Resumo das aulas do curso em vídeo, eu já estou na aula número 6 e já resolvi os três exercícios propostos estou agora estudando os tipos básicos de variáveis sendo no python: Int; para números inteiros (7, -4, 0...): float; para números reais ou números de ponto flutuante (4.5, 0.076, -15.223, 7.0): bool; para valores lógicos ou boleanos (True ou False): str; para valores caracteres ou strens (PALAVRAS, ou NÚMEROS entre aspas. Tendo a estrem vazia ou as aspas sem nada dentro).

Os exercícios; até agora eu fiz os 4 primeiros sendo o mais difícil o 4º, ele utiliza o “.format” ou “format()” com a variável dentro dos parênteses ou antes do ponto. Iniciarei o a aula 7 no momento.

Operações aritméticas: + soma (2+2==4). – subtração (4-2==2). \* multiplicação (2\*2==4). / divisão (4/2==2). \*\* potencia (2\*\*2==4. // divisão inteira (5//2==2). % resto da divisão (5%2==1).

Bibilteca math

Biblioteca Random

Operadores de atribuição: ‘=’ recebimento. ‘+=’ atribuição e recebimento. ‘-=’ subtração e recebimento. ‘\*=” multiplicação e recebimento. ‘/=’ divisão e recebimento. ‘%=’ resto de divisão e recebimento.

Operadores de comparação: ‘>’ menor que. ‘<’ maior que. ‘==’ igual a. ‘!=’ diferente de. ‘>=’ menor ou igual. ‘<=’ maior ou igual.

Operador logico: ‘and’ Retorna True se todas as condições forem verdadeiras, caso contrário retorna False.

‘or’ Retorna True se uma das condições for verdadeiras, caso contrário retorna False.

‘not’ Inverte o resultado: se o resultado da expressão for True, o operador retorna false.

Operadores de identidade: ‘is’ Retorna True se as variáveis comparadas forem o mesmo objeto.

‘is not’ Retorna True se as variáveis comparadas não forem o mesmo objeto.

Operadores de associação: ‘in’ Retorna True caso o valor seja encontrado na sequência.

‘not in’ Retorna True caso o valor não seja encontrado na sequência.

Ordem de precedência:

1° ()

2° \*\*

3° \*, /, //, %

4° +, -

Aula 8; sobre módulos/bibliotecas e como colocar ou usar elas, tipo importar; para incluir no código uma biblioteca utiliza-se ‘import’ para toda a biblioteca, para importar comandos únicos da biblioteca coloca-se ‘fron+nome da biblioteca+import+código desejado’.

Uma biblioteca padrão do python é a ‘math’ de comandos de matemática com várias funcionalidades extras como ceil “teto” para arredondar números para cima, floor “piso” arredonda números para baixo, trunc “truncar” elimina todos os números depois da virgula, pow de potência, sqrt para calcular a raiz² e factorial para calcular as fatoriais

Manipulando string (cadeias de texto ou caracteres), aulas 9 do curso de py.

O py insere essas frases na memoria do computador colocando elas em mini bloquinhos de cada letra,

Fatiamento de string: “nome da variável [numero]” ex “frase[9]” = apenas uma letra.

Ex. frase[9:13] = pega as letra do 9 ao 13 sendo que o 13 não entra na contagem, pega apenas o antecessor. (ranger)

Frase[9:21:2] = pega do 9 ao 21, mas pulando de 2 em 2

Frase[:5] = quando o primeiro número não está iniciasse do 0 e vai ate o 5

Frase[15:] = inicia do 15 e como não foi dado um limite ele vai ate o fim

Frase[9∷3] = começa do 9 e vai ate o final pulando de 3 em 3

Analisar string:

Len(frase) = mostrar o comprimento da frase em numero

Frase.count(‘o’) = contar quantos ‘o’ tem na frase---- frase.count(‘o’, 0, 13) = conta já fatiando de 0 a 13

Frase. Find(‘deo’) = encontra o ‘deo’ na frase e mostra seu inicio.

Frase.find(‘’ algo que não ta dentro da string) === retorna um valor -1 pois não existe e deve ser levado em conta que a contagem começa a partir de 0 então -1 significa inexistente.

‘curso’ in frase == caso exista a busca que ta entre ‘’ na variável haverá um retorno em “True ou False”

Transformação (por via de regra não se pode mudar uma lista de string) mas da pra mudar através de métodos

Frase.replace(‘python’, ‘android’) = substitui a ‘python’ por ‘android’ de modo superficial/secundaria

Frase.upper() = coloca em maiúsculo

Frase.lower() = coloca em minúsculo

Frase.capitalize() = coloca toda a string em minúsculo e depois coloca a primeira letra em maiúsculo

Frase.title() = analisa quantas palavras tem na string e coloca a primeira letra de cada palavra em maiúsculo

Frase.strip() = remove os espaços inúteis da frase

Frase.rstrip() = remove os espaços inúteis da frase da direita

Frase.lstrip() = remove os espaços inúteis da frase da esquerda

Divisão

Frase.split() = divide a frase em palavras separadas e a contagem de cada uma começa do 0

Junção

‘-‘.join(frase) = junta todas as palavras em uma única frase

Valores.sort() organiza os valores em ordem crescente

Valores.sort(reverse = True) ordem decrescente

Aula 10

Métodos devem conter sempre os () parentes no final.

Condições

**IF……:**

Bolo True

**ELSE……:**

Bloco False

**ELIF….**

Ex…

tempo = int(input("quantos ano tem seu carro?"))  
#if tempo<=5:  
 # print("seu carro ainda é novo!")  
#else:  
 # print(" seu carro ja está velho!")  
print("seu carro ainda está novo"if tempo<=5 else"seu carro ja está velho")  
print("-----fim------")

adicionando cores no python sem modeulos:

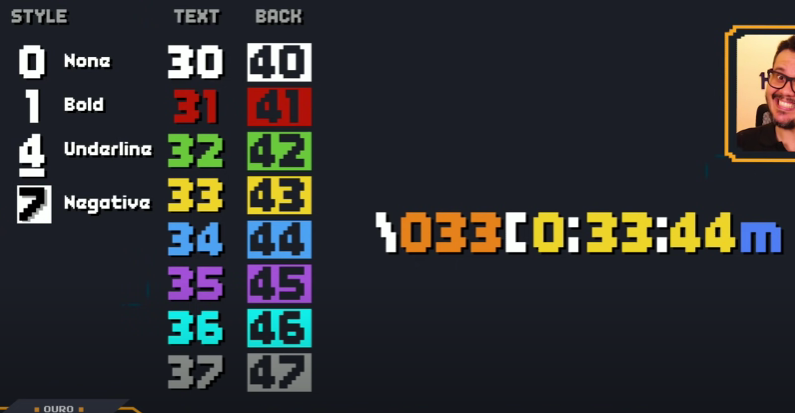
ansi- escape sequence

\033[m \033[ style ; cor do txt; back m

Codigos de style(estilo):: 0=sem estilo; 1=**negrito**; 4=underline; 7=inverter as cofuraçoes de fundo e letra.

Cor do txt :: 30 branco: 31 vermelho; 32 verde; 33 amarelo; 34 azul; 35 roxo; 36 ciano; 37 cinza:

Back, cor de fundo, mesma ordem das cores de texto partindo do 40



Criar de dicionário de cores:

cores = {  
 'limpa':'\033[m',  
 'azul':'\033[34m',  
 'amarelo':'\033[33',  
 'pretobranco':'\033[7;30',  
 'vermelho':'\033[31m',  
 'verde':'\033[32m'}

Pesquisar o modulo colorise do python

Acabou o modulo 1 de python, iniciar o modulo 2∷∷∷

ESTRUTURAS DE REPETIÇÃO:

**ELIF**

**FOR: for c in range(1, 8 ):** com a repetição identada no código: repete de 1 ate 8

for c in range(1, 10):  
 print(c)  
print("fim")

repete de 1 ate 501, pulando de 2 em 2

for p in range(1, 501, 2):  
 if p % 3 ==0:  
 count += 1  
 soma += p

WHILE: while c != 999:

while n != 999:  
 n=int(input("diigte um valor: "))  
 if n==999:  
 break  
 s+=n  
  
print(f"o resultado e {s}")

mundo 3 de python

tuplas – variável que guarda vários valores. (vetor)- imutável

lanche = ('hanburger','batata frita', 'suco', 'pizza', 'pudin', 'jaize')  
a = (2, 5, 7)  
b = (5, 9, 7, 1)  
for cont in range(0, len(lanche)):  
 print(f"eu vou comer {lanche[cont]}")  
print(" \n")  
 tuplas sao imutaveis  
for pos, comida in enumerate(lanche):  
 print(f'eu vou comer {comida}, na posicão {pos}')  
for comida in lanche:  
 print(f"eu vou comer {comida}")  
print(sorted(lanche)) #mostra o lance em hordem  
c = a + b  
print(c)  
#print(c.count(7))  
del(c)  
print(c.index(7, 3))  
print("saiuk")

lista – variável que guarda valores e é mutável

inserir/criar

valores = lista(range(4,11)) – cria uma lista e armazena em uma variável.

lache.append(‘ ’) – adiciona no final da lista

lanche.insert(0, ‘’) – adiciona no inicio ou em determinada posição

deletar

del lanche[3] – apaga tudo ou pelo indice

lanche.pop(3) -apaga o final, mas pode retirar pelo indice

lanche.remove(‘ ’) – deleta um determinado elemento passando o elemento

Código QR

Descrição gerada automaticamentedicinarios – variável que guarda valores com nomes

dados = {:} // nome da caravel = {nome da posição: elemento}

ex: alunos = {‘nome’: ‘joao’, ‘idade’: 20}

para criar uma nova posição basta refenciar

ex: aluno[‘sexo’]=’M’. assim criou a posição sexo

print(alunos.values()) // mostra todos os elementos(valores) do dicionário (joao, 20, M)

alunos.keys()//motra os nomes das posições onde os valores estão armazenadas

alunos.items()// mostra os valores e posições

\*funções

Começam com um DEF nome (paramentro)

Def define uma função

Paramentro, pode ser mais de um, ou um vetor de paramentros, pois se colocar um \* não frete da variável diz-se que vai receber um conjunto de valores que podem ser armazenados todos dentro de uma única variável

Funcoes podem ter retorno ou não

Interact help:

Help() é a função que explica os comandos, help(print). Ou pode usar o console python e entrar direto no help sendo um comando interno. Pode acessar a documentação de um comando print(input.\_\_doc\_\_) ou help(input) mostram as coisas

Doc strings:

Docstrings é uma forma de documentação resumida do código, fica na linha abaixo da função, e é escrita entre três aspas duplas, de modo a explicar o recebimento de argumentos da função. Texto

Descrição gerada automaticamente

Argumentos opcionais:

Os parâmetros opcionais são declarar as variaveis de recebimento com igual a um valor que não irá alterar o resultado do código, caso, apenas se elas não forem declaradas da chamada da função.

Texto

Descrição gerada automaticamente

Escopo de variáveis:

Global e local, refere-se ao lugar onde as varáveis podem ser acessadas diretamente. Pode-se variáveis locais e globais, por trabalhar as variáveis com os endereços é possível ter uma variável local com o mesmo nome da variável global. Podesse alterar a variável global ao invés de definir uma local, usando a palavra ‘global’ antes do nome da valiavel como se fosse uma declaração ex: ‘global a’

MODULARIZAÇÃO:

É criar pacotes e módulos (bibliotecas) no próprio diretório onde pode ser incluídas em outros códigos como as bibliotecas tradicionais, usando o comando “import”. Recomendado modularizar funções que são sempre usadas nos programas, de modo a reduzir a quantidade de linhas de códigos e a legibilidade dos programas.

TRATAMENTO DE ERROS:

Texto branco sobre fundo preto

Descrição gerada automaticamente com confiança média

Try: código que dará erro por algum motivo.

Except: código a ser executado caso haja erro, podendo ser mais de um except para o mesmo try.



Else: comando a ser executado caso não haja erro.

Finally: comando a ser executado ao final de tudo.

Else e finally são opcinais.

Texto

Descrição gerada automaticamente

Exception é uma classe de erro que pode mostrar o código do erro executado, mostrando assim o tipo do erro.