Python

Para alternar entre o terminal e o editor manualmente utilize o “ctrl + ‘ ” e o “ctrl + 1”

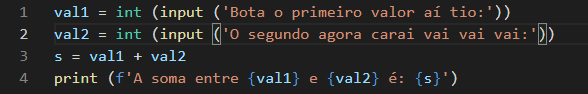
Para pegar um dado do usuário é possível utilizar uma variável com input:



O “,” é geralmente usado para separar quando há uma string na mensagem misturada com números, quando há apenas numeradores pode se usar o simples “+”

Lembrando: esse apóstrofo e mais não são recomendados para uso e sim o ‘f’ com chaves como no exemplo logo abaixo

Para fazer um cálculo com um input do usuário é melhor utilizar o f antes do apóstrofo e as variáveis em chaves, deixando assim o código mais otimizado e limpo do que colocar vários “,” e ‘



Tipos primitivos com exemplos de cada um:



Tipos de is que servem para detectar qual é o tipo que o usuário botou (string, float, maiúscula etc.)



Obs: uma palavra que começa com maiúscula em inglês é chamada de *capitalized* e para chamar um is que detecta isso você usa (variável).istitle

Tipos de operadores aritméticos:

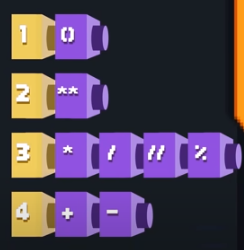


O “\*\*” é a potenciação, o “//” a divisão apenas mostrando inteiros e a “%” é o resto da divisão inteira

Também dá para fazer a potenciação através de uma função interna, porém nessa você perde a ordem de expressões



Ordem de expressões em Python:



Para fazer a raíz quadrada você só precisa exponenciar por ½



Para repetir uma string várias vezes é só colocar ela e a multiplicação pelo valor desejado na frente



Dentro das chaves você pode definir quantas casas decimais um número terá utilizando os dois pontos “:” e um ponto final com a quantidade de casas que você quer (ele conta também com os valores antes da vírgula)

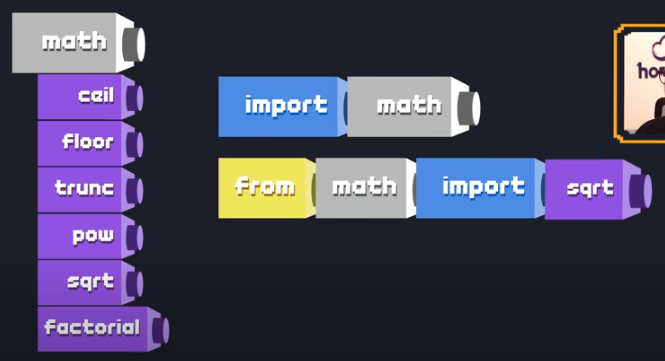


Bibliotecas

Para achar uma biblioteca e então adicioná-la em sua IDE, utilize o “Library reference” na parte de documentação no site do python, daí é só digitar o nome da biblioteca na IDE

Já para instalar um pacote com várias bibliotecas juntas, utilize o pypi.org

Como se importa uma biblioteca e como importar apenas certas funções dela:



Na prática: quando for usar uma função de um biblioteca coloque o nome dela com um ponto final antes de usá-la



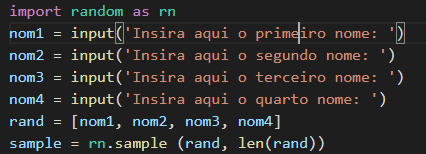
Se for usar apenas algumas funções já determinadas, não é necessário colocar o nome da biblioteca antes da função



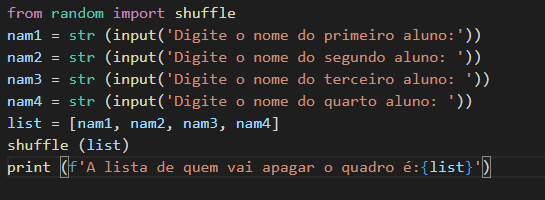
Para usar com o f antes do apóstrofo e {} em vez do formato acima, coloque a função dentro das chaves como por exemplo: {math.ceil(raiz)}')

A função pow é potência e a sqrt é raíz quadrada

Para mudar o nome da biblioteca para agilizar dentro do arquivo utilize “as” depois do import



Se for sortear uma lista com shuffle, não coloque ele dentro de uma variável porque se não irá retornar como “none”, use como uma função separa em ordem como no exemplo abaixo:



Manipulando strings

Fatiamento de strings

Para fatiar e selecionar uma parte da string você utiliza o número em colchetes, com o primeiro sendo onde a fatia começa a contar e o último sendo onde ela termina reduzindo uma casa no final (-1)



Se adicionar mais um número nesse colchete esse será a quantidade de casas que ele irá pular e contar, parando em cima da que foi correspondida



Se colocar apenas os dois pontos antes ou depois do número ele irá contar a partir do 0 ou a casa final se estiver na direita reduzindo uma casa



Funções de análise

**Len** (): serve para contar quantos caracteres tem na frase



Count (): mostra quantas vezes a letra apareceu na string



Find (): serve para mostrar onde está uma determinada palavra ou letra dentro da frase



In: essa diferente da find retorna para você se a palavra existe na frase ou não (true or false)



Funções de transformação

Replace (): como nome já diz, serve para alterar uma palavra dentro da string



Upper() e lower(): servem para deixar todas as letras em maiúsculas e minúsculas, respectivamente



Capitalize (): coloca uma maiúscula no início da string e deixa todo o resto em minúsculas (como se fosse o início de um texto após o pronto final)



Title (): funciona de uma forma parecida com o capitalize, porém nessa ele deixa em maiúscula todas as letras após um espaço



**Strip** (): uma importante função que é usada para ignorar espaços tanto no início quanto no final da frase



Da mesma forma dá para colocar um l (left) ou r (right) antes da strip para remover apenas os espaços da esquerda ou da direita

Lembrando, essa regra do l e r serve para várias funções ao decorrer do curso



Funções de divisão e junção

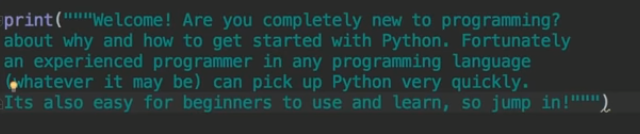
Split (): esse serve para separar em blocos as palavras de acordo com o espaço dado entre elas (é recomendado dar uma aprofundada mais para frente sobre para ver as possibilidades)



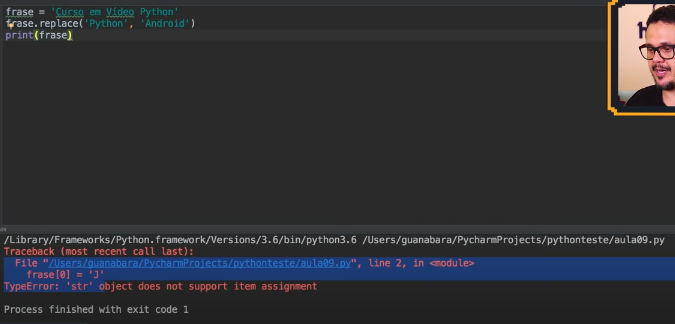
O join pelo contrário serve para juntar as palavras com um caractere que você quiser antes de chamá-lo, se quiser um espaço nessas partes você só precisa deixar uma área vazia entre apóstrofos: ‘ ’

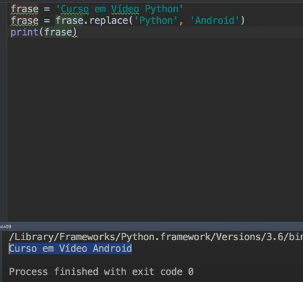


Dica aleatória: para fazer uma frase aparecer organizada embaixo da outra você pode usar aspas triplas no começo e início da frase, assim aparecerá organizado para o usuário em vez de ter que criar vários “print”



Lembrando: se for usar alguma dessas funções com uma variável, declare-a antes como variável da mesma forma, caso contrário ele apresentará um erro pois ou não vai entender o que precisa reproduzir ou irá reproduzir o que foi colocada primeiramente como variável



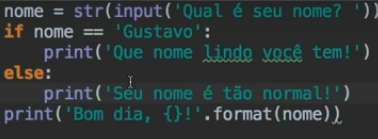


Condicionais

Indentação = nome do espaço inicial usando o TAB, obrigatório para executar o comando no Python

As condições simples são aquelas que possuem apenas o IF, já as compostas possuem também o ELSE no código

Exemplo de código usando condicionais:



Mudando as cores no terminal (**não funciona nativamente antes do Windows 10**)

Para mudar as cores do terminal você deve executar sempre o comando \033[...m e seguir os valores respectivos de cores da tabela abaixo:



Obs: O “:” da imagem na verdade é o ponto e vírgula “;”

Os valores em amarelo variam, enquanto o restante é obrigatório na hora de definir cores

Se você não quiser alterar algum valor que já está padrão você pode simplesmente pula-lo e colocar apenas o “;” logo de cara

Exemplos utilizando esses estilizadores:



Para pegar o ano/mês/dia atual do computador use o comando “date.today().year”

Obs: é necessário importar a biblioteca “datetime” e a função é “date”



Para converter qualquer valor para binário, octal ou hexadecimal, utilize os comandos abaixo dentro dos colchetes no format

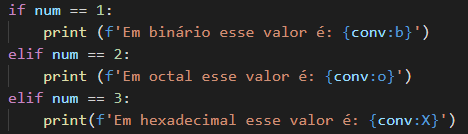
:b = binário

:o = octal

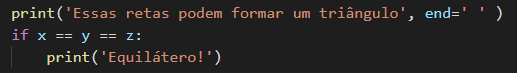
:x = hexadecimal com letras minúsculas

:X = hexadecimal com letras maiúsculas

Exemplo utilizando esses comandos (mais no exercício 37):

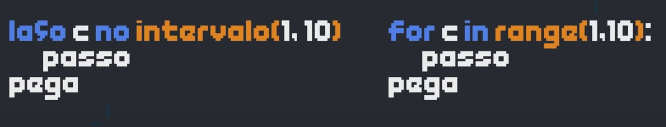


Quando quiser que uma frase/palavra apareça logo em seguida de outra em ser uma variável entre colchetes, utilize “end= ’ ‘)



Estruturas de repetição

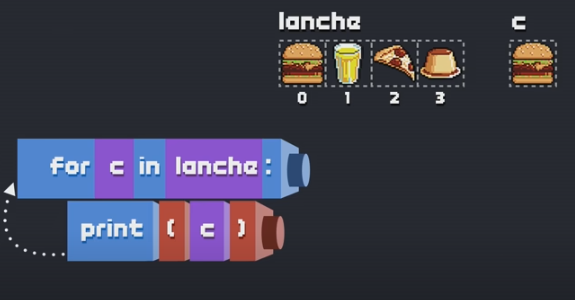
For

Como é um código de estrutura de repetição FOR 

Os números em parênteses dizem respectivamente onde ele começa e onde termina a repetição

Se houver um terceiro número separado por vírgula nos parênteses ele significa quantas casas estão sendo puladas por etapa

Como o loop for funciona teoricamente:



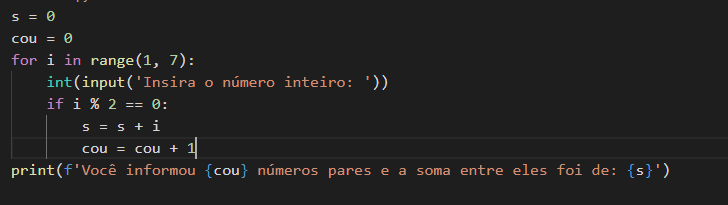
Exemplo com condicionais:



Se quiser que essa repetição seja regressiva, utilize “-1” nos passos (terceiro número)



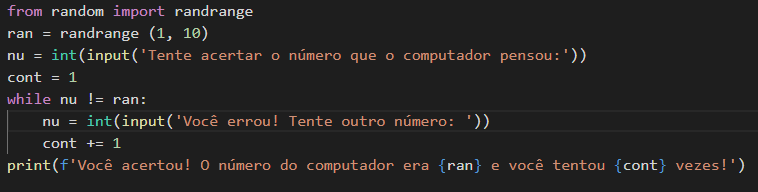
Se você quiser somar todos os números que tem em uma repetição for, coloque uma variável antes do loop dizendo que ela começa do zero, assim o loop vai pegar o valor 0 de início e somar 1, depois pegar esse valor anterior e somar com o próximo para formar 2 e assim sucessivamente até acabar o loop



Para contar a quantidade de vezes, você segue a mesma lógica, porém, acrescentando “+ 1” em vez da variável do loop, já que ele contará todos os valores presentes ignorando o 0

Na prática, você coloca o contador += 1 dentro do loop e depois o coloca nos colchetes na frase final já que ele também é considerado uma variável como os dois que você utilizou ao longo do loop

Exemplo desses contadores só que com o while loop:

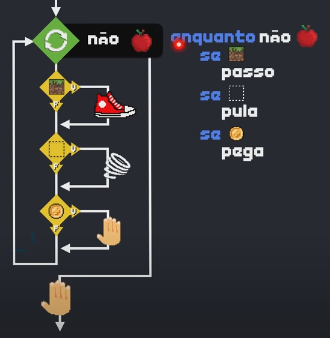


While

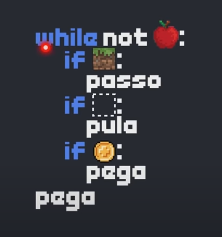
A principal diferença da estrutura while para a for é que a while pode ser usada mesmo se não souber um limite de contagem, enquanto no for esse limite obrigatoriamente precisa ser definido

Além disso, ele utiliza a lógica booleana (true e false) para determinar se a ação principal do loop deve ser executada ou não

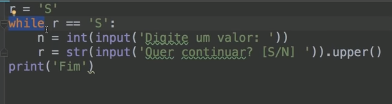
Exemplo lúdico de uso do while loop:



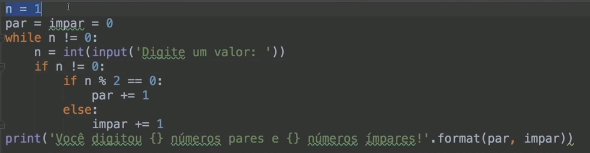
Obs: nos losangos amarelos é demonstrado o caminho verdadeiro e falso, se for falso ele continua, se for verdadeiro ele para e executa a ação



Exemplo de while no código:



Outro exemplo mais complexo utilizando o while e contando quantos números pares e ímpares foram digitados:



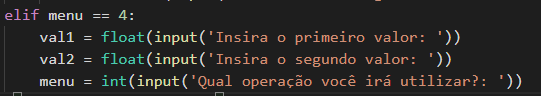
Se quiser validar dados com while utilize “not in” acompanhado das letra/números que você quer sem vírgula mesmo



Dica: para parar a execução de um código no VS Code você utiliza

Para contar a quantidade de vezes que foi repetido você precisa criar um contador antes do loop e depois colocar um cont += 1 (cont = cont + 1) dentro dele

Em um loop se for necessário você pode repetir as variáveis que ficaram fora dele, atribuindo novos valores a elas automaticamente mesmo dentro do loop



Nos loops você pode também ir substituindo e avançando as “casinhas” dos termos atribuindo um variável dentro de outra variável (t1=t2, t2=t3 no exemplo abaixo)



Exemplo lúdico de como isso ocorre na prática:

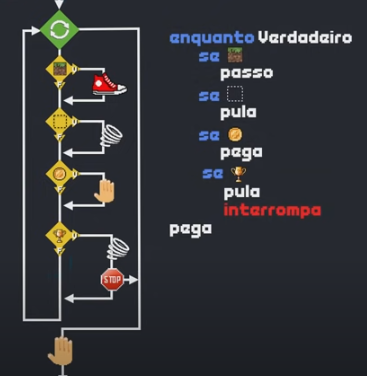


Se as variáveis dentro de um código tiverem todas o mesmo valor, você pode colocar exemplo1 = exemplo2 = exemplo3 = exemplo4 = 0 para economizar linhas e ter a mesma efetividade

Interrompendo repetições while

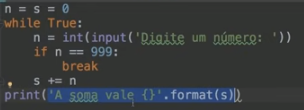
Para interromper um repetição dessa você utiliza o while true no começo e quando quiser que o programa encerre e volte para fora do loop você coloca break no meio do código

Exemplos lúdicos:

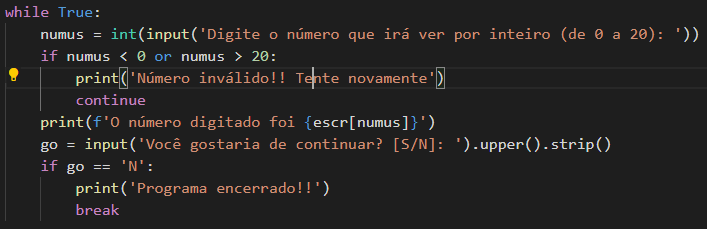




Você também pode usar essa interrupção nos casos de uma flag (comando que quando digitado interrompe o programa e por vezes não é contabilizado em uma eventual operação)



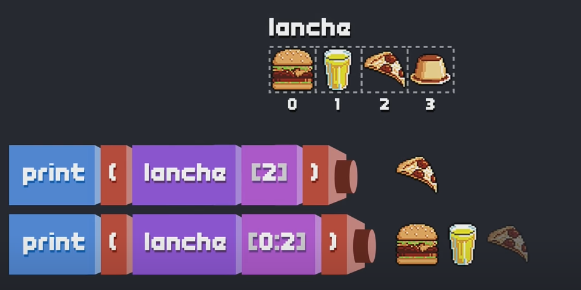
Dica: se quiser que uma parte volte ao prosseguimento do loop, utilize o “continue” sozinho no código, assim ele continuará o loop como já estava



Arrays

Tuplas

As tuplas são basicamente variáveis compostas que permitem organizar um conjunto de informações



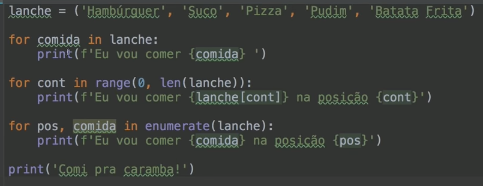
Obs: messe tipo você utiliza muito a manipulação de strings para organizar os dados

Exemplo utilizando essa manipulação:

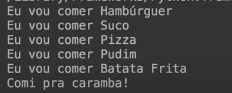


**Regra das tuplas**: os elementos dentro de uma tupla **nunca** conseguem ser alterados enquanto o programa estiver rodando

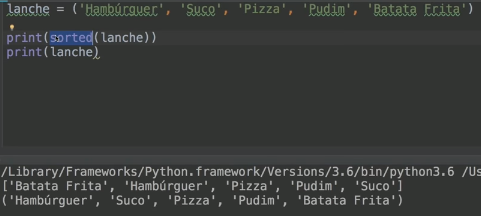
Formas de fazer um loop utilizando for para tuplas:



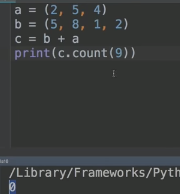
Obs: o enumerate te ajuda a mostrar automaticamente a respectiva posição do elemento a partir de dois critérios, com a primeira palavra (pos no exemplo acima mas pode ser qualquer uma) indicando a posição e a segunda (comida no exemplo mas pode ser qualquer outra) mostrando qual foi o elemento, com o enumerate essas duas já estão interligadas e basta apenas printa-las na tela



Função para organizar a tupla: “sorted”

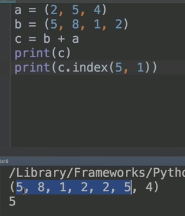


Função para contar quantos daquele elemento há na tupla (count):



Obs: essa forma de somar tuplas não segue a lógica matemática onde a ordem não importa, nesse caso, ele mostra os números na ordem que você colocou a soma

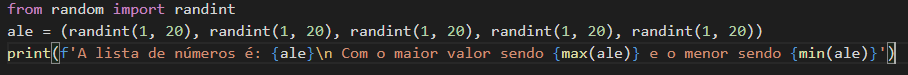
Função para encontrar em que posição está certo elemento (index):



Você também pode excluir uma variável composta utilizando a função “delete” (que serve para excluir qualquer parte no python também) se for necessário após realizar um ordenamento por exemplo



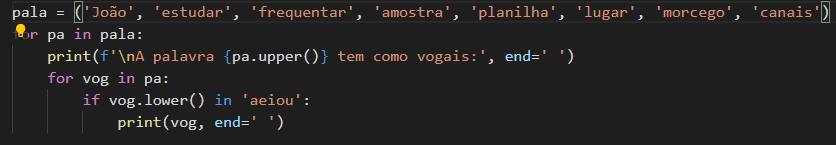
Para encontrar o maior ou o menor valor de um tupla (e em outras funcionalidades do python), simplesmente use max() e min()



Para deixar a quantidade de zeros após a vírgula que você deseja, utilize ‘.2f’ com o número mudando de acordo com a quantidade de zeros



Já para encontrar uma string dentro de uma tupla você pode utilizar “in” dentro de um loop como no exemplo abaixo (feature importante):



Listas

Diferentemente das tuplas, as listas podem ser alteradas durante a execução do programa (mutáveis)

Elas também são utilizadas através de colchetes “[ ]” em vez dos parênteses das tuplas ou através do comando “list()”

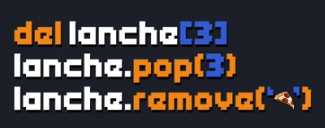
Para criar uma casa a mais com o elemento nas listas utlize a função “append”



Já para colocar no meio de algum intervalo você utiliza “insert”, indicando em qual casa o elemento irá ficar no primeiro elemento antes da vírgula



Para deletar algum elemento da lista você pode fazer de três jeitos:



Obs: o pop geralmente é usado apenas para apagar o último elemento, ficando com os parênteses vazios

Obs2: o .remove() por padrão remove o primeiro elemento indicado na lista, podendo remover todos através dos laços de repetição caso seja necessário

Além disso, é recomendável utilizar uma condição antes de executar com remove para verificar se o elemento está na lista, caso contrário se você não conferir e ele não estiver o programa dará erro

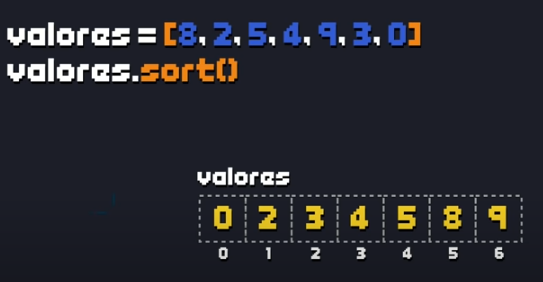


É possível também criar uma variável composta de lista através do range, a principal vantagem dele é que a lista já vem em ordem numérica

Lembrando: o último valor do range sempre exclui o último número, no caso abaixo seria exibido de 4 a 10

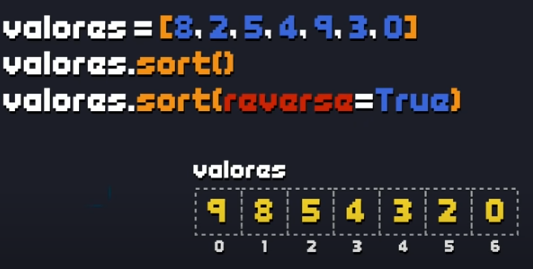


Se precisar organizar uma lista já criada você pode usar o comando sort,



Podendo inverter a ordem de trás para frente utilizando o comando “reverse=True” caso seja necessário como no exemplo abaixo:

Lembrando: a função true no python é sempre escrita com letra maiúscula no começo



Você também pode colocar o len para ver a quantidade de casas contidas na lista, similar as strings



Detalhe das listas em python:

Se você usar o sinal de igualdade entre duas listas você estará ligando ambas, ou seja, o que for alterado em uma será automaticamente alterado em outra



Para fazer essa cópia você utiliza a opção de fatiamento “ : ”, assim a segunda variável irá receber apenas a cópia da lista, não afetando a primeira



Dica para sumir com os colchetes no output:



Como funciona a estrutura de uma lista dentro de uma lista

Composição básica de uma lista:



Basicamente ele pega primeiramente o valor do blocão selecionado e o 2° valor que você colocou será oq vai encontrar dentro do bloco e printar

Se você não colocar nada como opção (ex da última linha) ele printará tudo que está naquele bloco

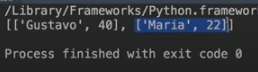


É nessa etapa de lista dentro de lista que a dica do “[:]” fica tão importante, já que ele facilita para limpar os dados e não juntar/sobrepor todos

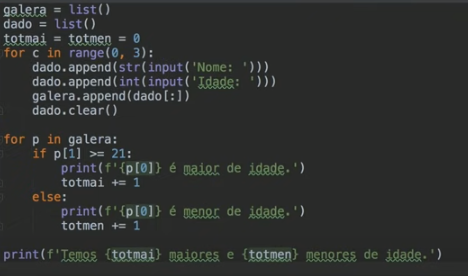
Exemplo utilizando o [:]

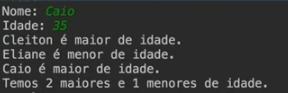


Como dá pra ver, o “[:]” limpou os dados e manteve o anterior guardado em outro lugar na memória sem sobrepor nem um nem o outro, resultando no seguinte output:



Exemplo utilizando os dois conceitos anteriores:





No caso acima o p[ ] selecionou a casa do bloco que queria mostrar(nome), e como os dados foram adicionados através do append ele foi sendo sequencial, facilitando pro loop acertar a localização desses dados em 0, 1 e 2

Inclusive esse esquema de “0, 1, 2, 3...” para encontrar algum dado poderá ser substituído por algum nome utilizando os dicionários, função que será melhor detalhada na próxima aula

Dica: sempre que quiser limpar os dados de uma lista, utilize .clear() depois do nome dela



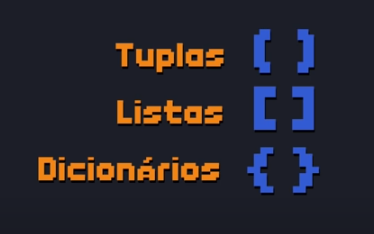
Detalhe das listas: o atalho de organizar os contadores em uma mesma linha (ex:3° da imagem abaixo) só pode ocorrer com variáveis simples (de apenas 1 elemento), as variáveis compostas como listas não podem ter esse atalho se não não funcionarão



Dicionários

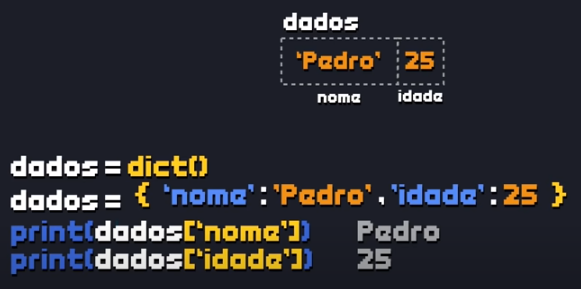
Os dicionários servem para definir nomes em cada uma daquelas seções compostas por números na listas, facilitando em vez de ter que ficar correspondendo com 0, 1, 2 etc

Diferença de como declarar cada tipo de array

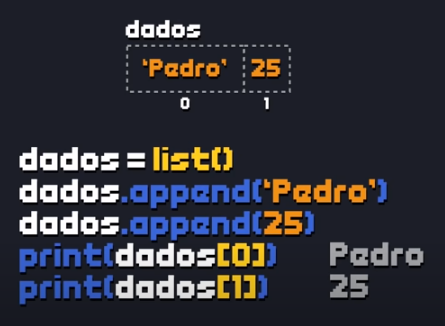


Para declarar um dicionário você também pode usar o dict() assim como pode usar o list() nas listas

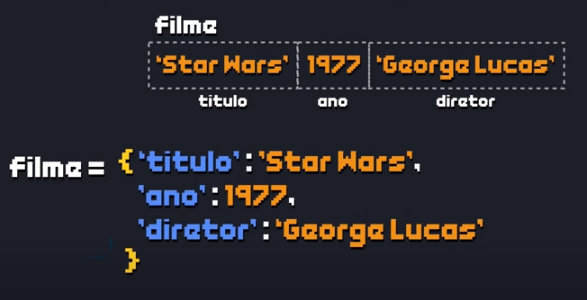
Sintaxe básica para declarar cada seção utilizando o dicionário:



Estrutura lá da lista apenas para comparar:



Exemplo básico utilizando filmes:



Exemplo prático de como utilizar os dicionários quando houver um format f:



Lembre-se: sempre que for referenciar o dicionário você utiliza os colchetes [ ] e quando for declarar você utiliza as chaves { }

Formas de selecionar os valores em um dicionário:

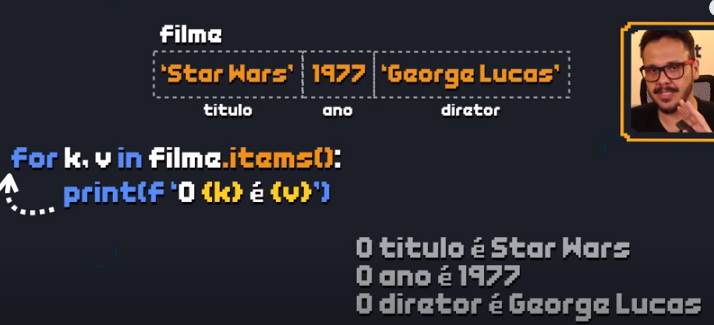


O .values() declara apenas a parte de cima (Star wars, 1977 e George Lucas no exemplo acima)

Já o .keys() declara apenas as seções (título, ano e diretor)

Enquanto o .items() declara todos juntos

Exemplo utilizando-o junto com laços de repetição:



Dica: Para fazer o fatiamento [ : ] em um dicionário você utiliza o comando .copy() (bem mais intuitivo já que o objetivo é realmente copiar a lista de itens)

