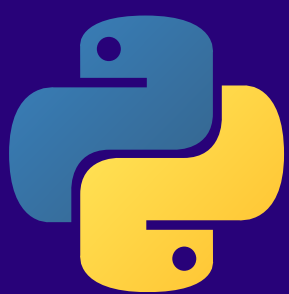
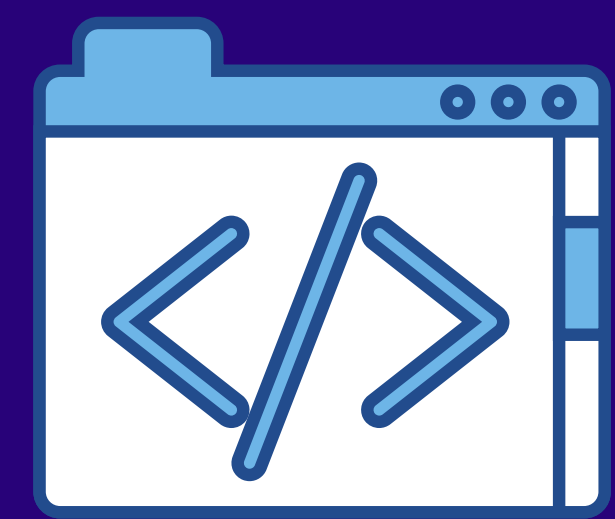
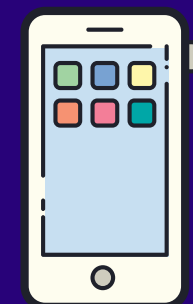
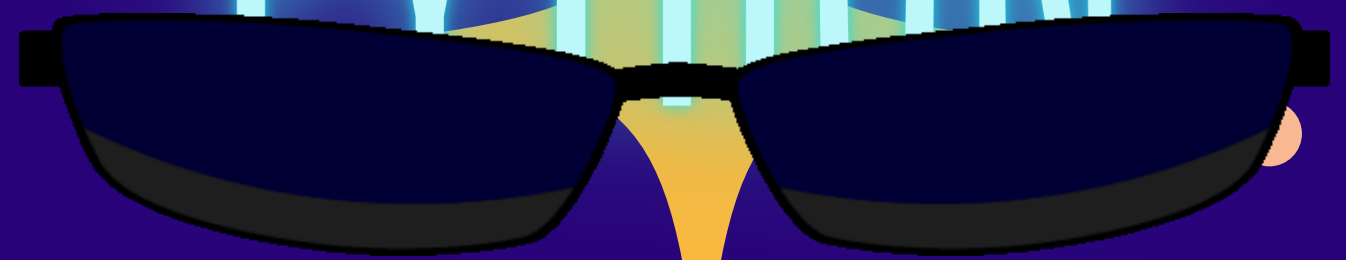



PYTHON




IMPRIMIR

 **`print('texto')`**

 **`print(variável)`**

 **`print(1+1)`** imprime 2

 **`print('1'+1)`** imprime 11


 **`print('texto', número)`** concatena


 **`print(variávelTexto, variávelNúmero)`**


 **`print('texto {} texto'.format(variavel))`**

 **`print(type(variável))`**


 **`print('texto {:.númerof}'.format())`**

 **`print('texto \n texto')`** pula linha

 **`print('texto' end= '')`** junta em uma linha
`print('texto')`

 **`print('texto \n texto')`** pula linha


 **`print("""texto
texto
texto""")`** 3 aspas duplas

 **`print(lista[][])`** imprime lista de lista

 **`print(dicionario['índice'])`** índice não numérico

 **`print(dicionario.values())`** retorna os valores

 **`print(dicionario.keys())`** retorna os índices

 **`print(dicionario.items())`** retorna os values e keys

VARIÁVEL


Toda variável é um objeto

 ***variável = tipo(input())*** int, float, bool, str

 ***tupla = ()*** imutável, input igual a variável comum


 ***tupla[posição]*** acessar dado na posição

 ***tupla[fatiamento]*** acessar alguns dados

 ***lista = []*** mutável

 ***variavel = list()*** outra forma de declarar

 ***lista.append()*** input


 ***listaA = listaB*** ligação, se alterar uma altera outra

 ***listaA = listaB[:]*** faz uma copia


 ***listaA.append(listaB[:])*** lista dentro da lista


 ***dicionário = { }***


 ***variavel = dict()***


 ***variável = { 'índice': 'x' }*** x = nome ou número

 ***variável['índice'] = tipo(input())*** recebe a informação










 ***def funcaocriada():*** ***o que vai fazer***
cria uma função com o que quero que ela faça

 ***def mensagem(msg):*** ***print(msg)***
mensagem('texto') substitui texto no lugar de msg

 ***def funcaocriada():*** ***o que vai fazer***
""" docstring a escrita do que a função faz """
sempre entre 3 aspas duplas na linha após o :

 ***def funcaocriada(para =0,me =0,tros =0):***
o que vai fazer
tornar os parametros opcionais evita problemas

STRING 1


-  ***string[posição]*** letra que está na posição dada por um número
-  ***string[posição inicial: posição final]*** letra de pi até pf -1
-  ***string[pi: pf: como anda]*** o número determina de quantas em quantas casas
-  ***string[: posição final]*** preenche automaticamente com zero antes dos :
-  ***string[posição inicial:]*** preenche automaticamente com a ultima posição da string após o :
-  ***string[posição inicial :: como anda]*** preenche automaticamente como nos anteriores
-  ***len(string)*** tamanho da string
-  ***string.count()*** conta quantas vezes aparece cada letra
-  ***string.count('letra', pi, pf)*** contagem com fatiamento


STRING 2

 ***string.find()*** procura o que voce colocar nos parênteses, volta a posição

 ***'texto' in string*** retorna True, ou False

 ***string.replace('palavra1', 'palavra2')*** troca a 1 pela 2

 ***string.upper()*** deixa tudo maiúsculo

 ***string.lower()*** deixa tudo minúsculo

 ***string.capitalize()*** só a primeira letra fica maiúsculo

 ***string.title()*** a primeira letra de cada palavra fica maiúsculo

 ***string.strip()*** tira espaços das laterais


 ***string.split()*** onde tem espaço ele divide


STRING 3

 **'-'.join(string)** une os elementos com o - entre eles


 **'texto' in string** retorna True, ou False


 **string.replace('palavra1', 'palavra2')** troca a 1 pela 2


 **string.upper()** deixa tudo maiúsculo


 **string.lower()** deixa tudo minúsculo

OPERADORES

 = recebe

 == iguala

 + soma

 - subtrai

 * multiplica

 / divide


 ** potência

 // divisão inteira

 % resto

 >= maior ou igual

 <= menor ou igual

 != diferente


 **and** e


 **or** ou

 **in** dentro

ORDEM DE PRECEDÊNCIA

1  `()` parenteses

2  `**` potências

3  `*` `/` `//` `%` nessa ordem

4  `+` `-`

CONDICIONAIS



if : se



elif : ou se



else: senão



for controle in range(número, número): intervalo ate ultimo numero -1



for controle in range(número, número, como anda): frente +1 ou nada, ou trás -1



for posição, item in enumerate (var composta): pegar cada posição e item



enquanto controle operador alguma coisa : faz enquanto



enquanto verdade : usar break pra sair

MÉTODOS

 ***variável.isnumeric()*** True/False


 ***variável.isalpha()*** True/False

 ***lista.pop(índice)*** para remover

 ***lista.remove(item)*** usado quando sabe o que esta dentro


 ***variável = list(range(,))*** forma de criar lista

 ***lista.append(dicionário.copy())*** forma de copiar dicionário sem fazer ligação


 ***try: operação except: algo que queira***
else: o que acontece se der certo a operação
finally: acontece independe de acerto ou erro

tratamento de erro bem igual do
javascript

FUNÇÕES

 ***pow()*** potência

 ***help()*** roda o código e depois escreve o que quer saber, biblioteca, função, etc...

 ***variável.sort(reverse=True)*** do maior pro menor

 ***variável.insert(posição, entra)*** inserção com escolha de posição

MÓDULOS / BIBLIOTECAS

 ***import biblioteca as apelido***

 ***from biblioteca import parte específica***

 ***from biblioteca import parte1, parte2***


 ***biblioteca.método()***

BIBLIOTECA MATH

 ***ceil()*** arredonda pra cima

 ***floor()*** arredonda pra baixo

 ***trunc()*** elimina da virgula pra frente

 ***pow()*** potência

 ***sqrt()*** raiz quadrada

 ***factorial()*** fatorial

BIBLIOTECA RANDOM

 ***.random()*** arredonda pra cima

 ***.randint()*** arredonda pra baixo