**Material de Referência.py**

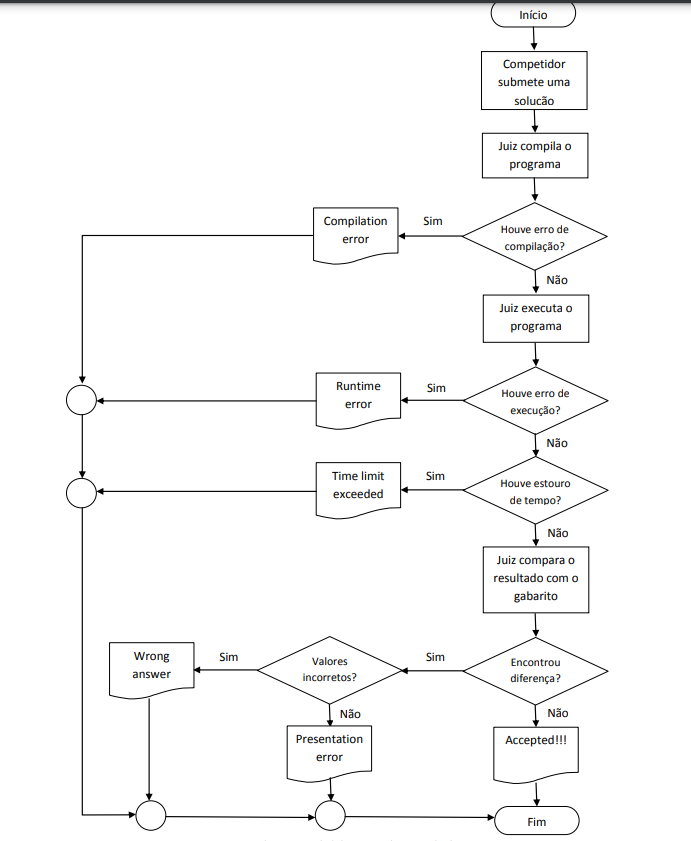
**Código Trifásico – FATEC Itapira**

Alexandre Alcindo  
Gabriel Almir  
Rodrigo Polastro

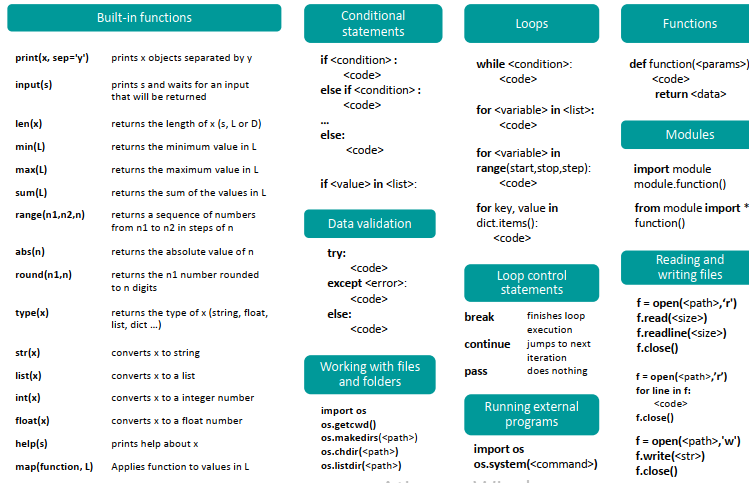
Técnico: Prof. Júnior Gonçalves

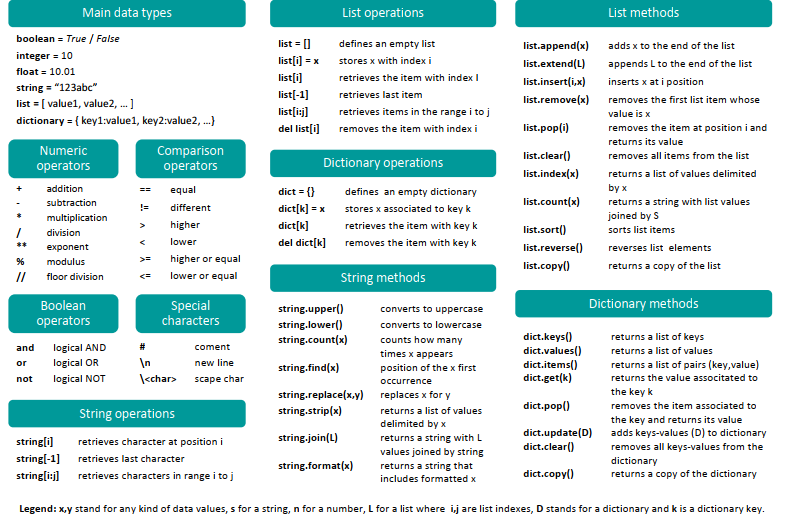
2025

**Fluxo de Correção no Interfatecs**

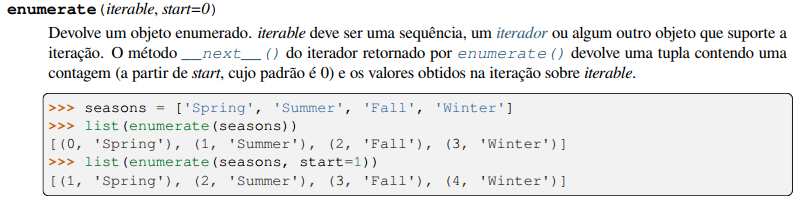
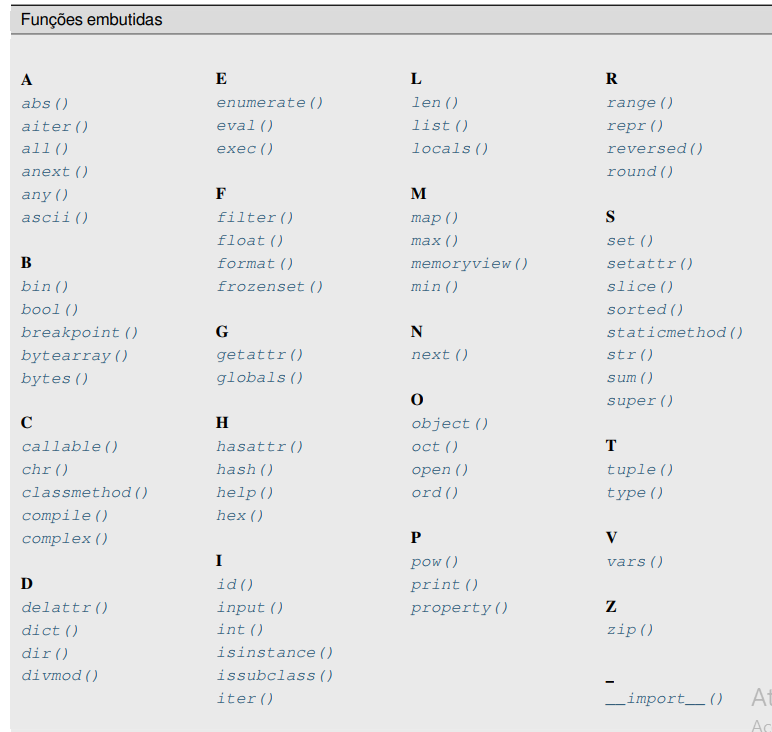


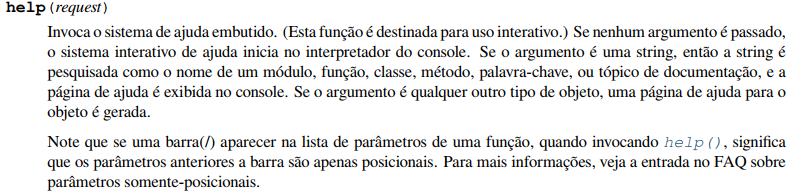
**Referência Básica Python**

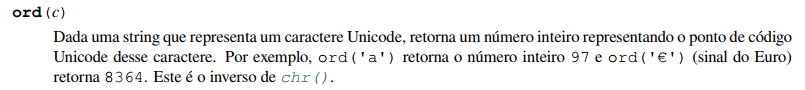


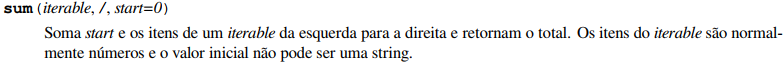


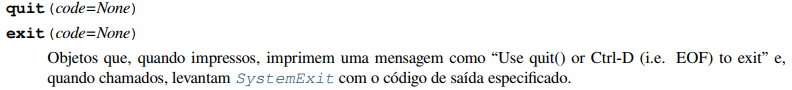
**Funções Embutidas**











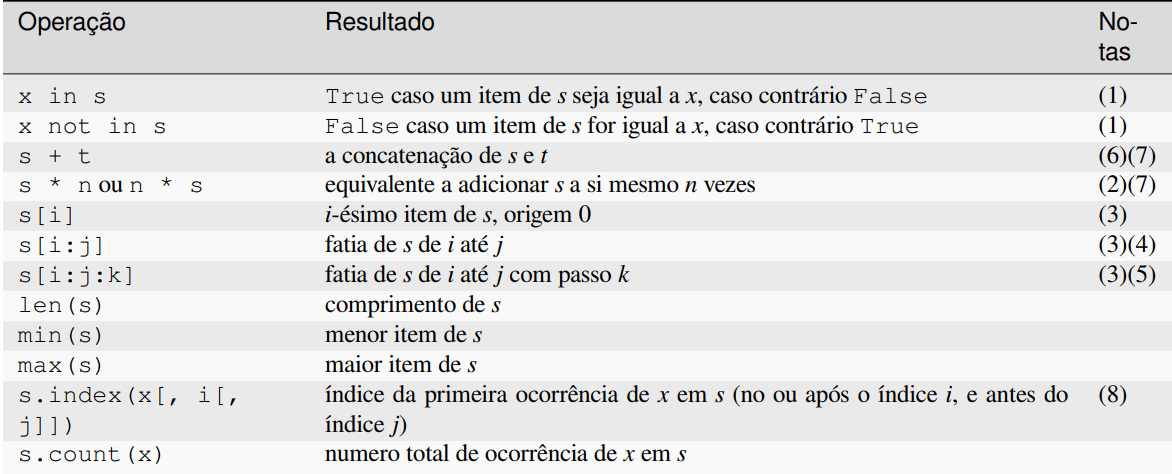
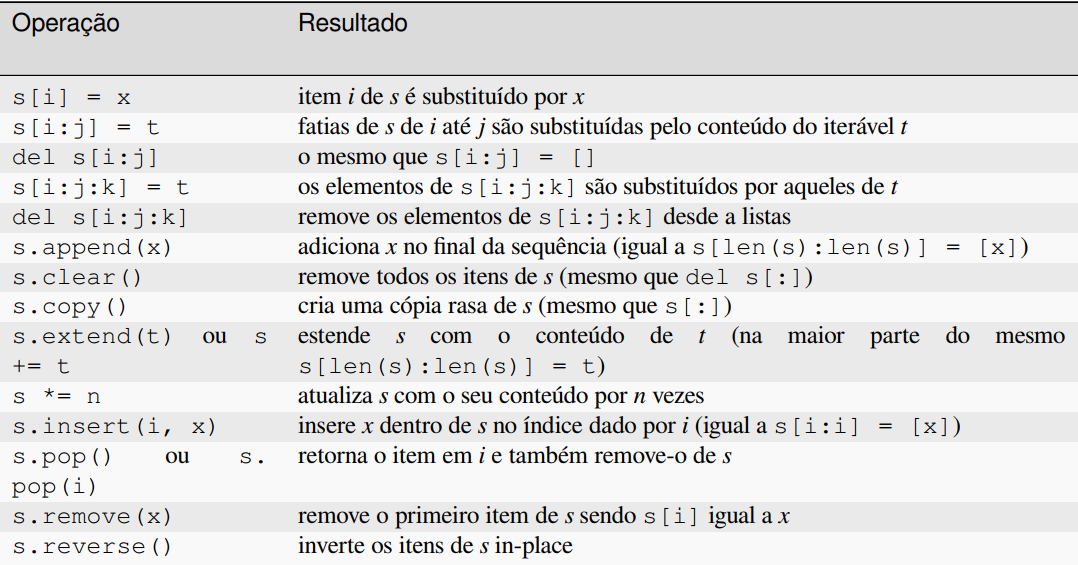
**filter**(função,iterável) = volta uma lista com todos os iteráveis que retornarem True na função

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 básicamente]

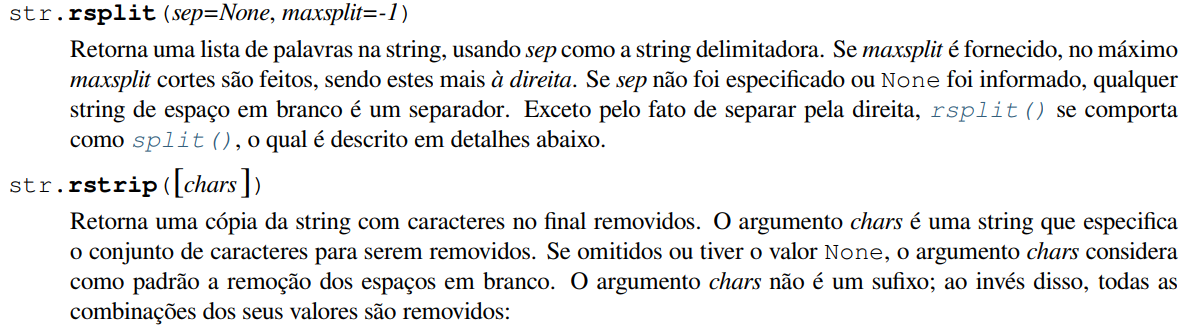
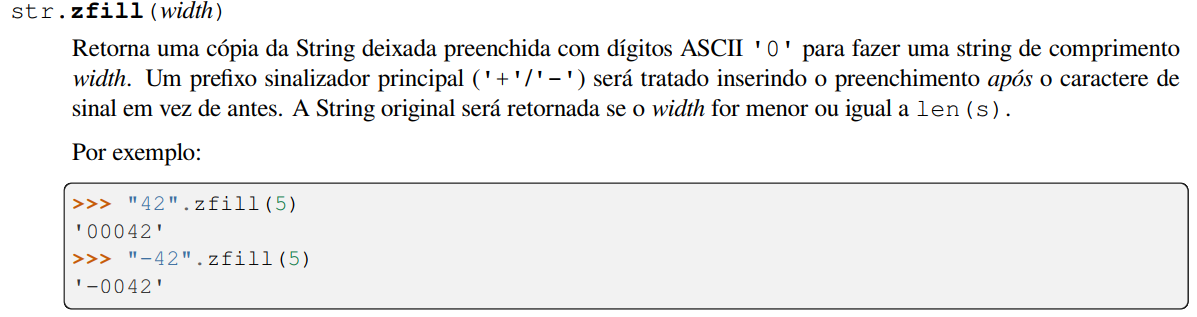
**map** (função, iteráveis) -> faz a mesma coisa do que o filter porem pode receber qualquer valor além de verdadeiro ou falso

**math.isqrt(n)** -> retorna inteiro da raiz (sqrt() normal retorna float da raiz)

**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

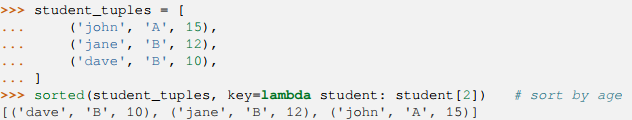
**str.ljust**(length, char) e **str.rjust**(length, char) – Alinha a string à esquerda ou direita utilizando o char informado

**Extra: Ferramenta “Sorted”**

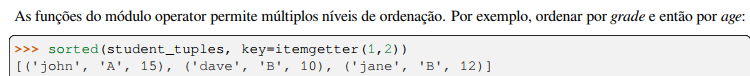
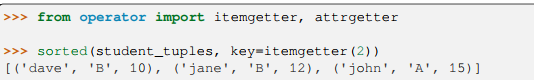
Sorted() -> é igual o .sort() de list porém funciona com qualquer iterável (sort LEVEMENTE mais eficiente que sorted), quanto sort e sorted tem parâmetro key que especifica em base oque deve ser organizado, ex:

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)



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 de teste:

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 so apareceram 1 vez, volta o que aparece primeiro ##extra> você pode usar index para pegar apenas as letras ou as quantidades q apareceram

**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]

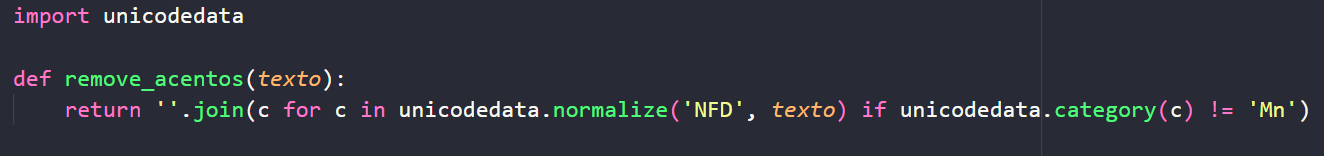
combinations(a,2) -> todas combinações com 2 valores de a, não repete, ou seja, 3,1 == 1,3, então não é uma combinação e sim uma variação de uma mesma combinação

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]

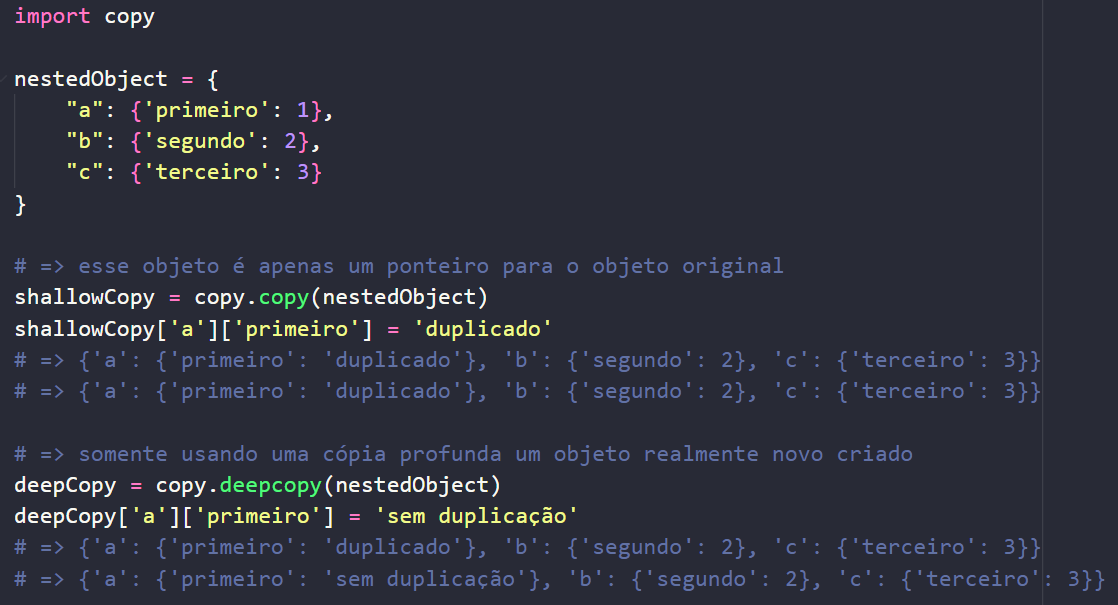
**Algoritmos Úteis**

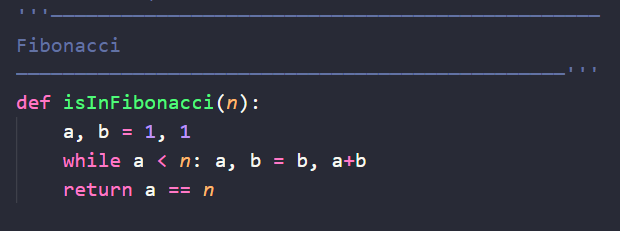
**Remover acentos de string**

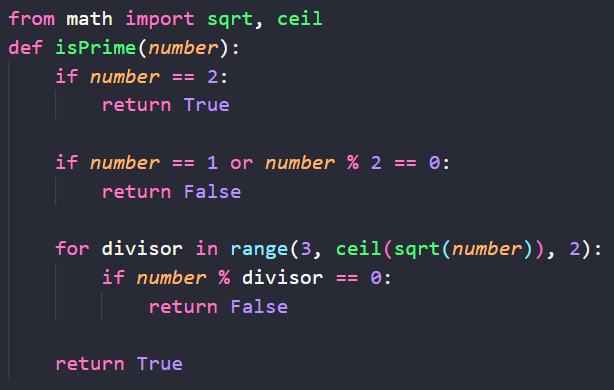


**Copiar Objetos Complexos:**

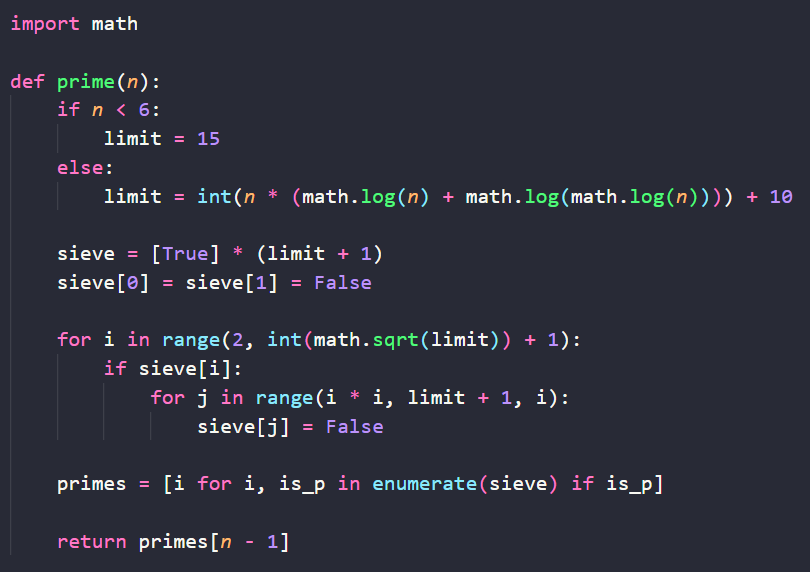
Quando precisar copiar objetos complexos (objetos com objetos dentro ou arrays com múltiplas dimensões), utilize a função “deepcopy” da biblioteca “copy”. Caso contrário, apenas criará um ponteiro para o objeto original



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

**Verificar se um número é primo**

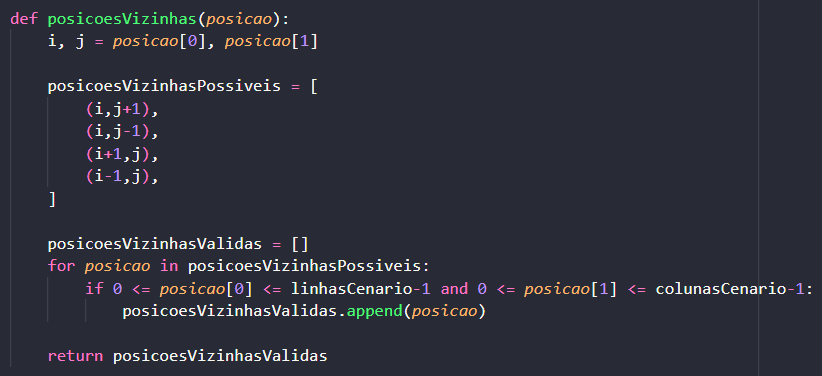
**Encontrar o enésimo primo (otimizado)**



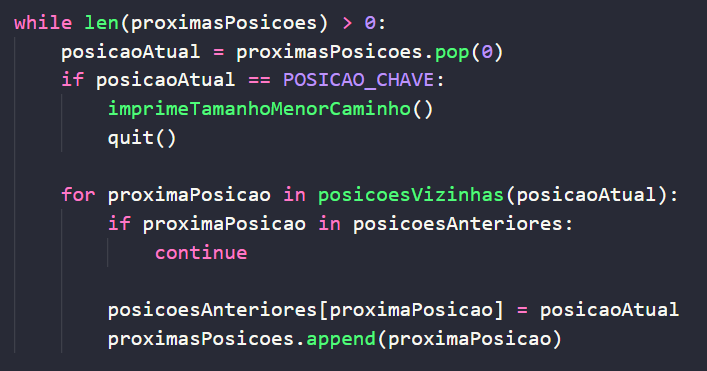
**Busca em profundidade em um grafo (Depth First Search – DFS)**



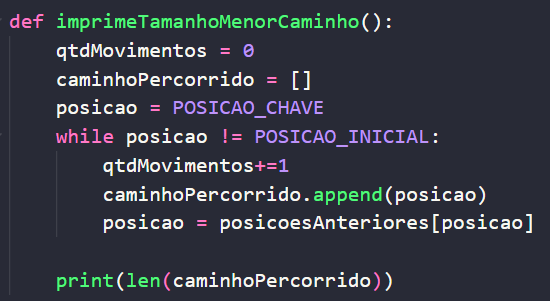
**Busca em largura em um grafo (Breadh First Search – BFS) – APLICADO EM GRID**

1 – Função para retornar posições vizinhas de um nó

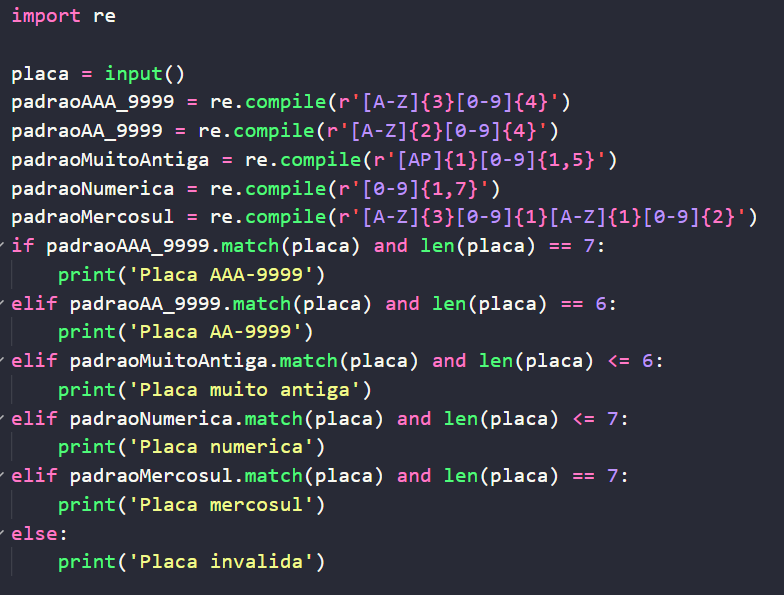
2 – Loop de busca utilizando “fila”



3 – Calcular e printar tamanho do menor caminho entre dois pontos



**Validar entradas com padrões de regex (compile e match)**

****