TRABALHO A3

Professor: Henrique Poyatos

UC: Gestão e Qualidade de Software

Integrantes:

Willian da Silva Silveira – **RA:** 1072221401

- ## DESCRIÇÃO DO TRABALHO.

Escolher um código-fonte legado com diversas más práticas de programação.

Identificar os problemas e realizar uma refatoração, utilizando as melhores práticas de código limpo.

- **INDICE**

- **CÓDIGO ORIGINAL**

Esse código foi desenvolvido nas aulas de Python que estou tendo.

Esse código foi desenvolvido para calcular áreas e volumes, ajudando pessoas que necessitam dessas informações no dia a dia.

```
action = int(input("Digite o número do item da a fórmula que você quer calcular.\n01 - Área do de um quadrado\n"
                   "02 - Área de um retângulo\n03 - Área do circulo\n04 - Área de um triângulo\n05 - Área do trapézio\n"
                   "06 - Volume de um cubo\n07 - Volume de um cilíndro\n08 - Volume de um prisma\n"
                   "09 - Volume de um trapézio\n"))

decTy = int(input("Digite o nome da decimal você quer trabalhar ?\n01 - Metros\n02 - Centímetros\n03 - Decímetros\n"
                  "04 - Milimetros\n"))

if decTy == 1:
    decTy = "Metros"
    decSq = "m²"
    decVol = "m³"

if decTy == 2:
    decTy = "Centímetros"
    decSq = "cm²"
    decVol = "cm³"

if decTy == 3:
    decTy = "Decímetros"
    decSq = "dec²"
    decVol = "dec³"

if decTy == 4:
    decTy = "Milímetros"
    decSq = "mm²"
    decVol = "mm³"

if action == 1:
    print("A fórmula para descobrir a área de um quadrado é A=a*b\n")
    a = float(input("Qual é o valor de 'a' em {} ?\n".format(decTy)))
    b = float(input("Qual é o valor de 'b' em {} ?\n".format(decTy)))
    sqAr = a*b
    print("A área desse quadrado é de {:.2f} {}".format(sqAr, decSq))

if action == 2:
    print("A fórmula para descobrir a área de um retângulo é A=a*b\n")
    a = float(input("Qual é o valor de 'a' em {} ?\n".format(decTy)))
    b = float(input("Qual é o valor de 'b' em {} ?\n".format(decTy)))
    recAr = a*b
```

Commented [Wd1]: Aqui foi utilizado um nome para a variável em outra língua, sendo que todo o código foi feito em português.

Commented [Wd2]: Aqui foi Criado um menu de escolha com tudo junto.
Da para criar o menu separado para ser chamado a qualquer momento.

Commented [Wd3]: Aqui tem um nome de variável que é uma abreviação de duas palavras em inglês.
Isso dificulta o entendimento e não condiz com a linguagem geral do código.

Commented [Wd4]: Erro de português

Commented [Wd5]: Variável com nome abreviado de 2 palavras em inglês.
Isso está dificultando o entendimento para ser utilizado depois.

Commented [Wd6]: Utilização excesiva de condições simples.

Commented [Wd7]: O nome dessas variáveis estão digitadas em inglês, sendo que todo o código está em portugês

Commented [Wd8]: Essa string se repete várias vezes ao longo do código.
Se tiver que fazer alguma alteração ou correção, o programador terá muito trabalho desnecessário para corrigir isso.

Commented [Wd9]: Nome de variável em outra língua e ainda está abreviada de forma muito difícil de entender.

```
    print("A área desse retângulo é de {:.2f} {}".format(recAr, decSq))

if action == 3:
    print("A fórmula para descobrir a área de um circulo é A=r²*PI/2")
    r = float(input("Qual é o raio desse circulo em {} ?\n".format(decTy)))
    cirAr = (r**2)*3.147/2
    print("A área desse circulo é de {:.2f} {}".format(cirAr, decSq))

if action == 4:
    print("A fórmula para descobrir a área de um triângulo é A=a*b/2")
    a = float(input("Qual é o valor de 'a' em {} ?\n".format(decTy)))
    b = float(input("Qual é o valor de 'b' em {} ?\n".format(decTy)))
    triAr = a*b/2
    print("A área desse triângulo é de {:.2f} {}".format(triAr, decSq))

if action == 5:
    print("A fórmula para descobrir a área de um trapézio é A=(B+b)*h/2")
    B = float(input("Qual é o valor de 'B' em {} ?\n".format(decTy)))
    b = float(input("Qual é o valor de 'b' em {} ?\n".format(decTy)))
    h = float(input("Qual é o valor de 'h' em {} ?\n".format(decTy)))
    trapAr = (B+b)*h/2
    print("A área desse trapézio pe de {:.2f} {}".format(trapAr, decSq))

if action == 6:
    print("A fórmula para descobrir o volume de um cubo é V=(a*b)*h")
    a = float(input("Qual é o valor de 'a' em {} ?\n".format(decTy)))
    b = float(input("Qual é o valor de 'b' em {} ?\n".format(decTy)))
    h = float(input("Qual é o valor de 'h' em {} ?\n".format(decTy)))
    cubVol = (a*b)*h
    print("O volume desse cubo é de {:.2f} {}".format(cubVol, decVol))

if action == 7:
    print("A fórmula para descobrir o volume de um cilíndro é V=(r²*PI/2)*h")
    r = float(input("Qual é o valor de 'r' (r é o raio, ok !) em {} ?\n".format(decTy)))
    h = float(input("Qual é o valor de 'h' em {} ?\n".format(decTy)))
    cilVol = ((r**2)*3.147/2)*h
    print("O volume desse cilíndro é de {:.2f} {}".format(cilVol, decVol))

if action == 8:
    print("A fórmula para descobrir o volume de um prisma é V=(1/2)*b*h*i\n")
    b = float(input("Qual é o valor de 'b' em {} ?\n".format(decTy)))
    h = float(input("Qual é o valor de 'h' em {} ?\n".format(decTy)))
    i = float(input("Qual é o valor de 'i' em {} ?\n".format(decTy)))
    triVol = (1/2)*b*h*i
    print("O volume desse prísma é de {:.2f} {}".format(triVol, decVol))

if action == 9:
    print("A fórmula para descobrir o volume de um trapézio é V=((1/2)*h*(B+b))*w")
    h = float(input("Qual é o valor de 'h' em {} ?\n".format(decTy)))
```

```
B = float(input("Qual é o valor de 'B' em {} ?\n".format(decTy)))
b = float(input("Qual é o valor de 'b' em {} ?\n".format(decTy)))
w = float(input("Qual é o valor de 'w' em {} ?\n".format(decTy)))
trapVol = ((1/2)*h*(B+b))*w
print("O volume desse trapézio é de {:.2f} {}".format(trapVol, decVol))
```

Commented [Wd10]: Novamente.
Aqui tem muita repetição de condições simples

- **IDENTIFICAÇÃO GERÃO DOS PROBLEMAS**

O código contém erros de português.

O código contém palavras em outra língua.

Muitas variáveis estão abreviadas de palavras de outra língua.

Contém muitas condições simples, ao invés de utilizar aninhamentos.

Abreviações muito simples, que dificultam o entendimento.

O código contém muitos textos e funções repetidas.

Não tem condições que tratam retornos que não são esperados, caso o usuário digite algo que não está na lista, ou uma palavra digitada errada.

- **CÓDIGO REFATORADO**

```
#Aqui estou importando a função exit para finalizar o programa, caso precise
from sys import exit

#Essa área é dedicada para organizar os menus e demais informações para chamar futuramente.
listaUnidadeMedida = '''
01 - Metros
02 - Centímetros
03 - Decímetros
04 - Milímetros
05 - Litros
06 - Mililitros
'''

menu = '''
01 - Área de quadrado
02 - Área de retângulo
03 - Área de triângulo
04 - Área de Trapézio
05 - Área do círculo
06 - Volume de cubo
07 - Volume de cilindro
08 - Volume de prisma
09 - Volume de trapézio
'''


class metros:
  unidade = "metros"
  simboloArea = "m²"
  simboloVolume = "m³"

class centimetros:
  unidade = "centímetros"
  simboloArea = "cm²"
  simboloVolume = "cm³"

class decimetros:
  unidade = "decímetros"
  simboloArea = "dm²"
  simboloVolume = "dm³"

class milimetros:
  unidade = "mm"
  simboloArea = "mm²"
  simboloVolume = "mm³"
```

Commented [Wd11]: Desta vez, estou utilizando nomes na mesma língua do programa e também utilizei o nome que se refere ao conteúdo dela.

Commented [Wd12]: Preferi criar uma variável para cada menu.
Assim eu consigo deixar organizado e de fácil localização.
Aqui eu consigo fazer alterações, melhorias e correções, diminuindo as chances de danificar o código e gerar erros, por ter apagado alguma letra ou outra coisa.

```
class litros:
  unidade = "litros"
  simboloVolume = "lt"

class mililitros:
  unidade = "mililitros"
  simboloVolume = "mi"

#Aqui ficam os textos que são padrão
escolherUnidade = "Digite o nome da unidade que você deseja trabalhar.\n{}"
escolherFormula = "Digite o número do item que você quer calcular.\n{}"

#Essa área é dedicada para o usuário definir qual conta deseja fazer.
unidadeEscolhida = str(input(escolherUnidade.format(listaUnidadeMedida)))

unidadeEscolhida.lower()

if unidadeEscolhida == "metros":
  unidadeEscolhida = metros()
  formulaEscolhida = int(input(escolherFormula.format(menu)))

elif unidadeEscolhida == "centímetros" or unidadeEscolhida == "centimetros":
  unidadeEscolhida = centimetros()
  formulaEscolhida = int(input(escolherFormula.format(menu)))

elif unidadeEscolhida == "decímetros" or unidadeEscolhida == "decimetros":
  unidadeEscolhida = decimetros()
  formulaEscolhida = int(input(escolherFormula.format(menu)))

elif unidadeEscolhida == "milímetros" or unidadeEscolhida == "milimetros":
  unidadeEscolhida = milimetros()
  formulaEscolhida = int(input(escolherFormula.format(menu)))

elif unidadeEscolhida == "litros":
  unidadeEscolhida = litros()
  formulaEscolhida = int(input(escolherFormula.format(menu)))

elif unidadeEscolhida == "mililitros":
  unidadeEscolhida = mililitros()
  formulaEscolhida = int(input(escolherFormula.format(menu)))

else:
  print("Essa opção não existe !\nPor favor, reinicie o programa.")
  exit()
```

Commented [Wd13]: Separei as unidades que serão disponibilizadas para o usuário, em classes.
Casa classe terá variáveis com o mesmo nome, mas com conteúdos específicos para aquela classe.
Isso vai me facilitar fazer manutenções futuras e adições.
Isso também minimiza a quantidade de nome de variáveis.

Commented [Wd14]: Para não ficar repetindo demais esses textos, eu criei variáveis com esse conteúdo.
Assim, eu só preciso chamar a variável.
Isso me facilita a digitar menos, fazer manutenções, correções e adições.

Commented [Wd15]: Nome de variável na mesma língua do programa e com nome que identifica a propósito dela.

Commented [Wd16]: Aqui eu estou tratando o retorno que o usuário vai me dar.
Evitando problemas onde o usuário não digitar exatamente da forma que eu programei, mesmo sendo a mesma palavra.
Um exemplo é em casos do usuário digitar tudo em minúsculo e o programa estiver esperando uma string com a primeira letra em maiúscula.

Commented [Wd17]: Desta vez, eu trabalhei com aninhamento.
Todas as condições estão chamando as variáveis que eu defini anteriormente.
Isso deixou o código mais encurtado e com poucas repetições.
Aqui estou reutilizando o mesmo nome de variável, mas que eslas recebem conteúdos automaticamento, dependendo do que o usuário me retornar.

Commented [Wd18]: Desta vez, estou tratando retornos que não estão sendo esperados pelo programa.
Decidi apresentar uma mensagem de erro e finalizar o código.
Fazendo o usuário reiniciar o programa.
Fiz isso porque ainda não aprendi outras formas de tratar isso, nas aulas.

```
#Essa área é dedicada para criar as classes de cada fórmula.
class areaQuadrado:
  formula = "A=a*b"
  def calcular(self):
    valorDe_a = float(input("Qual é o valor de a em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_b = float(input("Qual é o valor de b em {} ?\n".format(unidadeEscolhida.unidade)))
    print("A área é de {:.2f}{}".format(valorDe_a * valorDe_b, unidadeEscolhida.simboloArea))

class areaRetangulo:
  formula = "A=a*b"
  def calcular(self):
    valorDe_a = float(input("Qual é o valor de a em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_b = float(input("Qual é o valor de b em {} ?\n".format(unidadeEscolhida.unidade)))
    print("A área é de {:.2f}{}".format(valorDe_a * valorDe_b, unidadeEscolhida.simboloArea))

class areaTriangulo:
  formula = "A=a*b/2"
  def calcular(self):
    valorDe_a = float(input("Qual é o valor de a em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_b = float(input("Qual é o valor de b em {} ?\n".format(unidadeEscolhida.unidade)))
    print("A área é de {:.2f}{}".format(valorDe_a * valorDe_b / 2, unidadeEscolhida.simboloArea))

class areaTrapezio:
  formula = "A=(B+b)*h/2"
  def calcular(self):
    valorDe_B = float(input("Qual é o valor de B em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_b = float(input("Qual é o valor de b em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_h = float(input("Qual é o valor de h em {} ?\n".format(unidadeEscolhida.unidade)))
    print("A área é de {:.2f}{}".format((valorDe_B + valorDe_b) * valorDe_h / 2,
                                        unidadeEscolhida.simboloArea))

class areaCirculo:
  formula = "A=r²*PI/2"
  def calcular(self):
    valorDe_r = float(input("Qual é o valor de r em {} ?\n".format(unidadeEscolhida.unidade)))
    print("A área é de {:.2f}{}".format((valorDe_r ** 2) * 3.1415 / 2, unidadeEscolhida.simboloArea))

class volumeCubo:
  formula = "V=(a*b)*h"
  def calcular(self):
    valorDe_a = float(input("Qual é o valor de a em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_b = float(input("Qual é o valor de b em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_h = float(input("Qual é o valor de h em {} ?\n".format(unidadeEscolhida.unidade)))
    print("O volume é de {:.2f}{}".format((valorDe_a * valorDe_b) * valorDe_h,
                                          unidadeEscolhida.simboloVolume))

class volumeCilindro:
  formula = "V=(r²*PI/2)*h"
```

Commented [Wd19]: Estou utilizando nomes na mesma língua do programa e sem abraviação, facilitando o entendimento do que ela faz.

Commented [Wd20]: Aqui é um exemplo do que acontece ao longo de todo o código.
Estou chamando aquelas variáveis que defini anteriormente. O conteúdo é alterado de acordo com o retorno que tivermos do usuário.

```
  def calcular(self):
    valorDe_r = float(input("Qual é o valor de r em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_h = float(input("Qual é o valor de h em {} ?\n".format(unidadeEscolhida.unidade)))
    print("O volume é de {:.2f}{}".format(((valorDe_r ** 2) * 3.1415 / 2) * valorDe_h,
                                          unidadeEscolhida.simboloVolume))

class volumePrisma:
  formula = "V=(1/2)*b*h*i"
  def calcular(self):
    valorDe_b = float(input("Qual é o valor de b em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_h = float(input("Qual é o valor de h em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_i = float(input("Qual é o valor de i em {} ?\n".format(unidadeEscolhida.unidade)))
    print("O volume é de {:.2f}{}".format((1 / 2) * valorDe_b * valorDe_h * valorDe_i,
                                          unidadeEscolhida.simboloVolume))

class volumeTrapezio:
  formula = "V=((1/2)*h*(B+b))*w"
  def calcular(self):
    valorDe_h = float(input("Qual é o valor de h em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_B = float(input("Qual é o valor de B em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_b = float(input("Qual é o valor de b em {} ?\n".format(unidadeEscolhida.unidade)))
    valorDe_w = float(input("Qual é o valor de w em {} ?\n".format(unidadeEscolhida.unidade)))
    print("O volume é de {:.2f}{}".format((1 / 2) * valorDe_h * (valorDe_B + valorDe_b) * valorDe_w,
                                          unidadeEscolhida.simboloVolume))

#Essa área é dedicada para executar os calculos, de acordo com o retorno do usuário
if formulaEscolhida == 1:
  print("A fórmula para descobrir a área do quadrado é {}".format(areaQuadrado.formula))
  areaQuadrado.calcular(areaQuadrado)

elif formulaEscolhida == 2:
  print("A fórmula para descobrir a área do retângulo é {}".format(areaRetangulo.formula))
  areaRetangulo.calcular(areaRetangulo)

elif formulaEscolhida == 3:
  print("A fórmula para descobrir a área do triângulo é {}".format(areaTriangulo.formula))
  areaTriangulo.calcular(areaTriangulo)

elif formulaEscolhida == 4:
  print("A fórmula para descobrir a área do trapézio é {}".format(areaTrapezio.formula))
  areaTrapezio.calcular(areaTrapezio)

elif formulaEscolhida == 5:
  print("A fórmula para descobrir a área do círculo é {}".format(areaCirculo.formula))
  areaCirculo.calcular(areaCirculo)

elif formulaEscolhida == 6:
  print("A fórmula para descobrir o volume do cubo é {}".format(volumeCubo.formula))
  volumeCubo.calcular(volumeCubo)
```

Commented [Wd21]: Criei classes para cada tipo de fórmula.
Dentro delas, defini as funções com os mesmos nomes.
Isso me facilitou a deixar o código mais limpo e organizado.
Agora o programador consegue chamar as variáveis e funções, tranquilamente.
Isso também ajuda na manutenção e adição de mais funções e variáveis.

Commented [Wd22]: Estou chamando as classes para fazer os calculos.
Evitando digitar repetidamente e conteúdos diferentes, por conta do que o usuário nos retornar.
Consegui deixar o código mais limpor e organizado, desta forma.

```
elif formulaEscolhida == 7:
  print("A fórmula para descobrir o volume do cilindro é {}".format(volumeCilindro.formula))
  volumeCilindro.calcular(volumeCilindro)

elif formulaEscolhida == 8:
  print("A fórmula para descobrir o volume de um prisma é {}".format(volumePrisma.formula))
  volumePrisma.calcular(volumePrisma)

elif formulaEscolhida == 9:
  print("A fórmula para descobrir o volume de um trapezio é {}".format(volumeTrapezio.formula))
  volumeTrapezio.calcular(volumeTrapezio)

else:
  print("Essa função não existe!\nPor favor, reinicie o programa.")
  exit()
```

Commented [Wd23]: Aqui eu utilizei o aninhamento. Note que agora, está mais fácil de entender o que cada condição faz.
Aqui, a manutenção e correção é praticamente zero, de necessidade.
Qualquer necessidade de ajustar, adicionar ou corrigir, fica direcionada no início do código.

Commented [Wd24]: Agora temos um tratamento para retornos que o programa não está esperando

- **TESTES UNITARIOS**

```
import unittest
from unittest.mock import patch
from io import StringIO

# Mock para unidadeEscolhida
class UnidadeMock:
    unidade = "m"
    simboloArea = " m²"
    simboloVolume = " m³"

# Injetando o mock no escopo global
import builtins
builtins.unidadeEscolhida = UnidadeMock()

# Importar ou definir aqui as classes: areaQuadrado, areaRetangulo, etc.

class TestCalculos(unittest.TestCase):

    @patch('builtins.input', side_effect=["3", "4"])
    @patch('sys.stdout', new_callable=StringIO)
    def test_area_quadrado(self, mock_stdout, mock_input):
        obj = areaQuadrado()
        obj.calcular()
        self.assertIn("A área é de 12.00 m²", mock_stdout.getvalue())

    @patch('builtins.input', side_effect=["10", "2", "3"])
    @patch('sys.stdout', new_callable=StringIO)
    def test_volume_cubo(self, mock_stdout, mock_input):
```

```
        obj = volumeCubo()
        obj.calcular()
        self.assertIn("O volume é de 60.00 m³", mock_stdout.getvalue())

    @patch('builtins.input', side_effect=["5", "2"])
    @patch('sys.stdout', new_callable=StringIO)
    def test_area_triangulo(self, mock_stdout, mock_input):
        obj = areaTriangulo()
        obj.calcular()
        self.assertIn("A área é de 5.00 m²", mock_stdout.getvalue())

if __name__ == '__main__':
    unittest.main()
```

- **CONCLUSÃO**

Utilizar o clean code ajuda demais a organizar o código, porque além de ficar mais fácil de ler, ajuda na manutenção, nas correções e adições de linhas de códigos.

A facilidade que se tem para entender, se estende para outros programadores.

Qualquer um vai conseguir ler e entender o seu código.

Uma das coisa que mais notei, é a diminuição do trabalho excesivel de ficar digitando muito e códigos repetidos.

Deixar comentários ajuda demais no entendimento de outros programadores e até nós mesmo, na hora de relembrar porque foi feito tal linha de código.