Manual básico para Python

1 - Instalação;

2 - Uso de IDE;

3 – Uso e importação de bibliotecas;

4 - Criação do primeiro programa “Olá mundo” e introdução a comandos;

5 – Conhecimento de “INPUT”, “IF”, ”ELIF” e “ELSE”;

6 - Loop(repetição), uso do “While TRUE” e “break”;

7 – Definição de objetos;

8 - Criação e alteração de arquivos de texto;

9 - Uso de comandos de CMD;

10 - Criação de interface gráfica(uso da biblioteca “tKinter”);

11 - Criação e uso de automação(uso das bibliotecas “keyboard”, ”mouse” e “pyautogui”);

12 – Agendar tarefas (uso das bibliotecas: “schedule” e “time”);

13 – Editar planilhas (uso da biblioteca “pywin32”);

14 – Criação de um arquivo executável (uso da biblioteca “pyinstaller”);

15 – Comando de/por áudio;

16 – Exemplos de códigos;

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1 – Instalação:

Para realizar a instalação de Python, acesse <https://www.python.org/downloads/> e realize o download do instalador compatível com o seu sistema operacional

Obs.: o modelo usado nesse manual é 3.11.4

Na instalação:

1 - Após baixar o instalador compatível com seu sistema operacional, clique no executável

2 - Marque a caixa "Add python.exe to path" e clique em "Customize installation"

3 - Não desmarque nada e clique em "next";

4 - Clique nas caixas

"Install Python (versão baixada) for all users"

"Download debugging symbols"

"Download debug binaries (requires VS 2017 or later)"

4.1 - Tendo todas as caixa selecionadas, clique em "Install"

5 - Após a instalação, clique em "Disable path length limit" e depois "close"

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

2 – Uso de “IDE”:

Python, contém diversas IDEs, como pycharm, Júpiter, entre outros.

Para uso desse manual, será citado o Visual Studio Code, IDE gratuito lançado pela Microsoft.

Instalação do visual studio code:

1 - Acesse o link "https://code.visualstudio.com" e clique no botão de "download" para o seu sistema operacional

2 - Clique no instalador, concorde com os termos e clique em "Próximo";

3 - Não mude o caminho da instalação, e clique em "Próximo";

4 - Não mude o caminho da pasta que irá criar, não clique em "não criar uma pasta no Menu Inicial" e clique em "Próximo";

5 - Defina quais caixas quer marcar e clique em "Próximo"

6 - Clique em "Instalar";

7 - Aguarde a instalação e clique em "Concluir"

8 - Após a instalação, clique em "Extensions" na caixa de pesquisa procure por "python", e clique no botão azul "Install";

9 - Criar um primeiro código de teste, clique em "Select Python interprete", no canto inferior direito que irá aparecer;

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3 - Uso e importação de bibliotecas:

IMPORT, -> serve para importar bibliotecas, muitas das vezes tem que exercer a função "PIP INSTALL...(escreva a biblioteca após o pip install)" no prompt de comando para instalar a biblioteca, como o caso do "pyautogui"

obs¹.: Para executar a função "pip install..." precisa ser administrador no CMD, ou pode ser executado como usuário no Visual Studio Code, na área de terminal de deles.

obs².: quando for usado import, ele pode ser substituído por "from (biblioteca referida) import \*", o "\*" serve para baixar toda a biblioteca sem precisar ter que escrevê-la sempre.

Para desinstalar, utilize o comando “uninstall”, como no exemplo “pip uninstall (biblioteca)”;

Para atualizar, utilize o comando “-upgrade” seguido do comando “install”, como no exemplo “pip install -upgrade (biblioteca)”

Para listar as bibliotecas instaladas e suas versões, utilize apenas o comando “pip list”.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4 – Criação do primeiro programa “Olá mundo” e introdução a comandos:

Para comentar nos códigos de python, utilize “#” antes da frase a palavra escrita, ou “””(3 aspas) na linha inicial e final, caso queira comentar mais de uma linha.

PRINT("") -> escreve em tela a frase dentro das aspas;

Ex.:

print("olá mundo!")

STRINGS -> são letras e palavras que são escritas

Ex.:

carro='charger'

INT e VAR -> são os números que são escritos, "int" são os números escritos inteiros, e "var" são os números quebrados, e vale ressaltar que não se pode escrever número quebrado usando virgula e sim ponto.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

5 – Conhecimento de “INPUT”, “IF”, ”ELIF” e “ELSE”:

INPUT -> seguindo a palavra para definir a classe, com a função "input", define qual o objeto, e após isso, vem as funções "if", "elif" e "else", que servem para definir escolha.

obs.: quando colocar o input, logo abaixo tem de ter o if, no mesmo angulo.

IF -> É a primeira opção sempre. Obs. sempre terminar com dois pontos ":" após o uso do IF para definir a consequência.

Se o IF for usado para String, ou no caso palavras, coloque-as dentro de parênteses, se for para números ou equações matemáticas, não.

obs.: após o uso do primeiro IF, tanto quanto no elif e no else, tem que ter no final os ":" e abaixo começar com um tab de diferença da linha de cima.

ELIF -> pode ser usada como segunda opção, e pode ser usada inúmeras vezes, mas nunca como a última opção, e pode ser também substituída pela opção IF.

ELSE -> É sempre a última opção, ou caso já tenha definido todas as opções usando "IF" e "ELIF", o "ELSE" serve para caso não tenha sido escolhido nenhuma das opções apresentadas.

Ex.:

resposta = input("identifique-se: ")

if(resposta=="marcelo"):

print("autorizado")

elif(resposta==”carol”):

print(“autorizada”)

else:

print("negado")

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

6 - Loop(repetição), uso do “While TRUE” e “break”:

WHILE -> É a função criada para o que ocorre enquanto o código principal está rodando sendo seguido de "True" ou "False", Obs. é importante observar que a primeira letra dessas duas opções é maiúscula.

True -> traduzido literalmente como "verdadeira", pode ser após a função "WHILE" que mostra que está ativo

False -> "falso", seguido da função "WHILE", define que a função está desligada, ou não iniciada, até após alguma escolha mostrar "while true", que pode mudar isso, podendo ser usada como “return False”;

Break -> Serve para interromper a função "WHILE" se executada previamente.

Ex.:

while True:

resposta = input("identifique-se: ")

if(resposta=="marcelo"):

print("autorizado")

elif(resposta==”carol”):

print(“autorizada”)

else:

print("negado")

break

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

7 – Definição de objetos:

DEF "objetos" -> É para criar objetos complexos, que possam ser usados nos códigos, sem a necessidade de repeti-los continuamente:

Ex.:

def aviso():

resposta = input("identifique-se: ")

if(resposta=="marcelo"):

print("autorizado")

elif(resposta==”carol”):

print(“autorizada”)

else:

print("negado")

while True:

aviso()

break

Repare que no exemplo, "def aviso" é seguido de "():", isso é necessário, para mostrar que o código acima, está contido dentro dos parênteses.

e da mesma forma, para referenciar o objeto durante o código, é necessário repetir "()", mas sem a necessidade dos dois pontos para executar o código previamente definido.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

8 - Criação e alteração de arquivos de texto:

É possível criar, ler e alterar arquivos de texto, seguindo os seguintes comandos:

w - Usado somente para escrever

w+ - Usado para escrever sobre algo

r - Usado somente para ler

r+ - Para ler e escrever algo

a - Para acrescentar

- Criar arquivo:

Escreva o comando “open” seguido das informações dentro parênteses e aspas, sendo que o arquivo precisa estar devidamente escrito o caminho e tipo de arquivo após o ponto, e em seguida o “W” entre as aspas, pois é o comando para criar o arquivo;

Ex.:

open(“c:/users/usuário\_exemplo/desktop/arquivo.txt”,”w”)

obs.: na criação do arquivo, ele não precisa ser necessariamente “txt”, podendo ser em qualquer formato, mas a resposta terá retorno apenas se a criação do arquivo, pode ser feita por texto

Ex.: bat, html, py, json, doc... entre outros

- Escrever dentro do arquivo criado;

Execute o comando anterior como um objeto a ser definido, e após isso, com esse objeto, execute em conjunto o comando “write”, e após isso, o que quer escrever dentro de parênteses e aspas.

Ex.:

arquivo = open(“c:/users/usuário\_exemplo/desktop/arquivo.txt”,”w”)

arquivo.write(“Olá mundo! “)

- Editar dados em arquivo já existente:

Da mesma forma como criar um novo arquivo, o comando irá apagar os dados existente, e escrever a nova linha informada, com a diferença de haver um “+”, após o “W”, e seguir o comando “read”;

Ex.:

arquivo = open(“c:/users/usuário\_exemplo/desktop/arquivo.txt”,”w+”)

conteúdo = arquivo.read()

arquivo.write(“Olá mundo! “)

- Ler conteúdo de arquivos.

Para ler o conteúdo do arquivo, segue o padrão dado anteriormente, mas mudando o comando “W”, para “R”, sendo necessário também o comando “print”, para mostrar o conteúdo lido;

Ex.:

arquivo = open(“c:/users/usuário\_exemplo/desktop/arquivo.txt”,”R”)

conteudo = arquivo.read()

print(conteudo)

- Para acrescentar informação:

Para acrescentar informação, há duas maneiras, colocar o comando “A” ou colocando o comando “R+”;

- Usando o comando “A”, é necessário apenas o comando com a localização do arquivo e o comando para escrever o dado a adicionar.

Ex.:

arquivo = open(“c:/users/usuário\_exemplo/desktop/arquivo.txt”,”A”)

arquivo.write(“texto a ser acrescentado”)

O arquivo citado acima, irá ter a informação acrescentada, após a informação já escrita no arquivo não apagando o dado escrito,

- Usando o comando “R+”, é necessário o comando com a localização do arquivo, o comando para escrever o dado a adicionar e o comando “read”, para ler o dado já estabelecido.

Ex.:

arquivo = open(“c:/users/usuário\_exemplo/desktop/arquivo.txt”,”R+”)

conteudo = arquivo.read()

arquivo.write(“texto a ser acrescentado”)

O código citado acima, pode escrever por cima dos dados escritos, se não posto o comando “read”, diferente quando usado o comando “A”.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

9 - Uso de comandos de CMD:

Execute a primeira linha: import os, não é necessário instalar, pois ele já é nativo do python

Depois disso execute a linha de comando:

os.system('o comando de CMD que você conhece').

Ex.:

import os

os.system('shutdown /s /t 1')

#Serve para desligar o computador.

os.system('msg \* "Aviso do prompt de comando"')

#Serve para criar mensagens de aviso padrão do prompt de comando.

os.system("start Chrome.exe")

#Serve para abrir o browser Chrome.

os.remove(“arquivo.txt”)

#Serve para apagar um arquivo.

os.system("start c:\\users\\marcelo.pugan\\downloads\\%DATE:~0,2%-%DATE:~3,2%-%DATE:~-2%")

#Serve para abrir a pasta com a data de atual.

Obs.: quando direcionado ou citado pastas dentro de python(independente da IDE), é necessário usar duas barras “\\” ao invés de uma como no CMD.

os.system("taskkill /im code.exe /f")

#o comando é para a finalização do Visual Studio Code, podendo ser trocado por outro executável em questão.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

10 - Criação de interface gráfica

Para criação de interface gráfica em python, utilizaremos a biblioteca “tkinter”, nativa do python desde a versão 3.8, não necessitando de realizar a instalação com o comando “pip”, mas ainda necessitando do comando “import tkinter” nos códigos;

Para criar uma janela simples, seguir o padrão importar a biblioteca, definir o objeto com a biblioteca, e no final, colocar esse objeto em loop, para manter a janela aberta.

Como para criar uma janela simples:

Ex.:

Import tkinter

janela = tkinter.Tk()

janela.mainloop()

Comandos para personalizar a janela criada:

Ex.:

janela.geometry(“300x200”) # para definir o tamanho da janela em pixel, sendo X e Y respectivamente as medidas;

janela.title(“titulo do aplicativo”) # para definir titulo;

janela.configure(background=”#000”) # para definir a cor de fundo, sendo usado hexadecimal para escrever a cor, também pode ser substituído por “bg”

janela.attributes(‘-fullscreen’, True) # para definir a janela cheia, mas é necessário a configurações de outros objetos na interface, como local dos botões e rótulos.

Para configurar texto na interface:

Para a criação de textos ou descrições dentro da interface é necessário o uso do comando “Label” e dentro dele as especificações.

Ex.:

label = tkinter.Label(text=”texto do label”)

Mas não bastando o comando acima, é necessário também, informar a posição do label, utilizando a definição X e Y;

Ex.:

label = tkinter.Label(text=”texto do label”)

label.place(x=120, y=100)

ou

label.place(relx=0.5, rely=0.4, anchor=tkinter.CENTER)

# sendo esse modelo, usado, quando a janela estiver em modo “fullscream”, os “relx” e “rely”, são adaptáveis a janela, enquanto que o comando “anchor=tkinter.CENTER”, mantem o objeto no centro.

O comando Label, pode ser personalizado com comandos internos, para se adaptar ou se destacar dentro da interface;

Ex.:

text=”texto do label”;

bg=”numero hexadecimal ou cor especifica” # para mostrar a cor de fundo;

fg=” numero hexadecimal ou cor especifica” # para mostrar a cor das letras;

font=numero inteiro # serve para definir o tamanha, mas também pode decidir o tipo de fonte quando especificado.

Ex.:

label = Label(text="texto", font=("Arial", 20), bg="black" ,fg="red")

label.pack(padx=20, pady=20)

Adicionando Frames:

Para a criação de Frames(janelas dentro das janelas), defina o objeto com o frame, e suas características:

Ex.:

janela\_frame = Frame(width=200, height=100)

janela\_frame.place(x=25, y=25)

Configurando botões:

Para a criação de botões, é a mesma logico do Label, mas com o acréscimo de executar funções;

Ex.:

butao = Button()

butao.place(x=50, y=100)

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

11 - Criação e uso de automação(uso das bibliotecas “keyboard”, ”mouse” e “pyautogui”);

Para uso dessas bibliotecas, realize as instalações:

Ex.:

pip install keyboard

pip install mouse

pip install pyautogui

E importe as bibliotecas em seus códigos:

Ex.:

import keyboard

import mouse

import pyautogui

Para os comandos há diferenças, que serão mostradas aqui:

11.1 – Comandos para uso da biblioteca “keyboard”:

Pressionar uma tecla:

keyboard.press(‘a’)

keyboard.release(‘a’)

combinação de comandos para pressionar uma tecla:

keyboard.press\_and\_release(‘a’)

combinação de comandos para pressionar mais de uma tecla:

keyboard.press\_and\_release(‘ctrl+alt+delete’)

comando para esperar entre comandos:

import keyboard

import time

keyboard.press(‘w’)

time.sleep(3)

keyboard.release(‘w’)

Obs.: o comando “time.sleep(3)” foi usado para esperar por 3 segundos

comando para reagir quando uma tecla for apertada:

while True:

if keyboard.is\_pressed(‘q’):

print(“a tecla ‘q’ foi apertada, saindo...”)

break

Obs.: para o código identificar que uma tecla ou comando foi executado, ele tem que estar em loop (“while True”).

Exemplo de comando de monitoramento de múltiplas teclas:

Import keyboard

While True:

If keyboard.is\_pressed(‘a’):

print(“Tecla ‘a’ pressionada.”)

elif keyboard.is\_pressed(‘b’):

print(“Tecla ‘b’ pressionada.”)

elif keyboard.is\_pressed(‘esc’):

print(“Tecla ‘esc’ pressionada, saindo...”)

break

11.2 – Comandos para uso da biblioteca “mouse”:

Comando para clicar em uma posição específica:

import mouse

x, y = 100, 200

mouse.click(x, y)

# outros exemplos com as posições definidas:

mouse.right\_click(x, y)

mouse.double\_click(x, y)

11.3 – Comandos para uso da biblioteca “pyautogui”:

import pyautogui

# função para mover o mouse para a localização na tela

pyautogui.moveTo(x= y=)

# Necessário colocar após os caracteres "x" e "y" a localização na tela (plano cartesiano)

# função para clicar com o mouse na tela

pyautogui.click(x=número ,y=número) / também vale para a função "doubleClick" / "rightClick"

Obs.: pode ser necessário, usar antes a função "moveTo", para gerar a função "Click".

# função para esperar o mouse por (4) segundos

pyautogui.sleep(4)

# função para mostrar a posição do mouse na tela

print(pyautogui.position())

# escreve, desde que dentro ("")

pyautogui.write

# pressionar a tecla(e manter a tecla pressionada)

pyautogui.keyDonw

# soltar a tecla que antes foi pressionada

pyautogui.keyUp

# pressionar de forma rápida

pyautogui.press

obs.: dentro dos ("",), após a virgula, ao usar o comando "presses=" seguido de um número, você pode definir a quantidade de vezes será pressionado.

Ex.: pyautogui.press("tab", presses=4)

# tampo de pausa entre as funções(X = segundo)

pyautogui.pause = X

# para pressionar conjuntos de teclas simultaneamente

Ex.: pa.hotkey("ctrl", "l")

# função para localizar figura na tela

pyautogui.locateOnScreen('colocar a figura aqui')

Obs.: caso ('The Pillow package is required to use this function.') esse erro apareça,

utilize o código: pip install Pillow --upgrade no CMD, para corrigi-lo

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

12 - Agendar tarefas:

Para o agendamento de códigos em python, utilizando a biblioteca “schedule” e a biblioteca “time”, mas a instalação é apenas da biblioteca “schedule”.

pip install schedule

a biblioteca "schedule" serve para iniciar a atividade no momento requisitado, enquanto a biblioteca "time" serve para definir o momento

comando para executar repetidamente:

import schedule

import time

def tarefa():

print(“executando a tarefa”)

schedule.every(5).seconds.do(tarefa)

# o número após “schedule.every” define durante quanto tempo irá rodar o loop;

# “seconds” para segundo / “minutes” para minutes / “hours” para horas / “days” para dias /”month” para meses;

# na posição de “seconds”, pode ser substituído por dias da semana: “monday”, “wednesday”, “friday”...

while True:

schedule.run\_pending()

time.sleep(1)

comando para executar em um horário específico:

import time

import schedule

def tarefa():

print(“executando tarefa agendado”)

schedule.every().day.at(“11:00”).do(tarefa)

while True:

schedule.run\_pending()

Obs.: não há a necessidade do uso do “time.sleep” para a finalização do código.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

13 - Editar planilhas:

Para editar, copiar e colar, mover, e executar macros ou tabelas dinâmicas, é usado nesse manual a biblioteca “pywin32”.

Para realizar a instalação da biblioteca, execute o comando “pip install pywin32”, seja no terminal do VSC ou no CMD.

13.1 – Utilização da biblioteca:

Para utilizar os comandos do pywin32, coloque o seguinte comando “import win32com.client” e após isso defina os seguintes códigos como mostrado no exemplo a seguir:

Ex.:

import win32com.client as w32

excel = w32.Dispatch('Excel.Application') # Serve para definir a execução do excel;

excel.Visible = True # Serve para tornar visível as automações que serão realizadas, caso, seja “= False” elas ficaram em segundo plano.

excel.DisplayAlerts = False # Serve para evitar os avisos quando fecha uma planilha, como se deseja que salve ou algo similar;

workbook = excel.Workbooks.Add() # serve para abrir uma planilha nova;

workbook = excel.Workbook.Open('caminho/para/seu/arquivo.xlsx') # serve para abrir uma planilha especificas;

worksheet = workbook.Sheets(1) # para especificar a aba da planilha, sendo “1” a primeira;

worksheet = workbook.Sheets(“aba escrita”) # é outro exemplo, mas é especificando a aba escrita;

workbook.Save() # serve para salvar a alteração do arquivo;

workbook.SaveAs('caminho/para/seu/arquivo.xlsx') # serve para salvar a planilha em lugar especifico;

workbook.close() # serve para fechar essa planilha, normalmente no final do código;

excel.Quit() # serve para fechar o todo excel após a finalização dos comandos;

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

14 – Criação de um arquivo executável:

# Caso você queira compartilhar seu arquivo, com alguém que não tenha python instalado em sua máquina, pode fazer isso, transformando o arquivo em .exe

# primeiro vá no seu CMD como administrador e execute o seguinte comando para instalar o conversor:

pip install pyinstaller

# após isso no CMD, com o pyinstaller tendo concluído a instalação, execute o seguinte comando:

pyinstaller –onefile --noconsole "arquivo que você queira criar o executável"

Obs¹.: O comando “--onefile” serve para criar todo o código em um único arquivo executável.

O comando “--noconsole” serve para, que o arquivo criado, não abra em conjunto uma janela do prompt ao executar

Obs¹ ².: O comando “--noconsole”, pode ativar o antivírus do Windows, e para criar o arquivo executável sem a janela do prompt, é necessário desativar o antivírus do Windows, durante a criação do executável.

# Após isso, irá aparecer em sua área de trabalho duas pastas: "builder" e "dist".

# Seu arquivo executável estará na pasta "dist", e você poderá compartilhar com outros para testar, os arquivos na pasta "builder" são os arquivos criados na conversão, você poderá exclui-los se preferir.

Obs².: Caso aconteça o seguinte erro:

O termo 'bloco\_de\_notas2' não é reconhecido como nome de cmdlet, função, arquivo de script ou programa operável. Verifique a grafia do nome ou, se um caminho tiver

sido incluído, veja se o caminho está correto e tente novamente.

Solução 1:

Execute os seguintes comandos:

1 - Acesse o explorer, botão direito no meu computador, clique em "propriedades";

2 - Clique em "Configuração avançadas do sistema";

3 - Clique em "variável de ambiente...";

4 - Na parte "Variáveis do sistema", procure a variável "Path" e clique em "Editar...";

5 - Clique em "Novo" e coloque o seguinte caminho "C:\Program Files(x86)\Python\Python36-32\";

6 - Clique em ok;

7 - Reinicie o terminal (seja ele no CMD ou Visual Studio Code).

Solução 2:

1 - No terminal do VSC(Visual Studio Code);

2 - Acesse a pasta onde se encontra o arquivo .py salvo;

3 - Execute o seguinte comando: python -m PyInstaller --onefile (nome do arquivo .py).

obs.: importante que o "P" e "I" do pyinstaller estejam em caixa alta.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

15 – Comando de e por áudio;

Em python, há a possibilidade de executar comandos por áudio ou que o código responda com comandos falados

Para que o código responda por vós, utilize a biblioteca “pyttsx3”, executando o comando “pip install pyttsx3”;

Comando simples para falar:

Ex.:

Import pyttsx

engine = pyttsx3.init()

engine.say(“olá mundo!”)

engine.runAndWait()

No comando acima, é necessário o comando “engine = pyttsx3.init()” para iniciar o mecanismo de síntese de vóz;

O comando “engine.say(“olá mundo!”)” fala o texto citado entre os parênteses, mas é necessário após isso o comando “engine.runAndWait()”, para que execute.

Para configurar ou especificar a fala no comando:

Ex.:

Import pyttsx

engine = pyttsx3.init()

engine.setProperty(‘volume’, 0.9) # configura volume a 90%

engine.setProperty(‘rate’, 150) # configura a velocidade para 150 palavras por minuto

engine.say(“olá mundo!”)

engine.runAndWait()

------------------------------------------------------------------------------------------------------------------------------------------------------------------------