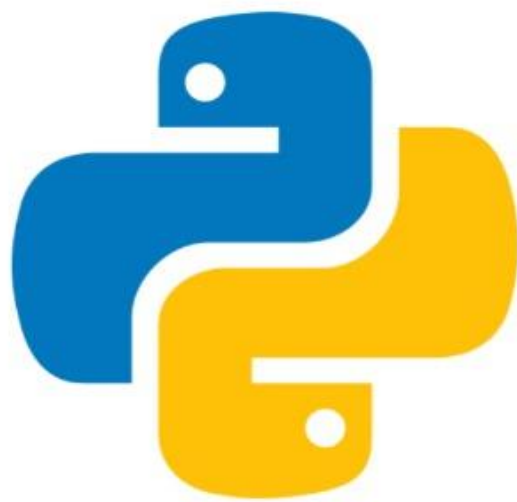


APOSTILA PYTHON



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

1. Capítulo 1	2
1.1 História e características do python.....	2
1.2 Possibilidades de ide para python.....	3
1.3 Ide escolhida para o trabalho.....	4
1.4 Processo de download e instalação da linguagem e ide vs code.....	5
1.5 Configurando o visual studio code para programar em python.....	9
2. Capítulo 2	12
2.1 Exercícios - Parte 1.....	12
2.2 Exercícios - Parte 2.....	14
3. Capítulo 3	22
4. Capítulo 4	31
5. Capítulo 5	53
6. Capítulo 6	58

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 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 exhibe os conteúdos

IDEs PARA PYTHON 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

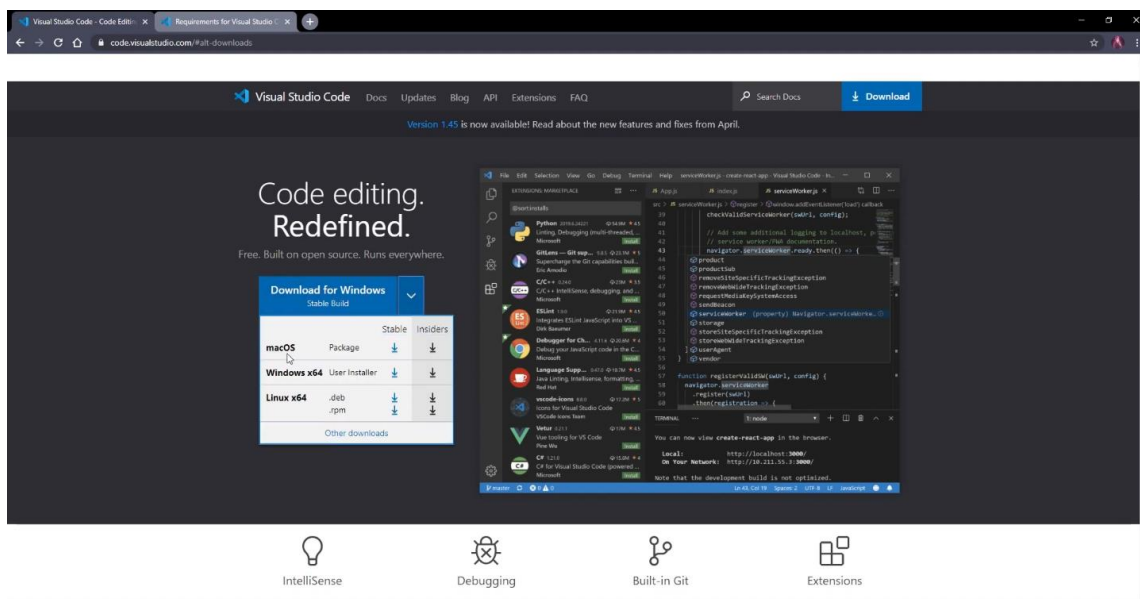
com ele ao decorrer da apostila.

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

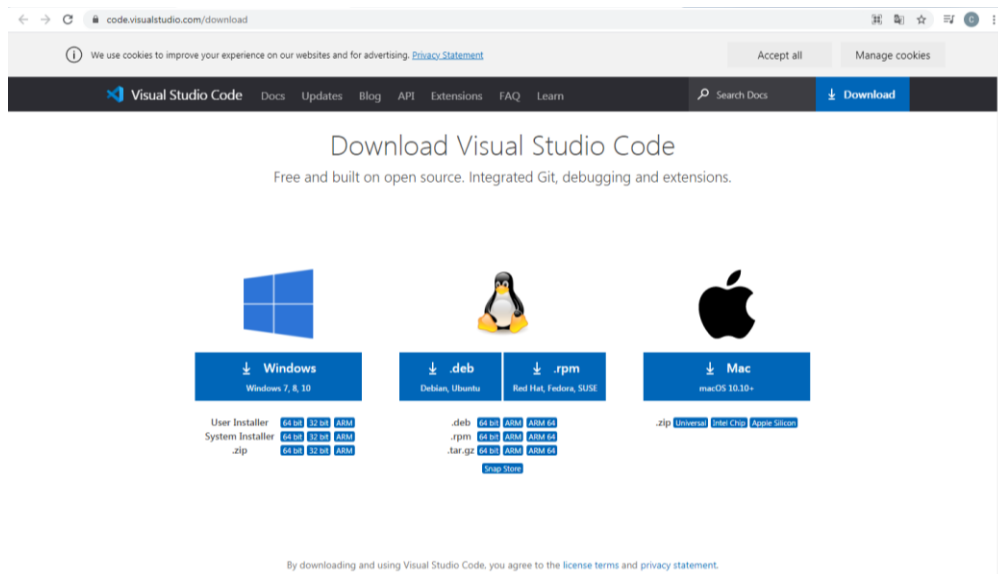
Visual Studio code é um editor de código, gratuito da Microsoft, onde há diversas extensões para que sejam adicionadas funcionalidades extras.

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.

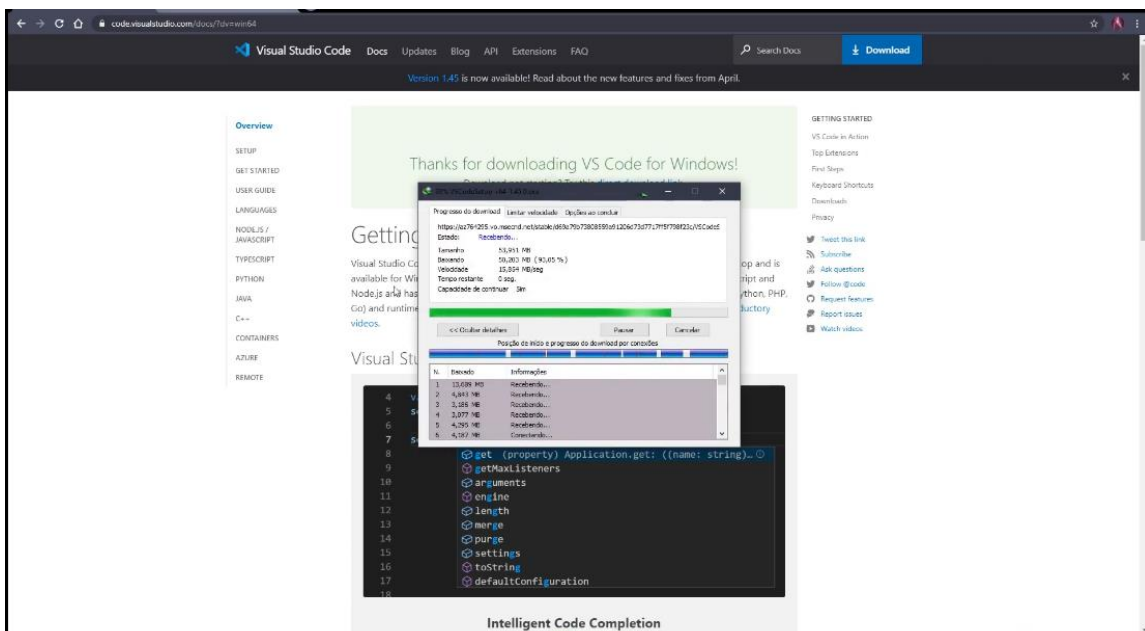
1. Link para download: <https://code.visualstudio.com/download>



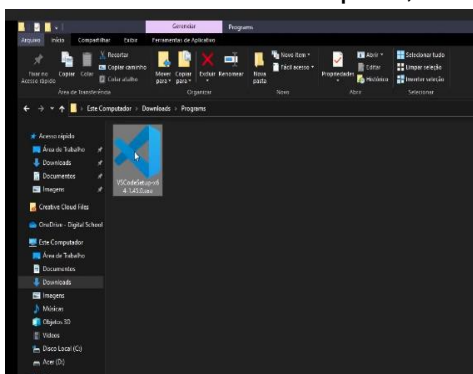
2. Abrindo o site há as opções de sistemas operacionais onde pode ser instalado o programa.



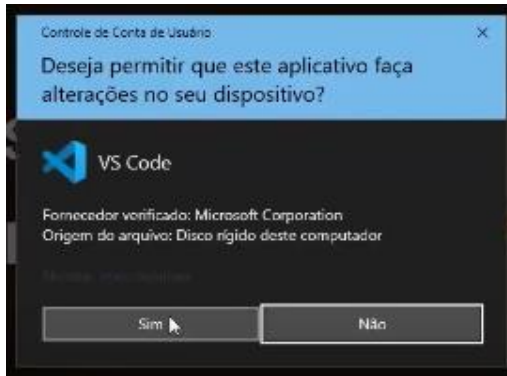
3. Após ser selecionada a opção que se adequa ao seu computador, será iniciado o download.



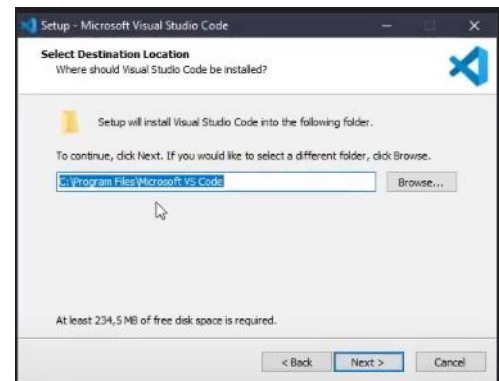
4. Já no diretório:
4.1. Selecione o arquivo;



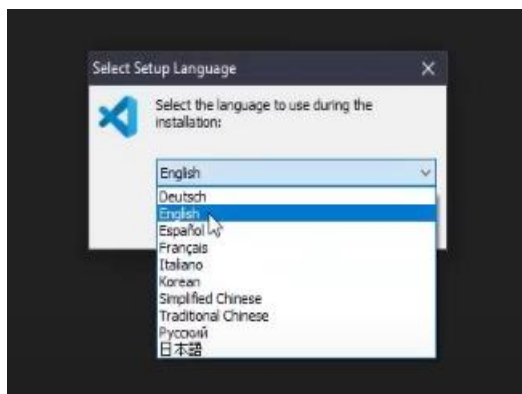
4.2. Conceda as permissões solicitadas;



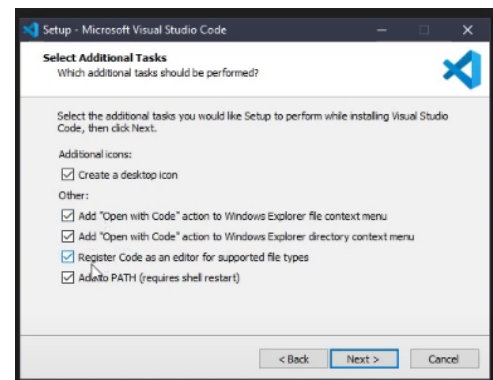
4.5. Altere o diretório, caso desejado;



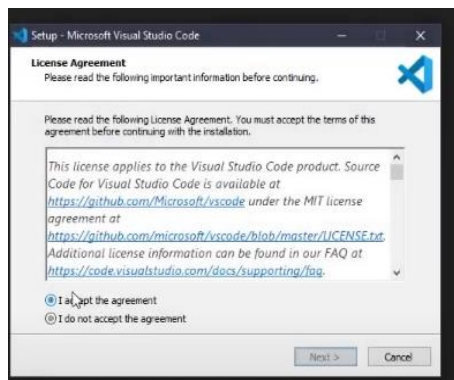
4.3. Selecione o idioma.



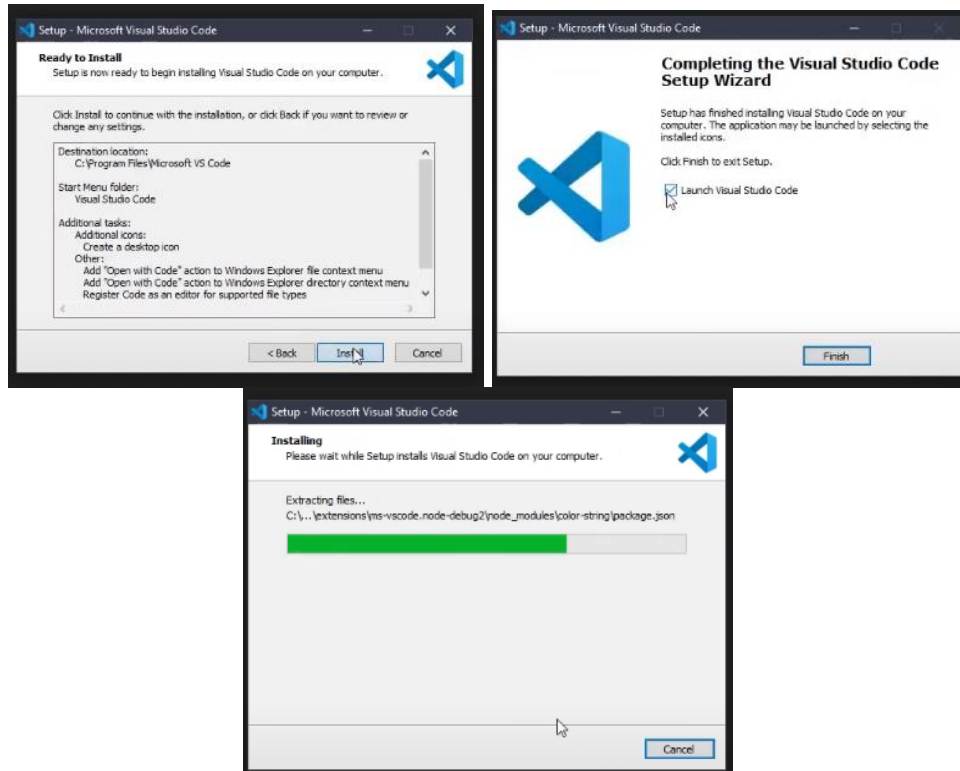
4.6. Selecione toda as opções e next;



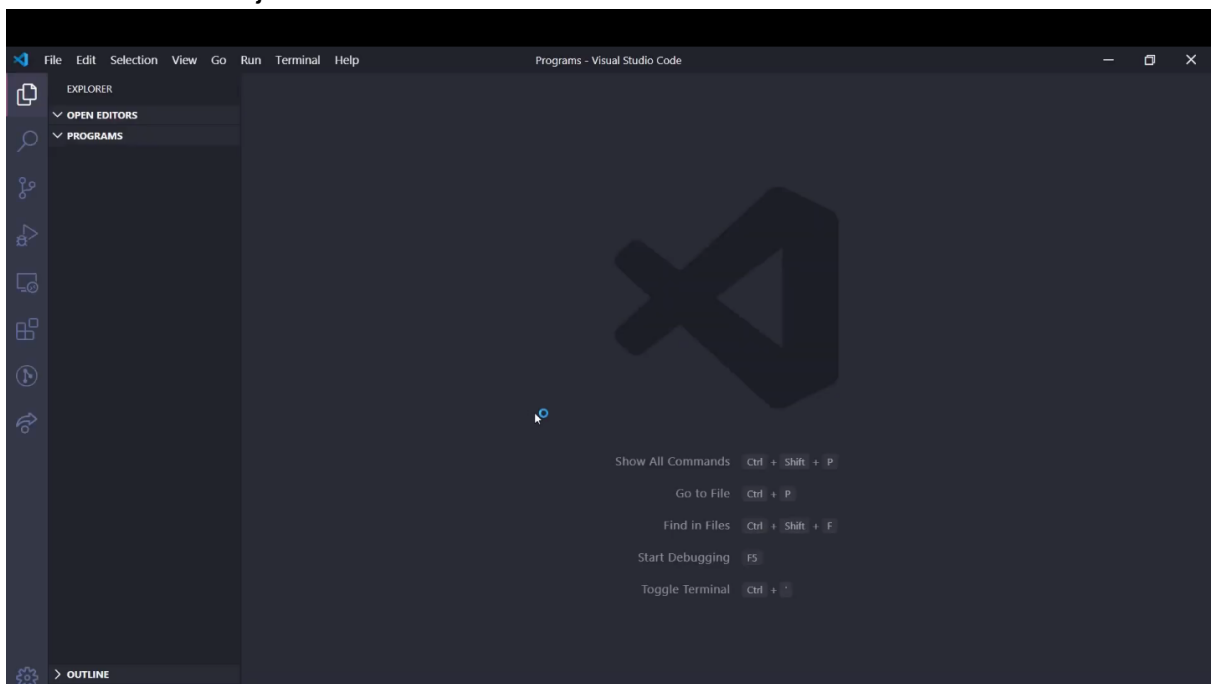
4.4. Aceite os termos;



4.7. Instale;

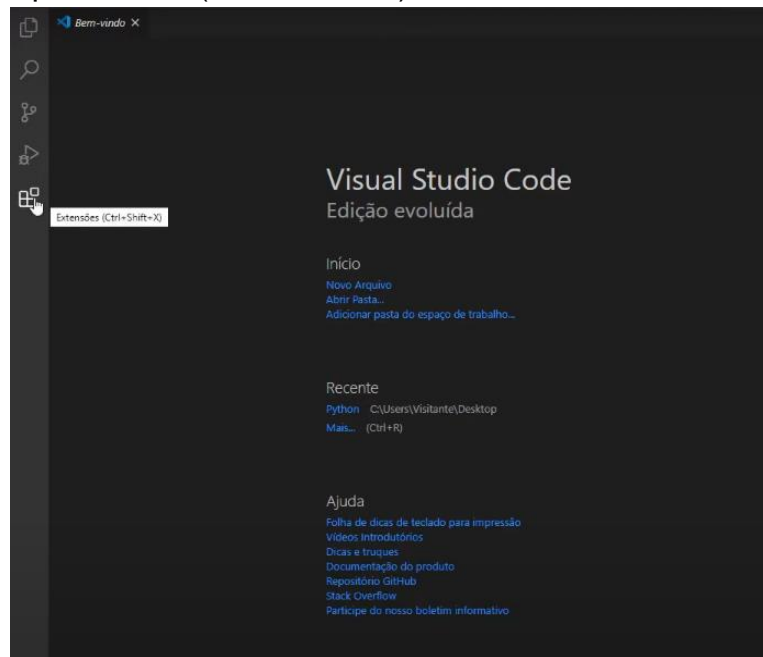


5. Interface já instalada:

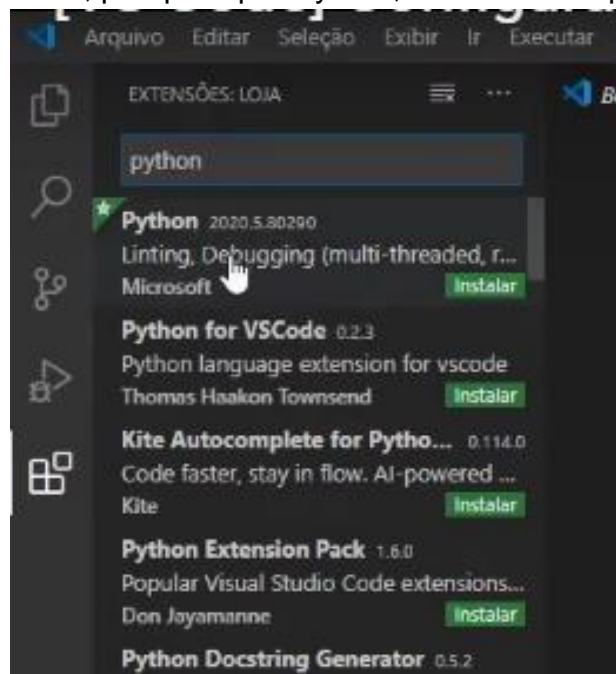


CONFIGURANDO O VISUAL STUDIO CODE PARA PROGRAMAR EM PYTHON

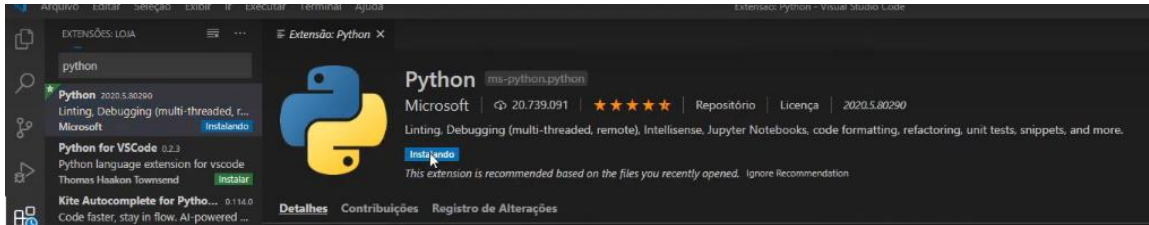
1. Abra o programa e na lateral esquerda, selecione a o último ícone, de extensões ou pelo atalho (ctrl + shift + x);



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

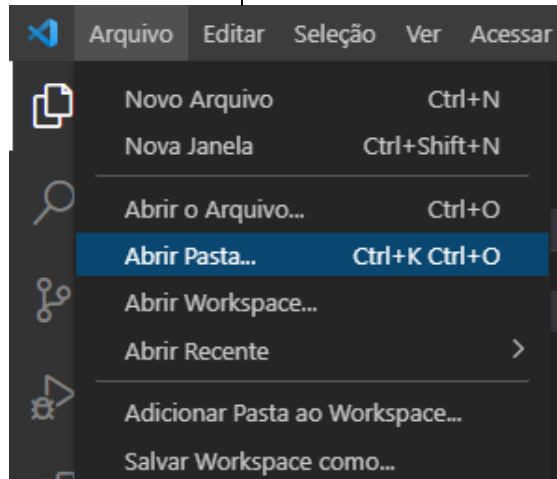


3. Instale;

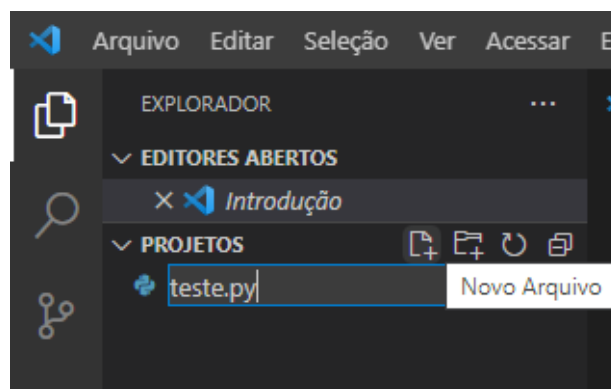


Primeiramente, é necessário escolher uma pasta previamente para salvar o código que será desenvolvido.

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

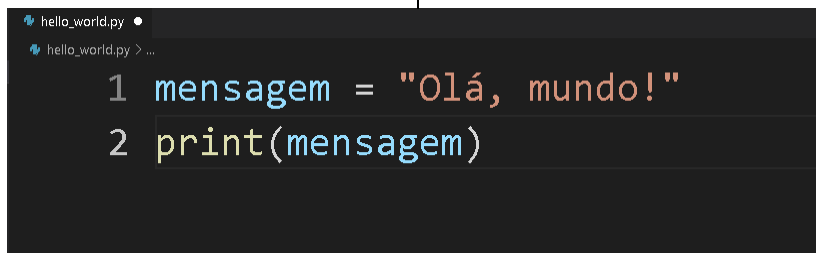
caractere nova linha ou “\n”, semelhantes à função tecla *enter* do teclado.



```
hello_world.py •
hello_world.py
1 print("hello,world!")
```

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.



```
hello_world.py •
hello_world.py > ...
1 mensagem = "Olá, mundo!"
2 print(mensagem)
```

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.

```
1 #Digitação dos dados
2 altura = int(input('Digite a altura do triângulo: '))
3 base = int(input('Digite a base do triângulo: '))
4
5 #Calculo da área
6 area = (base * altura)/2
7
8 #Exibição do resultado
9 print('A área do triângulo é:', area)
10
```

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.

```

1 #importação da biblioteca de matemática
2 import math
3
4 #leitura do ângulo armazenando na variável
5 angulo = float (input("Digite o ângulo desejado:"))
6
7 #Estrutura de calculos e conversão do seno
8 seno = math.sin(math.radians (angulo))
9
10 #Exibição do resultado do seno
11 print ("O ângulo de {} tem o SENO de {:.2f}".format(angulo, seno))
12
13 #Estrutura de calculos e conversão do cosseno
14 cosseno = math.cos(math.radians (angulo))
15
16 #Exibição do resultado do cosseno
17 print ("O ângulo de {} tem o COSSENO de {:.2f}".format(angulo,cosseno))
18

```

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.

```

1 #Digitação das notas, armazenando em variáveis
2 nota1 = float( input("Digite a nota do 1º bimestre: ") )
3 nota2 = float( input("Digite a nota do 2º bimestre: ") )
4 nota3 = float( input("Digite a nota do 3º bimestre: ") )
5 nota4 = float( input("Digite a nota do 4º bimestre: ") )
6
7 # Cálculo da média final
8 media = (nota1 + nota2 + nota3 + nota4) / 4
9
10 #Exibição da média
11 print("A média final é: ",media)
12
13 # Laço de repetição para analisar se o aluno está reprovado,
14 # aprovado ou de exame
15 if (media>=6):
16     print("Você está aprovado!")
17 elif (media<3):
18     print("Você está reprovado!")
19 else :
20     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.

```

1 # Entrada de dados
2 num = int(input('Digite um numero inteiro e positivo: '))
3
4 # Estrutura de repetição para exibir o resultado
5 for c in range (1, 11):
6     print('{} x {:2} = {}'.format(num, c, num*c))
7

```

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 $(x^2 + 1)$ onde $x = \{1, 2, 3, 4, 5 \dots\}$.

```

1 #Pedido de valor
2 numero = int(input('digite um valor: '))
3 termo = []
4
5 #Estrutura de repetição
6 for termo in range(numero):
7     x = ((pow(termo,2)) + 1)
8     print('X: ', termo, ' | F(x): ', x)
9

```

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.


```

1  #Designer
2  print ('Sequência de Fibonacci' )
3  print ('__'*30)
4
5  # Declaração de variáveis
6  t1 = 0
7  t2 = 1
8
9  # Exibição dos dois primeiros números
10 print ('__'*30)
11 print ('{} -> {}'.format(t1,t2), end=' ')
12
13 # Declaração de variável que auxiliará como contador
14 cont = 3
15
16 # Estrutura de repetição para exibir o resultado
17 while cont <= 999:
18     t3 = t1 + t2
19     print ('-> {}'.format(t3), end=' ')
20     t1 = t2
21     t2 = t3
22     cont +=1
23 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.

```

1  #Pedido dos valores
2  valor1 = int(input('digite o primeiro valor: '))
3  valor2 = int(input('digite o segundo valor: '))
4
5  #Estrutura de repetição
6  while valor2 < valor1 :
7      valor2 = int(input('digite o segundo valor: '))
8
9  #Exibição na tela
10 print('Primeiro Valor: ', valor1)
11 print('Segundo Valor: ', valor2)
12

```

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.

```

1  #Armazenando a digitação na variável sexo
2  sexo = input("digite seu sexo -- (M) masculino (F) feminino: ")
3
4  #Estrutura de repetição para conferir se foi digitado certo
5  while True:
6      if sexo != 'M' and sexo != 'F':
7          print("resposta invalida")
8          sexo = input("digite seu sexo -- (M) masculino (F) feminino: ")
9      else:
10         break
11

```

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.

```

1 #enquanto o loop for igual a S, a consulta continua
2 continua = "s"
3 while(continua == "s"):
4     n = int(input('digite um numero positivo e inteiro: '))
5     #Estrutura de repetição para verificar se o numero é inteiro e positivo
6     if not n >= 0:
7         print('o numero deve ser inteiro e positivo ')
8         n = int(input("Digite um valor que seja aceito: "))
9     #Estrutura para realizar o fatorial
10    valida = True
11    fat = 1
12    i = 2
13    while i <= n:
14        fat = fat*i
15        i = i + 1
16    #Exibição do resultado na tela
17    print("O valor de %d! eh =" %n, fat)
18    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.

```

1 #Declaração das variáveis
2 num = []
3 a = 1
4
5 #Laço para criar os números
6 for i in range (10):
7     num.append(input('Digite o número: '))
8     a = a +1
9
10 #Imprime na tela a mensagem da ordem inversa
11 print('Ordem Inversa: ')
12
13 #Laço para imprimir na tela a ordem inversa
14 for x in reversed(num):
15     print(x)
16

```

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.

```

1  #Declaração de variáveis
2  values = []
3  soma = 0
4  media = 0
5
6  #Laço de repetição para alimentação do vetor
7  for i in range(10):
8      values.append(input('Número... '))
9
10 #Laço de repetição para calcular a soma dos números digitados
11
12 for i in range(10):
13     soma += int(values[i])
14     media = (soma / 10)
15
16 #Imprime na tela o resultado da média dos valores
17 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.

```

1  #Declaração de variáveis
2  listanum = []
3  maior = 0
4  menor = 0
5
6  #Estrutura de repetição para a entrada de dados
7  for c in range(0, 10):
8      listanum.append(int(input(f'Digite um valor para a posição {c}')))
9      #Estrutura de repetição para a verificação do maior e menor número
10     if c == 0:
11         maior = menor = listanum[c]
12     else:
13         if listanum[c] > maior:
14             maior = listanum[c]
15         if listanum[c] < menor:
16             menor = listanum[c]
17
18 #Exibição dos resultados
19 print(f'O maior valor digitado foi: {maior}')
20 print(f'O menor valor digitado foi: {menor}')
21

```

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.

```

1  #percorre array e verifica se o nome existe, se existir,
2  # cai no return no for e volta uma string com a frase 'nomeUSer Ano(s)',
3  # caso n tenha o nome, retorna 'Pessoa não localizada'
4  def verificaNome(nome):
5      for x in range(0, 10):
6          for y in range(0, 1):
7              if matrizTb[x][y] == nome:
8                  return str(matrizTb[x][1] + " Ano(s)")
9
10     return "Pessoa não localizada"
11
12     matrizTb = []
13     linhatt = []
14
15     #pula a linha
16     print()
17
18     #array com 10 posições
19     for x in range(0, 10):
20         linhatt = []
21         print("----- Pessoa {} ----- ".format((x+1)))
22         print()
23
24         for y in range(0, 2):
25
26             # valorVerdadeito if condicao else valorFalso
27             # ternario define qual valor vai receber e atribuna na matriz
28             # valor = ( Se y = 0 -> nome )
29             # valor = ( Se y = 1 -> idade )
30             valor = str(input("Informe um nome: ")) if y == 0 else str(int(input("Informe a idade: ")))
31             linhatt.append( valor )
32
33         matrizTb.append( linhatt )
34
35     #enquanto o loop for igual a S, continua consultando
36     continua = "s"
37     while(continua == "s"):
38         nome = str(input("Pesquise por um nome: "))
39         print(verificaNome(nome))
40
41         valida = True
42
43         while valida:
44             continua = str(input("Deseja realizar uma nova consulta?")).lower().strip()
45             if(continua == "s" or continua == "S"):
46                 valida = False
47             else:
48                 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.

```
1 #cria array com a string 'vazio' e todas as posicoes e retorna o array criado
2 def defineCinema(ln, cl):
3
4     cinema = []
5     linhatt = []
6
7     for x in range(0, ln):
8         linhatt = []
9         for y in range(0, cl):
10            linhatt.append( "vazio" )
11        cinema.append( linhatt )
12
13    return cinema
14
15 #exibe as opções, para o usuário continuar ou não
16 def exhibeMenu():
17     menu = "\n1 - Nova reserva. \n4 - Finalizar."
18
19     return menu
20
21 #percorre cada posicao do array e concatenar na variavel lugares
22 def exhibeLugares():
23
24     lugares = ""
25     #for para rodar todo o array
26     for x in range(0, linha):
```

```

27     lugares += "\n"
28     for y in range(0, coluna):
29         lugares += "["+str((x+1))+ " "+str((y+1))+"]->" +str(cinemaArray[x][y])+ " | "
30     return lugares
31
32 #verifica se o lugar existe na matriz (combinacao dos indexes)
33 def verificaLugar(flU, clU):
34     if flU > -1 and flU < linha and clU > -1 and clU < coluna:
35         return True
36     else:
37         return False
38
39 linha = 20
40 coluna = 15
41
42 # atribui o retorno da funcao (array solicitado) na variavel
43 cinemaArray = defineCinema(linha, coluna)
44
45 #mostra o menu de opções
46 print(exibeMenu())
47 opcaoMenu = 0
48
49 #loop para resevar os lugares
50 while opcaoMenu != 4 :
51     opcaoMenu = int(input("\nEscolha uma opcao: "))
52     if opcaoMenu == 1:
53
54         print("Escolha um lugar: ")
55         #mostra os lugares
56         print(exibeLugares())
57         print("\n")
58
59         flUser = int(input("Informe o numero da Fileira (fl): "))
60         clUser = int(input("Informe o numero da poltrona (cl): "))
61         #pega o numero da fileira e da coluna tirando 1 por causa do array
62         flUser = flUser - 1
63         clUser = clUser - 1
64         nome = str(input("Informe seu nome: "))
65         print("\n")
66
67         #verifica se o lugar está reservado
68         rtVf = verificaLugar(flUser, clUser)
69         print(rtVf)
70         #se a posição estiver entre os indices da matriz, vai para a próxima verificação
71         if rtVf :
72             #se o lugar estiver indicando valor == vazio, pode reservar
73             if cinemaArray[flUser][clUser] == "vazio":
74
75                 cinemaArray[flUser][clUser] = nome
76                 print("\n ----- \n")
77                 print("Seu lugar foi reservado com sucesso!!\n")
78                 print("\n ----- \n")
79                 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 !!")
```

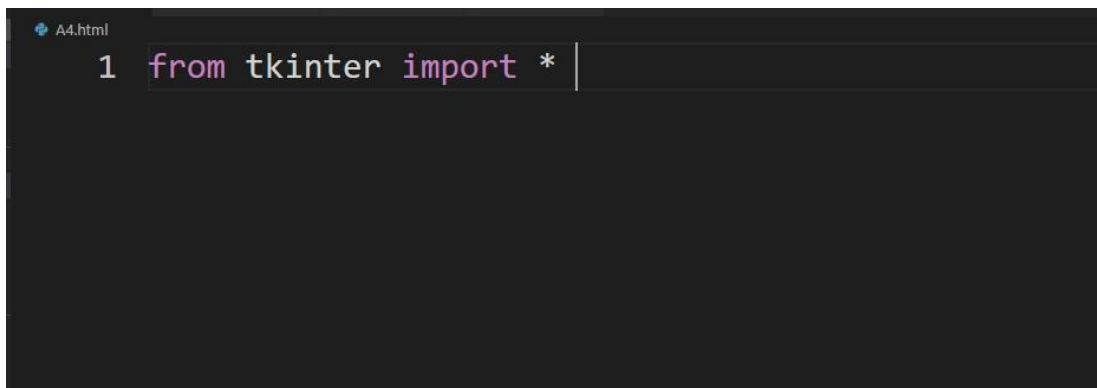

CAPÍTULO 3

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.

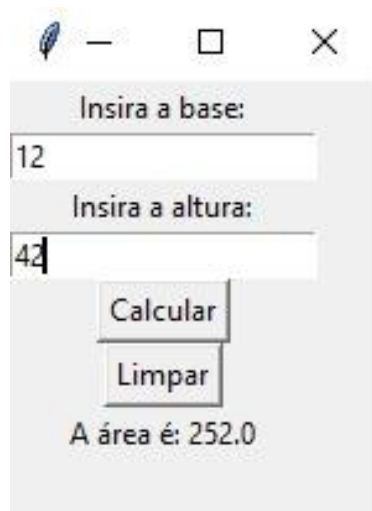
Uma vez que a linguagem é nativa do Python, bastante executar o comando a seguir para ter acesso a biblioteca:



```
A4.html
1 from tkinter import *
```

Exercício 1.

```
1  from tkinter import *
2
3  #calcular a area de um triango.A = (b*h)/2
4
5  #funcoes
6  def calcular():
7      B = float(textbox1.get())
8      H = float(textbox2.get())
9      R = (B*H)/2
10     final.set("A área é: "+str(R))
11
12 def limpar():
13     textbox1.delete(0, 'end')
14     textbox2.delete(0, 'end')
15
16 #tela
17 menuInicial = Tk()
18 menuInicial.title("Cálculo de Área do Triângulo");
19
20
21 #variaveis
22 final = StringVar()
23
24 #design
25 label1 = Label(menuInicial,text="Insira a base:")
26 textbox1 = Entry(menuInicial);
27 label2 = Label(menuInicial,text="Insira a altura:")
28 textbox2 = Entry(menuInicial);
29 button1 = Button(menuInicial,text="Calcular",command=calcular)
30 button2 = Button(menuInicial,text="Limpar",command=limpar)
31 labelResultado = Label(menuInicial,textvariable=final)
32
33 #graphs
34 label1.grid()
35 textbox1.grid()
36 label2.grid()
37 textbox2.grid()
38 button1.grid()
39 button2.grid()
40 labelResultado.grid()
41
42 #loop
43 menuInicial.mainloop()
```



Exercício 2.

```
1  from tkinter import *
2
3  #calcular o salario
4  # Exercício 02
5
6  #funcoes
7  def calcular():
8
9      qtd = float(textbox1.get())
10     valor = float(textbox2.get())
11     sb = float()
12
13     aux=profissao.get()
14
15     if aux == "h":
16         sb = qtd*valor
17     elif aux == "p":
18         sb = (qtd * valor) * 1.25
19
20     final.set("O salário é: "+str(sb))
21
22  def limpar():
23     textbox1.delete(0, 'end')
24     textbox2.delete(0, 'end')
25
26  #tela
27  menuInicial = Tk()
28  menuInicial.title("Cálculo de Salário");
29
```

```

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()

```

Honorista
 Professor

Digite a quantidade de horas/aulas

72

Digite o valor da hora/aula

8

Calcular Salário Bruto

Limpar

O salário é: 576.0

Honorista
 Professor

Digite a quantidade de horas/aulas

72

Digite o valor da hora/aula

8

Calcular Salário Bruto

Limpar

O salário é: 720.0

Exercício 3.

```
1 from tkinter import *
2 from tkinter import ttk
3 #calcular o salaerio
4 # Exercício 03
5
6 #funcoes
7 def calcular():
8
9     qtd = float(textbox1.get())
10    valor = float(textbox2.get())
11    sb = float()
12
13    aux=cb_profissao.get()
14
15    aux=cb_profissao.current()
16
17    if aux == 0:
18        sb = qtd*valor
19    elif aux == 1:
20        sb = (qtd * valor) * 1.25
21
22    final.set("O salário é: "+str(sb))
23
24 def limpar():
25     textbox1.delete(0, 'end')
26     textbox2.delete(0, 'end')
27
28 #tela
29 menuInicial = Tk()
30 menuInicial.title("Cálculo de Salário");
```

```

32 #variaveis
33 final = StringVar()
34 profissao=StringVar()
35 listaprofissoes =["Honorista", "Professor"]
36 #design
37
38 lb_Profissao=Label(menuInicial,text= "Você é:")
39 lb_Profissao.pack()
40 cb_profissao = ttk.Combobox(menuInicial, values= listaprofissoes )
41 cb_profissao.pack()
42 label1 = Label(menuInicial,text="Digite a quantidade de horas/aulas")
43 textbox1 = Entry(menuInicial);
44 label2 = Label(menuInicial,text="Digite o valor da hora/aula")
45 textbox2 = Entry(menuInicial);
46 button1 = Button(menuInicial,text="Calcular Salário Bruto",command=calcular)
47 button2 = Button(menuInicial,text="Limpar",command=limpar)
48 labelResultado = Label(menuInicial,textvariable=final)
49
50 #graphs
51 label1.pack()
52 textbox1.pack()
53 label2.pack()
54 textbox2.pack()
55 button1.pack()
56 button2.pack()
57 labelResultado.pack()
58
59 #loop
60 menuInicial.mainloop()

```

Você é:

Honorista

Digite a quantidade de horas/aulas

30

Digite o valor da hora/aula

12

Calcular Salário Bruto

Limpar

O salário é: 360.0

Você é:

Professor

Digite a quantidade de horas/aulas

30

Digite o valor da hora/aula

14

Calcular Salário Bruto

Limpar

O salário é: 450.0

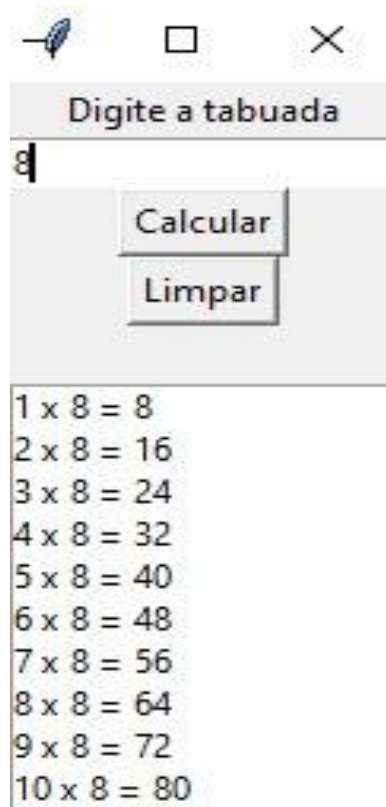
Exercício 4.

```
1  from tkinter import *
2
3  #calcular tabuada
4  # Exercício 04
5
6  #funcoes
7  def calcular():
8      qtd = int(textbox1.get())
9
10     for c in range(1, 11):
11         Lista.insert (END,f'{c} x {qtd} = {qtd*c}')
12
13
14  def limpar():
15     textbox1.delete(0, 'end')
16     Lista.delete(0, 'end')
17
18  #tela
19  menuInicial = Tk()
20  menuInicial.title("Calcular tabuada");
21
22
23  #variaveis
24  final = StringVar()
25  profissao=StringVar()
26
```

```

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

```
1  from Ex1 import triangulo
2  import Ex1BLL
3  import Erro
4  from tkinter import *
5  from tkinter import messagebox
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"] = 10
16         self.container2["pady"] = 5
17         self.container2.pack()
18         self.container3 = Frame(master)
19         self.container3["padx"] = 10
20         self.container3["pady"] = 5
21         self.container3.pack()
22         self.container4 = Frame(master)
23         self.container4["padx"] = 10
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["pady"] = 15
32     self.container6.pack()
33
34     self.titulo = Label(self.container1, text="Área do triângulo:")
35     self.titulo["font"] = ("Calibri", "9", "bold")
36     self.titulo.pack ()
37
38     self.lblBase = Label(self.container2, text="Base do triângulo:",
39     font=self.fonte, width=25)
40     self.lblBase.pack(side=LEFT)
41
42     self.txtBase = Entry(self.container2)
43     self.txtBase["width"] = 25
44     self.txtBase["font"] = self.fonte
45     self.txtBase.pack(side=LEFT)
46
47     self.lblAltura = Label(self.container3, text="Altura do triângulo:",
48     font=self.fonte, width=25)
49     self.lblAltura.pack(side=LEFT)
50
51     self.txtAltura = Entry(self.container3)
52     self.txtAltura["width"] = 25
53     self.txtAltura["font"] = self.fonte
54     self.txtAltura.pack(side=LEFT)
55
56     self.lblArea = Label(self.container4, text="Área do triângulo:",
57     font=self.fonte, width=25)
58     self.lblArea.pack(side=LEFT)
59
60     self.txtArea = Entry(self.container4)
61     self.txtArea["width"] = 25
62     self.txtArea["font"] = self.fonte
63     self.txtArea.pack(side=LEFT)
64
65     self.btnCalcular = Button(self.container5, text="Calcular",
66     font=self.fonte, width=12)
67     self.btnCalcular["command"] = self.Calcular
68     self.btnCalcular.pack (side=LEFT)
69
70
71     self.btnLimpar = Button(self.container5, text="limpar",
72     font=self.fonte, width=12)
73     self.btnLimpar["command"] = self.limpaCampos
74     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')
98
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 x
Ex1.py > triangulo
1 class triangulo():
2
3     def __init__(self):
4         self.b = 0
5         self.h = 0
6
7     def setB(self, _b):
8         self.b = _b
9
10    def setH(self, _h):
11        self.h = _h
12
13    def getB(self):
14        return self.b
15
16    def getH(self):
17        return self.h
18
19    def triangulo(self):
20        return str((float(self.b) * float(self.h)) / 2)
21
```

Classe Erro

```
Erro.py x
Erro.py > ...
1 erro = False
2 mens = ''
3
4 def setErro(_erro):
5     global erro
6     global mens
7
8     if isinstance(_erro, str):
9         erro = True
10        mens = _erro
11    else:
12        erro = _erro
13
14    def getErro():
15        return erro
16
17    def getMens():
18        return mens
```

Classe BLL

```
Ex1BLL.py x
Ex1BLL.py > validaDados
1 import Erro
2
3
4 def validaDados(triangulo):
5
6     Erro.setErro(False)
7
8     if len(triangulo.getB()) == 0:
9         Erro.setErro("A base tem que ter preenchimento obrigatório.")
10        return None
11    else:
12        try:
13            float(triangulo.getB())
14        except:
15            Erro.setErro("A base deve ser numérico.")
16            return None
17
18    if len(triangulo.getH())==0:
19        Erro.setErro("A altura tem que ter preenchimento obrigatório.")
20        return None
21    else:
22        try:
23            float(triangulo.getH())
24        except:
25            Erro.setErro("A altura deve ser numérico.")
26            return None
```

Exercício 2

```
Ex2.py x
Ex2.py
1 class Salario:
2     def __init__(self):
3         self.qtd = "0"
4         self.valor = "0"
5
6     def setQtd(self, _qtd):
7         self.qtd = _qtd
8
9     def setVal(self, _valor):
10        self.valor = _valor
11
12       def getQtd(self):
13           return self.qtd
14
15       def getVal(self):
16           return self.valor
17
18       def getSalarioBruto(self):
19           return str((int(self.qtd) * int(self.valor)) / 2)
20
21
```

Classe BLL

```
Ex2BLL.py x
Ex2BLL.py > validaDados
1 import Erro
2
3 def validaDados(Salario):
4
5     Erro.setErro(False)
6
7     if len(Salario.getQtd()) == 0:
8         Erro.setErro("O campo QUANTIDADE DE HORAS é de preenchimento obrigatório...")
9         return None
10    else:
11        try:
12            int(Salario.getQtd())
13        except:
14            Erro.setErro("O campo QUANTIDADE DE HORAS deve ser numérico...")
15            return None
16
17    if int(Salario.getQtd()) <= 0:
18        Erro.setErro("O campo QUANTIDADE DE HORAS deve ser maior que zero.");
19        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)

```

```

51     self.label_3.setObjectName("label_3")
52     self.qtdH = QtWidgets.QLineEdit(Dialog)
53     self.qtdH.setGeometry(QtCore.QRect(350, 20, 101, 51))
54     self.qtdH.setText("")
55     self.qtdH.setObjectName("qtdH")
56     self.valH = QtWidgets.QLineEdit(Dialog)
57     self.valH.setGeometry(QtCore.QRect(350, 90, 101, 51))
58     self.valH.setText("")
59     self.valH.setObjectName("valH")
60     self.ValSalario = QtWidgets.QLineEdit(Dialog)
61     self.ValSalario.setGeometry(QtCore.QRect(350, 230, 101, 51))
62     self.ValSalario.setText("")
63     self.ValSalario.setObjectName("ValSalario")
64
65     self.retranslateUi(Dialog)
66     QtCore.QMetaObject.connectSlotsByName(Dialog)
67
68     self.calcButton.clicked.connect(self.Calcular)
69     self.cleanButton.clicked.connect(self.Limpar)

```

```

71     def retranslateUi(self, Dialog):
72         _translate = QtCore.QCoreApplication.translate
73         Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
74         self.calcButton.setText(_translate("Dialog", "Calcular"))
75         self.cleanButton.setText(_translate("Dialog", "Limpar"))
76         self.label.setText(_translate("Dialog", "Salário Bruto:"))
77         self.label_2.setText(_translate("Dialog", "Qtd de horas trabalhadas:"))
78         self.label_3.setText(_translate("Dialog", "Valor da Hora:"))
79
80     def Calcular(self):
81         msg = QtWidgets.QMessageBox()
82         salario = Salario()
83
84         salario.setQtd(self.qtdH.text())
85         salario.setVal(self.valH.text())
86
87         Ex2BLL.validaDados(salario)
88         if Erro.getErro():
89             msg.setText(Erro.getMens())
90             msg.exec()
91         else:
92             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 X
Equacao2G_inicial.py
1 import Erro
2 from Equacao2g import Equacao2G
3 import Equacao2GBLL
4 from tkinter import *
5 from tkinter import messagebox
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.container7 = Frame(master)
35     self.container7["padx"] = 20
36     self.container7["pady"] = 10
37     self.container7.pack()
38     self.container8 = Frame(master)
39     self.container8["pady"] = 15
40     self.container8.pack()
41
42     self.titulo = Label(self.container1, text="Equação de 2º grau :")
43     self.titulo["font"] = ("Calibri", "9", "bold")
44     self.titulo.pack ()
45
46     self.lblValorA = Label(self.container2, text="Valor a:",
47     font=self.fonte, width=10)
48     self.lblValorA.pack(side=LEFT)
49
50     self.txtValorA = Entry(self.container2)
51     self.txtValorA["width"] = 25
52     self.txtValorA["font"] = self.fonte
53     self.txtValorA.pack(side=LEFT)
54
55     self.lblValorB = Label(self.container3, text="Valor b:",
56     font=self.fonte, width=10)
57     self.lblValorB.pack(side=LEFT)
58
59     self.txtValorB = Entry(self.container3)
60     self.txtValorB["width"] = 25
61     self.txtValorB["font"] = self.fonte
62     self.txtValorB.pack(side=LEFT)
63
64     self.lblValorC = Label(self.container4, text="Valor c:",
65     font=self.fonte, width=10)
66     self.lblValorC.pack(side=LEFT)
67
68     self.txtValorC = Entry(self.container4)
69     self.txtValorC["width"] = 25
70     self.txtValorC["font"] = self.fonte
71     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

```

```

121         else:
122
123             self.txtValorX1.insert(INSERT, equacao2G.get_x1())
124             self.txtValorX2.insert(INSERT, equacao2G.get_x2())
125
126
127             self.txtValorA.config(state = 'disabled')
128             self.txtValorB.config(state = 'disabled')
129             self.txtValorC.config(state = 'disabled')
130
131     def limpaCampos(self):
132
133         self.txtValorA.config(state = 'normal')
134         self.txtValorB.config(state = 'normal')
135         self.txtValorC.config(state = 'normal')
136
137         self.txtValorA.delete(0, END)
138         self.txtValorB.delete(0, END)
139         self.txtValorC.delete(0, END)
140         self.txtValorX1.delete(0, END)
141         self.txtValorX2.delete(0, END)

```

```

144 #Instância
145 root = Tk()
146
147 root.title("POO COM PYTHON :)")
148 Application(root)
149 root.mainloop()
150

```

```
Equacao2g.py X
Equacao2g.py > ...
1  from math import *
2
3  class Equacao2G:
4      def __init__(self):
5          self.a = 0
6          self.b = 0
7          self.c = 0
8
9      def get_a(self):
10         return self.__a
11     def get_b(self):
12         return self.__b
13     def get_c(self):
14         return self.__c
15
16     def set_a(self, valor_a):
17         self.__a = valor_a
18     def set_b(self, valor_b):
19         self.__b = valor_b
20     def set_c(self, valor_c):
21         self.__c = valor_c
22
23     def get_delta(self):
24         auxa = float(self.__a)
25         auxb = float(self.__b)
26         auxc = float(self.__c)
27         calc = float((auxb * auxb) - (4 * auxa * auxc))
28         return str(calc)
29
30     def get_x1(self):
31         auxa = float(self.__a)
32         auxb = float(self.__b)
33         auxc = float(self.__c)
34         auxdelta = float(self.get_delta())
35         calc = float((-auxb + sqrt(auxdelta)) / (2 * auxa))
36         return str(calc)
37
38     def get_x2(self):
39         auxa = float(self.__a)
40         auxb = float(self.__b)
41         auxc = float(self.__c)
42         auxdelta = float(self.get_delta())
43         calc = float((-auxb - sqrt(auxdelta)) / (2 * auxa))
44         return str(calc)
```

Classe BLL

```
Equacao2GBLL.py X
Equacao2GBLL.py > ...
1 import Erro
2
3 def validaDados(equacao2G):
4     Erro.setErro(False)
5
6     if len(equacao2G.get_a()) == 0:
7         Erro.setErro("O valor de A é de preenchimento obrigatório...")
8         return None
9     else:
10        try:
11            float(equacao2G.get_a())
12        except:
13            Erro.setErro("O valor de A deve ser numérico...")
14            return None
15
16    if len(equacao2G.get_b()) == 0:
17        Erro.setErro("O valor de B é de preenchimento obrigatório...")
18        return None
19    else:
20        try:
21            float(equacao2G.get_b())
22        except:
23            Erro.setErro("O valor de B deve ser numérico...")
24            return None
25
26    if len(equacao2G.get_c()) == 0:
27        Erro.setErro("O valor de C é de preenchimento obrigatório...")
28        return None
29    else:
30        try:
31            float(equacao2G.get_c())
32        except:
33            Erro.setErro("O valor de C deve ser numérico...")
34            return None
35
36    if float(equacao2G.get_delta()) < 0:
37        Erro.setErro("Delta negativo, não existem raízes reais para esta equação..")
38
```

Exercício 4

```
Livros.py x
Livros.py > Livro
1 class Livro(object):
2
3     def __init__(self, codigo = "", titulo = "", autor = "", editoria = "", ano = ""):
4         self.livros = []
5         self.codigo = codigo
6         self.titulo = titulo
7         self.autor = autor
8         self.editoria = editoria
9         self.ano = ano
10
11
12     def insertLivro(self):
13
14         for x in self.livros:
15             if( x[0] == self.codigo ):
16                 return "Código já cadastrado"
17
18         self.livros.append([self.codigo, self.titulo, self.autor, self.editoria, self.ano])
19         return "Livro salvo com sucesso";
20
21
22     def buscaLivro(self, codigoLv):
23
24         texto = ""
25         for x in self.livros:
26             texto += "x -> " + str(x) + " cod -> " + str(codigoLv)
27             if( x[0] == codigoLv ):
28                 self.codigo = x[0]
29                 self.titulo = x[1]
30                 self.autor = x[2]
31                 self.editoria = x[3]
32                 self.ano = x[4]
33
34             return ""
35
36         return "Código não encontrado"
```


casdastroLivro.py X

casdastroLivro.py > ...

```
1 from Livros import Livro
2 from tkinter import *
3
4 class Application:
5     def __init__(self, master=None):
6         self.fonte = ("Verdana", "8")
7
8         self.container1 = Frame(master)
9         self.container1["pady"] = 10
10        self.container1.pack()
11        self.container2 = Frame(master)
12        self.container2["padx"] = 20
13        self.container2["pady"] = 5
14        self.container2.pack()
15        self.container3 = Frame(master)
16        self.container3["padx"] = 20
17        self.container3["pady"] = 5
18        self.container3.pack()
19        self.container4 = Frame(master)
20        self.container4["padx"] = 20
21        self.container4["pady"] = 5
22        self.container4.pack()
23        self.container5 = Frame(master)
24        self.container5["padx"] = 20
25        self.container5["pady"] = 5
26        self.container5.pack()
```

```

27     self.container6 = Frame(master)
28     self.container6["padx"] = 20
29     self.container6["pady"] = 5
30     self.container6.pack()
31     self.container8 = Frame(master)
32     self.container8["padx"] = 20
33     self.container8["pady"] = 10
34     self.container8.pack()
35     self.container9 = Frame(master)
36     self.container9["pady"] = 15
37     self.container9.pack()
38
39     self.titulo = Label(self.container1, text="Cadastro de Livros :")
40     self.titulo["font"] = ("Calibri", "9", "bold")
41     self.titulo.pack ()
42
43     self.lblcodigo = Label(self.container2,
44     text="Código:", font=self.fonte, width=10)
45     self.lblcodigo.pack(side=LEFT)
46
47     self.txtcodigo = Entry(self.container2)
48     self.txtcodigo["width"] = 10
49     self.txtcodigo["font"] = self.fonte
50     self.txtcodigo.pack(side=LEFT)
51
52     self.btnBuscar = Button(self.container2, text="Buscar",
53     font=self.fonte, width=10)
54     self.btnBuscar["command"] = self.buscarUsuario
55     self.btnBuscar.pack(side=RIGHT)
56
57     self.lbltituloLv = Label(self.container3, text="Titulo:",
58     font=self.fonte, width=10)
59     self.lbltituloLv.pack(side=LEFT)
60
61     self.txttituloLv = Entry(self.container3)
62     self.txttituloLv["width"] = 25
63     self.txttituloLv["font"] = self.fonte
64     self.txttituloLv.pack(side=LEFT)
65
66     self.lblautor = Label(self.container4, text="Autor:",
67     font=self.fonte, width=10)
68     self.lblautor.pack(side=LEFT)
69
70     self.txtautor = Entry(self.container4)
71     self.txtautor["width"] = 25
72     self.txtautor["font"] = self.fonte
73     self.txtautor.pack(side=LEFT)
74
75     self.lbleditoria= Label(self.container5, text="Editoria:",
76     font=self.fonte, width=10)
77     self.lbleditoria.pack(side=LEFT)

```

```

79     self.txteditoria = Entry(self.container5)
80     self.txteditoria["width"] = 25
81     self.txteditoria["font"] = self.fonte
82     self.txteditoria.pack(side=LEFT)
83
84     self.lblano= Label(self.container6, text="Ano:",
85     font=self.fonte, width=10)
86     self.lblano.pack(side=LEFT)
87
88     self.txtano = Entry(self.container6)
89     self.txtano["width"] = 10
90     self.txtano["font"] = self.fonte
91     self.txtano.pack(side=LEFT)
92
93
94     self.btnSalvar = Button(self.container8, text="Salvar",
95     font=self.fonte, width=12)
96     self.btnSalvar["command"] = self.salvaLivro
97     self.btnSalvar.pack (side=LEFT)
98
99
100    self.btnLimpar = Button(self.container8, text="limpar",
101    font=self.fonte, width=12)
102    self.btnLimpar["command"] = self.limpaCampos
103    self.btnLimpar.pack(side=LEFT)

```

```

105    self.lblmsg = Label(self.container9, text="")
106    self.lblmsg["font"] = ("Verdana", "9", "italic")
107    self.lblmsg.pack()
108
109    def salvaLivro(self):
110        #Valida dados
111        codigo = self.txtcodigo.get().strip()
112        titulo = self.txttituloLv.get().strip()
113        autor = self.txtautor.get().strip()
114        editoria = self.txteditoria.get().strip()
115        ano = self.txtano.get().strip()
116
117        if(codigo == "" or titulo == "" or autor == "" or editoria == "" or ano == ""):
118            self.lblmsg["text"] = "Todos os campos são obrigatórios!"
119            return
120        else:
121            try:
122                ano = int(ano)
123                if ano <= 0:
124                    self.lblmsg["text"] = "Ano inválido!"
125                    return
126                else:
127                    ano = str(ano) #a class espera receber uma string no parametro
128            except:
129                self.lblmsg["text"] = "Informe um ano válido!"
130            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

```
File Edit Selection View Go Run Terminal Help Banco.py - cadastroLivroCrudBanco - Visual Studio Code

Banco.py x
Banco.py
1 import mysql.connector
2 from mysql.connector import errorcode
3
4 ##### CRIAÇÃO DO BANCO DE DADOS E TABELA #####
5 # CREATE DATABASE IF NOT EXISTS bd_livro;
6 # USE bd_livro;
7 # CREATE TABLE IF NOT EXISTS livros(
8 #     idLivro int primary key unique not null,
9 #     titulo text,
10 #     autor text,
11 #     editoria text,
12 #     ano int
13 # );
14
15 class Banco():
16
17     def __init__(self):
18         self.conexao = mysql.connector.connect(host='localhost', user='root', password='', database='bd_livro')
```

Comandos sql

```
File Edit Selection View Go Run Terminal Help Livros.py - cadastroLivroCrudBanco - Visual Studio Code

Livros.py x
Livros.py > Livros
1 from Banco import Banco
2
3 class Livro(object):
4
5     def __init__(self, codigo = "", titulo = "", autor = "", editoria = "", ano = ""):
6         #self.livros = []
7         self.codigo = codigo
8         self.titulo = titulo
9         self.autor = autor
10        self.editoria = editoria
11        self.ano = ano
12
13
14    def insertLivro(self):
15
16        banco = Banco()
17        try:
18            c = banco.conexao.cursor()
19            c.execute("insert into livros (idLivro,titulo,autor,editoria,ano) values (' + self.codigo + ', ' + self.titulo + ', ' + self.autor + ', ' + self.editoria + ', ' + self.ano + ' ")")
20
21            banco.conexao.commit()
22            c.close()
23
24            return "Livro salvo com sucesso!"
25        except:
26            return "Ocorreu um erro no cadastro do livro!"
27
28    def updateLivro(self):
29
30        banco = Banco()
31        try:
32
33            c = banco.conexao.cursor()
34            c.execute("update livros set titulo = ' + self.titulo + ', autor = ' + self.autor + ', editoria = ' + self.editoria + ', ano = ' + self.ano + ' where idLivro = ' + self.codigo + ' ")
35
36            banco.conexao.commit()
37            c.close()
38
39            return "Livro atualizado com sucesso!"
40        except:
41            return "Ocorreu um erro na alteração do livro!"
42
43    def deleteLivro(self, codigolv):
44
45        banco = Banco()
46        try:
47
48            c = banco.conexao.cursor()
49            c.execute("delete from livros where idLivro = " + codigolv + " ")
```

```

51         banco.conexao.commit()
52         c.close()
53
54         return "Livro excluído com sucesso!"
55     except:
56         return "Ocorreu um erro na exclusão do livro!"
57
58
59     def buscaLivro(self, codigolv):
60
61         banco = Banco()
62
63         try:
64
65             c = banco.conexao.cursor()
66             c.execute("select * from livros where idlivro = " + codigolv + " ")
67
68             row = c.fetchall()
69             #VERIFICA SE TEM RETORNO
70             if len(row) == 0 :
71                 self.codigo = ""
72                 self.titulo = ""
73                 self.autor = ""
74                 self.editoria = ""
75                 self.ano = ""
76                 return "Código não encontrado!"
77             else:
78                 for linha in row:
79                     self.codigo = linha[0]
80                     self.titulo = linha[1]
81                     self.autor = linha[2]
82                     self.editoria = linha[3]
83                     self.ano = linha[4]
84                     c.close()
85                 return ""
86         except:
87             self.codigo = ""
88             self.titulo = ""
89             self.autor = ""
90             self.editoria = ""
91             self.ano = ""
92             return "Código não encontrado"
93

```

Código principal

```

Terminal Help
casdastroLivro.py - casdastroLivro

Livros.py
casdastroLivro.py X
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                                cadastroLivro.py - C
└─ Livros.py ─┬─ cadastroLivro.py X
└─ cadastroLivro.py > ...
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="Codigo:", font=self.fonte, width=10)
48     self.lblcodigo.pack(side=LEFT)
49
50     self.txtcodigo = Entry(self.container2)
51     self.txtcodigo["width"] = 10
52     self.txtcodigo["font"] = self.fonte
53     self.txtcodigo.pack(side=LEFT)
54
55     self.btnBuscar = Button(self.container2, text="Buscar",
56     font=self.fonte, width=10)
57     self.btnBuscar["command"] = self.buscarLivro
58     self.btnBuscar.pack(side=RIGHT)
59
60     self.lbltitulov = Label(self.container3, text="Titulo:",
61     font=self.fonte, width=10)
62     self.lbltitulov.pack(side=LEFT)
63
64     self.txttitulov = Entry(self.container3)
65     self.txttitulov["width"] = 25
66     self.txttitulov["font"] = self.fonte
67     self.txttitulov.pack(side=LEFT)
68
69     self.lblautor = Label(self.container4, text="Autor:",
70     font=self.fonte, width=10)
71     self.lblautor.pack(side=LEFT)
72
73     self.txtautor = Entry(self.container4)
74     self.txtautor["width"] = 25
75     self.txtautor["font"] = self.fonte
76     self.txtautor.pack(side=LEFT)
77
78     self.lbleditoria = Label(self.container5, text="Editoria:",
79     font=self.fonte, width=10)
80     self.lbleditoria.pack(side=LEFT)

```

```

Terminal Help                                cadastroLivro.py
└─ Livros.py ─┬─ cadastroLivro.py X
└─ cadastroLivro.py > ...
81
82     self.txteditoria = Entry(self.container5)
83     self.txteditoria["width"] = 25
84     self.txteditoria["font"] = self.fonte
85     self.txteditoria.pack(side=LEFT)
86
87     self.lblano = Label(self.container6, text="Ano:",
88     font=self.fonte, width=10)
89     self.lblano.pack(side=LEFT)
90
91     self.txtano = Entry(self.container6)
92     self.txtano["width"] = 10
93     self.txtano["font"] = self.fonte
94     self.txtano.pack(side=LEFT)
95
96
97     self.btnSalvar = Button(self.container8, text="Salvar",
98     font=self.fonte, width=12)
99     self.btnSalvar["command"] = self.salvaLivro
100    self.btnSalvar.pack(side=LEFT)
101
102
103    self.btnLimpar = Button(self.container8, text="limpar",
104    font=self.fonte, width=12)
105    self.btnLimpar["command"] = self.limpaCampos
106    self.btnLimpar.pack(side=LEFT)
107
108
109    self.btnExcluir = Button(self.container8, text="Excluir",
110    font=self.fonte, width=12)
111    self.btnExcluir["command"] = self.excluiLivro
112    self.btnExcluir.pack(side=LEFT)
113
114    self.btnAtualizar = Button(self.container8, text="Alterar",
115    font=self.fonte, width=12)
116    self.btnAtualizar["command"] = self.atualizaLivro
117    self.btnAtualizar.pack(side=RIGHT)
118
119
120
121
122    self.lblmsg = Label(self.container9, text="")
123    self.lblmsg["font"] = ("Verdana", "9", "italic")
124    self.lblmsg.pack()
125

```



```
Terminal Help
casdastrolivro.py - casdastroLivroCrudBanco

Livros.py
casdastrolivro.py X
casdastroLivro.py > ...

127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175

def salvarLivro(self):
    #valida dados
    codigo = self.txtcodigo.get().strip()
    titulo = self.txttitulo.v.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

    #chama metodo e atribui retorno
    self.lblmsg["text"] = livro.insertLivro()

    #limpa campos
    self.txtcodigo.delete(0, END)
    self.txttitulo.v.delete(0, END)
    self.txtautor.delete(0, END)
    self.txteditoria.delete(0, END)
    self.txtano.delete(0, END)

def buscarLivro(self):
    codigolv = self.txtcodigo.get()
    self.lblmsg["text"] = livro.buscarLivro(codigolv)
```

```
Terminal Help
casdastrolivro.py - casdastroLivroCrudBanco

Livros.py
casdastrolivro.py X
casdastroLivro.py > ...

170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205

def buscarLivro(self):
    codigolv = self.txtcodigo.get()
    self.lblmsg["text"] = livro.buscarLivro(codigolv)

    self.txtcodigo.delete(0, END)
    self.txtcodigo.insert(INSERT, livro.codigo)

    self.txttitulo.v.delete(0, END)
    self.txttitulo.v.insert(INSERT, livro.titulo)

    self.txtautor.delete(0, END)
    self.txtautor.insert(INSERT, livro.autor)

    self.txteditoria.delete(0, END)
    self.txteditoria.insert(INSERT, livro.editoria)

    self.txtano.delete(0, END)
    self.txtano.insert(INSERT, livro.ano)

def excluirlivro(self):
    codigolv = self.txtcodigo.get()

    self.lblmsg["text"] = livro.deleteLivro(codigolv)

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

```
Terminal Help cadastroLivro.py - cadastroLivroCru@Banco
Livros.py cadastroLivro.py X
cadastroLivro.py > ...
287
288 def atualizaLivro(self):
289
290     #Valida dados
291     codigo = self.txtcodigo.get().strip()
292     titulo = self.txttitulo.get().strip()
293     autor = self.txtautor.get().strip()
294     editoria = self.txteditoria.get().strip()
295     ano = self.txtano.get().strip()
296
297     if(codigo == "" or titulo == "" or autor == "" or editoria == "" or ano == ""):
298         self.lblmsg["text"] = "Todos os campos são obrigatórios!"
299         return
300     else:
301         try:
302             ano = int(ano)
303             if ano <= 0:
304                 self.lblmsg["text"] = "Ano inválido!"
305                 return
306             else:
307                 ano = str(ano) #a class espera receber uma string no parametro
308         except:
309             self.lblmsg["text"] = "Informe um ano válido!"
310             return
311
312     #define parametros
313     livro.codigo = codigo
314     livro.titulo = titulo
315     livro.autor = autor
316     livro.editoria = editoria
317     livro.ano = ano
318
319     self.lblmsg["text"] = livro.updatelivro()
320
321     self.txtcodigo.delete(0, END)
322     self.txttitulo.delete(0, END)
323     self.txtautor.delete(0, END)
324     self.txteditoria.delete(0, END)
325     self.txtano.delete(0, END)
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
256
257 #Instância
258 root = Tk()
259 livro = Livro()
260
261 root.title("POO COM PYTHON :)")
262 Application([root])
263 root.mainloop()
264
265
```

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:

CEP não encontrado

Código:

Título:

Autor:

Editoria:

Ano:

POO COM PYTHON :)

Cadastro de Livros :

Cep:

Rua Eduardo Alves - São Jorge - Santos/SP

Código:

Título:

Autor:

Editoria:

Ano: