**Python em Tópicos – Semana 01**

**Aula 01:**

# Saída de Dados: print( )

# Variáveis são objetos.

# Tipos primitivos de variáveis: *string* (**str**), *number* (**int**, **float**), *booleano* (**bool**).

# *String* 🡪 “aspas duplas” ou ‘simples’. Se usar operador **+** com *string* gera concatenação.

‘’’ Entrada de dados: input( ) 🡪 por padrão retorna *string* como argumento se não definirmos o tipo primitivo antes. Ex.: variavel = int( input( ) ) recebe o dado do usuário e vai transformar em número inteiro, que será atribuído à variável. ’’’

# Vírgula 🡪 separa variáveis. Ponto 🡪 separa casas decimais em número *float*.

# \n 🡪 separa as linhas na saída de dados (escrever dentro das aspas).

‘’’ Função .format( ) 🡪 aceita vários argumentos como parâmetro, que devem ser digitados na ordem em que quer que apareçam no texto. No texto, colocar { } onde os argumentos devem ser inseridos. Ex.: print(“Olá, { }! Você tem { } anos!”.format(nome, idade)) ‘’’

# Alternativa de grafia para esta função 🡪 print( f”texto {argumento} .”)

# Função type( ) 🡪 verifica dados. Ex.: print( type (N1) )

‘’’ Funçao .is 🡪 mesma coisa, retorna *true* ou *false* para o que quiser testar. Ex.: .istitle( ), .isupper( ), tem muitas outras opções para testar. ’’’

# Operadores Aritméticos: **+** (soma) **–** (subtrai) **\*** (multiplica) **/** (divide)

**\*\*** (potência) **//** (divisão inteira) **%** (resto da divisão)

# **==** operador de igualdade (diferente de **=**, que é de atribuição)

# **!=** operador de diferença (diferente de)

‘’’ Ordem de Precedência: o que estiver entre parênteses vem primeiro. Depois, em ordem: potência **\*\***, depois multiplicar e dividir **\* / // %** , por último somar e subtrair **+ -** ‘’’

# Bizus de *output*: print(“-“\*20) 🡪 vai imprimir na tela 20 vezes o que estiver dentro das aspas.

# O { :20} reserva 20 espaços, com opções de formatação:

nome = "Fulano"  
print(f"Prazer, {nome:20}") -> Prazer, Fulano (espaços)  
print(f"Prazer, {nome:>20}") -> Prazer, Fulano  
print(f"Prazer, {nome:<20}") -> Prazer, Fulano (idem 1º)  
print(f"Prazer, {nome:^20}") -> Prazer, Fulano (centraliza)  
print(f"Prazer, {nome:=^20}") -> Prazer, =======Fulano=======

# Usar o número que quiser, 20 é só exemplo.

# O { :.2f} faz imprimir apenas duas casas decimais após o ponto em números *float*.

# O end=’ ‘ faz imprimir na mesma linha do terminal:

print("texto ", end=' ')  
print(" mais texto") -> texto mais texto

**Aula 02:**

# Módulos 🡪 pacotes de funcionalidades da linguagem. É preciso importar eles no código.

**import** <nomemodulo> Vai importar tudo

**from** <nomemodulo> **import** <funcionalidade> Importa algo específico

# Documentação: <https://docs.python.org/pt-br/3/library/index.html#library-index>

# Módulo **Math:** Após importar, usar onde quiser aplicar 🡪 **math.***funcao***( )**

<https://docs.python.org/pt-br/3/library/math.html>

Principais: math.**ceil( )** para arredondar para cima, math.**floor( )** para arredondar para baixo, math.**trunc( )** para cortar o que tem após o ponto decimal, math.**pow(x,y)** para potência, math.**sqrt( )** para raiz quadrada, math.**factorial( )** para fatorial.

Para seno, cosseno e tangente: math**.sin**( ), **.cos**( ) e **.tan**( ) 🡪 usar em radianos.

# Módulo **Random**: para gerar números pseudoaleatórios.

<https://docs.python.org/pt-br/3/library/random.html>

Principais: random.**random( )** Retorna o próximo *float* aleatório entre [0.0, 1.0), random.**uniform(a,b)** Retorna *float* aleatório entre a e b, random.**randrange(**a, b,passo**)** Retorna um *int* aleatório entre a e b com salto, random.**randint(a,b)** Retorna *int* aleatório entre a e b, random.**choice([a,b,c])** Retorna elemento aleatório dentre os elementos de uma sequência, random.**shuffle([a,b,c])** Embaralha a sequência.

# Módulo **Time:**

time**.sleep(s) :** Temporizador para a saída de dados, colocar tempo em segundos.

# Módulo **Datetime:**

datetime**.date.today( ).year** Retorna o ano atual, datetime**.date.today( ).month** Retorna o mês atual, datetime**.date.today( ).day** Retorna o dia de hoje.

**Aula 03:**

# Manipular *strings*:

*Strings* são cadeias de caracteres lidas da Esq-Dir do **0** ao último índice, sendo 0 o primeiro caractere, ou lidas da Dir-Esq do **-1** ao último índice, sendo -1 o último caractere.

# **Fatiamento:**

# Pegar caractere específico: string [ índice ] 🡪 frase [0]

# Pegar intervalo específico: string [indiceInicial : indiceFinal] 🡪 frase [5:10] (último não incluso)

# Pegar intervalo com salto: string [inicio : fim : salto] 🡪 frase [5:15:2] (lê o 5, 7, 9, 11 e 13)

# Ler do começo até um ponto específico: string [ : indiceFinal] 🡪 frase [ :15] (lê do 0 ao 14)

# Ler de um ponto específico até o final: string[inicio:] 🡪 frase [3: ] (lê do 3 até o fim)

‘’’ Ler de um ponto específico até o final com salto: string [ inicio :: salto] 🡪 frase [3: :2] (lê do 3 até o fim pulando de 2 em 2). ‘’’

# **Análise:**

# Função **len**(string) 🡪 vai dar o *length* (comprimento) da *string.*

# Função string**.count**(“caractere”) 🡪 conta quantos do que tem entre aspas existem na *string.*

‘’’ Função string**.count**(“caractere”, indiceInicio, indiceFim) 🡪 conta também mas direcionado para intervalo específico. O último índice não é incluso na análise.‘’’

‘’’ Função string**.find**(“caractere”) 🡪 busca se o que está entre as aspas existe na *string*. Se encontrar, vai retornar o índice em que o trecho buscado inicia. Se não encontrar, retorna -1. ‘’’

‘’’ Função de busca com **in**: print(“texto” **in** frase) 🡪 retorna *true* se existir o trecho entre aspas na (*in*) *string* em questão, ou *false* se não encontrar. ‘’’

# **Transformações:**

‘’’ Função string**.replace**(“antigo”,”novo”) 🡪 vai substituir o primeiro termo entre aspas pelo segundo. Vai alterar o tamanho da *string* se os termos tiverem tamanhos diferentes. ‘’’

# string**.upper**( ) 🡪 transforma caracteres em maiúsculos.

# string**.lower**( ) 🡪 transforma caracteres em minúsculos.

# string**.capitalize**( ) 🡪 transforma apenas primeiro caracter da *string* em maiúsculo.

# string**.title**( ) 🡪 transforma primeiro caracter de cada palavra da *string* em maiúsculo.

‘’’ string**.strip**( ) 🡪 remove espaços em branco antes e depois da *string.* Não remove os espaços entre as palavras. Adicionando um **r** de *right* (direita) ou um **l** de *left* (esquerda) na frente do *strip*, remove apenas os espaços à direita **.rstrip**( ) ou à esquerda **.lstrip**( ) da *string*, respectivamente. ‘’’

# **Divisão:**

‘’’ Função string**.split**( ) 🡪 divide as palavras da *string*: “Este é um texto” 🡪 [‘Este’, ‘é’, ‘um’, ‘texto’] . Assim, é possível acessar uma palavra específica através do seu índice: **.split**( )[indice] 🡪 **.split**( )[0] retornaria a primeira palavra (‘Este’), por exemplo. Ainda é possível acessar o índice de cada caracter dentro desta palavra 🡪 por exemplo, **.split**( )[0][0] retornaria a primeira letra da primeira palavra (‘E’). ‘’’

# **Junção:**

‘’’ Função “elemento”**.join**( string**.split**( ) ) 🡪 une as palavras através do elemento que está entre aspas. Pode ser um espaço em branco, uma letra, um símbolo, etc. Ex.: Este=é=um=texto‘’’

# **Cores no terminal:**

# **“\033[m** string **\033[m” 🡪** Entre a chave [ e o m é que vai o número da cor que se quer.

# Se for adicionar estilo e cor de fundo, seguir a ordem \033[ estilo; cortexto; corfundo m

# Estilo: 1 negrito

# Texto: Azul: 34; Vermelho: 31; Amarelo: 33;

# Fundo: inicia em 40 em vez de 30 (44; 41; 43)

**Aulas 04 e 05:**

# Estruturas condicionais: *lembrar dos dois pontos e da identação!*

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média# **if** <condição verdadeira>**:**

**ação**

# **else** <condição falsa>**:**

**ação**

# Saída de dados na mesma linha: print(**“**verdadeiro” **if** <condição verdadeira> **else** “falso”)

Texto

Descrição gerada automaticamente# **if** <condição verdadeira 01>**:**

**ação**

# **elif** <condição verdadeira 02>**:**

**ação**

# **elif** <condição verdadeira n>**:**

**ação**

# **else:**

**ação**

‘’’ Saída de dados pseudoternária: print(<condição teste> **and** “verdadeiro” **or** “falso”) retorna ou a primeira expressão ou a segunda, conforme a condição teste *true* ou *false*. ‘’’

**Aula 06:**

# Estruturas de repetição: *lembrar dos dois pontos e da identação!*

# **for** <contador> **in range (**<inicio>, <fim>**):**

**ação**

# O número de fim no *range* (intervalo) não é incluído.

# Por padrão, o salto do intervalo é de 1 em 1. Para outros valores, informar:

**for** <contador> **in range (**<inicio>, <fim>, <salto>**):**

Texto

Descrição gerada automaticamente

# Exemplo de 1 a 09 de 2 em 2:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

# Exemplo de 10 a 1 de 1 em 1:

Texto

Descrição gerada automaticamente

**Aulas 07 e 08:**

# **while** <teste lógico> **:**

**ação**

# While é uma estrutura de repetição com teste lógico.

# O for é uma estretura de repeticação com variável de controle.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente