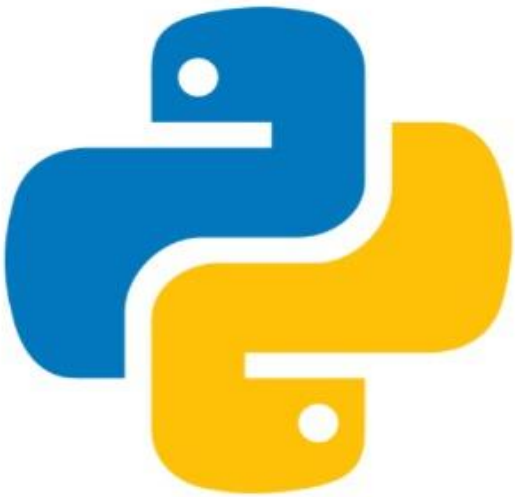
APOSTILA PYTHON



ÁLVARO DE MELO SANTIAGO

AMANDA ELISE SILVA DE OLIVEIRA

BEATRIZ VIEIRA DA SILVA ANDRADE

ISABELA DAMACENA DA CRUZ

IZABELLY APARECIDA SANTOS DA SILVA

JUAN PABLO DA SILVA MACHADO

LUIZ FELIPE SILVA MIRANDA

MARCELO GUSTAVO JESUS DE GÓIS

MARIA EDUARDA DE OLIVEIRA SALVADOR RUIS

MARIA EDUARDA BARBOSA MARTINEZ

VITÓRIA LINDA DA SILVA OLIVEIRA

Sumário

CAPÍTULO 1

1. HISTÓRIA E CARACTERÍSTICAS DO PYTHON

A criação da linguagem Python foi feita por Guido van Rossum (matemático e programador), no final dos anos 80 enquanto trabalhava na CWI (Centro de Matemática e Ciência da Computação) na Holanda. O nome vem da comédia britânica Monty Python.

Quando Guido iniciou o seu trabalho na CWI, trabalhando na equipe do sistema operacional Amoeba, ele percebeu que era necessário uma linguagem melhor, que fosse entre C e Shell Script, e que melhorasse o desempenho do sistema, tornando-o mais rápido e eficiente, substituindo assim a linguagem ABC. E Python foi o substituto perfeito, pois atendeu todos os requisitos que era esperado, sendo lançado em 1991, quando Guido publicou o código em uma comunidade.

A linguagem possui uma "filosofia" que a guia, o poema Zen of Python, que foi escrito na própria linguagem por Tim Peters, é possível vê-lo através do comando

">>> import this"

A primeira versão foi lançada em janeiro de 1994, logo depois foi lançado o Python 1.2, o último lançamento de Guido na CWI.

Após isso, começou a trabalhar na CNRI nos Estados Unidos, e lançou novas atualizações, como a versão 1.4 e a versão 1.6, última versão de Python na CNRI.

No ano 2000, toda a equipe do Python saiu da CNRI para a BeOpen, para montar o PythonLabs.

Na BeOpen foi lançado o 2.0, e após ele, o time mais uma vez se mudou, dessa vez para Digital Creations, tendo sua licença alterada para Python Software Foundation License, tendo como versão desta licença a 2.1, depois foram lançadas a 2.2 e 2.6, com novas atualizações para deixar o código cada vez mais acessível e prático.

Em 2008 foi lançado o Python3, para mudar erros nas versões anteriores. A última versão lançada foi a 3.9

Hoje em dia, Python é uma das linguagens mais famosas e mais utilizadas em diversos locais, seja em

grandes empresas como o Google ou pequenas empresas, sendo também linguagem padrão de vários S.O. como o Linux.

O que caracteriza a linguagem Python e a torna diferente das demais são a fácil aprendizagem e a simplicidade do código, podendo ser usada por qualquer pessoa. Ela também não precisa do uso das chaves durante o código, como nas demais linguagens, fazendo com que fique mais limpa e com melhor visualização

2. POSSIBILIDADES DE IDEs PARA PYTHON

IDE, Ambiente de Desenvolvimento Integrado, em português, é um programa que reúne ferramentas necessárias para a construção de outros softwares. Com ele é possível criar janelas, botões e outros elementos de interação de usuários com poucos cliques. Eles são um grande impulso na produtividade, sendo fundamentais para o desenvolvimento de grandes projetos.
Algumas das principais IDEs para usar Python são:

- Spyder: é uma ferramenta leve, simples e ao mesmo tempo poderosa, que conta com elementos avançados de edição, depuração, testes interativos e recurso de exploração de variáveis, que exibe os conteúdos armazenados dentro de cada uma;
- PyCharm: é um dos IDEs para Python mais completos. Existe na versão profissional (paga) e comunitária (código aberto). Tem uma interface muito limpa e personalizável e possui suporte para controle de fontes e projetos. Além disso, é ideal para iniciantes, pois é possível acessar passos a passos diretamente dentro da aplicação;
- Jupyter: é um IDE Python gratuito, utilizado principalmente na análise e ciência de dados**.** Ele é fácil e intuitivo, proporcionando um bom ambiente para iniciantes

em Python, e, além disso, conta com muitos materiais de referência;

- Eclipse: é talvez a IDE de código aberto para desenvolvimento Java mais famosa entre os programadores. Ele possui uma grande variedade de extensões, uma delas é a PyDev, que permite a programação em Python. Entre outras funcionalidades, ela permite compilação de código e debug;
- Visual Studio Code: leve e completo, é um editor de código disponível para todas as plataformas. Ele possui código aberto, é extensível e pode ser configurado para praticamente qualquer tarefa, além de ter uma interface totalmente customizável. Ademais, é muito fácil de ser instalado.

3. IDE ESCOLHIDA PARA O TRABALHO

O Visual Studio code, é uma multiplataforma, que tem um suporte para mais de 30 linguagens diferentes e possui código aberto, ou seja todos os usuários tem acesso a fonte de código. Ele nos permite programar sem limitações, com diversos arquivos, extensões etc. É expansível e customizável, sendo assim um dos IDE mais utilizados pelos programadores, e pelo fato dele ser completo e otimizado, ajuda a desenvolver aplicações de modo fácil e eficaz, e se encaixa muito bem para o desenvolvimento em Python.

Ele também permite que você deixe seu programa mais clean, pois permite mexer no código direto do editor, facilitando a correção de erros e bugs. Além disso, tem uma série de funcionalidades sendo elas: Informações de parâmetro, completar palavra, entre outras.

Para mais, além dos atalhos regulares, o editor também possui um painel de comando incrivelmente rápido. No geral, o VS Code contém milhares de teclas de atalho e comandos, que podem melhorar significativamente a eficiência de nosso

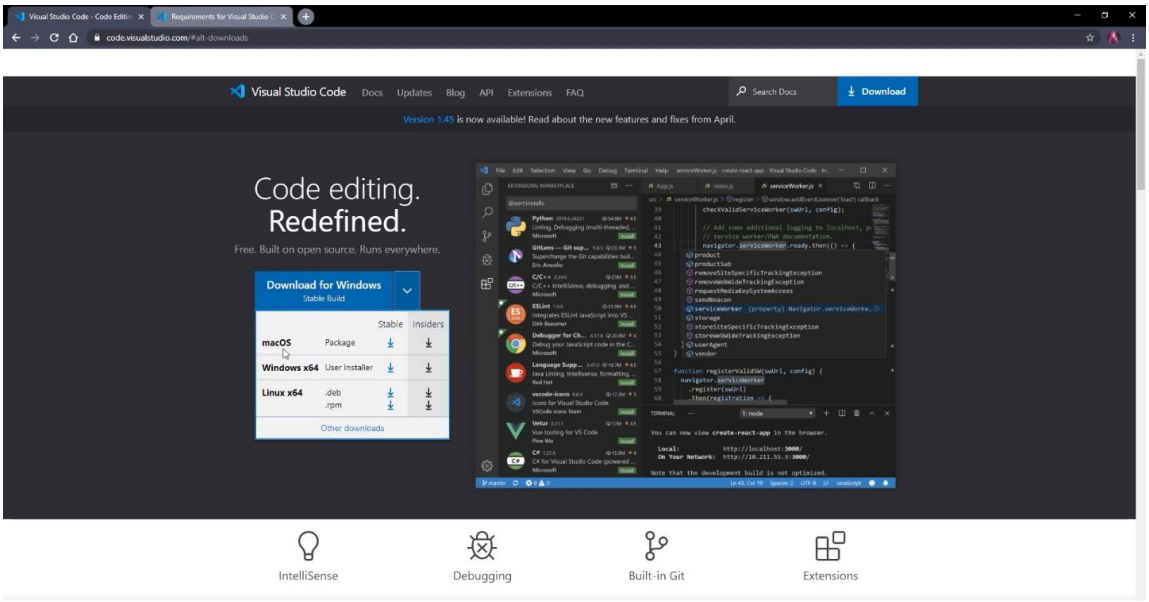
trabalho durante o processo de codificação, e por isso trabalharemos com ele ao decorrer da apostila.

4. PROCESSO DE DOWNLOAD E INSTALAÇÃO DA LINGUAGEM E IDE – VSCODE

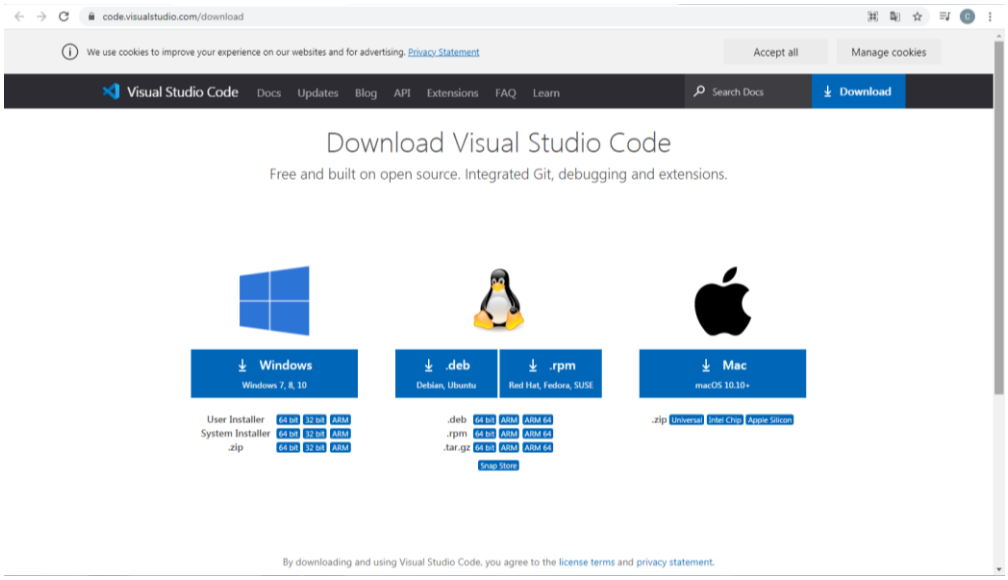
Visual Studio code é um editor de código, gratuito da Microsoft, onde há diversas extensões para que sejam adicionadas funcionalidades extras.

Para que seja instalado, basta você possuir um hardware de 1,6 Ghz e 1gb de Ram, e com isso já é possível instalá-lo.

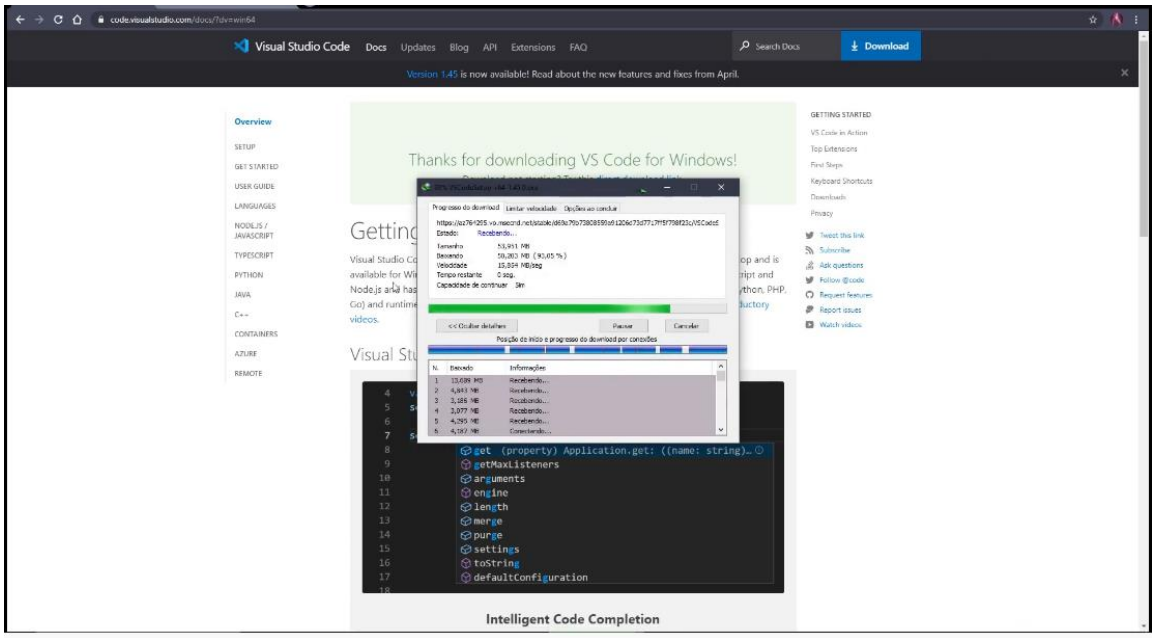
1. Link para download: https://code.visualstudio.com/download



2. Abrindo o site há as opções de sistemas operacionais onde pode ser instalado o programa.

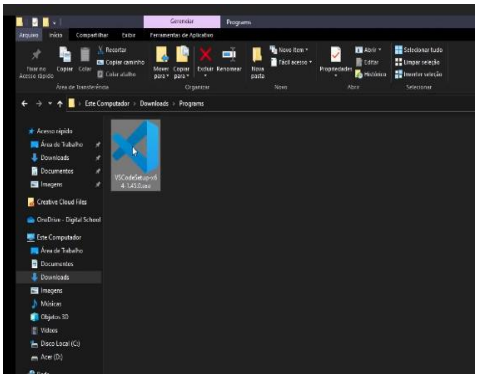


3. Após ser selecionada a opção que se adequa ao seu computador, será iniciado o download.

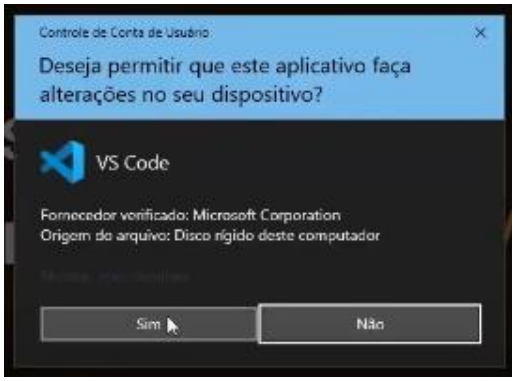


4. Já no diretório:

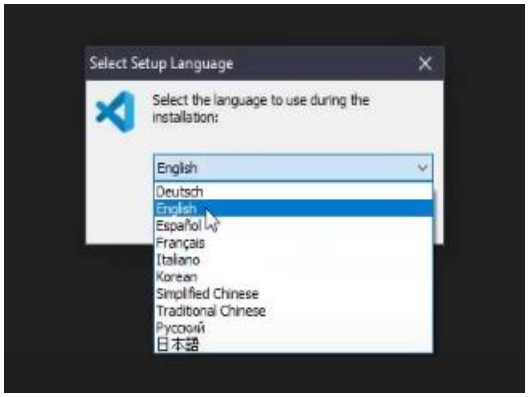
4.1. Selecione o arquivo;



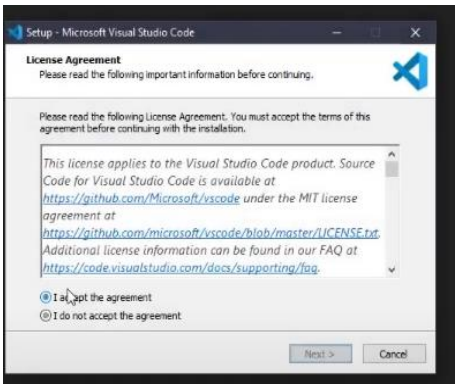
4.2. Conceda as permissões solicitadas;



4.3. Selecione o idioma.

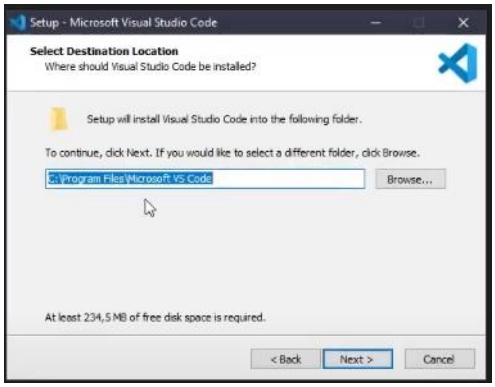


4.4. Aceite os termos;

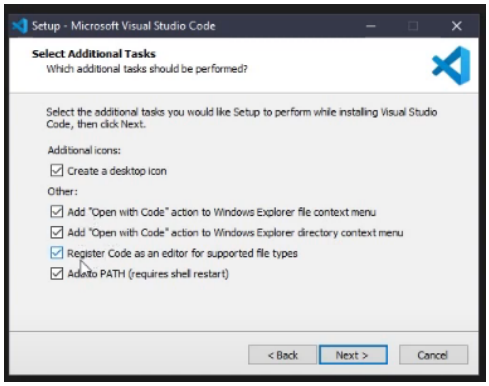


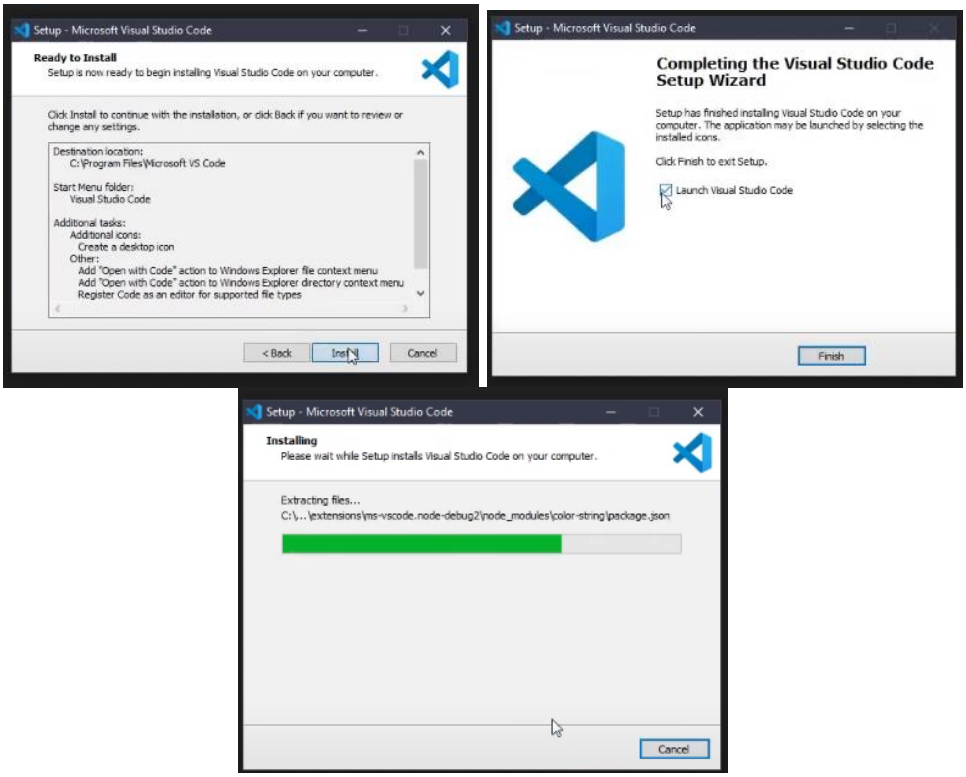
4.7. Instale;

4.5. Altere o diretório, caso desejado;

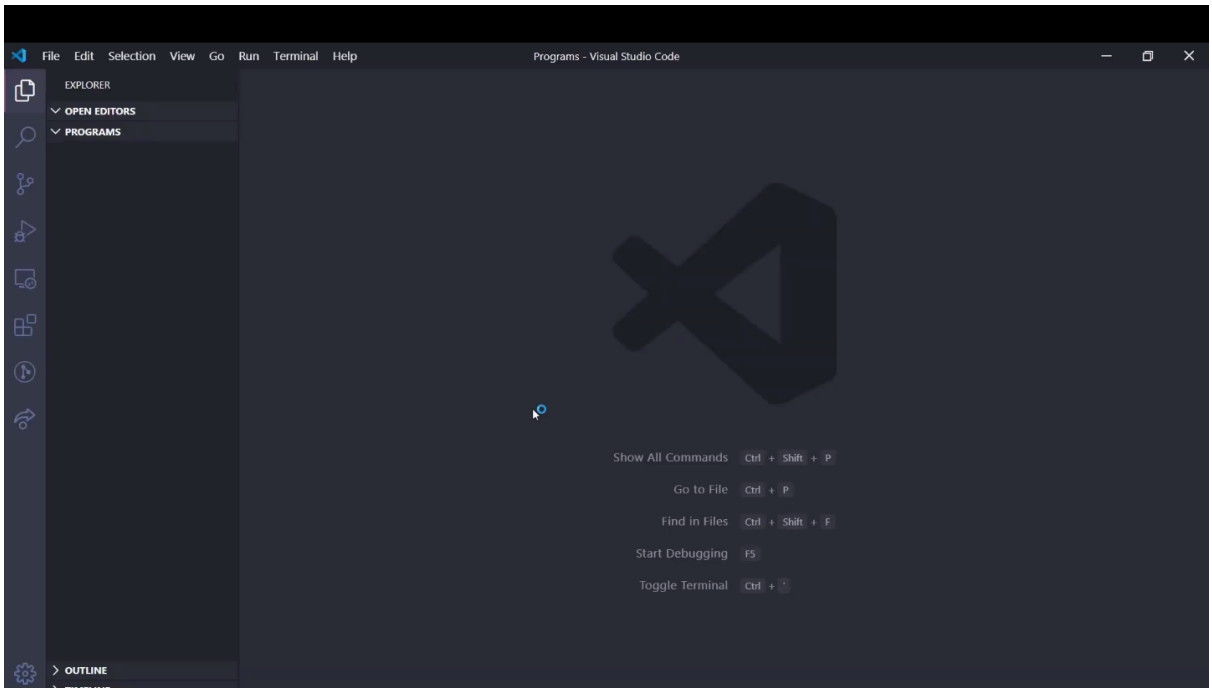


4.6. Selecione toda as opções e next;



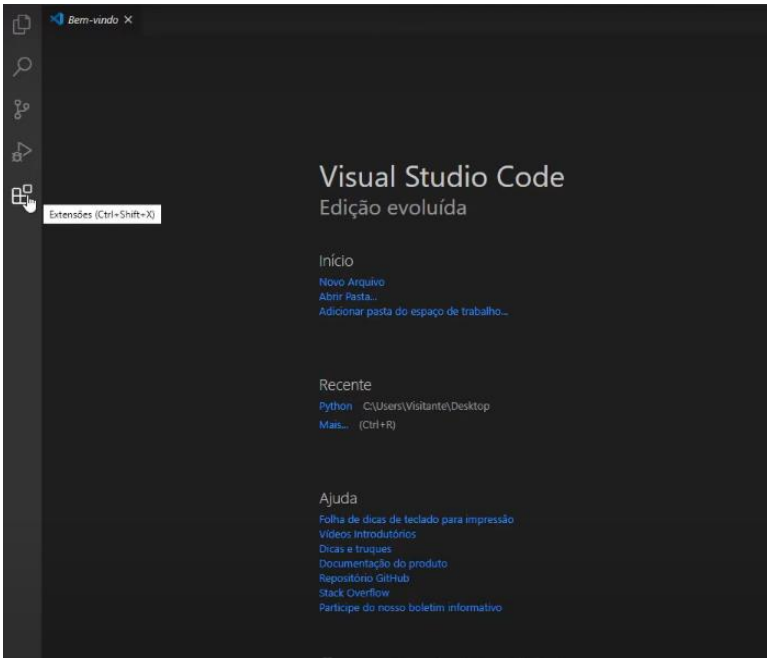


5. Interface já instalada:

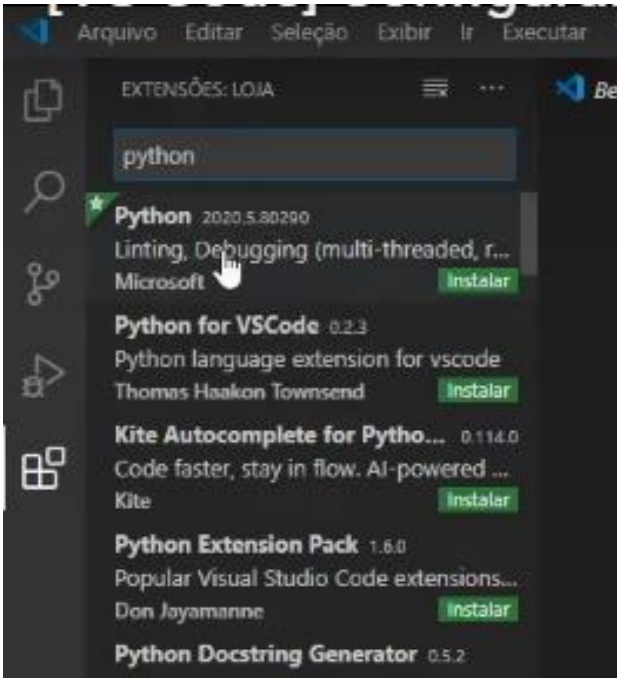


CONFIGURANDO O VISUAL STUDIO CODE PARA PROGRAMAR EM PYTHON

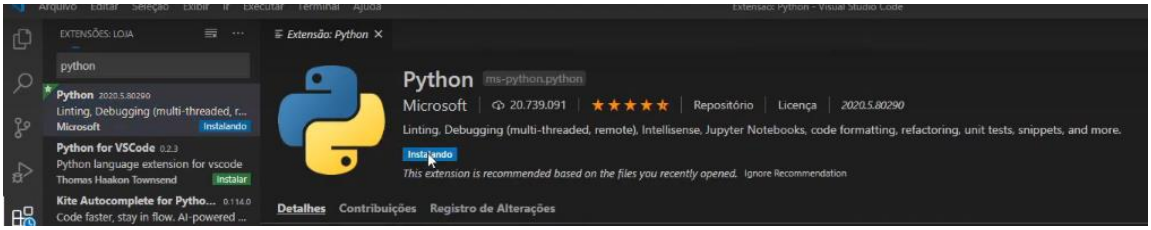
1. Abra o programa e na lateral esquerda, selecione a o último ícone, de extensões ou pelo atalho (ctrl + shift + x);



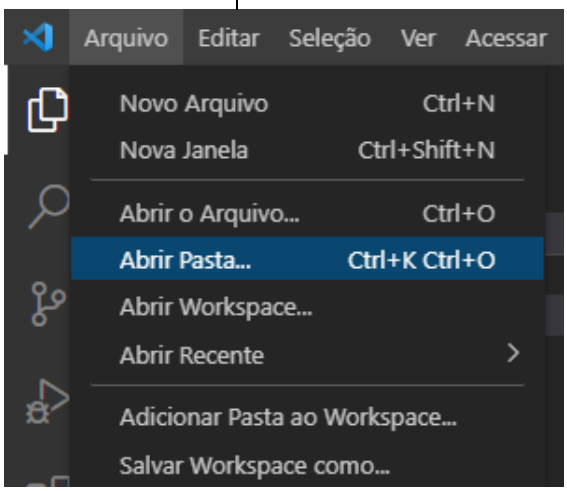
2. Na caixa de buscas, pesquise por Python, e selecione a primeira opção;



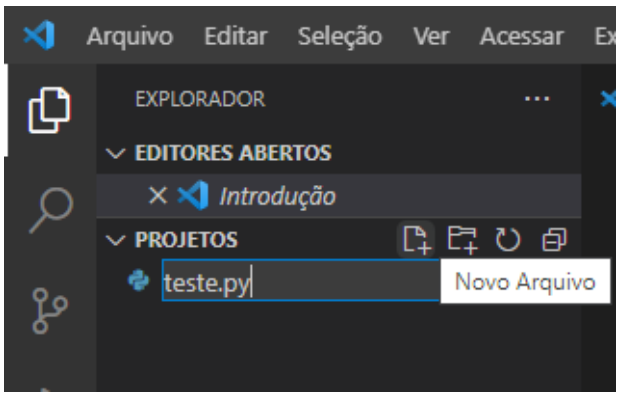
3. Instale;



Primeiramente, é necessário escolher uma pasta previamente para salvar o código que será desenvolvido. Nesse caso, é necessário escolher a opção “Abrir Pasta” ou acionar o atalho “Ctrl+K Ctrl+O”.



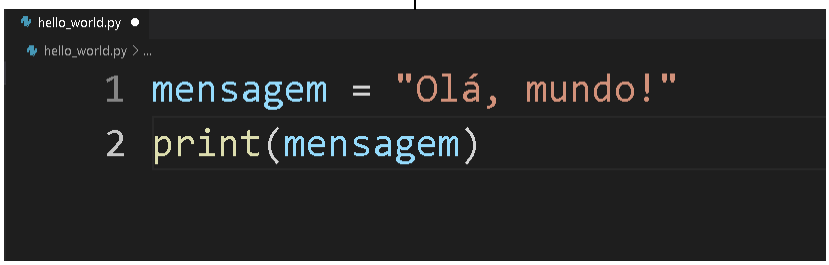
Posteriormente, é necessário adicionar um arquivo que servirá de base para desenvolver o código na opção “Novo Arquivo”, escolhendo um nome e uma extensão — nesse caso, a extensão será ‘.py’, a qual refere-se à linguagem *python*.



Para realizar a exibição do conteúdo no terminal da IDE ou para a execução no *output*, é necessário utilizar a função *print*. Essa função tem como finalidade exibir uma determinada mensagem como uma *string* — caso o valor não seja uma *string*, será convertido em uma, independente do valor atribuído; entretanto, se nenhum valor for

passado a ela, será produzida uma linha invisível ou linha em branco (ao referir-se o terminal) composta pelo caractere nova linha ou “/n”, semelhantes à função tecla *enter* do teclado.

Nesse exemplo, a mensagem é passada como um argumento de maneira direta da função *print*. Entretanto, em uma estrutura de códigos maior e mais complexa, essa forma de informar o valor poderia tornar a leitura e manutenção do código mais difícil.



No código descrito acima, há uma variável com o valor que será exibido no método de saída, tendo como finalidade uma possível substituição da mensagem.

CAPÍTULO 2

Exercícios - Parte 1

Exercício 1: A partir da digitação da base e altura de um triângulo o programa deverá calcular sua área e exibi-la no monitor.

```
#Digitação dos dados
altura = int(input('Digite a altura do triângulo: '))
base = int(input('Digite a base do triângulo: '))

#Calculo da área
area = (base * altura)/2

#Exibição do resultado
print('A área do triângulo é:', area)

```

Exercício 2: O programa deverá pedir para você digitar o valor de um ângulo em graus e na sequência mostrar o valor do seno e cosseno deste ângulo.

```python
#importação da biblioteca de matemática
import math

#leitura do ângulo armazenando na variável
angulo = float (input("Digite o ângulo desejado:"))

#Estrutura de calculos e conversão do seno
seno = math.sin(math.radians (angulo))

#Exibição do resultado do seno
print ("O ângulo de {} tem o SENO de {:.2f}".format(angulo, seno))

#Estrutura de calculos e conversão do cosseno
cosseno = math.cos(math.radians (angulo))

#Exibição do resultado do cosseno
print ("O ângulo de {} tem o COSSENO de {:.2f}".format(angulo,cosseno))

```

Exercício 3: O programa deverá solicitar a digitação de suas 4 notas bimestrais, feito isso deverá calcular e exibir a sua média final (média aritmética entre as 4 notas). Feito isso deverá também mostrar as mensagens: “Você está aprovado!”, “Você está reprovado!” ou “Você está de exame” de acordo com o seguinte critério:
Média final maior ou igual a seis o aluno esta aprovado, menor que três reprovado, entre 3 e 6 de exame.

```python
#Digitação das notas, armazenando em variaveis
nota1 = float( input("Digite a nota do 1º bimestre: ") )
nota2 = float( input("Digite a nota do 2º bimestre: ") )
nota3 = float( input("Digite a nota do 3º bimestre: ") )
nota4 = float( input("Digite a nota do 4º bimestre: ") )

# Cálculo da média final
media = (nota1 + nota2 + nota3 + nota4) / 4

#Exibição da média
print("A média final é: ",media)

# Laço de repetição para analisar se o aluno está reprovado,
# aprovado ou de exame
if (media>=6):
    print("Você está aprovado!")
elif (media<3):
    print("Você está reprovado!")
else :
    print("Você está de exame!")
```

Exercício 4: O programa deverá nos solicitar a digitação de dois números e um caractere, sendo que este poderá ser “+”, “-”, “” ou “/”. Mediante o caractere digitado fazer o respectivo cálculo e exibir o resultado, se o caractere não corresponder a nenhum dos 4 caracteres em questão exibir mensagem de erro. Este programa obrigatoriamente deverá ser feito usando um ninho de ifs.

Exercício 5: Idem ao exercício anterior, porém agora a solução deverá ser desenvolvida usando um switch-case.

Exercícios - Parte 2

Exercício 1: O programa deve começar nos solicitando a digitação de um número, inteiro e positivo, a partir de então o programa deverá exibir na tela a tabuada deste número. A entrada de dados não deverá ser validada, vamos acreditar que o usuário sempre digitará um valor devido.

```python
# Entrada de dados
num = int(input('Digite um numero inteiro e positivo: '))

# Estrutura de repetição para exibir o resultado
for c in range (1, 11):
    print('{} x {:2} = {}'.format(num, c, num*c))

```

Exercício 2: O programa deverá exibir na tela os “n” primeiros termos da série: 2, 5, 10, 17, 26... onde o valor de “n” deverá ser inicialmente digitado. Em tempo esclareço que tal série tem como termo geral (x2 + 1) onde x = {1, 2, 3, 4, 5 ...}.

```python
#Pedido de valor
numero = int(input('digite um valor: '))
termo = []

#Estrutura de repetição
for termo in range(numero):
    x =  ((pow(termo,2)) + 1)
    print('X: ',termo, '| F(x): ', x)

```

Exercício 3: Temos aqui um clássico, o programa deverá listar no vídeo os termos da série de Fibonacci (1, 1, 2, 3, 5, 8, 13, 21 ...) menores que 1000.

```python
#Designer
print ('Sequência de Fibonacci' )
print ('__'*30)

# Declaração de variavéis
t1 = 0
t2 = 1

# Exibição dos dois primeiros números
print ('__'*30)
print ('{} -> {}'.format(t1,t2), end=' ')

# Declaração de variavél que auxiliará como contador
cont = 3

# Estrutura de repetição para exibir o resultado
while cont <= 999:
    t3 = t1 + t2
    print ('-> {}'.format(t3), end=' ')
    t1 = t2
    t2 = t3
    cont +=1
print (' ->FIM')
```

Exercício 4: Entrar com dois valores via teclado, onde o segundo deverá ser maior que o primeiro. Caso contrário solicitar novamente a digitação do segundo valor, o que deve ser repetido até que o usuário atenda a condição definida.

```python
#Pedido dos valores
valor1 = int(input('digite o primeiro valor: '))
valor2 = int(input('digite o segundo valor: '))

#Estrutura de repetição
while valor2 < valor1 :
    valor2 = int(input('digite o segundo valor: '))

#Exibição na tela
print('Primeiro Valor: ', valor1)
print('Segundo Valor: ', valor2)

```

Exercício 5: Entrar via teclado com o sexo de determinado usuário, aceitar somente “F” ou “M” como respostas válidas, caso o valor digitado seja indevido repetir o processo até que se digite um dos dois caracteres válidos.

```python
#Armazenando a digitação na variável sexo
sexo = input("digite seu sexo -- (M) masculino (F) feminino: ")

#Estrutura de repetição para conferir se foi digitado certo
while True:
  if sexo != 'M' and sexo != 'F':
    print("resposta invalida")
    sexo = input("digite seu sexo -- (M) masculino (F) feminino: ")
  else:
    break

```

Exercício 6: O programa deverá nos permitir a digitação de um número inteiro e positivo, essa entrada deverá ser repetida até que o usuário satisfaça a condição. Feito isso o programa deverá calcular e exibir o fatorial deste número. Ao fim questionar se desejamos continuar ou não e essa entrada só deverá aceitar como resposta “S” ou “N” e a pergunta deverá ser repetida até que o usuário responda corretamente. Se a resposta for “S” então deveremos voltar ao início do programa e repetir tudo novamente, caso contrário o programa deverá ser encerrado.

```python
#enquanto o loop for igual a S, a consulta continua
continua = "s"
while(continua == "s"):
    n = int(input ('digite um numero positivo e inteiro: '))
    #Estrutura de repetição para verificar se o numero é inteiro e positivo
    if not n >= 0:
        print('o numero deve ser inteiro e positivo ')
        n = int(input("Digite um valor que seja aceito: "))
    #Estrutura para realizar o fatorial
    valida = True
    fat = 1
    i = 2
    while i <= n:
        fat = fat*i
        i = i + 1
    #Exibição do resultado na tela
    print("O valor de %d! eh =" %n, fat)
    continua = str(input("Deseja realizar uma nova consulta? ")).lower().strip()
```

Exercício 7: O programa deverá nos permitir digitar e armazenar dez números na memória do computador. Feito isso exibir os valores digitados em tela na ordem inversa à da digitação.

```python
#Declaração das variáveis
num = []
a = 1

#Laço para criar os numéros
for i in range (10):
    num.append(input('Digite o número: '))
    a = a +1

#Imprime na tela a mensagem da ordem inversa
    print('Ordem Inversa: ')

#Laço para imprimir na tela a ordem inversa
    for x in reversed(num):
        print(x)

```

Exercício 8: O programa deverá nos permitir digitar e armazenar dez números na memória do computador. Feito isso criar um laço capaz de calcular a somatória desses 10 valores e então exibir a média desses valores.

```python
#Declaração de variáveis
values = []
soma = 0
media = 0

#Laço de repetição para alimentação do vetor
for i in range(10):
    values.append(input('Número... '))

#Laço de repetição para calcular a soma dos números digitados

for i in range(10):
    soma += int(values[i])
    media = (soma / 10)

#Imprime na tela o resultado da média dos valores
print(media)
```

Exercício 9: O programa deverá nos permitir digitar e armazenar dez números na memória do computador. Feito isso criar um laço capaz de identificar o maior e o menor dos números digitados e exibi-los ao final.

```python
#Declaração de variáveis
listanum = []
maior = 0
menor = 0

#Estrutura de repetição para a entrada de dados
for c in range(0, 10):
    listanum.append(int(input(f'Digite um valor para a posição {c}')))
    #Estrutura de repetição para a verificação do maior e menor número
    if c == 0:
        maior = menor = listanum[c]
    else:
        if listanum[c] > maior:
            maior = listanum[c]
        if listanum[c] < menor:
            menor = listanum[c]

#Exibição dos resultados
print(f'O menor valor digitado foi: {maior}')
print(f'O menor valor digitado foi: {menor}')

```

Exercício 10: O programa deverá nos permitir digitar e armazenar o nome e idade de dez pessoas. Feito isso deverá nos solicitar a digitação de um nome e então proceder a pesquisa e informar a idade do sujeito pesquisado caso ele se encontre armazenado, caso contrário informar o fato através de mensagem “Pessoa não localizada”, ao final verificar se nova consulta é desejada, validar a resposta do usuário no sentido de só aceitar “S” ou “N”. Obviamente que no caso de nova consulta a digitação dos dez nomes/idades não deve ser repetido.

```python
#percorre array e verifica se o nome existe, se existir,
# cai no return no for e volta uma string com a frase 'nomeUSer Ano(s)',
# caso n tenha o nome, retorna  'Pessoa não localizada'
def verificaNome(nome):
    for x in range(0, 10):
        for y in range(0, 1):
            if matrizTb[x][y] == nome:
                return str(matrizTb[x][1] + " Ano(s)")

    return "Pessoa não localizada"

matrizTb = []
linhatt = []

#pula a linha
print()

#array com 10 posições
for x in range(0, 10):
    linhatt = []
    print("----- Pessoa {} ----- ".format((x+1)))
    print()

    for y in range(0, 2):
        # valorVerdadeito if condicao else valorFalso
        # ternario define qual valor vai receber e atribuna na matriz
        #   valor = ( Se y = 0 -> nome )
        #    valor = ( Se y = 1 -> idade )
        valor = str(input("Informe um nome: ")) if y == 0 else str(int(input("Informe a idade: ")))
        linhatt.append( valor )

    matrizTb.append( linhatt )

#enquanto o loop for igual a S, continua consultando
continua = "s"
while(continua == "s"):
    nome = str(input("Pesquise por um nome: "))
    print(verificaNome(nome))

    valida = True

    while valida:
        continua = str(input("Deseja realizar uma nova consulta?")).lower().strip()
        if(continua == "s" or continua == "S"):
            valida = False
        else:
            valida = True
```

Exercício 11: Determinado cinema tem 20 fileiras (de 1 a 20) com 15 cadeiras (de 1 a 15) cada uma, portanto estamos falando de uma sala com 300 assentos, que deve ser reproduzida através de um array de 20 linhas por 15 colunas. O programa deve começar solicitando do usuário a digitação de seu nome, o número da fileira e cadeira em que deseja se sentar, se o assento estiver vazio reservá-lo registrando no array seu nome, caso contrário informar que o assento esta ocupado. Feito isso o programa deverá nos questionar se desejamos nova reserva, validando nossa resposta e repetindo todo o processo.

```
#cria array com a string 'vazio' e todas as posicoes e retorna o array criado
def defineCinema(ln, cl):

    cinema = []
    linhatt = []

    for x in range(0, ln):
        linhatt = []
        for y in range(0, cl):
            linhatt.append( "vazio" )
        cinema.append( linhatt )

    return cinema

#exibe as opções, para o usuário continuar ou não
def exibeMenu():
    menu = "\n1 - Nova reserva. \n4 - Finalizar."

    return menu

#percorre cada posicao do array e concatenar na variavel lugares
def exibeLugares():

    lugares = ""
    #for para rodar todo o array
    for x in range(0, linha):
```

```
        lugares += "\n"
        for y in range(0, coluna):
            lugares += "["+str((x+1))+" "+str((y+1))+"]->"+str(cinemaArray[x][y])+" | "
    return lugares

#verifica se o lugar existe na matriz (combinacao dos indexes)
def verifcaLugar(flU, clU):
    if flU > -1 and flU < linha and clU > -1 and clU < coluna:
        return True
    else:
        return False

linha = 20
coluna = 15

# atribui o retorno da funcao (array solicitado) na variavel
cinemaArray = defineCinema(linha, coluna)

#mostra o menu de opções
print(exibeMenu())
opcaoMenu = 0

#loop para resevar os lugares
while opcaoMenu != 4 :
    opcaoMenu = int(input("\nEscolha uma opcao: "))
    if opcaoMenu == 1:
        print("Escolha um lugar: ")
        #mostra os lugares
        print(exibeLugares())
        print("\n")

        flUser = int(input("Informe o numero da Fileira (fl): "))
        clUser = int(input("Informe o numero da poltrona (cl): "))
        #pega o numero da fileira e da coluna tirando 1 por causa do array
        flUser = flUser - 1
        clUser = clUser - 1
        nome = str(input("Informe seu nome: "))
        print("\n")

        #verifica se o lugar está reservado
        rtVf = verifcaLugar(flUser, clUser)
        print(rtVf)
        #se a posição estiver entre os indices da matriz, vai para a próxima verificação
        if  rtVf :
            #se o lugar estiver indicando valor == vazio, pode reservar
            if cinemaArray[flUser][clUser] == "vazio":

                cinemaArray[flUser][clUser] = nome
                print("\n ---------------------------------------------------------- \n")
                print("Seu lugar foi reservado com sucesso!!\n")
                print("\n ---------------------------------------------------------- \n")
                print(exibeLugares())
```

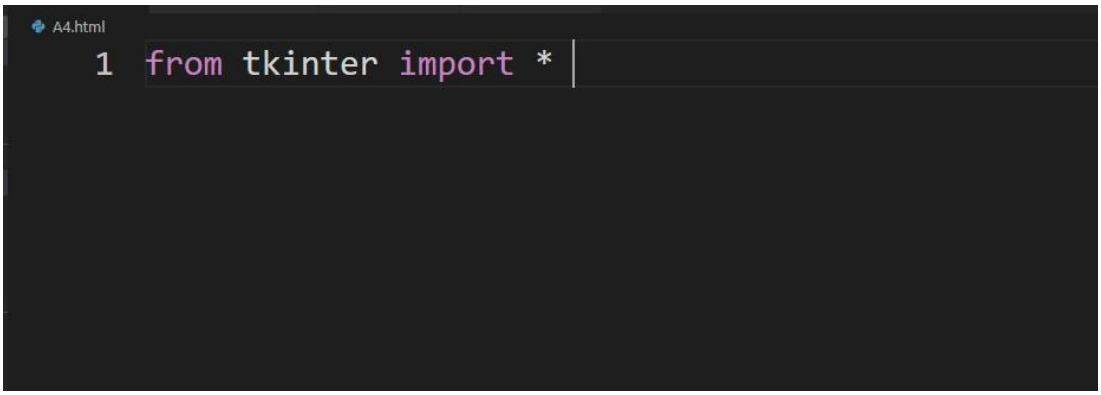
```
80                  else:
81                      print("\n ---------------------------------------------------------- \n")
82                      print("Esse lugar ja foi reservado!!\n")
83                      print("\n ---------------------------------------------------------- \n")
84              else:
85                  print("\n ---------------------------------------------------------- \n")
86                  print("Posicao invalida!!!")
87                  print("\n ---------------------------------------------------------- \n")
88
89          #mostra o menu de novo
90          print(exibeMenu())
91
92      print("\n")
93      print("Até + ^-^ Obrigado !!")
```

CAPÍTULO 3

No capítulo 3 veremos como realizar os programas com um framework, e para isso utilizaremos o Tkinter.

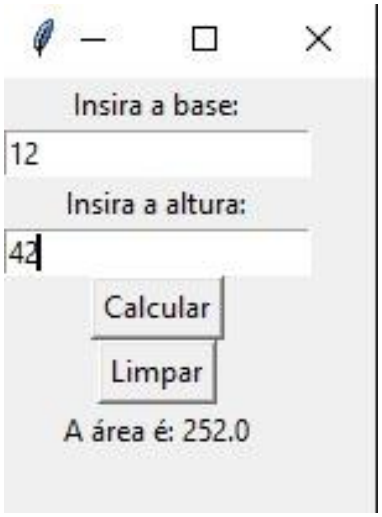
Para entender o Tkinter, é necessário conhecer o Tk GUI toolkit, uma serie de ferramentas gráficas (o que explica 'toolkit', que significa "caixa de ferramentas", em inglês) que auxiliam o desenvolvimento de interfaces voltadas ao usuário. Nesse sentindo, Tkinter é uma biblioteca nativa da linguagem Python — por ser nativa, está presente na instalação da linguagem, o que a torna melhor em comparação a outras. Ademais, demonstra-se como uma ferramenta de fácil uso.

Uma vez que a linguagem é nativa do Python, bastante executar o comando a seguir para ter acesso a biblioteca:



Exercício 1.

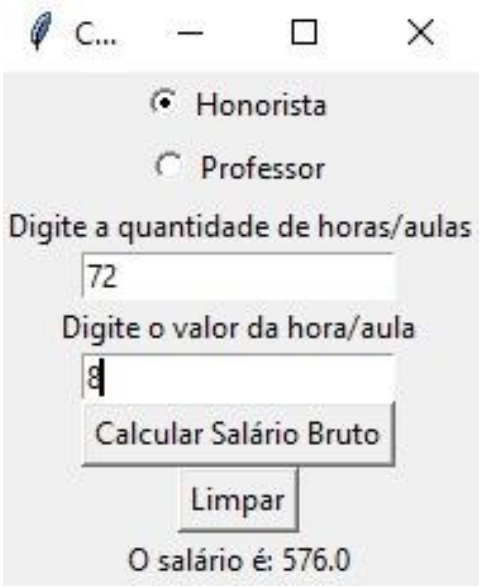
```python
from tkinter import *

#calcular a area de um triango.A = (b*h)/2

#funcoes
def calcular():
    B = float(textbox1.get())
    H = float(textbox2.get())
    R = (B*H)/2
    final.set("A área é: "+str(R))

def limpar():
   textbox1.delete(0, 'end')
   textbox2.delete(0, 'end')

#tela
menuInicial = Tk()
menuInicial.title("Cálculo de Área do Triângulo");


#variaveis
final = StringVar()

#design
label1 = Label(menuInicial,text="Insira a base:")
textbox1 = Entry(menuInicial);
label2 = Label(menuInicial,text="Insira a altura:")
textbox2 = Entry(menuInicial);
button1 = Button(menuInicial,text="Calcular",command=calcular)
button2 = Button(menuInicial,text="Limpar",command=limpar)
labelResultado = Label(menuInicial,textvariable=final)

#graphs
label1.grid()
textbox1.grid()
label2.grid()
textbox2.grid()
button1.grid()
button2.grid()
labelResultado.grid()

#loop
menuInicial.mainloop()
```

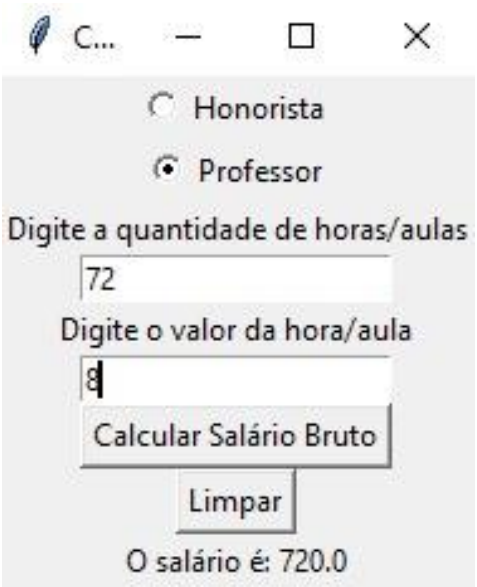


Exercício 2.

```
from tkinter import *

#calcular o salaerio
# Exercicio 02

#funcoes
def calcular():

    qtd = float(textbox1.get())
    valor = float(textbox2.get())
    sb = float()

    aux=profissao.get()

    if aux == "h":
     sb = qtd*valor
    elif aux == "p":
         sb = (qtd * valor) * 1.25

    final.set("O salário é: "+str(sb))

def limpar():
   textbox1.delete(0, 'end')
   textbox2.delete(0, 'end')

#tela
menuInicial = Tk()
menuInicial.title("Cálculo de Salário");

```

```
26  #tela
27  menuInicial = Tk()
28  menuInicial.title("Cálculo de Salário");
29
30  #variaveis
31  final = StringVar()
32  profissao=StringVar()
33
34  #design
35  rb_Honorista=Radiobutton(menuInicial, text="Honorista",value ="h",variable=profissao)
36  rb_Honorista.pack()
37  rb_Professor=Radiobutton(menuInicial, text="Professor", value ="p",variable=profissao)
38  rb_Professor.pack()
39  label1 = Label(menuInicial,text="Digite a quantidade de horas/aulas")
40  textbox1 = Entry(menuInicial);
41  label2 = Label(menuInicial,text="Digite o valor da hora/aula")
42  textbox2 = Entry(menuInicial);
43  button1 = Button(menuInicial,text="Calcular Salário Bruto",command=calcular)
44  button2 = Button(menuInicial,text="Limpar",command=limpar)
45  labelResultado = Label(menuInicial,textvariable=final)
46
47  #graphs
48  label1.pack()
49  textbox1.pack()
50  label2.pack()
51  textbox2.pack()
52  button1.pack()
53  button2.pack()
54  labelResultado.pack()
55
56  #loop
57  menuInicial.mainloop()
```

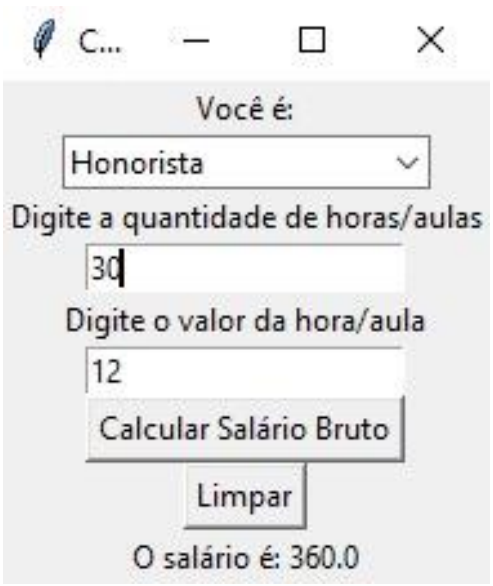


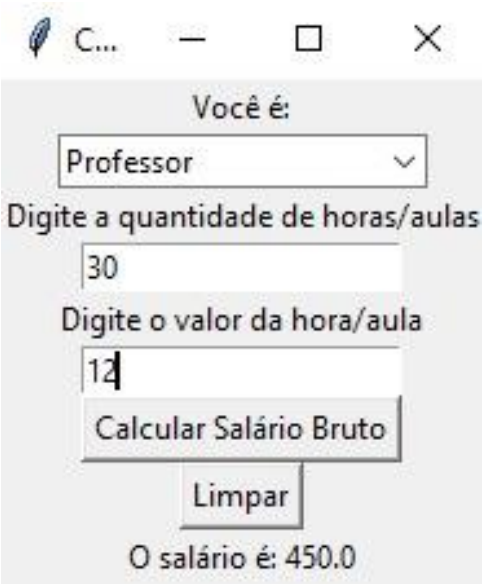


Exercício 3.

```
from tkinter import *
from tkinter import ttk
#calcular o salaerio
# Exercicio 03

#funcoes
def calcular():

    qtd = float(textbox1.get())
    valor = float(textbox2.get())
    sb = float()

    aux=cb_profissao.get()

    aux=cb_profissao.current()

    if aux == 0:
        sb = qtd*valor
    elif aux == 1:
        sb = (qtd * valor) * 1.25

    final.set("O salário é: "+str(sb))

def limpar():
   textbox1.delete(0, 'end')
   textbox2.delete(0, 'end')

#tela
menuInicial = Tk()
menuInicial.title("Cálculo de Salário");
```

```
#variaveis
final = StringVar()
profissao=StringVar()
listaprofissoes =["Honorista", "Professor"]
#design

lb_Profissao=Label(menuInicial,text= "Você é:")
lb_Profissao.pack()
cb_profissao = ttk.Combobox(menuInicial, values= listaprofissoes )
cb_profissao.pack()
label1 = Label(menuInicial,text="Digite a quantidade de horas/aulas")
textbox1 = Entry(menuInicial);
label2 = Label(menuInicial,text="Digite o valor da hora/aula")
textbox2 = Entry(menuInicial);
button1 = Button(menuInicial,text="Calcular Salário Bruto",command=calcular)
button2 = Button(menuInicial,text="Limpar",command=limpar)
labelResultado = Label(menuInicial,textvariable=final)

#graphs
label1.pack()
textbox1.pack()
label2.pack()
textbox2.pack()
button1.pack()
button2.pack()
labelResultado.pack()

#loop
menuInicial.mainloop()
```

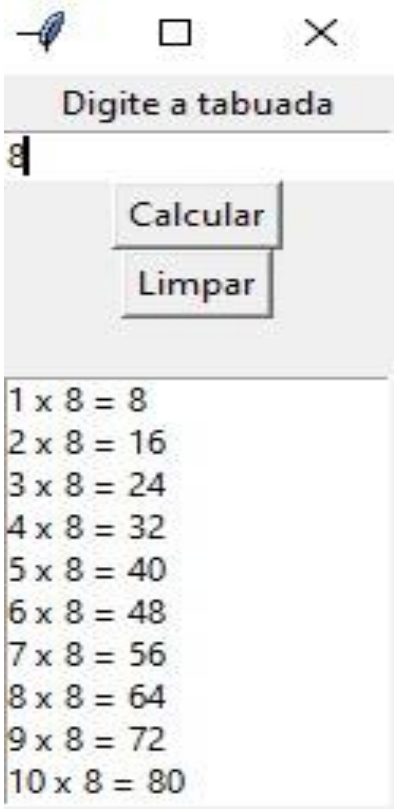




Exercício 4.

```python
from tkinter import *

#calcular tabuada
# Exercicio 04

#funcoes
def calcular():
    qtd = int(textbox1.get())

    for c in range(1, 11):
        Lista.insert (END,f'{c} x {qtd} = {qtd*c}')


def limpar():
   textbox1.delete(0, 'end')
   Lista.delete(0, 'end')

#tela
menuInicial = Tk()
menuInicial.title("Calcular tabuada");


#variaveis
final = StringVar()
profissao=StringVar()

```

```
27  #design
28  Lista = Listbox (menuInicial)
29  label1 = Label(menuInicial,text="Digite a tabuada")
30  textbox1 = Entry(menuInicial);
31  button1 = Button(menuInicial,text="Calcular",command=calcular)
32  button2 = Button(menuInicial,text="Limpar",command=limpar)
33  labelResultado = Label(menuInicial,textvariable=final)
34
35  #graphs
36  label1.pack()
37  textbox1.pack()
38  button1.pack()
39  button2.pack()
40  labelResultado.pack()
41  Lista.pack()
42
43  #loop
44  menuInicial.mainloop()
```



CAPÍTULO 4

Neste próximo capítulo veremos, com exemplo a utilização da POO (Programação Orientada a Objetos) na linguagem Python.

Exercício 1

```
from Ex1 import triangulo
import Ex1BLL
import Erro
from tkinter import *
from tkinter import messagebox

class Application:
    def __init__(self, master=None):
        self.fonte = ("Verdana", "8")

        self.container1 = Frame(master)
        self.container1["pady"] = 10
        self.container1.pack()
        self.container2 = Frame(master)
        self.container2["padx"] = 10
        self.container2["pady"] = 5
        self.container2.pack()
        self.container3 = Frame(master)
        self.container3["padx"] = 10
        self.container3["pady"] = 5
        self.container3.pack()
        self.container4 = Frame(master)
        self.container4["padx"] = 10
        self.container4["pady"] = 5
        self.container4.pack()
        self.container5 = Frame(master)
```

```
        self.container5["padx"] = 20
        self.container5["pady"] = 5
        self.container5.pack()
        self.container6 = Frame(master)
        self.container6["pady"] = 15
        self.container6.pack()

        self.titulo = Label(self.container1, text="Área do triângulo:")
        self.titulo["font"] = ("Calibri", "9", "bold")
        self.titulo.pack ()

        self.lblBase = Label(self.container2, text="Base do triângulo:",
        font=self.fonte, width=25)
        self.lblBase.pack(side=LEFT)

        self.txtBase = Entry(self.container2)
        self.txtBase["width"] = 25
        self.txtBase["font"] = self.fonte
        self.txtBase.pack(side=LEFT)

        self.lblAltura = Label(self.container3, text="Altura do triângulo:",
        font=self.fonte, width=25)
        self.lblAltura.pack(side=LEFT)
```

```
        self.txtAltura = Entry(self.container3)
        self.txtAltura["width"] = 25
        self.txtAltura["font"] = self.fonte
        self.txtAltura.pack(side=LEFT)

        self.lblArea = Label(self.container4, text="Área do triângulo:",
        font=self.fonte, width=25)
        self.lblArea.pack(side=LEFT)

        self.txtArea = Entry(self.container4)
        self.txtArea["width"] = 25
        self.txtArea["font"] = self.fonte
        self.txtArea.pack(side=LEFT)

        self.btnCalcular = Button(self.container5, text="Calcular",
        font=self.fonte, width=12)
        self.btnCalcular["command"] = self.Calcular
        self.btnCalcular.pack (side=LEFT)


        self.btnLimpar = Button(self.container5, text="limpar",
        font=self.fonte, width=12)
        self.btnLimpar["command"] = self.limpaCampos
        self.btnLimpar.pack(side=LEFT)
```

```
76          self.lblmsg = Label(self.container6, text="")
77          self.lblmsg["font"] = ("Verdana", "9", "italic")
78          self.lblmsg.pack()
79
80
81      def Calcular(self):
82          Calcular = triangulo()
83
84          Calcular.setB(self.txtBase.get().strip())
85          Calcular.setH(self.txtAltura.get().strip())
86
87          Ex1BLL.validaDados(Calcular)
88
89
90          if Erro.getErro():
91              messagebox.showerror('error', Erro.getMens())
92          else:
93              self.txtArea.insert(INSERT, Calcular.triangulo())
94
95              self.txtArea.config(state = 'disabled')
96              self.txtBase.config(state = 'disabled')
97              self.txtAltura.config(state = 'disabled')
```

```
99      def limpaCampos(self):
100
101         self.txtArea.config(state = 'normal')
102         self.txtBase.config(state = 'normal')
103         self.txtAltura.config(state = 'normal')
104
105         self.txtArea.delete(0, END)
106         self.txtBase.delete(0, END)
107         self.txtAltura.delete(0, END)
108
109 #Instância
110 root = Tk()
111
112 root.title("POO COM PYTHON :)")
113 Application(root)
114 root.mainloop()
```

Ex1.py

Ex1.py > triangulo

```python
class triangulo():

    def __init__(self):
        self.b =0
        self.h =0

    def setB(self,_b):
        self.b = _b

    def setH(self,_h):
        self.h = _h

    def getB(self):
        return self.b

    def getH(self):
        return self.h

    def triangulo(self):
        return str((float(self.b) * float(self.h)) / 2)

```

Classe Erro

Erro.py

Erro.py > ...

```python
erro = False
mens = ''

def setErro(_erro):
    global erro
    global mens

    if isinstance(_erro, str):
        erro = True
        mens = _erro
    else:
        erro = _erro

def getErro():
    return erro

def getMens():
    return mens
```

Classe BLL

Ex1BLL.py

Ex1BLL.py > validaDados

```python
import Erro


def validaDados(triangulo):

    Erro.setErro(False)

    if len(triangulo.getB()) == 0:
        Erro.setErro("A base tem que ter preenchimento obrigatório.")
        return None
    else:
        try:
            float(triangulo.getB())
        except:
            Erro.setErro("A base deve ser numérico.")
            return None

    if len(triangulo.getH())==0:
        Erro.setErro("A altura tem que ter preenchimento obrigatório.")
        return None
    else:
        try:
            float(triangulo.getH())
        except:
            Erro.setErro("A altura deve ser numérico.")
            return None
```

Exercício 2

Ex2.py

```python
class Salario:
    def _init_(self):
        self.qtd = "0"
        self.valor = "0"

    def setQtd(self, _qtd):
        self.qtd = _qtd

    def setVal(self, _valor):
        self.valor = _valor

    def getQtd(self):
        return self.qtd

    def getVal(self):
        return self.valor

    def getSalarioBruto(self):
        return str((int(self.qtd) * int(self.valor)) / 2)


```

Classe BLL

Ex2BLL.py

```python
import Erro

def validaDados(Salario):

    Erro.setErro(False)

    if len(Salario.getQtd()) == 0:
        Erro.setErro("O campo QUANTIDADE DE HORAS é de preenchimento obrigatório...")
        return None
    else:
        try:
            int(Salario.getQtd())
        except:
            Erro.setErro("O campo QUANTIDADE DE HORAS deve ser numérico...")
            return None

    if int(Salario.getQtd()) <= 0:
        Erro.setErro("O campo QUANTIDADE DE HORAS deve ser maior que zero.");
        return None
```

```
21      if len(Salario.getVal()) == 0:
22          Erro.setErro("O campo VALOR DA HORA é de preenchimento obrigatório...")
23          return None
24      else:
25          try:
26              int(Salario.getVal())
27          except:
28              Erro.setErro("O campo VALOR DA HORA deve ser numérico...")
29              return None
30
31      if int(Salario.getVal()) <= 0:
32          Erro.setErro("O campo VALOR DA HORA deve ser maior que zero.");
33          return None
```

Classe IHM

```
18  class Ui_Dialog(object):
19      def setupUi(self, Dialog):
20          Dialog.setObjectName("Dialog")
21          Dialog.resize(458, 312)
22          self.calcButton = QtWidgets.QPushButton(Dialog)
23          self.calcButton.setGeometry(QtCore.QRect(30, 150, 161, 61))
24          font = QtGui.QFont()
25          font.setPointSize(18)
26          self.calcButton.setFont(font)
27          self.calcButton.setObjectName("calcButton")
28          self.cleanButton = QtWidgets.QPushButton(Dialog)
29          self.cleanButton.setGeometry(QtCore.QRect(260, 150, 161, 61))
30          font = QtGui.QFont()
31          font.setPointSize(16)
32          self.cleanButton.setFont(font)
33          self.cleanButton.setObjectName("cleanButton")
34          self.label = QtWidgets.QLabel(Dialog)
35          self.label.setGeometry(QtCore.QRect(30, 240, 161, 31))
36          font = QtGui.QFont()
37          font.setPointSize(20)
38          self.label.setFont(font)
39          self.label.setObjectName("label")
40          self.label_2 = QtWidgets.QLabel(Dialog)
41          self.label_2.setGeometry(QtCore.QRect(30, 40, 321, 31))
42          font = QtGui.QFont()
43          font.setPointSize(20)
44          self.label_2.setFont(font)
45          self.label_2.setObjectName("label_2")
46          self.label_3 = QtWidgets.QLabel(Dialog)
47          self.label_3.setGeometry(QtCore.QRect(30, 100, 171, 31))
48          font = QtGui.QFont()
49          font.setPointSize(20)
50          self.label_3.setFont(font)
```

```
        self.label_3.setObjectName("label_3")
        self.qtdH = QtWidgets.QLineEdit(Dialog)
        self.qtdH.setGeometry(QtCore.QRect(350, 20, 101, 51))
        self.qtdH.setText("")
        self.qtdH.setObjectName("qtdH")
        self.valH = QtWidgets.QLineEdit(Dialog)
        self.valH.setGeometry(QtCore.QRect(350, 90, 101, 51))
        self.valH.setText("")
        self.valH.setObjectName("valH")
        self.ValSalario = QtWidgets.QLineEdit(Dialog)
        self.ValSalario.setGeometry(QtCore.QRect(350, 230, 101, 51))
        self.ValSalario.setText("")
        self.ValSalario.setObjectName("ValSalario")

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

        self.calcButton.clicked.connect(self.Calcular)
        self.cleanButton.clicked.connect(self.Limpar)
```

```
    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
        self.calcButton.setText(_translate("Dialog", "Calcular"))
        self.cleanButton.setText(_translate("Dialog", "Limpar"))
        self.label.setText(_translate("Dialog", "Salário Bruto:"))
        self.label_2.setText(_translate("Dialog", "Qtd de horas trabalhadas:"))
        self.label_3.setText(_translate("Dialog", "Valor da Hora:"))

    def Calcular(self):
        msg = QtWidgets.QMessageBox()
        salario = Salario()

        salario.setQtd(self.qtdH.text())
        salario.setVal(self.valH.text())

        Ex2BLL.validaDados(salario)
        if Erro.getErro():
            msg.setText(Erro.getMens())
            msg.exec()
        else:
            self.ValSalario.setText(salario.getSalarioBruto())
```

```
94                  self.qtdH.setEnabled(False)
95                  self.valH.setEnabled(False)
96                  self.ValSalario.setEnabled(False)
97
98
99
100     def Limpar(self):
101         self.qtdH.setText("")
102         self.valH.setText("")
103         self.ValSalario.setText("")
104
105         self.qtdH.setEnabled(True)
106         self.valH.setEnabled(True)
107         self.ValSalario.setEnabled(True)
108
109 if __name__ == "__main__":
110     import sys
111     app = QtWidgets.QApplication(sys.argv)
112     Dialog = QtWidgets.QDialog()
113     ui = Ui_Dialog()
114     ui.setupUi(Dialog)
115     Dialog.show()
116     sys.exit(app.exec_())
117
```

Exercício 3

Equacao2G_inicial.py

```
import Erro
from Equacao2g import Equacao2G
import Equacao2GBLL
from tkinter import *
from tkinter import messagebox

class Application:
    def __init__(self, master=None):
        self.fonte = ("Verdana", "8")

        self.container1 = Frame(master)
        self.container1["pady"] = 10
        self.container1.pack()
        self.container2 = Frame(master)
        self.container2["padx"] = 20
        self.container2["pady"] = 5
        self.container2.pack()
        self.container3 = Frame(master)
        self.container3["padx"] = 20
        self.container3["pady"] = 5
        self.container3.pack()
        self.container4 = Frame(master)
        self.container4["padx"] = 20
        self.container4["pady"] = 5
        self.container4.pack()
        self.container5 = Frame(master)
```

```
        self.container5["padx"] = 20
        self.container5["pady"] = 5
        self.container5.pack()
        self.container6 = Frame(master)
        self.container6["padx"] = 20
        self.container6["pady"] = 5
        self.container6.pack()
        self.container7 = Frame(master)
        self.container7["padx"] = 20
        self.container7["pady"] = 10
        self.container7.pack()
        self.container8 = Frame(master)
        self.container8["pady"] = 15
        self.container8.pack()

        self.titulo = Label(self.container1, text="Equação de 2º grau :")
        self.titulo["font"] = ("Calibri", "9", "bold")
        self.titulo.pack ()

        self.lblValorA = Label(self.container2, text="Valor a:",
        font=self.fonte, width=10)
        self.lblValorA.pack(side=LEFT)
```

```
        self.txtValorA = Entry(self.container2)
        self.txtValorA["width"] = 25
        self.txtValorA["font"] = self.fonte
        self.txtValorA.pack(side=LEFT)

        self.lblValorB = Label(self.container3, text="Valor b:",
        font=self.fonte, width=10)
        self.lblValorB.pack(side=LEFT)

        self.txtValorB = Entry(self.container3)
        self.txtValorB["width"] = 25
        self.txtValorB["font"] = self.fonte
        self.txtValorB.pack(side=LEFT)

        self.lblValorC = Label(self.container4, text="Valor c:",
        font=self.fonte, width=10)
        self.lblValorC.pack(side=LEFT)

        self.txtValorC = Entry(self.container4)
        self.txtValorC["width"] = 25
        self.txtValorC["font"] = self.fonte
        self.txtValorC.pack(side=LEFT)
```

```
73          self.lblValorX1= Label(self.container5, text="Valor X1:",
74          font=self.fonte, width=10)
75          self.lblValorX1.pack(side=LEFT)
76
77          self.txtValorX1 = Entry(self.container5)
78          self.txtValorX1["width"] = 10
79          self.txtValorX1["font"] = self.fonte
80          self.txtValorX1.pack(side=LEFT)
81
82          self.lblValorX2= Label(self.container6, text="Valor X2:",
83          font=self.fonte, width=10)
84          self.lblValorX2.pack(side=LEFT)
85
86          self.txtValorX2 = Entry(self.container6)
87          self.txtValorX2["width"] = 10
88          self.txtValorX2["font"] = self.fonte
89          self.txtValorX2.pack(side=LEFT)
90
91
92          self.btnCalcular = Button(self.container7, text="Calcular",
93          font=self.fonte, width=12)
94          self.btnCalcular["command"] = self.Calcular
95          self.btnCalcular.pack (side=LEFT)
```

```
97
98          self.btnLimpar = Button(self.container7, text="limpar",
99          font=self.fonte, width=12)
100         self.btnLimpar["command"] = self.limpaCampos
101         self.btnLimpar.pack(side=LEFT)
102
103         self.lblmsg = Label(self.container8, text="")
104         self.lblmsg["font"] = ("Verdana", "9", "italic")
105         self.lblmsg.pack()
106
107
108
109     def Calcular(self):
110         equacao2G = Equacao2G()
111
112         equacao2G.set_a(self.txtValorA.get().strip())
113         equacao2G.set_b(self.txtValorB.get().strip())
114         equacao2G.set_c(self.txtValorC.get().strip())
115
116         Equacao2GBLL.validaDados(equacao2G)
117
118         if(Erro.getErro()):
119             messagebox.showerror('error', Erro.getMens())
120
```

```
        else:

            self.txtValorX1.insert(INSERT, equacao2G.get_x1())
            self.txtValorX2.insert(INSERT, equacao2G.get_x2())


            self.txtValorA.config(state = 'disabled')
            self.txtValorB.config(state = 'disabled')
            self.txtValorC.config(state = 'disabled')

    def limpaCampos(self):

        self.txtValorA.config(state = 'normal')
        self.txtValorB.config(state = 'normal')
        self.txtValorC.config(state = 'normal')

        self.txtValorA.delete(0, END)
        self.txtValorB.delete(0, END)
        self.txtValorC.delete(0, END)
        self.txtValorX1.delete(0, END)
        self.txtValorX2.delete(0, END)
```

```
#Instância
root = Tk()

root.title("POO COM PYTHON :)")
Application(root)
root.mainloop()

```

Equacao2g.py

```python
from math import *

class Equacao2G:
    def _init_(self):
        self.a = 0
        self.b = 0
        self.c = 0

    def get_a(self):
        return self.__a
    def get_b(self):
        return self.__b
    def get_c(self):
        return self.__c

    def set_a(self, valor_a):
        self.__a = valor_a
    def set_b(self, valor_b):
        self.__b = valor_b
    def set_c(self, valor_c):
        self.__c = valor_c
```

```python
    def get_delta(self):
        auxa = float(self.__a)
        auxb = float(self.__b)
        auxc = float(self.__c)
        calc = float((auxb * auxb) - (4 * auxa * auxc))
        return str(calc)

    def get_x1(self):
        auxa = float(self.__a)
        auxb = float(self.__b)
        auxc = float(self.__c)
        auxdelta = float(self.get_delta())
        calc = float((-auxb + sqrt(auxdelta)) / (2 * auxa))
        return str(calc)

    def get_x2(self):
        auxa = float(self.__a)
        auxb = float(self.__b)
        auxc = float(self.__c)
        auxdelta = float(self.get_delta())
        calc = float((-auxb - sqrt(auxdelta)) / (2 * auxa))
        return str(calc)
```

Classe BLL

Equacao2GBLL.py

```python
import Erro

def validaDados(equacao2G):
    Erro.setErro(False)

    if len(equacao2G.get_a()) == 0:
        Erro.setErro("O valor de A é de preenchimento obrigatório...")
        return None
    else:
        try:
            float(equacao2G.get_a())
        except:
            Erro.setErro("O valor de A deve ser numérico...")
            return None

    if len(equacao2G.get_b()) == 0:
        Erro.setErro("O valor de B é de preenchimento obrigatório...")
        return None
    else:
        try:
            float(equacao2G.get_b())
        except:
            Erro.setErro("O valor de B deve ser numérico...")
            return None

    if len(equacao2G.get_c()) == 0:
        Erro.setErro("O valor de C é de preenchimento obrigatório...")
        return None
    else:
        try:
            float(equacao2G.get_c())
        except:
            Erro.setErro("O valor de C deve ser numérico...")
            return None

    if float(equacao2G.get_delta()) < 0:
        Erro.setErro("Delta negativo, não existem raízes reais para esta equação..")

```

Exercício 4

Livros.py

Livros.py > Livro

```python
class Livro(object):

    def __init__(self, codigo = "", titulo = "", autor = "", editoria = "", ano = ""):
        self.livros = []
        self.codigo = codigo
        self.titulo = titulo
        self.autor = autor
        self.editoria = editoria
        self.ano = ano


    def insertLivro(self):

        for x in self.livros:
            if( x[0] == self.codigo ):
                return "Código já cadastrado"

        self.livros.append([self.codigo, self.titulo, self.autor, self.editoria, self.ano])
        return "Livro salvo com sucesso";


    def buscaLivro(self, codigoLv):
        texto = ""
        for x in self.livros:
            texto += "x -> " + str(x) + " cod -> " + str(codigoLv)
            if( x[0] == codigoLv ):
                self.codigo = x[0]
                self.titulo = x[1]
                self.autor = x[2]
                self.editoria = x[3]
                self.ano = x[4]

                return ""

        return "Código não encontrado"
```

casdastroLivro.py

```python
from Livros import Livro
from tkinter import *

class Application:
    def __init__(self, master=None):
        self.fonte = ("Verdana", "8")

        self.container1 = Frame(master)
        self.container1["pady"] = 10
        self.container1.pack()
        self.container2 = Frame(master)
        self.container2["padx"] = 20
        self.container2["pady"] = 5
        self.container2.pack()
        self.container3 = Frame(master)
        self.container3["padx"] = 20
        self.container3["pady"] = 5
        self.container3.pack()
        self.container4 = Frame(master)
        self.container4["padx"] = 20
        self.container4["pady"] = 5
        self.container4.pack()
        self.container5 = Frame(master)
        self.container5["padx"] = 20
        self.container5["pady"] = 5
        self.container5.pack()
```

```
        self.container6 = Frame(master)
        self.container6["padx"] = 20
        self.container6["pady"] = 5
        self.container6.pack()
        self.container8 = Frame(master)
        self.container8["padx"] = 20
        self.container8["pady"] = 10
        self.container8.pack()
        self.container9 = Frame(master)
        self.container9["pady"] = 15
        self.container9.pack()

        self.titulo = Label(self.container1, text="Cadastro de Livros :")
        self.titulo["font"] = ("Calibri", "9", "bold")
        self.titulo.pack ()

        self.lblcodigo = Label(self.container2,
        text="Código:", font=self.fonte, width=10)
        self.lblcodigo.pack(side=LEFT)

        self.txtcodigo = Entry(self.container2)
        self.txtcodigo["width"] = 10
        self.txtcodigo["font"] = self.fonte
        self.txtcodigo.pack(side=LEFT)
```

```
        self.btnBuscar = Button(self.container2, text="Buscar",
        font=self.fonte, width=10)
        self.btnBuscar["command"] = self.buscarUsuario
        self.btnBuscar.pack(side=RIGHT)

        self.lbltituloLv = Label(self.container3, text="Titulo:",
        font=self.fonte, width=10)
        self.lbltituloLv.pack(side=LEFT)

        self.txttituloLv = Entry(self.container3)
        self.txttituloLv["width"] = 25
        self.txttituloLv["font"] = self.fonte
        self.txttituloLv.pack(side=LEFT)

        self.lblautor = Label(self.container4, text="Autor:",
        font=self.fonte, width=10)
        self.lblautor.pack(side=LEFT)

        self.txtautor = Entry(self.container4)
        self.txtautor["width"] = 25
        self.txtautor["font"] = self.fonte
        self.txtautor.pack(side=LEFT)

        self.lbleditoria= Label(self.container5, text="Editoria:",
        font=self.fonte, width=10)
        self.lbleditoria.pack(side=LEFT)
```

```
        self.txteditoria = Entry(self.container5)
        self.txteditoria["width"] = 25
        self.txteditoria["font"] = self.fonte
        self.txteditoria.pack(side=LEFT)

        self.lblano= Label(self.container6, text="Ano:",
        font=self.fonte, width=10)
        self.lblano.pack(side=LEFT)

        self.txtano = Entry(self.container6)
        self.txtano["width"] = 10
        self.txtano["font"] = self.fonte
        self.txtano.pack(side=LEFT)


        self.btnSalvar = Button(self.container8, text="Salvar",
        font=self.fonte, width=12)
        self.btnSalvar["command"] = self.salvaLivro
        self.btnSalvar.pack (side=LEFT)


        self.btnLimpar = Button(self.container8, text="limpar",
        font=self.fonte, width=12)
        self.btnLimpar["command"] = self.limpaCampos
        self.btnLimpar.pack(side=LEFT)
```

```
        self.lblmsg = Label(self.container9, text="")
        self.lblmsg["font"] = ("Verdana", "9", "italic")
        self.lblmsg.pack()

    def salvaLivro(self):
        #Valida dados
        codigo = self.txtcodigo.get().strip()
        titulo = self.txttituloLv.get().strip()
        autor = self.txtautor.get().strip()
        editoria = self.txteditoria.get().strip()
        ano = self.txtano.get().strip()

        if(codigo == "" or titulo == "" or autor == "" or editoria == "" or ano == ""):
            self.lblmsg["text"] = "Todos os campos são obrigatórios!"
            return
        else:
            try:
                ano = int(ano)
                if ano <= 0:
                    self.lblmsg["text"] = "Ano inválido!"
                    return
                else:
                    ano = str(ano) #a class espera receber uma string no parametro
            except:
                self.lblmsg["text"] = "Informe um ano válido!"
                return
```

```
132         #define parametros
133         livro.codigo = codigo
134         livro.titulo = titulo
135         livro.autor = autor
136         livro.editoria = editoria
137         livro.ano = ano
138 
139         #chama metodo e atribui retorno
140         self.lblmsg["text"] = livro.insertLivro()
141 
142         #limpa campos
143         self.txtcodigo.delete(0, END)
144         self.txttituloLv.delete(0, END)
145         self.txtautor.delete(0, END)
146         self.txteditoria.delete(0, END)
147         self.txtano.delete(0, END)
```

```
150     def buscarUsuario(self):
151 
152         codigolv = self.txtcodigo.get()
153 
154         self.lblmsg["text"] = livro.buscaLivro(codigolv)
155 
156         self.txtcodigo.delete(0, END)
157         self.txtcodigo.insert(INSERT, livro.codigo)
158 
159         self.txttituloLv.delete(0, END)
160         self.txttituloLv.insert(INSERT, livro.titulo)
161 
162         self.txtautor.delete(0, END)
163         self.txtautor.insert(INSERT, livro.autor)
164 
165         self.txteditoria.delete(0, END)
166         self.txteditoria.insert(INSERT, livro.editoria)
167 
168         self.txtano.delete(0, END)
169         self.txtano.insert(INSERT, livro.ano)
```

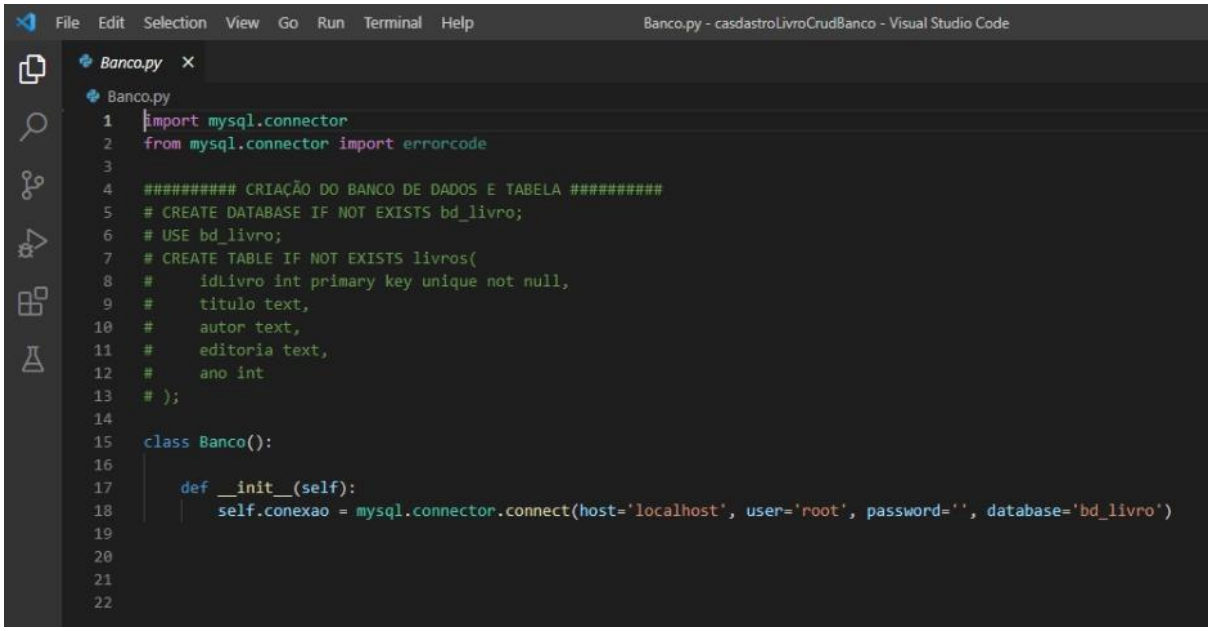
```
173     def limpaCampos(self):
174 
175         self.txtcodigo.delete(0, END)
176         self.txttituloLv.delete(0, END)
177         self.txtautor.delete(0, END)
178         self.txteditoria.delete(0, END)
179         self.txtano.delete(0, END)
180 
181 #Instância
182 root = Tk()
183 livro = Livro()
184 
185 root.title("POO COM PYTHON :)")
186 Application(root)
187 root.mainloop()
```

CAPÍTULO 5

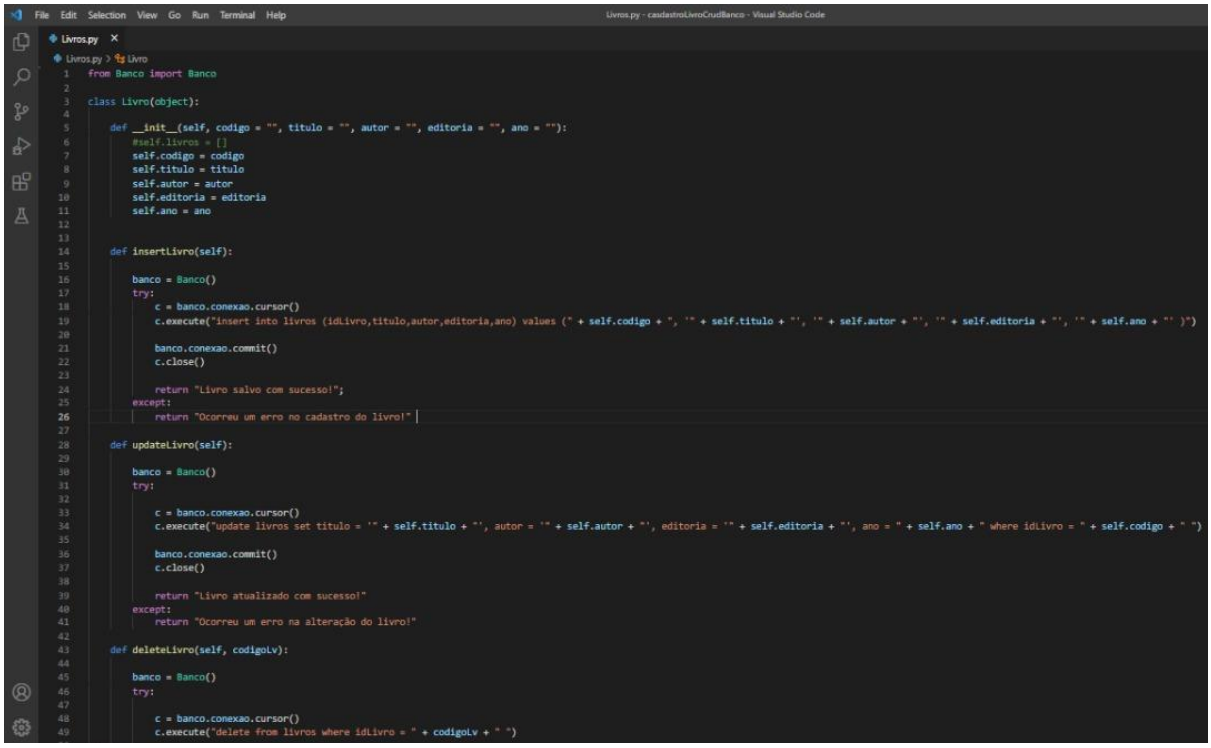
Neste próximo capítulo veremos o acesso e implementação do banco de dados MySql no exercício 4 do capítulo 4.

Projeto CRUD

Banco de dados



Comandos sql



```
            banco.conexao.commit()
            c.close()

            return "Livro excluído com sucesso!"
        except:
            return "Ocorreu um erro na exclusão do livro!"


    def buscaLivro(self, codigoLv):

        banco = Banco()

        try:

            c = banco.conexao.cursor()
            c.execute("select * from livros where idLivro = " + codigoLv + "  ")

            row = c.fetchall()
            #VERIFICA SE TEM RETORNO
            if len(row) == 0 :
                self.codigo = ""
                self.titulo = ""
                self.autor = ""
                self.editoria = ""
                self.ano = ""
                return "Código não encontrado!"
            else:
                for linha in row:
                    self.codigo = linha[0]
                    self.titulo = linha[1]
                    self.autor = linha[2]
                    self.editoria = linha[3]
                    self.ano = linha[4]
                    c.close()
                return ""
        except:
            self.codigo = ""
            self.titulo = ""
            self.autor = ""
            self.editoria = ""
            self.ano = ""
            return "Código não encontrado"
```

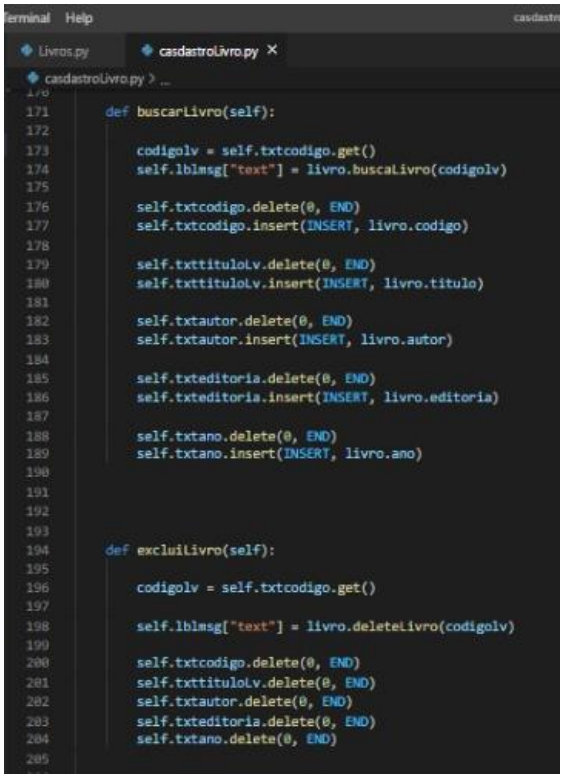
Código principal

```
Terminal  Help                                              casdastroLivro.py - casdastroLivro

 Livros.py    casdastroLivro.py ×

 casdastroLivro.py > ...
1   # !ANTES DE EXECUTAR! -> (CMD) pip install mysql-connector-python==8.0.13
2   # LIGAR SERVIDOR MYSQL
3
4   from Livros import Livro
5   from tkinter import *
6
7   class Application:
8       def __init__(self, master=None):
9           self.fonte = ("Verdana", "8")
10
11          self.container1 = Frame(master)
12          self.container1["pady"] = 10
13          self.container1.pack()
14          self.container2 = Frame(master)
15          self.container2["padx"] = 20
16          self.container2["pady"] = 5
17          self.container2.pack()
18          self.container3 = Frame(master)
19          self.container3["padx"] = 20
20          self.container3["pady"] = 5
21          self.container3.pack()
22          self.container4 = Frame(master)
23          self.container4["padx"] = 20
24          self.container4["pady"] = 5
25          self.container4.pack()
26          self.container5 = Frame(master)
27          self.container5["padx"] = 20
28          self.container5["pady"] = 5
29          self.container5.pack()
30          self.container6 = Frame(master)
31          self.container6["padx"] = 20
32          self.container6["pady"] = 5
33          self.container6.pack()
34          self.container8 = Frame(master)
35          self.container8["padx"] = 20
36          self.container8["pady"] = 10
37          self.container8.pack()
38          self.container9 = Frame(master)
39          self.container9["pady"] = 15
40          self.container9.pack()
41
42          self.titulo = Label(self.container1, text="Cadastro de Livros :")
43          self.titulo["font"] = ("Calibri", "9", "bold")
44          self.titulo.pack ()
45
46          self.lblcodigo = Label(self.container2,
47          text="Código:", font=self.fonte, width=10)
48          self.lblcodigo.pack(side=LEFT)
```


Terminal Help
Livros.py
casdastroLivro.py
self.container8.pack()
self.container9 = Frame(master)
self.container9["pady"] = 15
self.container9.pack()
self.titulo = Label(self.container1, text="Cadastro de Livros
self.titulo["font"] = ("Calibri", "9", "bold")
self.titulo.pack ()
self.lblcodigo = Label(self.container2,
text="Código:", font=self.fonte, width=10)
self.lblcodigo.pack(side=LEFT)
self.txtcodigo = Entry(self.container2)
self.txtcodigo["width"] = 10
self.txtcodigo["font"] = self.fonte
self.txtcodigo.pack(side=LEFT)
self.btnBuscar = Button(self.container2, text="Buscar",
font=self.fonte, width=10)
self.btnBuscar["command"] = self.buscarLivro
self.btnBuscar.pack(side=RIGHT)
self.lbltituloLv = Label(self.container3, text="Titulo:",
font=self.fonte, width=10)
self.lbltituloLv.pack(side=LEFT)
self.txttituloLv = Entry(self.container3)
self.txttituloLv["width"] = 25
self.txttituloLv["font"] = self.fonte
self.txttituloLv.pack(side=LEFT)
self.lblautor = Label(self.container4, text="Autor:",
font=self.fonte, width=10)
self.lblautor.pack(side=LEFT)
self.txtautor = Entry(self.container4)
self.txtautor["width"] = 25
self.txtautor["font"] = self.fonte
self.txtautor.pack(side=LEFT)
self.lbleditoria= Label(self.container5, text="Editoria:",
font=self.fonte, width=10)
self.lbleditoria.pack(side=LEFT)


Terminal Help
Livros.py
casdastroLivro.py
self.txteditoria = Entry(self.container5)
self.txteditoria["width"] = 25
self.txteditoria["font"] = self.fonte
self.txteditoria.pack(side=LEFT)
self.lblano= Label(self.container6, text="Ano:",
font=self.fonte, width=10)
self.lblano.pack(side=LEFT)
self.txtano = Entry(self.container6)
self.txtano["width"] = 10
self.txtano["font"] = self.fonte
self.txtano.pack(side=LEFT)
self.btnSalvar = Button(self.container8, text="Salvar",
font=self.fonte, width=12)
self.btnSalvar["command"] = self.salvaLivro
self.btnSalvar.pack (side=LEFT)
self.btnLimpar = Button(self.container8, text="limpar",
font=self.fonte, width=12)
self.btnLimpar["command"] = self.limpaCampos
self.btnLimpar.pack(side=LEFT)
self.btnExcluir = Button(self.container8, text="Excluir",
font=self.fonte, width=12)
self.btnExcluir["command"] = self.excluiLivro
self.btnExcluir.pack(side=LEFT)
self.btnAtualizar = Button(self.container8, text="Alterar",
font=self.fonte, width=12)
self.btnAtualizar["command"] = self.atualizaLivro
self.btnAtualizar.pack(side=RIGHT)
self.lblmsg = Label(self.container9, text="")
self.lblmsg["font"] = ("Verdana", "9", "italic")
self.lblmsg.pack()


```
127
128     def salvaLivro(self):
129
130         #valida dados
131         codigo = self.txtcodigo.get().strip()
132         titulo = self.txttituloLv.get().strip()
133         autor = self.txtautor.get().strip()
134         editoria = self.txteditoria.get().strip()
135         ano = self.txtano.get().strip()
136
137         if(codigo == "" or titulo == "" or autor == "" or editoria == "" or ano == ""):
138             self.lblmsg["text"] = "Todos os campos são obrigatórios!"
139             return
140         else:
141             try:
142                 ano = int(ano)
143                 if ano <= 0:
144                     self.lblmsg["text"] = "Ano inválido!"
145                     return
146                 else:
147                     ano = str(ano) #a class espera receber uma string no parametro
148             except:
149                 self.lblmsg["text"] = "Informe um ano válido!"
150                 return
151
152         #define parametros
153         livro.codigo = codigo
154         livro.titulo = titulo
155         livro.autor = autor
156         livro.editoria = editoria
157         livro.ano = ano
158
159         #chama metodo e atribui retorno
160         self.lblmsg["text"] = livro.insertLivro()
161
162         #limpa campos
163         self.txtcodigo.delete(0, END)
164         self.txttituloLv.delete(0, END)
165         self.txtautor.delete(0, END)
166         self.txteditoria.delete(0, END)
167         self.txtano.delete(0, END)
168
169
170
171     def buscarLivro(self):
172
173         codigolv = self.txtcodigo.get()
174         self.lblmsg["text"] = livro.buscaLivro(codigolv)
175
```




```
170
171     def buscarLivro(self):
172
173         codigolv = self.txtcodigo.get()
174         self.lblmsg["text"] = livro.buscaLivro(codigolv)
175
176         self.txtcodigo.delete(0, END)
177         self.txtcodigo.insert(INSERT, livro.codigo)
178
179         self.txttituloLv.delete(0, END)
180         self.txttituloLv.insert(INSERT, livro.titulo)
181
182         self.txtautor.delete(0, END)
183         self.txtautor.insert(INSERT, livro.autor)
184
185         self.txteditoria.delete(0, END)
186         self.txteditoria.insert(INSERT, livro.editoria)
187
188         self.txtano.delete(0, END)
189         self.txtano.insert(INSERT, livro.ano)
190
191
192
193
194     def excluiLivro(self):
195
196         codigolv = self.txtcodigo.get()
197
198         self.lblmsg["text"] = livro.deleteLivro(codigolv)
199
200         self.txtcodigo.delete(0, END)
201         self.txttituloLv.delete(0, END)
202         self.txtautor.delete(0, END)
203         self.txteditoria.delete(0, END)
204         self.txtano.delete(0, END)
205
```


```
    def atualizaLivro(self):

        #Valida dados
        codigo = self.txtcodigo.get().strip()
        titulo = self.txttituloLv.get().strip()
        autor = self.txtautor.get().strip()
        editoria = self.txteditoria.get().strip()
        ano = self.txtano.get().strip()

        if(codigo == "" or titulo == "" or autor == "" or editoria == "" or ano == ""):
            self.lblmsg["text"] = "Todos os campos são obrigatórios!"
            return
        else:
            try:
                ano = int(ano)
                if ano <= 0:
                    self.lblmsg["text"] = "Ano inválido!"
                    return
                else:
                    ano = str(ano) #a class espera receber uma string no parametro
            except:
                self.lblmsg["text"] = "Informe um ano válido!"
                return

        #define parametros
        livro.codigo = codigo
        livro.titulo = titulo
        livro.autor = autor
        livro.editoria = editoria
        livro.ano = ano

        self.lblmsg["text"] = livro.updateLivro()

        self.txtcodigo.delete(0, END)
        self.txttituloLv.delete(0, END)
        self.txtautor.delete(0, END)
        self.txteditoria.delete(0, END)
        self.txtano.delete(0, END)



    def limpaCampos(self):

        self.txtcodigo.delete(0, END)
        self.txttituloLv.delete(0, END)
        self.txtautor.delete(0, END)
        self.txteditoria.delete(0, END)
        self.txtano.delete(0, END)
```

```
#Instância
root = Tk()
livro = Livro()

root.title("POO COM PYTHON :)")
Application(root)
root.mainloop()
```

CAPÍTULO 6

Neste capítulo faremos uma aplicação em que digitemos um CEP e a partir de uma consulta externa tenhamos como retorno o endereço correspondente (rua, bairro, cidade e estado);

```
def consultaCep(self, cep):

    url = 'https://viacep.com.br/ws/%s/json/' % (cep)
    r = requests.get( url )

    if r.status_code == 200:
        resultado = r.json()
        resultado = '%s - %s - %s/%s ' % (resultado['logradouro'],
        resultado['bairro'], resultado['localidade'], resultado['uf'])

        return resultado
    else:
        return "CEP não encontrado"
```

```
self.btnConsulta = Button(self.container0, text="Consultar",
font=self.fonte, width=10)
self.btnConsulta["command"] = self.consultaCep
self.btnConsulta.pack(side=RIGHT)
```

```
    def consultaCep(self):

        cep = self.txtcep.get()
        self.lblmsgcep["text"] = livro.consultaCep(cep)
```


POO COM PYTHON :)
Cadastro de Livros :
Cep:
11085689
Consultar
CEP não encontrado
Código:
Buscar
Titulo:
Autor:
Editoria:
Ano:
Salvar
limpar
Excluir
Alterar


POO COM PYTHON :)
Cadastro de Livros :
Cep:
11085680
Consultar
Rua Eduardo Alves - São Jorge - Santos/SP
Código:
Buscar
Titulo:
Autor:
Editoria:
Ano:
Salvar
limpar
Excluir
Alterar