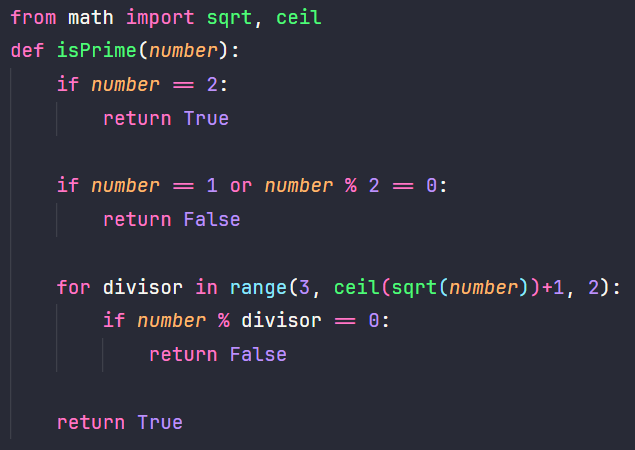






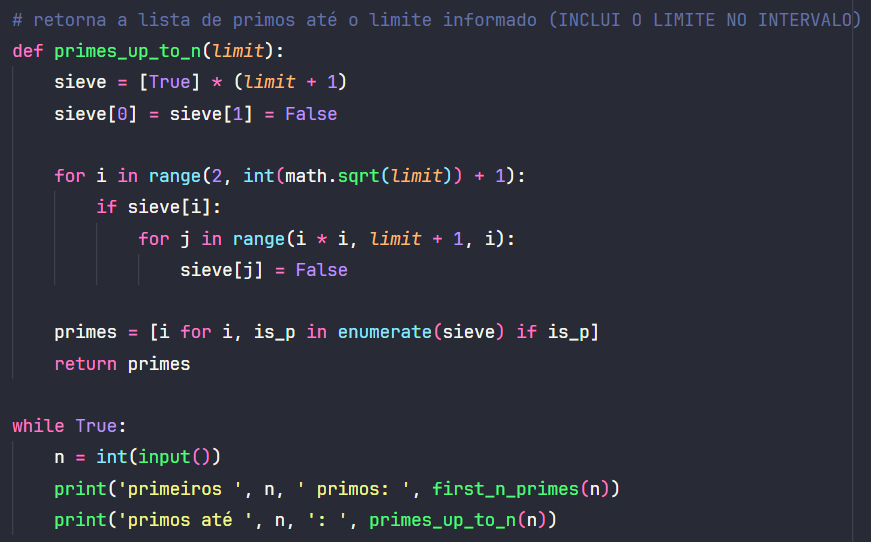
**Algoritmos Úteis**

**PRIMOS - Verificar se um número é primo**



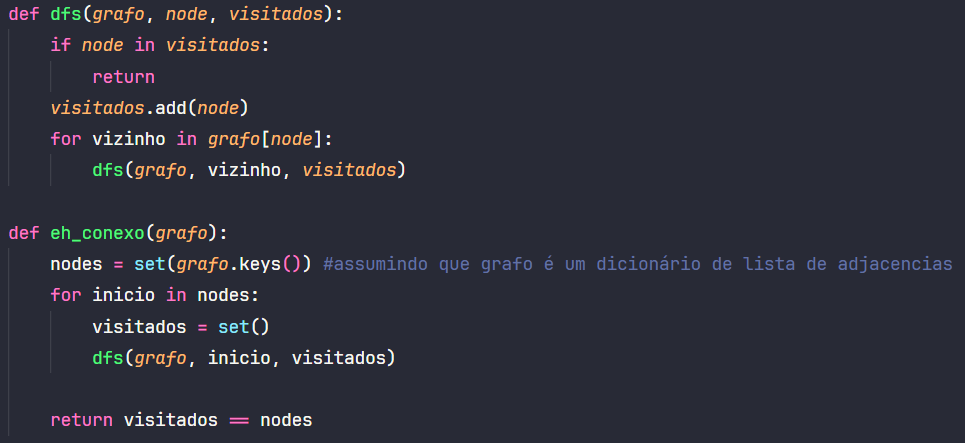
**PRIMOS - Gerar lista dos N primeiros primos/ Encontrar enésimo primo**

**PRIMOS - Gerar lista de primos até um número N (inclui o N se N for primo)**

****

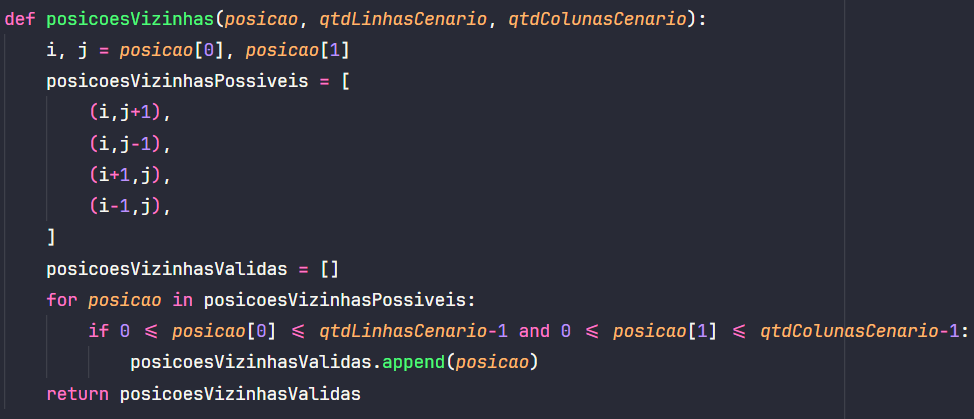
**GRAFOS – Verificar se grafo é conexo com DFS**

Grafo Conexo = Grafo “conectado”; Quer dizer que é possível viajar entre quaisquer dois pontos.



**GRAFOS – Encontrar menor caminho entre dois pontos em GRID com BFS**

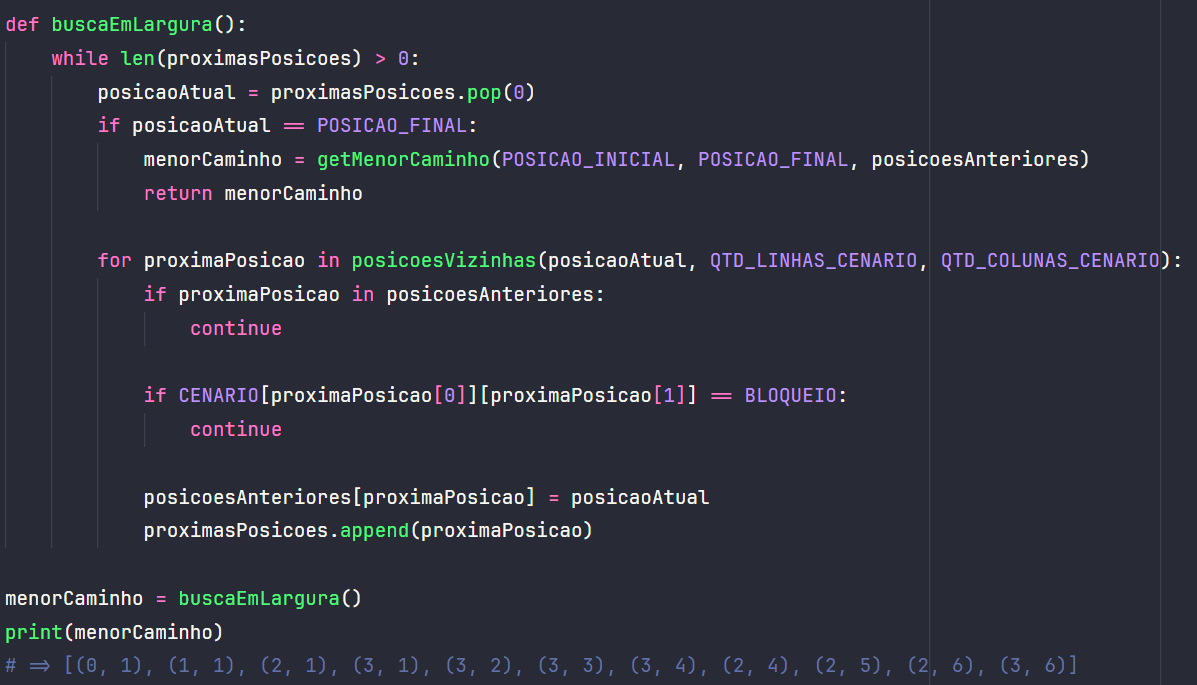
BFS Passo 1 - Função para retornar posições vizinhas de um nó

****

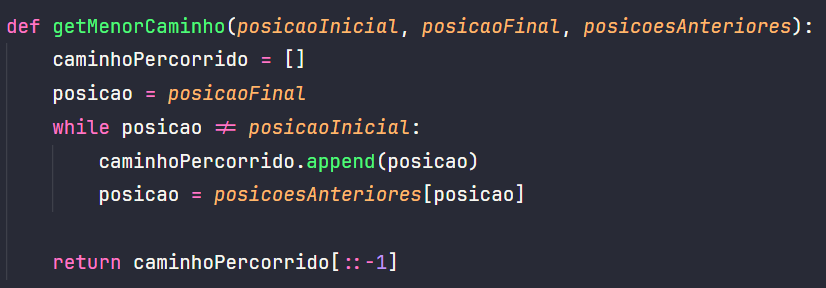
BFS Passo 2 - Definições iniciais



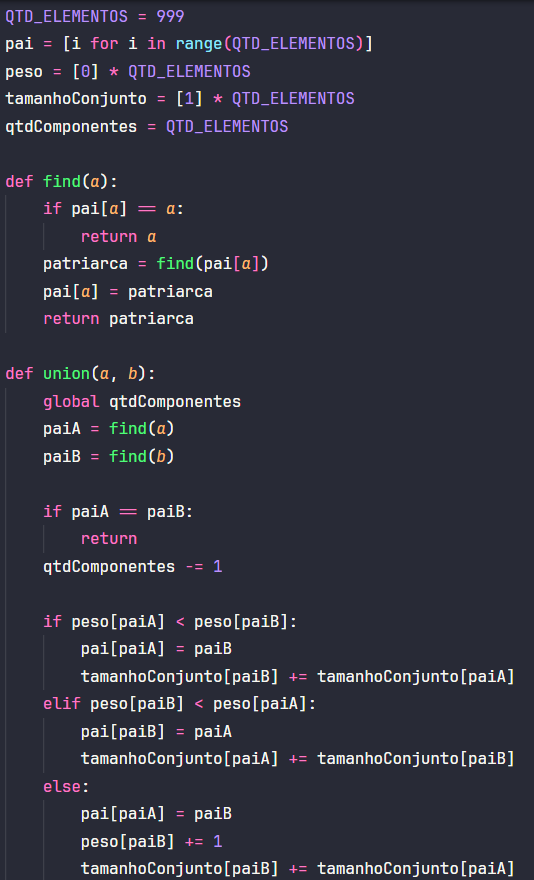
BFS Passo 3 - Busca em largura utilizando fila de próximas posições



BFS Passo 4 – Função para retornar menor caminho

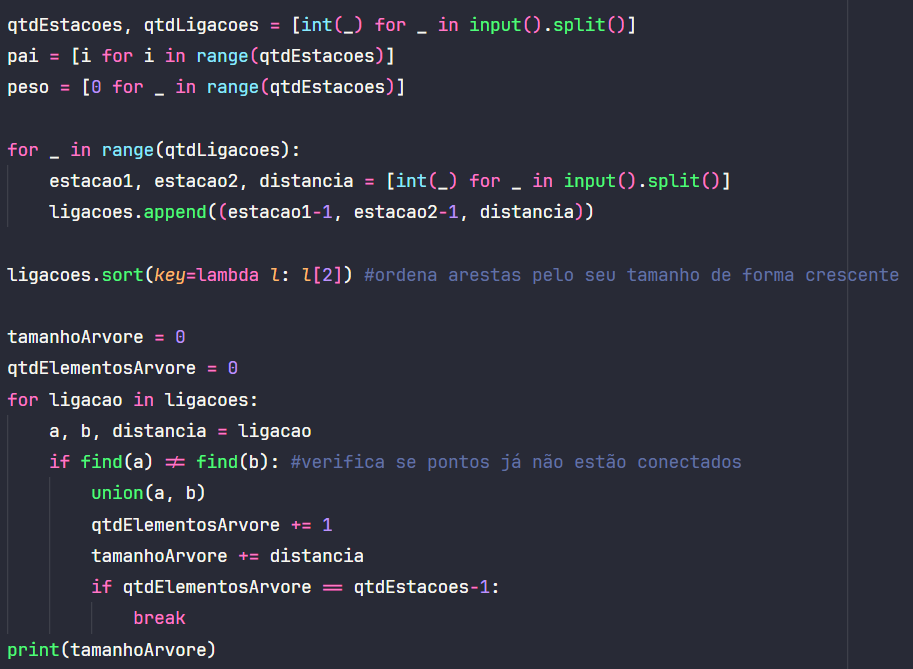


**GRAFOS – Union Find (União de conjuntos disjuntos)**

****

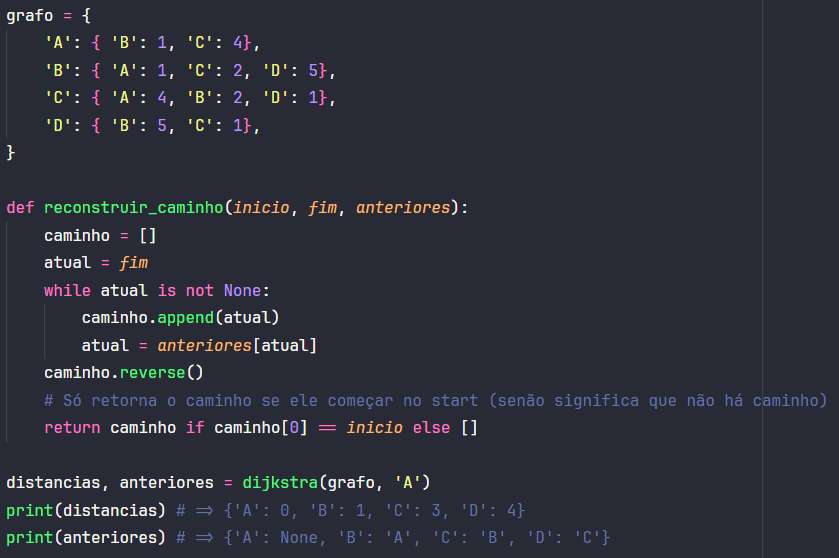
**GRAFOS – Árvore Geradora Mínima (MST) com algoritmo de Kruskal**

A ideia é percorrer a lista de arestas ordenada pelo tamanho de forma crescente e unir os pontos dessas arestas se eles já não estiverem na mesma componente. Utiliza Union-Find para fazer a união e a verificação se dois pontos estão no mesmo conjunto.

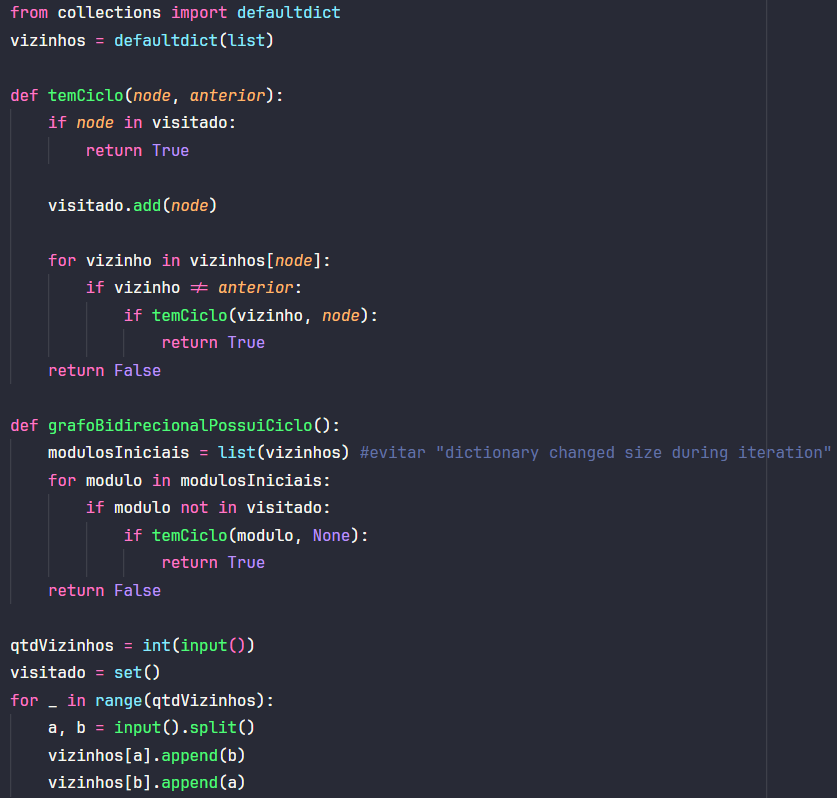
****

**GRAFOS – Menor caminho entre dois pontos com Dijkstra**

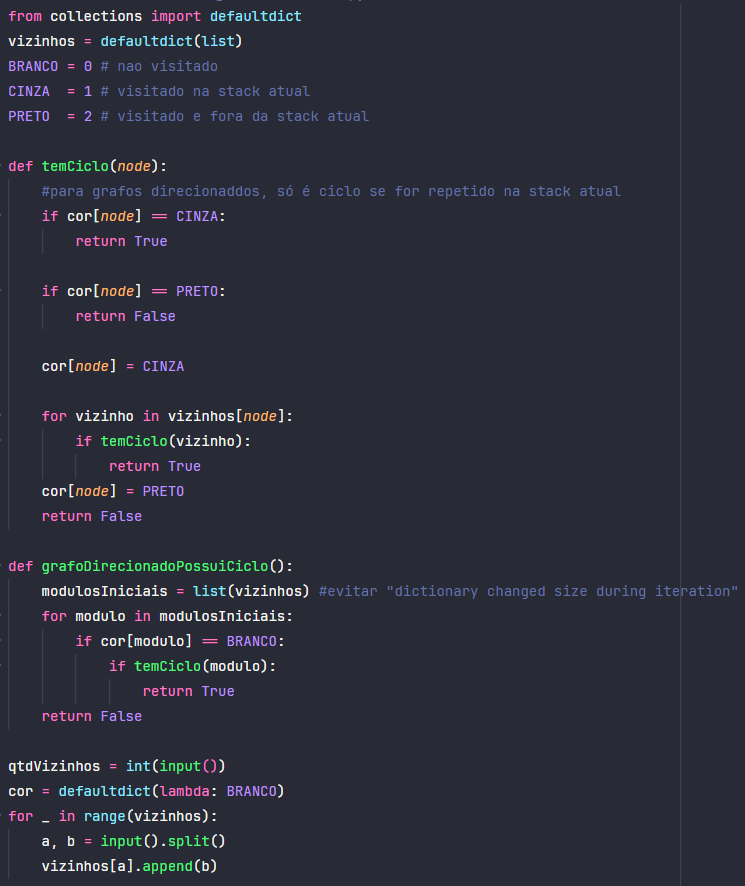
****



**GRAFOS – Verificar loop em grafo NÃO DIRECIONADO**

****

**GRAFOS – Verificar loop em grafo DIRECIONADO**



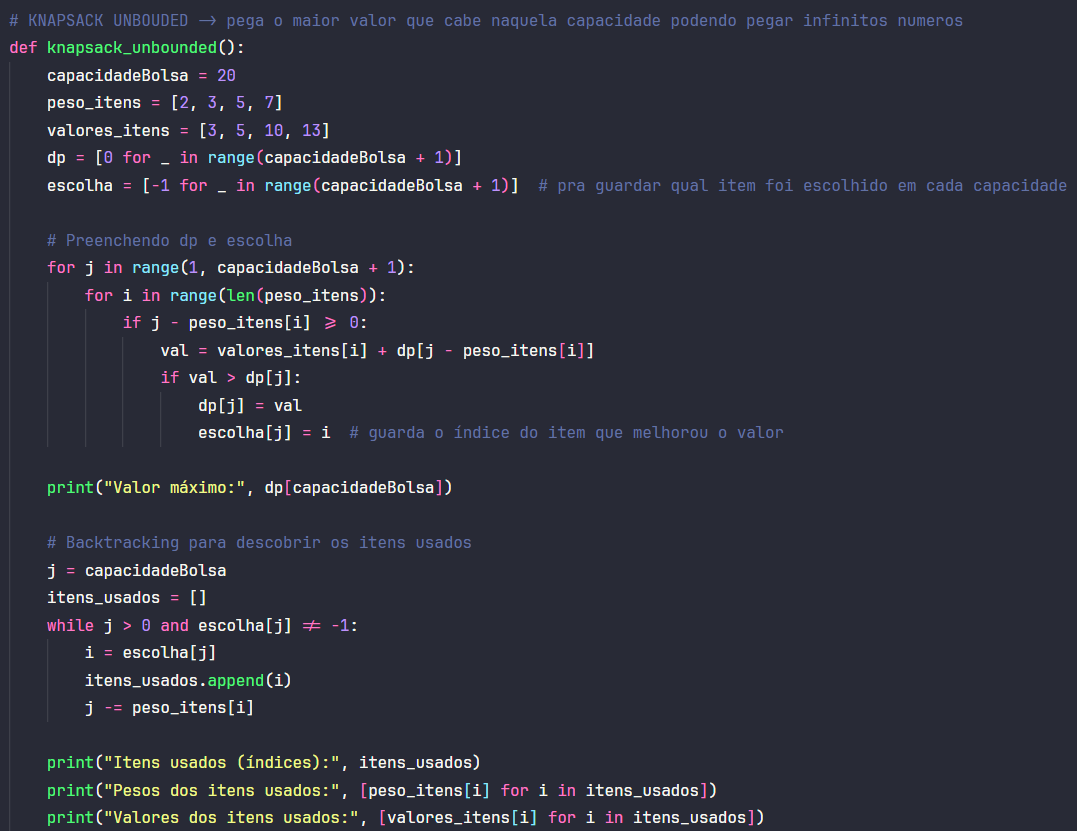
**KNAPSACK – 1/0 – Com Backtracking**



**KNAPSACK – Bounded**

****

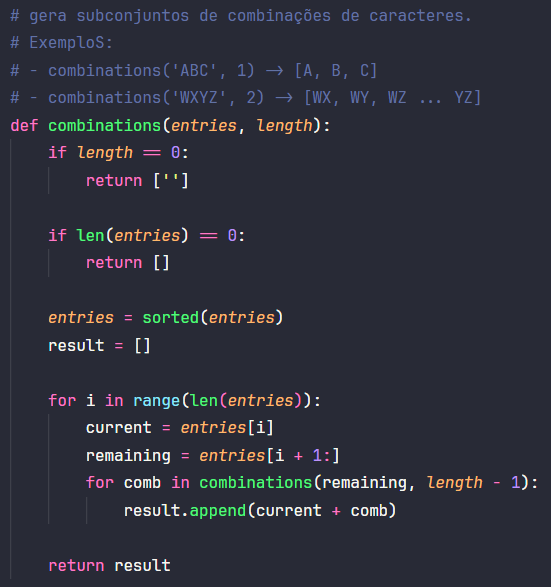
**KNAPSACK – Unbounded**

****

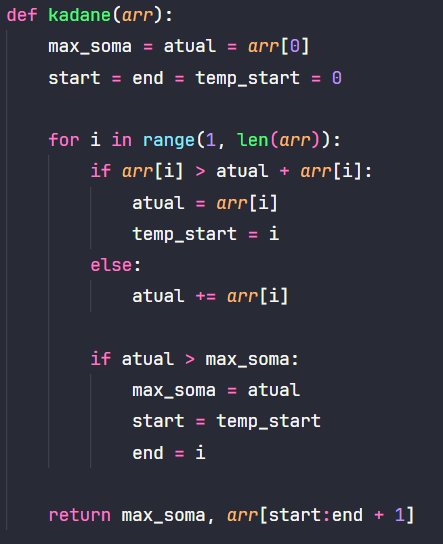
**KNAPSACK – Subset Sum**

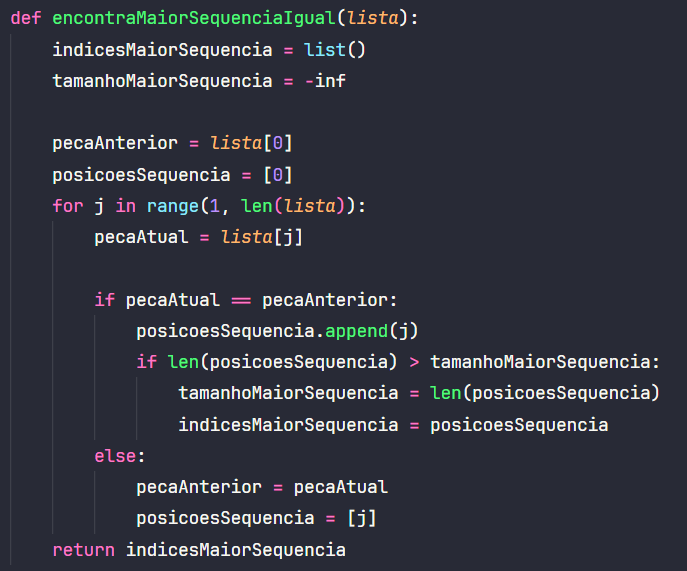
****

**Gera subconjuntos de elementos de array**

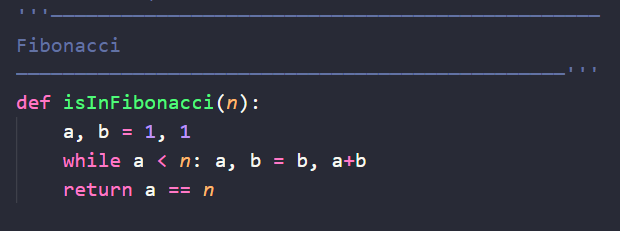
****

**Maior subsequência contígua (Kadane)**

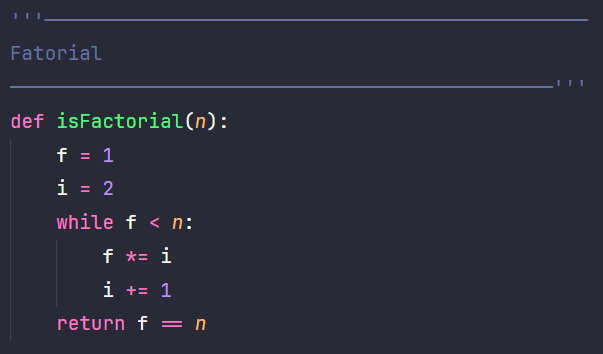
****

**Maior sequência de elementos iguais**

**Verificar se um número pertence à sequência Fibonacci**



**Verificar se um número é fatorial**

****