**Aula 6: Tipos primitivos e saída de dados**

**type(variável)** >>> mostra qual o tipo primitivo da variável. Quando não se faz a conversão sempre vai mostrar como str.

**variável.isspace()** >>>>> Mostra se a variável é só tem espaço

**variável.isnumeric()** >>> Se é um número.

**variável.isalpha()** >>> Se é alfabético

**variável.isalnum()** >>> Se é alfanumérico

**variável.isupper()** >>> Se está toda em maiúscula

**variável.islower()** >>> Se está toda em minúscula

**variável.istitle()** >>> Se começa em maiúscula

**Métodos =** quando tiver parênteses no final é um método.

**Aula 7: Operadores aritmétricos**

+ = soma

- = subtração

\* = multiplicação

/ = divisão

\*\* = potência

// = divisão inteira

% = resto da divisão

**Ordem de Precedência:**

1° - ()

2° - \*\*

3° - \*, /, //, %

4° - +, -

**Aula 08: Utilizando Módulos**

import biblioteca001 >>>>> vai importar todos os elementos da biblioteca001 (toda a biblioteca)

from biblioteca001 import x >>>>> vai importar o elemento x da biblioteca001 (uma parte da biblioteca)

**Aula 09: Manipulando texto**

**Fatiamento:**

Quando se atribui uma string a uma variável o Python conta quantos espaços existe nessa string, começando do zero. Ex:

frase = 'Curso em Video Python'

print(frase[4]) >>>>>> O programa vai printar a quarta letra, no caso : 'o'

print(frase[:13] >>>>>>> Vai printar as letras desde do 0 até a 12, ele para de contar no 13

print(frase[1:15:2] >>>>> Vai printar as letras do 1 até o 14, pulando de dois em dois

print(frase[1::2] >>>>>> Vai do 1 até o final pulando de dois em dois

print(frase[::2] >>>>> Vai printar a frase toda pulando de dois em dois

print(""" """) >>> Usar trés aspas duplas permite que você print um texto maoir com quebras de linhas...

print(frase.count('o') >>>> Vai contar quantas vezes tem a letra 'o' na string e printar o número.Obs: uma letra maiúscula tem diferença da mínuscula.Ex:

print(frase.upper().count('O') >>>>> O comando upper() coloca toda a frase em maiúsculo, logo, nesse caso, iria aprecer 3.

print(len(frase)) >>>> O comando 'len()' permite ver qual é o tamanho da frase

print(len(frase.strip()) >>>> O comando '.strip()' tira espaços sobrando no início e no final da frase.

print(frase.replace('Python', 'Android') >>>> O comando '.replace()' substituí, só na instância que está,uma palavara por outra. Mas se colocar:

frase = frase.replace('Python', 'Android') aí trocaria permanentemente

print('Curso' in frase) >>> O comando 'in' vai mostrar se a palavra 'Curso' está na frase

print(frase.find('Curso') >>>> Mostra se uma palavra existe e quando ela começa. (Quando não existe mostra -1). Pode coloar r, .rfind, para começar da direita para esquerda

frase.lower() >>>> deixa toda a frase minúscula

frase.split() >>>> Divide todas as palavras e armazena em forma de lista

d = frase.split()  
print(d[0][3])>>> no primeiro colchete, você pode colocar qual o número da palavra que você quer, que começa no zero, no segundo, qual letra

**Aula 11 : Cores no terminal**



print(\033[0,33,44mOlá mundo!\033[m) >>>>> Jeito mais básico de colocar cor.

==============================================================================

a = 3  
b = 5  
print('Os valores são \033[36;47m{}\033[m e \033[32;45m{}\033[m'.format(a, b)) >>>> mais complicado

==============================================================================  
nome = 'Lucas'  
print('Prazer em te conhecer {}{}{}'.format('\033[4;31m', nome, '\033[m')) >>>>> mais organizado

==============================================================================nome = 'Lucas'  
cores = {'limpa' : '\033[m',  
 'cinza': '\033[34m',  
 'rosa' : '\033[35m',  
 'pretoebranco' : '\033[7;30m'}  
print('Prazer em te conhecer {}{}{}'.format(cores['pretoebranco'], nome, cores['limpa']))  
>>>> Jeito mais complexo e organizado

**Aula 12: Condições aninhadas**

# O comando "elif" permite que você coloque várias condições dentro de um if, usando o elif repetidas vezes

**Aula 15: Interronpendo repetições while**

while True: >>>>>> repetição infinitas

break >>>>> interromper laços de repetições

f string >>>>> atualizção do python que n precisa mais usar o '.format', mas coloca um 'f' antes da string e coloca a variável dentro do colchete. Todas as formatações funcionam normalmente.

**Aula 16: Tuplas**

Tuplas são variáveis compostas, ou seja, contém mais de um valor e são IMUTÁVEIS, ou seja, não podem ser mudadas durante a execussão do programa. Tuplas são representadas por colchetes () e com vírgulas entre os valores e esses são numerados a partir do 0. Segue exemplos e algumas funcionalidades:

lanche = ('Hambúrguer', 'Suco', 'Pizza', 'Pudim', 'Batata frita')

print(lanche[1]) >>>> Os parênteses retos permite printar a variável que deseja, nesse caso vai ser 'Suco'

print(lanche[-2]) >> Vai começar a contar do último começando do -1. Nesse caso, vai printar 'Pudim'

print(lanche[1:3]) >> Vai mostra do 1 até o 3 ignorando o último. Nesse caso, ('Suco', 'Pizza')

for comida in lanche:

print(comida) >>>> Vai mostrar cada item da tupla a cada linha

for cont in range(0, len(lanche)):

print(cont) >>>Vai a posição dos itens na tupla.

for cont in range(0, len(lanche)):

print(lanche(cont)) >>> Vai mostrar os itens da tupla.

for pos, comida in enumerate(lanche):

print(f'Eu vou comer {comida} na posição {pos}) >>> Vai mostrar o iten e a posição

print(sorted(lanche)) >>>> Vai mostrar em ordem a tupla.

a = (2, 5, 4)  
b = (5, 8, 1, 2)  
c = a + b   
print(c) c))>> Vai juntar as tuplas a e b. Nesse caso a ordem em que foram somadas importa.

a = (2, 5, 4)  
b = (5, 8, 1, 2)  
c = a + b   
print(c.count(5)) >>> .count conta quantas vezes aparece um item na tupla. Nesse caso 2.

a = (2, 5, 4)  
b = (5, 8, 1, 2)  
c = a + b   
print(c.index(5)) >>>> .index() vai mostrar em qual posição um determinado item está, ele pega a primeira posição que achar. Nesse caso vai ser a pósição 1.

a = (2, 5, 4)  
b = (5, 8, 1, 2)  
c = a + b   
print(c.index(5, 2)) >>>>> Quando coloca uma vírgula e um número, o index vai mostrar qual é o número a partir do segundo que você colocou. Nesse caso vai ser na posição 3.

--Quando for usar o .index() para str e estiver dentro de aspas, usar aspas duplas ou .format(), porque o python não suporta aspas simples dentro de aspas simples.

-- Tuplas aceitam dados de tipos de diferentes, como letras e números.

pessoa = ('lucas', 17, 1.83, 'M')  
*del*(pessoa)  
print(pessoa) >>>>del() pode excluir uma tupla e é a única forma de 'modifica-lá' enquanto o programa está em execução. Se rodar o programa acima vai dar erro dizendo que 'pessoa is not defined'. Além disso não é possível deletar um único item.

print(f'\nO maior foi {max(numeros)} e o menor foi {min(numeros)}.') >>>> max() pega o maior valor e mim() pega o menor.

-- Dois pontos ':' permite editar um print. Exemplo ':^30 'vai mostrar o print centralizado em 30 caracteres, ':<30' vai mostrar o print com trinta caraceteres do lado direito

**Aula 17: Listas**

Listas são variávies que guardam vários valores. A principal diferença de uma tupla é que a lista é MUTÁVEL.

variável[2] = 3 >>> A variável que está na posição 2 passa a valer 3

.append() >>>>> adiciona o número que colocar entre parênteses no final da lista. Da pra colocar um 'input()' dentro.

.sort() >>>> Vai ordenar a lista.

.sort(reverse=True) >>>>> Vai ordenar a lista ao contrário.

.insert(2, 0) >>> Insere uma variável no lugar que você desejar. Primeiro digite o lugar e depois a variável, separados por vírgula. Nesse caso não excluí o valor, apenas 'chega os itens para a direita'.

-- É possivel usar o len() em listas.

.pop() >>>> Deleta o último iten, mas é possivel colocar o iten que você deseja.

.remove() >>>> Pega o valor que você colocar, e remove o primeiro que encontrar. Se colocar um valor que não existe vai dar erro, mas é possível resolver com o 'if in'.

*for* c, v *in* enumerate(valores):  
 print(f'Na posição {c} encontrei o valor {v}.') >>>> Mostra o valores de um jeito mais limpo.

-- Quando se iguala lista, ex:' a = b', cria uma conexão entre elas, ou seja, qualquer alteração feita em uma irá acontecer na outra. Porém, se colocar que 'b = a[:]' aí n cria essa ligação, b passa a ser apenas uma cópia de a, e uma mudança feita em uma não afetara a outra.

max() mim() >>>>> pega o maior ou menor valor de uma lista ou tlupa

**Aula 18: Listas (Parte 2)**

--- É possível colocar uma lista dentro de outra lista.

Pode simplesmente dar o .append[:], com o .append normal iria criar uma conexão entre as listas e qualquer alteração feita em seria feita na outra,

ou na hora de escrever mesmo:

pessoas = [['Lucas', 35], ['Maria', 51], ['João', 56]]

--- Também é possível usar mais de um índice:

print(pessoas[0][0]) >>> Vai printar 'Lucas', o primeiro índice escolheu da lista 'pessoas' e o segundo escolheu da lista 0 de pessoas.

--- .sort() vai ordenar a lista permanentemente e o sorted() ordena temporariamente.

**Aula 19: Dicionários**

Dicionários são basicamente listas, mas é possível nomear os índices. Para os dicionários usa-se chaves{}. Declara-se como:

dados = dict()

dados = {'nome':'Pedro', 'idade ':'25'}

print(dados['nome'] >>>> Vai printar: Pedro

O .append() não funciona para os dicionários. Para adicionar um novo item é necessário declarar o nome do item e colocar o valor da variável.

dados['sexo'] = 'M' >> vai ficar assim >> dados = {'nome':'Pedro', 'idade':'25', 'sexo':'M'}

Para remover um item é só usar o dell:

dell dados[idade]

.values() >>> Literalmente os valores do dicionário

.keys() >>> Seriam os índices

.items() >>> São os dois juntos

.copy() >>>>> Para copiar um dicionário (a mesma coisa do [:] para as listas)

*from* operator *import* itemgetter

key=itemgetter() >>>>> Para ordenar dicionários, entre o parênteses coloca se vai ordenar o índice[0] ou o valor[1]. Não foi explicado muito bem.

sum() >>> Server para somar os valores dentro de uma lista .