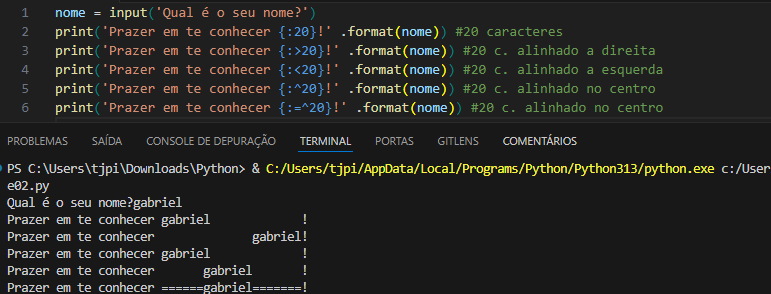
**PYTHON**

Arquivos python são salvos como .py

Para colocar algo formatado posso colocar f’: print(f'Olá {nome}! Seja bem-vindo!')

**Formatação:**

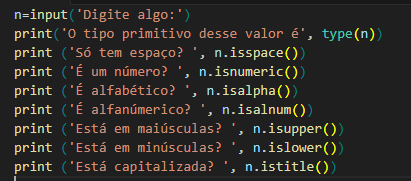
:*.2f == duas casas decimais* ***após o ponto flutuante****.*

**Tipos Primitivos:**

1. **int()** – nº inteiro.
2. **float()** – nº real.
3. **bool()** – valores lógicos.
4. **str()** – caracteres ou string.

Para verificar o tipo é apenas colocando print(type()).

**UTEIS:**



**Operadores Aritméticos:**



Obs: **Igual para Python se escreve ‘==’**

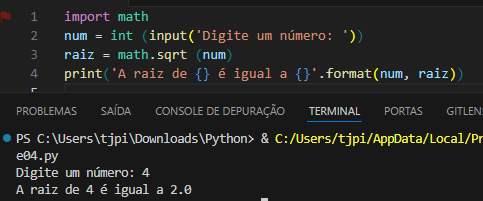
**Módulos:**

EX: import bebida

Mas se eu não quiser todas as bebidas? Eu utilizo from bebida import vodka

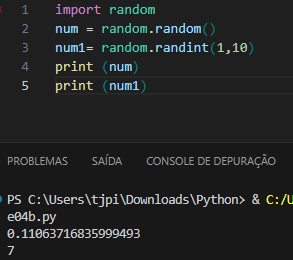
Uma grande biblioteca que é utilizada é a *math*:

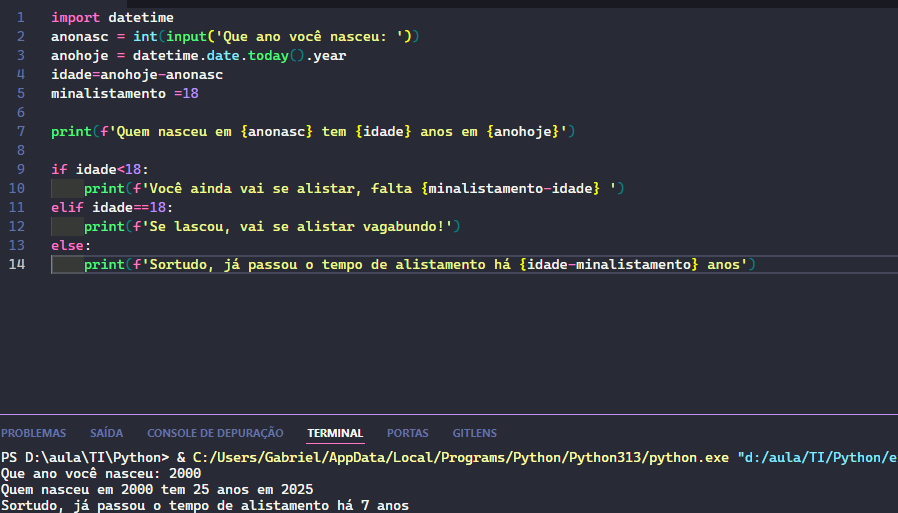
* ceil – Arredondar para cima
* floor – Arredondar para baixo
* trunc – Eliminar da virgula para frente
* pow – Potencia
* sqrt – Calcular raiz quadrada
* factorial – Calcular fatorial.
* hypot – Calcular hipotenusa.

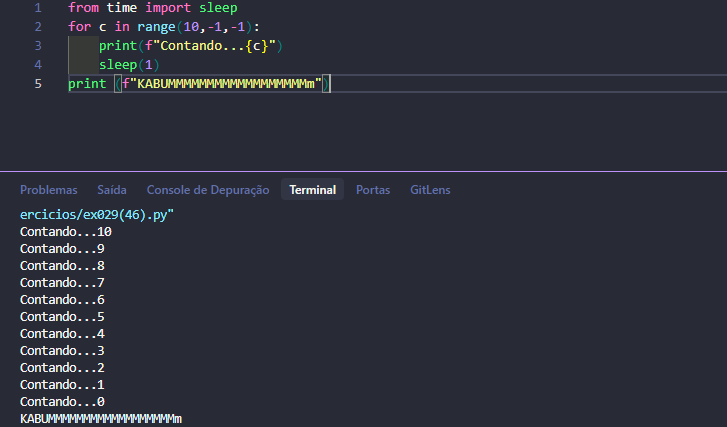


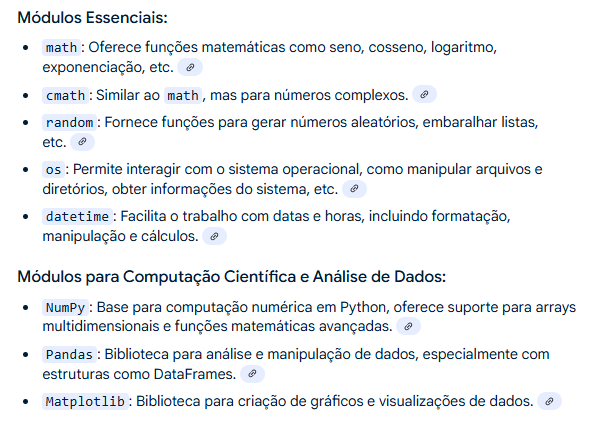
Bibliotecas podem ser importas: [*https://docs.python.org/3.13/library/index.html*](https://docs.python.org/3.13/library/index.html)- No qual esses são módulos feitos pelo Python.

OU <https://pypi.org/> - Módulos feitos pela comunidade.









. time — Temporizador

. pprint — Imprime listas e dicionários de forma **mais organizada**.

**Os módulos padrão mais populares são:**

* collections (coleções adicionais para complementar tupla, lista, conjunto, dic de builtinst)
* itertools (iteradores adicionais complementando iter, next, map, filter, zip, reversed, enumerate de builtins)
* string (essencialmente atributos str compartimentalizados)
* re (expressões regulares, correspondência de padrões de string)
* math (funções matemáticas)
* statistics (funções estatísticas)
* fractions (frações)
* decimal (números de ponto flutuante base 10)
* random (geração de números pseudoaleatórios)
* datetime (horas e datas)
* zoneinfo (fusos horários para usar com o datetime)
* os (sistema operacional, semelhante ao uso do bash para navegar pelo sistema operacional no terminal Linux)
* sys (o módulo sys é usado para interagir com o interpretador Python e recuperar detalhes sobre o ambiente Python)
* pathlib (gerencie facilmente caminhos de arquivos)
* csv (leia e escreva arquivos csv para objetos Python embutidos)
* json (leia e escreva arquivos json para objetos Python embutidos)
* pickle (serializa um objeto Python)
* shelve (arquive objetos serializados)
* pip (biblioteca padrão)
* venv (ambientes virtuais python)

Módulos de terceiros são desenvolvidos separadamente do Python. A pilha numpy é muito amplamente utilizada e se baseia no conhecimento adquirido com o aprendizado dos módulos padrão, pois transmite sua funcionalidade a dados de dimensões superiores.

**Pilha numpy:**

* numpy (Python numérico transmitindo int, float, complex, math, statistics, random e datetime)
* pandas (python e análise de dados, baseado em um DataFrame que pode ser conceituado como uma planilha Excel)
* matplotlib (biblioteca de plotagem matemática)
* seaborn (visualização de dados, biblioteca wrapper em torno do matplotlib simplificando gráficos mais complexos)
* conda (gerenciador de pacotes de ciência de dados e gerenciamento de ambiente, usado em vez de pip e venv para dependências Python e não Python)

**Formatadores e linters de terceiros:**

É útil saber o que cada formatador e linter faz, pois isso o tornará mais consciente sobre a maneira como você formata seu código Python [PEP 8: Guia de estilo Python](https://peps.python.org/pep-0008/)

* autopep8 (formata o código de acordo com o autopep8)
* isort (ordenação de importações)
* black (formatador opinativo black)
* ruff (formatador rápido rust)
* flake8 (linter)
* pyflakes (linter)

**Manipulando cadeia de caracteres:**



Operações que podem ser feitas:

1. **Fatiamento** – Pegar pedaços dela.
   1. frase[9] ele vai pegar só o V
   2. frase[9:13] ele vai pegar o V até o E (excluindo o 13), sempre 1 a menos no final.
   3. frase[9:21:2] ele vai pegar o V até o N pulando de 2 em 2.
2. **Analise** – Saber informação sobre a frase.
   1. *len*(frase) – saber a quantidade de caracteres (21 caracteres)
   2. frase.*coun*t*(‘o’)* – contar quantas vezes aparece o ‘o’ (3 caracteres)
      1. frase.count(‘o’,0,13) – pedindo para contar 0 até o 13 (excluindo o 13).
   3. frase.*find*(‘deo’) –aonde encontrou o texto mencionado. # Se ele responder -1 é pq não existe.
   4. ‘curso’ *in* frase – verificar se existe a palavra curso na frase. (true ou false)
3. **Transformação** – mudar atravês de metodos.
   1. frase.*replace* (‘Python’,’Android’) – troca a palavra 0 por 1.
   2. frase.*upper*() – colocar tudo em maisculo.
   3. frase.*lower*() – colocar tudo em minusculo.
   4. frase.*capitalize*() – coloca só a primeira letra em maisculo.
   5. frase.*title*() – coloca tudo após os espaços em maisculo.



1. frase.*strip*() – remove espaços inuteis (0:2 e 17:18)
2. frase.*rstrip*() – remove do lado direito.
3. frase.lstrip() – remove do lado esquerdo.
4. **Divisão**
   1. frase.split() – divisão na string entre os espaços.
   2. ‘-‘.join(frase) – juntar as strings.

**Condições:**



**OR**



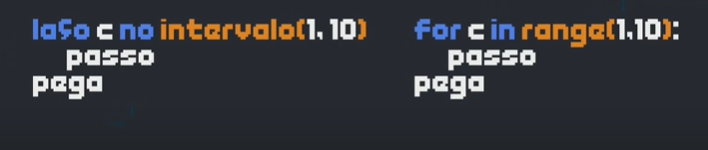
**Condições Aninhadas:**



A primeira condição verdadeira é executada

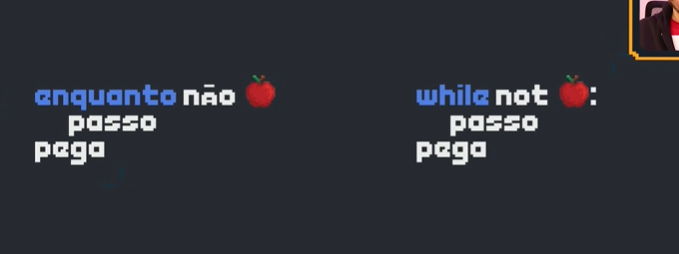
As outras são ignoradas, mesmo que também fossem verdadeiras

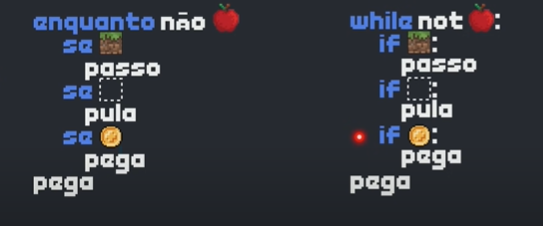
**Laço de Repetição:**



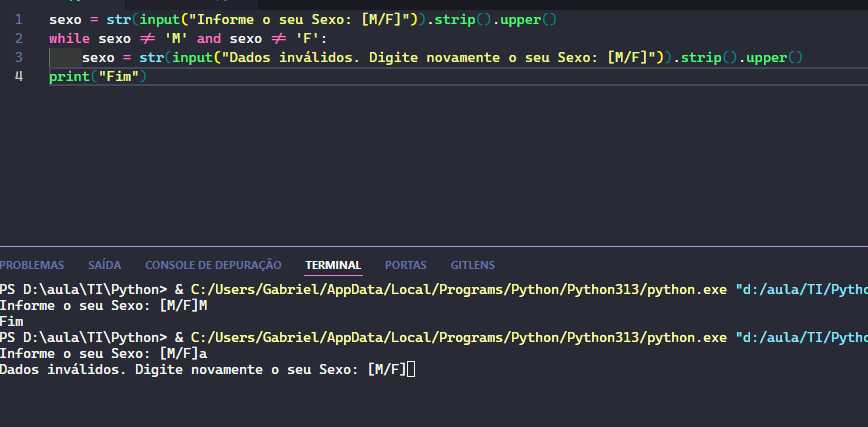
OBS: Se você colocar de 1 a 6 ele só repete 5 vez, pois ele não considera a ultima.

**Estrutura de Repetição WHILE: (quando não sabe o limite de repetição)**





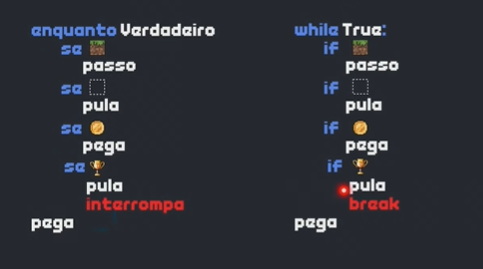
EX:



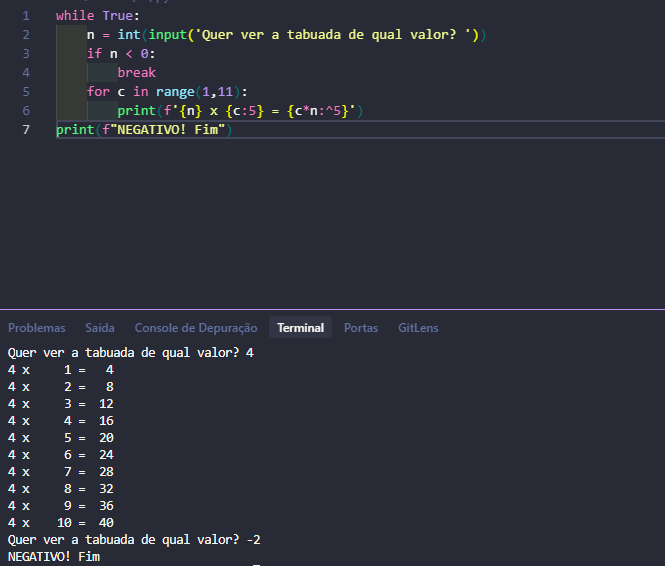
**Break para interromper loop:**

No python não tem o Do While.

Então para executar pelo menos uma vez precisa utilizar o loop infinito.



EX:



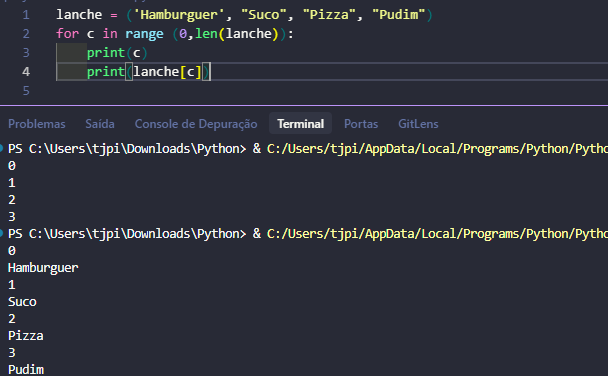
**Variaveis Compostas:**

**Tuplas são colocadas com “()”, listas são colocadas com “[]” e discionario com “{}”**

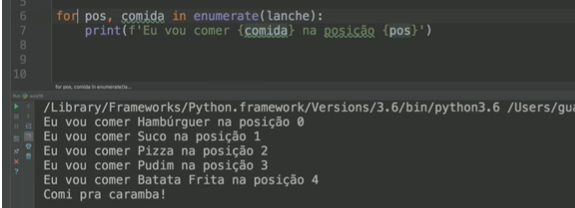
****

**Tuplas:**

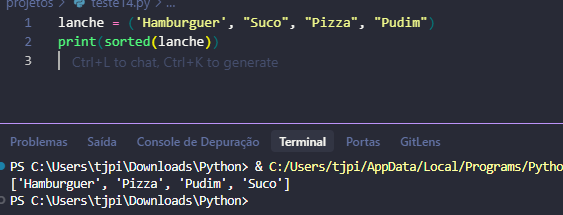
**AS TUPLAS SÃO IMUTÁVEIS!**



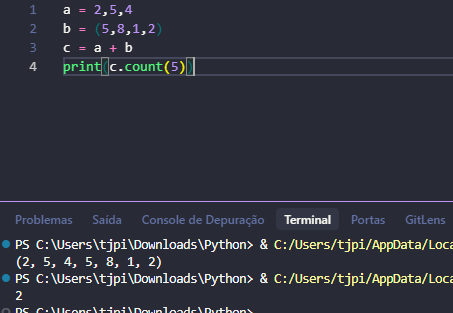
**enumerate()** é uma função que serve para iterar sobre itens de uma sequência (**como tuplas, listas ou strings**), retornando o índice (posição) e o valor ao mesmo tempo.



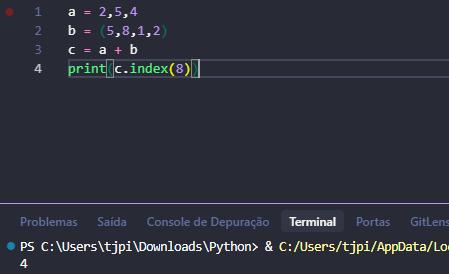
**Metodo Sorted: (em ordem)**



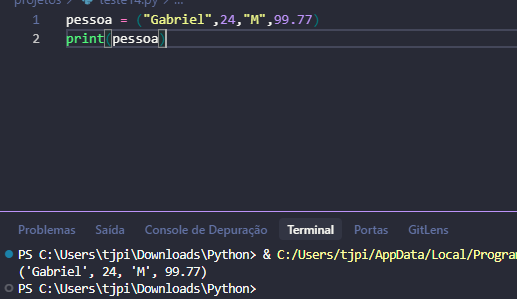
**Count: quantas vezes aparece o numero**



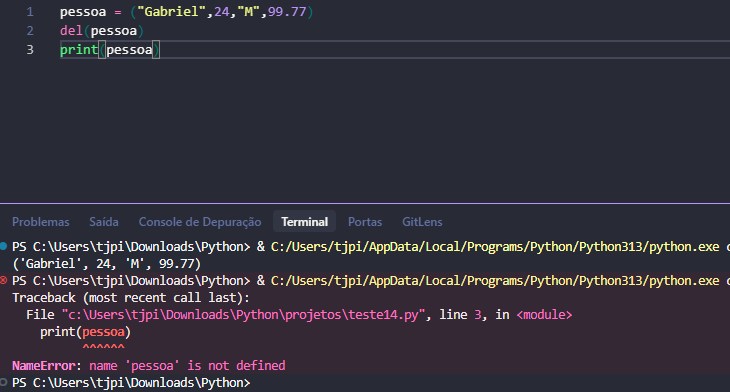
**Index: “Em qual posição”**



**Tupla aceita todas as variaveis:**

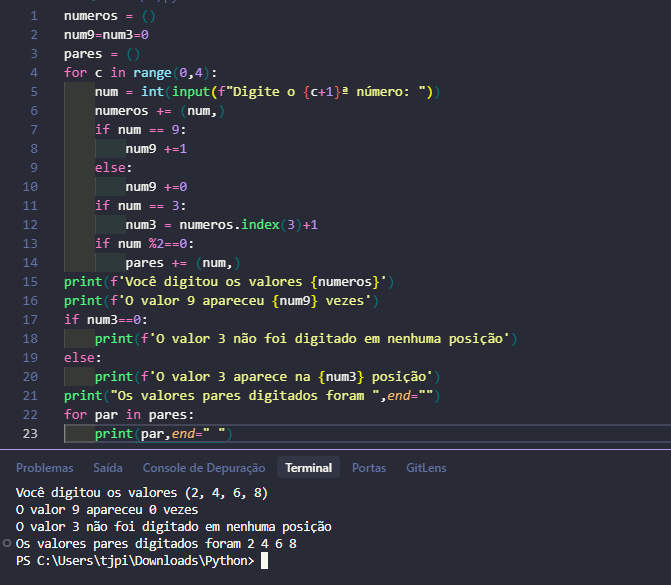


**Apagando tupla:**



Pode usar Max e Min para saber o maximo e minimo em uma tupla!

Pode add um numero na tupla assim:



**Listas:**

São feitas com ”[]”.

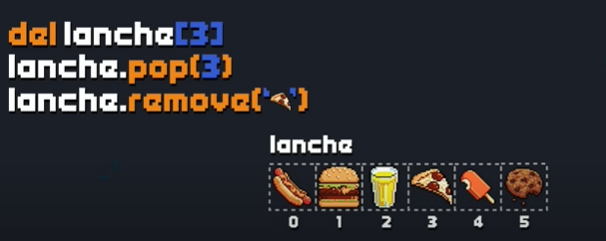
**Listas são mutaveis.**



Inclusive posso adicionar novos elementos. (**append**)

Adicionar elementos entre os elementos já existentes. (**insert**)

Comandos para eliminar. (**del** e **pop** *- geralmente para o ultimo se não indicar nada -*e **remove** – informar o elemento**-**)



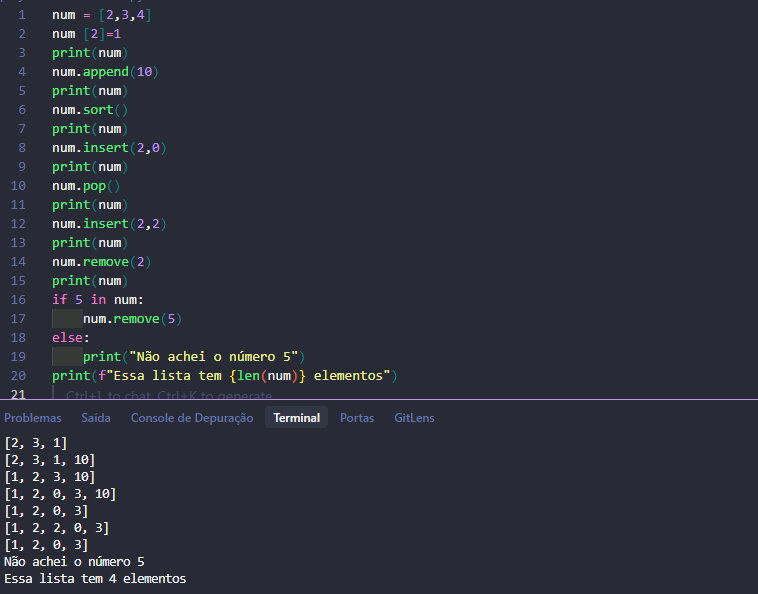
Verificar se existe o elemento. (**if “elemento” in lista:**)

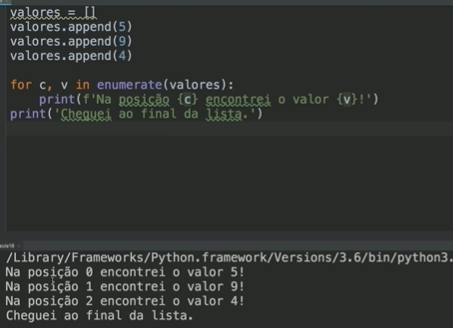
**Criar lista atraves de range.**



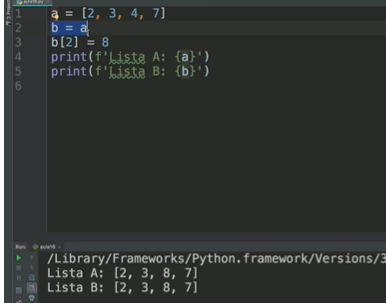
Colocar em ordem. (**sort**)

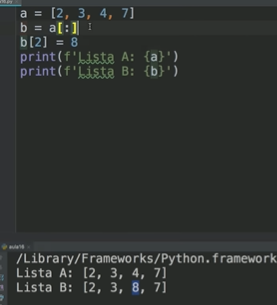
Saber quantos elementos tem. (**len**)









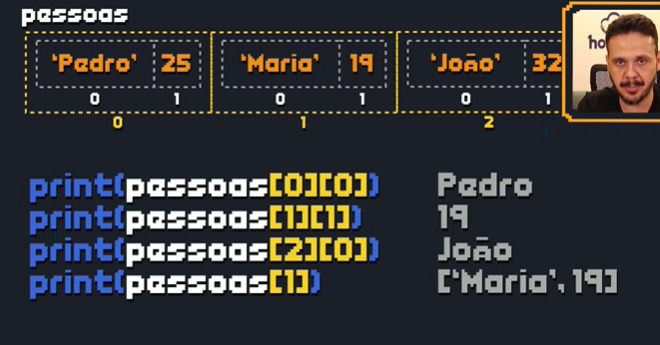


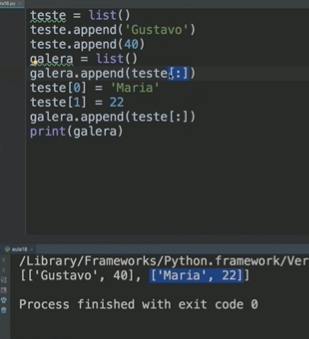
Assim estou copiando todos o elementos de A e fazendo o B

Também posso usar max, min e enumerate.

**Lista dentro da lista: (matrix)**

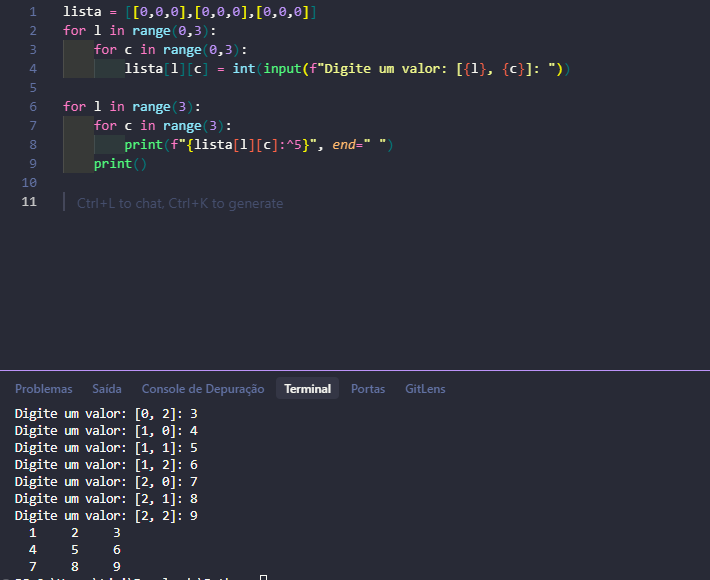








Criando matrix:



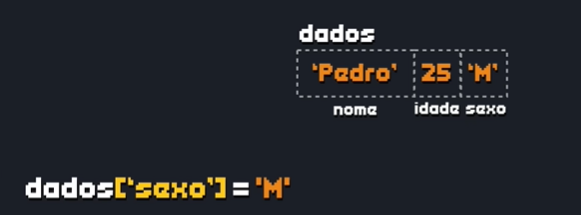
**Discionários:**

São feitos com “{}”

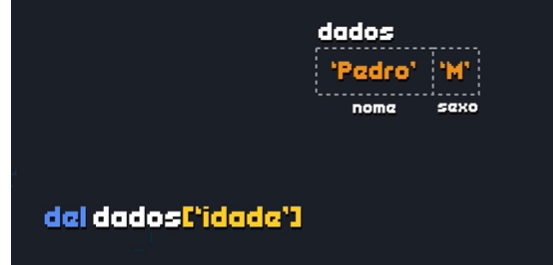
Semelhantes a lista só que colocamos os indices no nome que a gente deseja.

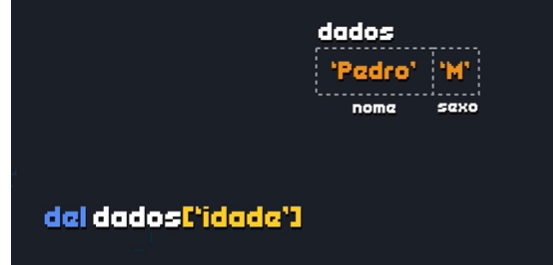


Para criar um elemento em discionarios não precisa fazer append. É só passar o dados. Ex:



**Remover** elemento eu utilizo del.





**Values x Keys x Items**







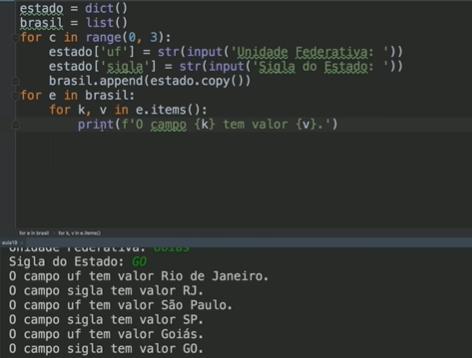
Não utiliza enumerate, ao inves:



Juntar todos o que a gente viu:

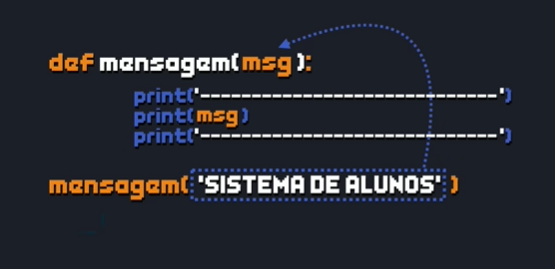


Fazer copia: Se eu não fizer uma copiar (.copy()) ele vai repetir tudo que inscrevi na lista, então é recomendado fazer uma copia!.

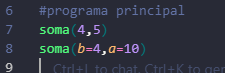


**Funções:**

Geralmente utiliza-se para uma rotina (algo que é utilizado varias vezes igualmente).



Posso discriminar os parametros:



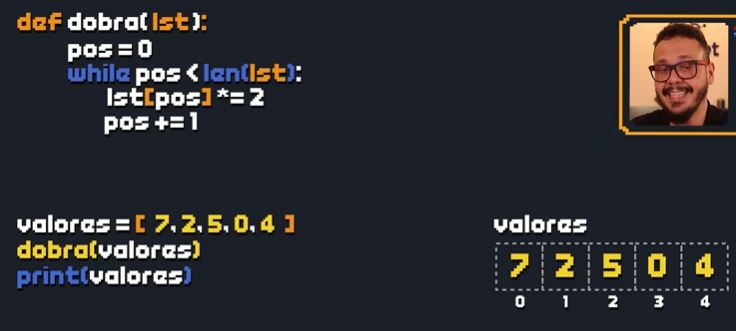
**Quando for varios parametros, posso colocar (\*):**



EX:



Posso criar uma lista com função:



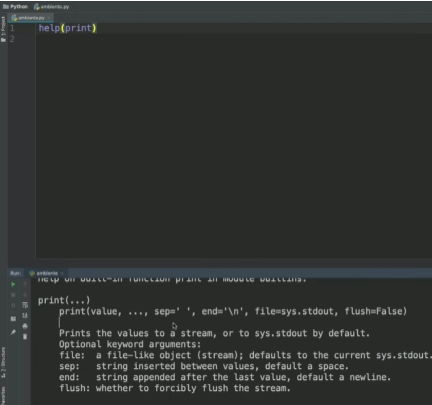


**Ajuda interativa: Interactive Help.**

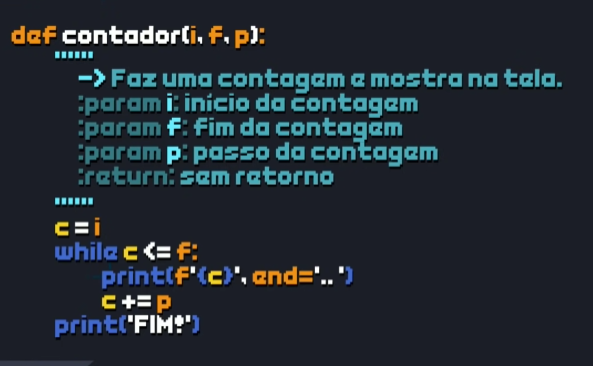
Basta apenas usar help().

É um manual das funções internas.

Para sair colocar só quit.

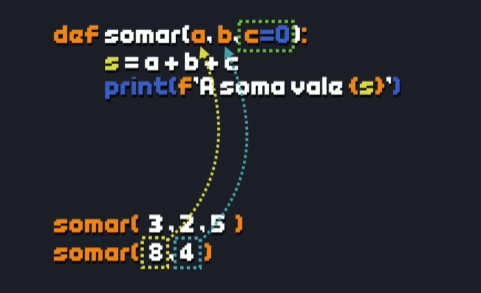


**Docstrings:**



É uma explicação para quem vai usar a função. Utiliza “””

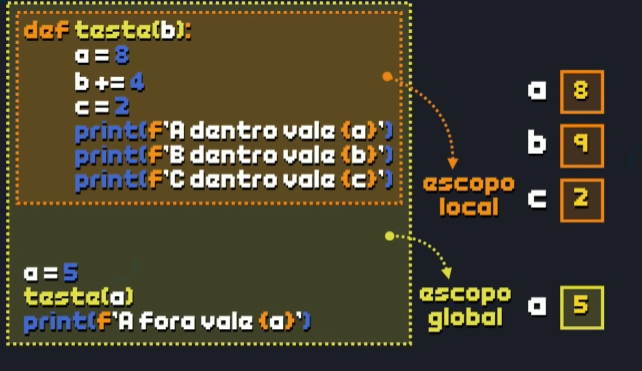
**Parâmetro Opcional:**



**Escopo das Variáveis:**



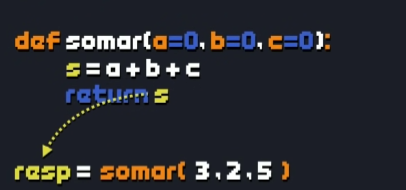
Cuidado se eu criar um A dentro da função, ele vai criar um outro “a” local

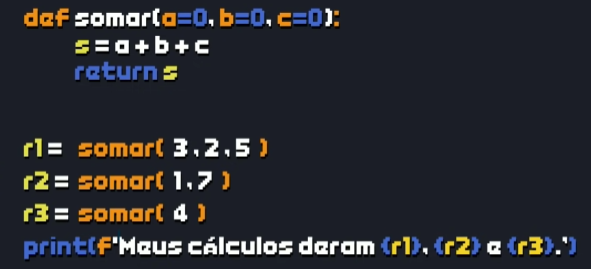


Mas tem como ser Global!



**Retorno de Valores:**

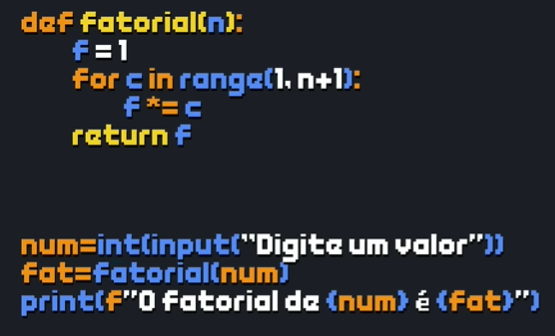




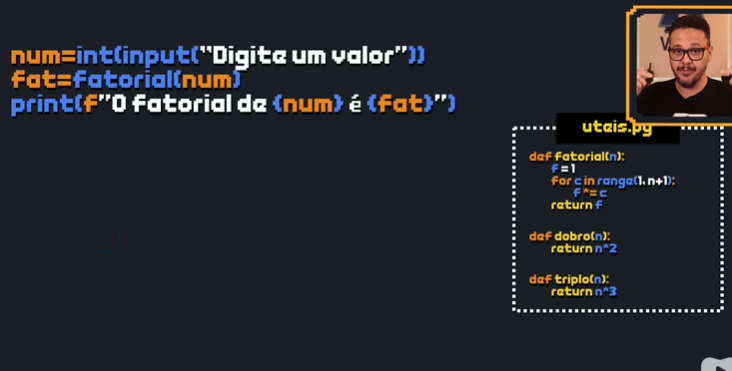
**Módulos e Pacotes:**

Dividir um programa em partes.

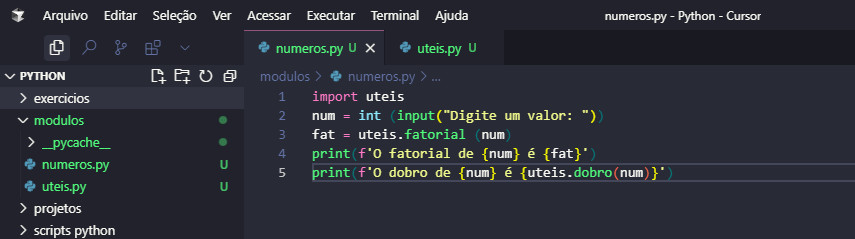
Aumentando a legibilidade, facilitando a manutenção.



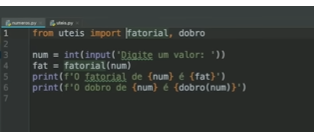
Criando o modulo:



Eu crio um arquivo na pasta e jogo as funções la. E importa, colocando “nome.função”

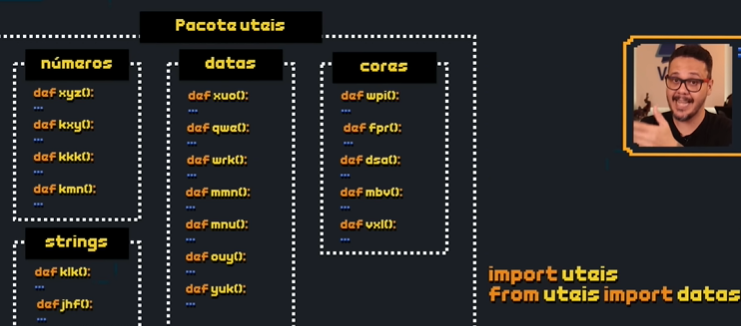


Também posso usar o import!

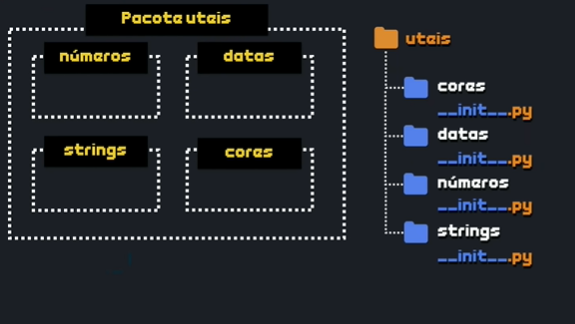


**Pacotes: Utiliza para quando se tem muito código!**

Junção de modulos!

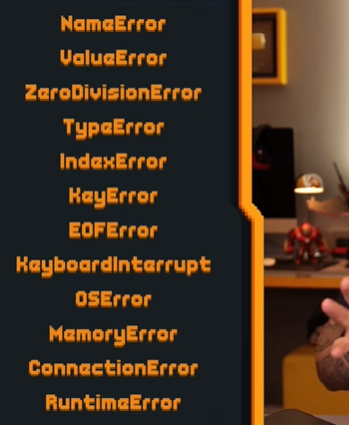


Para criar:Dentro de cada pasta precsa ter o \_\_init\_\_.py:

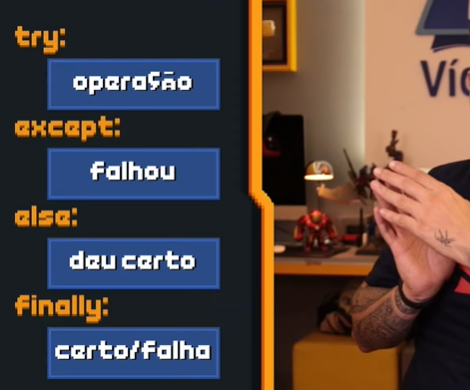


**Tratamento de Erros e Exceções:**

Exceções (mas tem muito mais) – erro sintatico.



Para tentar é utilizado o Try, Except.



\* Else e Finally é opcional

OBS: pode ter varios except:

