Curso em Vídeo – Python (com Gustavo Guanabara)

**Aula 4-**

‘olá mundo’ 🡪 as aspas simbolizam texto, que normalmente vem entre aspas

🡪(‘olá mundo’)

**Exemplo**: print(‘olá, mundo’)

**Computador responde 🡪** olá, mundo



No Python, números não precisam de aspas, caso usadas, serão reconhecidos como string

**Soma de números**



**Soma de números como strings**



Para mostrar uma mensagem, e em seguida um numero pode ser usado tanto o +, como a virgula



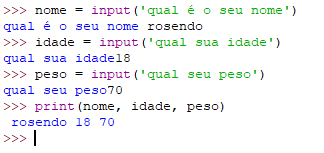
Toda variável é considerado um objeto no Python. Para adicionar valor a elas utilizamos o sinal de igual(=)



E para mostrar é semelhante ao que já vimos



Para gerar uma interação com o usuário podemos pedir para que ele insira os dados, dessa forma:



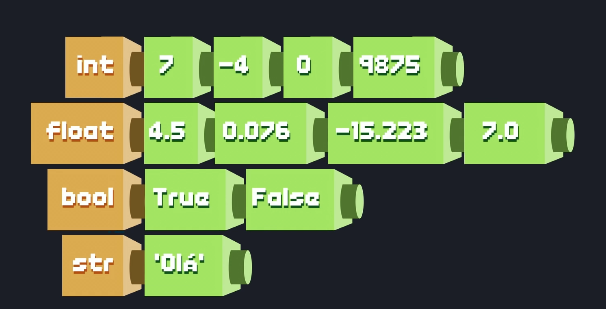
**Obs: Toda função é colocada entre parênteses**

Para somar variáveis(objetos) é necessário declara-las como números(int), pois quando apenas declaramos, **exemplo: idade = input(‘qual sua idade’)**, esse valor da idade será reconhecido como string

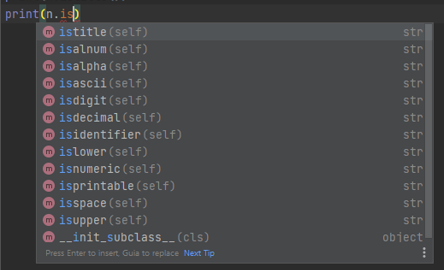
Uma opção para se usar no lugar da virgula para por uma variável no texto é a função de formatação, que funciona da seguinte forma:



No Python existem alguns tipos primitivos, que são comumente utilizados:



No Python existe uma função para descobrir o algum numero é(Mas como assim)



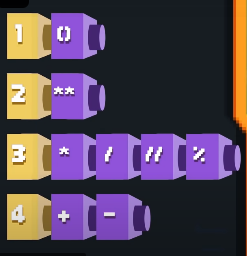
Usando o medo .is é possível descobrir

Indo um pouco além, agora vamos ver os operadores aritméticos disponíveis para uso no Python:



O sinal de igual é tratado como == (dois sinais de igual seguidos)

**Ordem de precedência**



Outra forma de utilizar a potencia (\*\*) é usando o comando pow(n,m), exemplo



Já para calcular a raiz quadrada de um numero é necessário eleva-lo a meio



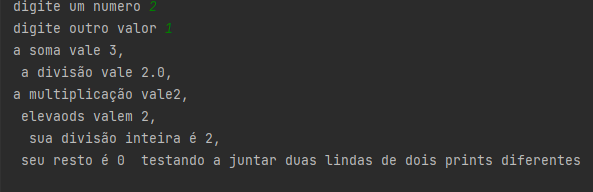
Para juntar duas linhas de print basta colocar end=” ” ao final do código





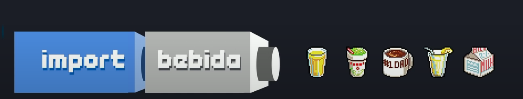
Já para quebrar um linha de texto basta por \n





Estamos chegando agora na parte em que surgem os módulos, que são ferramentas com a funcionalidade de expandir as possibilidades do Python

Para importar totalmente uma biblioteca é necessário usar o comando:



Já para importar apenas a ferramenta necessária utilizamos o **From**



**Manipulação de cadeia de textos**

No python é possível manipular o texto de forma a alterar detalhes, exemplo:



Esse comando faz com que apenas a letra 13 da frase “curso em vídeo Python” seja escrita



Outra forma de manupular os texto é no formato a seguir:



Onde são selecionados os caixas do 9 ao 21, mas pulando de dois em dois

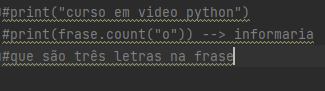


Quando não vir for utilizado um numero antes dos dois pontos, significa que a manipulação da String irá começar do inicio ( 0 )

No Python uma das funcionalidades mais utilizadas é a de analise de Strings, para isso, utilizamos a função len(), que informa quantos microespaços foram utilizados



Existe também a função count(), que informa quantos elementos do qual você perguntou tem numa frase, exemplo:



A mesma funcionalidade pode funcionar já com fatoramento



Que seria a contagem de quantas letras existem do 0 ao 13(mas não incluindo o 13) existem

No Python existem as chamadas **condições aninhadas: ELIF** que seria uma abreviação do **else if( senão se)**



No **Python** existem **3 tipos** de variáveis compostas, **Tuplas, Listas e os Dicionarios**



**São três as formas de declarar variáveis**

**Parênteses () para tuplas**

**Colchetes [] para listas**

**Chaves {} para dicionários**

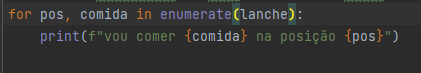
Existem três formas de usar o **for** juntamente da tupla:



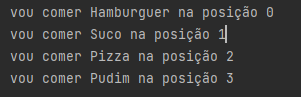




O comando **enumerate** é uma função que serve como contador que indica qual o índice o devido elemento da tupla representa**,** EX:



**Resultado:**



No **python** é possível, através de um comando, organizar as tuplas de acordo com a ordem alfabética utilizando o comando **sorted**



No **python** tembém é possível utilizar um comando semelhante ao **find(),** que é o **.index(),** que indica a primeira posição do elemento solicitado.

As **tuplas** são consideradas imutáveis, mas é possível deleta-las com comando **del**



Agora sobre listas. Elas são colocadas entre colchetes ‘[ ]’

Para criar um novo espaço na lista basta usar o comando **append**





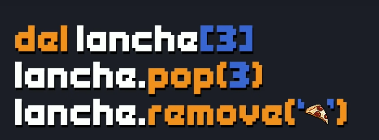


Também é possível escolher a posição de um novo item:





Agora sobre remover itens de uma lista. Existem 3 formas básicas de remover os itens de uma lista, duas delas são pelo índice e uma pelo valor dentro do índice







**Obs: O método pop é mais utilizado para remover o ultimo item da lista, ex: lanche.pop() remove o ultimo item da lista, mas o método ainda permite apagar outros índices quando indicado**

É possível ainda utilizar um método utilizado antes, o **range()**, ex:

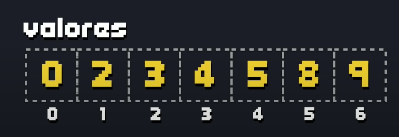




Que cria uma lista com os valores de 4 ate 10

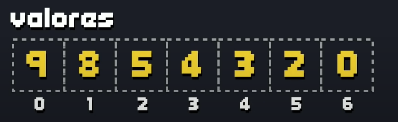
E assim como o método **sorted**, o método sort() organiza os itens de uma lista



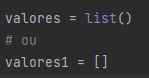


Já se o desejo for ordenar ao inverso, basta adicionar o parâmetro **reverse=True**





Existem duas formas de iniciar uma lista vazia:



 Nesse comando o Python cria uma ligação entre as listas, e não uma copia

Já se o seu objetivo é copiar uma lista dentro de outra o comando a ser utilizado é, Ex;

 Nesse momento a lista **b** recebeu uma copia da lista **a** sem que houvesse nenhuma ligação entre elas

Agora, finalmente, iremos aprender dicionários no Python. Sua principal diferença quando comparado as listas é que podemos defini-las com um valor literário. Ou seja, ao invés de se usar números como índice das listas, no dicionário podemos atribuir valores com nomes.

Para iniciar um dicionário vazio podem ser utilizadas duas formas:



A visualização do dicionário fica da seguinte forma:







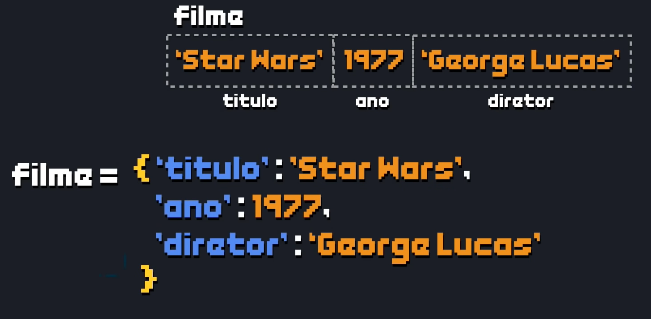
Diferente das listas que era necessário o comando append para criar novos itens, no dicionário basta utilizar o ‘recebe’ =



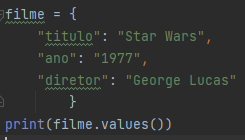


Já para deletar elementos é da mesma forma, del dados[“idade”]

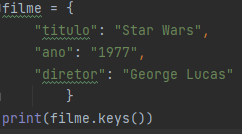
 



Se você deseja pegar apenas os valores adicionados ao dicionários, deve usar o método values()

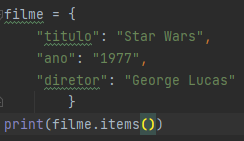


Agora se você deseja visualizar apenas as chaves utilizadas, deve usar o método keys()





Agora se você deseja pegar tanto as chaves utilizadas, mas também os valores, deve utilizar o método items()



**Diferente das listas, onde para fazer a copia era necessário indicar com o uso de [:], quando vamos copiar um dicionário basta utilizar um método interno, o copy()**

**Organizar Dicionários**

Para organizar dicionários é necessário importar um método da biblioteca **‘operator’,** o método itemgetter é responsável por solicitar a partir de qual elemento o dicionário será organizado:

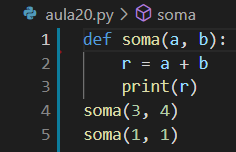


Ou seja, no exemplo, se deveria ser organizado pela ordem dos **jogadores**, ou pelos **valores inteiros** ligados a eles.



Com essa linha de código organizamos o dicionário **jogo** a partir da ordem dos valores inteiros(key=itemgetter(1)), e de forma decrescente(reverse=True)

**Finalmente** Chegamos ao conteúdo referente a funções no Python, Algo usado para diminuir as atividades repetitivas na programação. Vamos utilizar de exemplo uma função soma(), que soma dois valores a partir de dois parâmetros:



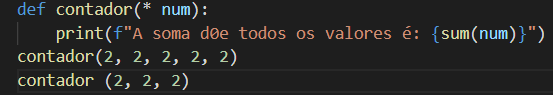
Soma(**a, b**), **a** e **b** foram os parâmetros passados que devem ser utilizados ao programar, se não o resultado será um erro

Uma forma de definir parâmetros de maneira mais clara é quando vamos dar seu valor

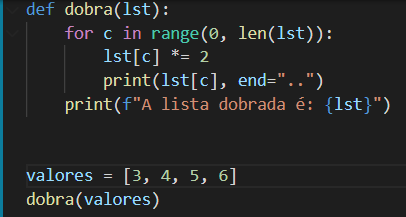


Dessa forma, fica mais fácil o entendimento do que está acontecendo no código

Uma coisa possível no Python que não acontece em outras linguagens de programação é a possibilidade de passar quantos parâmetros forem necessários, diferente de outras linguagens como java



Utilizando **listas** para dobrar o valor de cada espaço:





Manipulação de arquivos

