



# TECNOLOGIA DA INFORMAÇÃO NO CANTEIRO DE OBRAS/CALCULO DE CUSTO DE MATERIAIS

Caio Henrique Pinheiro Oliveira, Igor Pinto dos Santos

# INTRODUÇÃO

Tecnologia da informação na construção civil também se beneficia do uso da linguagem de programação Python. Python é amplamente utilizado para automatizar tarefas repetitivas, processar dados e criar ferramentas personalizadas para análise e modelagem. Sua flexibilidade e vasta biblioteca de recursos tornam Python uma escolha popular para desenvolver soluções de software que podem melhorar a eficiência no canteiro de obras. Além disso, a aplicação de aprendizado de máquina e análise de dados em Python permite prever problemas antes que ocorram, otimizando ainda mais os processos de construção. Assim, Python desempenha um papel importante na integração da tecnologia da informação na indústria da construção, contribuindo para a automação e aprimoramento contínuo do setor.

# OBJETIVO

O objetivo principal deste trabalho é investigar e demonstrar como a programação em Python pode ser aplicada para atingir os seguintes objetivos na construção civil:

- Automação de Tarefas Repetitivas: Utilizar scripts em Python para automatizar tarefas rotineiras, como o processamento de documentos, relatórios e medições. Isso economiza tempo e reduz erros humanos.
- Gerenciamento de Projetos: Desenvolver aplicativos em Python para gerenciar projetos de construção, incluindo o acompanhamento do progresso, alocação de recursos e agendamento de tarefas.
- Sustentabilidade e Eficiência Energética: Desenvolver soluções em Python para otimizar o consumo de energia e recursos naturais, tornando os edifícios mais sustentáveis.

# PROGRAMAS UTILIZADOS

Contudo, partindo desse ponto pode-se pensar no Python como uma ferramenta facilitadora desse processo, pois Python é uma linguagem de programação de nível alto, que usa instruções humanas e mais abstratas, incrementada através de um conjunto de instruções que permitem que uma máquina execute tarefas, como a calculadora de custo de materiais no canteiro de obras

# OUTRAS FERRAMENTAS

Além do Python, várias ferramentas desempenharam um papel fundamental no desenvolvimento deste projeto:

- Visual Studio Code (VSCode)
- Git e GitHub
- Trello
- Overleaf
- Bibliotecas Python ('tkinter', 'openpyxl', 'os' )


# PROPOSTA DE PROGRAMA

A proposta deste programa é fornecer aos profissionais da construção civil, engenheiros, arquitetos, empreiteiros e outros envolvidos na área uma ferramenta simples e eficaz para calcular o custo total de materiais de construção. A aplicação visa melhorar a eficiência, economizar tempo e reduzir erros comuns na estimativa de custos.

Ao desenvolver um trabalho com o tema "Tecnologia da Informação no Canteiro de Obras.", é essencial seguir uma metodologia estruturada para garantir a clareza, coesão e relevância do trabalho.

- Introdução
- Revisão Bibliográfica
- Fundamentação Teórica
- Metodologia de Pesquisa
- Desenvolvimento do projeto
- Resultados e Discussões
- Conclusão
- Referências Bibliográficas

# CONOGRAMA

Discriminação 	1º mês				2º mês				3º mês				4º mês				5º mês			
Aprf.teórico	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Coleta de dados			x	x	x	x														
Trat.de dados					x	x	x	x	x											
Análise de dados						x	x	x	x	x										
Dev. do trabalho									x	x	x	x	x	x	x					
Considerações finais													x	x	x	x				
Referências															x	x	x	x	x	
Trabalho Acadêmicos										x	x	x	x	x	x	x	x	x	x	x



# SCRIPT CÓDIGO 1

CODIGO PROJETO FINAL.py

C: > Users > Caio > Downloads > CODIGO PROJETO FINAL.py > ...

```
1  # Importa o modulo 'os' para interagir com funcionalidades específicas do sistema operacional
2  import os
3  # Importa a biblioteca openpyxl utilizada para trabalhar com arquivos Excel (.xlsx)
4  import openpyxl
5  # Importa as bibliotecas tkinter e ttk, que são usadas para criar a interface gráfica
6  import tkinter as tk
7  from tkinter import ttk
8
9  # Define a classe MaterialConstrucao
10 class MaterialConstrucao:
11     def __init__(self, nome, preco_unitario, quantidade):
12         # Inicializa um objeto MaterialConstrucao com nome, preço unitário e quantidade
13         self.nome = nome
14         self.preco_unitario = preco_unitario
15         self.quantidade = quantidade
16
17     def calcular_custo_total(self):
18         # Calcula o custo total multiplicando o preço unitário pela quantidade
19         return self.preco_unitario * self.quantidade
20
21 # Dicionário para armazenar informações sobre os materiais
22 materiais = {
23     "cimento": {"preco_unitario": 115.0},
24     "tijolos": {"preco_unitario": 2.5},
25     "madeira": {"preco_unitario": 18.5},
```

# SCRIPT CÓDIGO 2

```
24     "tijolos": {"preco_unitario": 2.5},
25     "madeira": {"preco_unitario": 18.5},
26     "telha ceramica": {"preco_unitario": 1.99},
27     "gesso": {"preco_unitario": 38.5}
28 }
29
30 historico_calculos = [] # Lista para armazenar o histórico de cálculos
31
32 # Função para calcular o custo
33 def calcular_custo():
34     nome_material = nome_entry.get() # Obtém o nome do material da entrada de texto
35     preco_unitario_str = preco_unitario_entry.get() # Obtém o preço unitário da entrada de texto
36     quantidade_str = quantidade_entry.get() # Obtém a quantidade da entrada de texto
37
38     if not nome_material or not preco_unitario_str or not quantidade_str:
39         # Verifica se algum dos campos está vazio e exibe uma mensagem de erro
40         resultado_label.config(
41             text="Por favor, preencha todos os campos.", foreground="red")
42         return
43
44     if nome_material not in materiais:
45         # Verifica se o material não está no dicionário de materiais e exibe uma mensagem de erro
46         resultado_label.config(text="Material desconhecido.", foreground="red")
```

# SCRIPT CÓDIGO 3

```
if nome_material not in materiais:
    # Verifica se o material não está no dicionário de materiais e exibe uma mensagem de erro
    resultado_label.config(text="Material desconhecido.", foreground="red")
    return

try:
    preco_unitario = float(preco_unitario_str) # Tenta converter o preço unitário em um número de ponto flutuante
except :
    # Se a conversão falhar, exibe uma mensagem de erro
    resultado_label.config(
        text="Preço unitário inválido. Insira um número válido.", foreground="red")
    return

try:
    quantidade = int(quantidade_str) # Tenta converter a quantidade em um número inteiro
except :
    # Se a conversão falhar, exibe uma mensagem de erro
    resultado_label.config(
        text="Quantidade inválida. Insira um número inteiro válido.", foreground="red")
    return

material = MaterialConstrucao(
    nome_material, materiais[nome_material][["preco_unitario"], quantidade)
custo_total = material.calcular_custo_total() # Calcula o custo total do material

nome_material = nome_material.upper() # Converte o nome do material para maiúsculas
resultado_label.config(text=f"O custo total do material {nome_material} é de R$ {custo_total:.2f}", foreground="green")
```

# SCRIPT CÓDIGO 4

```
# Limpa as entradas de texto
nome_entry.delete(0, tk.END)
preco_unitario_entry.delete(0, tk.END)
quantidade_entry.delete(0, tk.END)

# Adiciona a string ao histórico de cálculos
historico_calculos.append(
    f"Material: {nome_material}, Custo: R$ {custo_total:.2f}")

historico_text.config(state=tk.NORMAL)
historico_text.delete("1.0", tk.END) # Limpa o histórico anterior

for calculo in historico_calculos:
    # Adiciona os cálculos anteriores ao histórico
    historico_text.insert(tk.END, calculo + "\n")

historico_text.config(state=tk.DISABLED)

# Define uma função para adicionar o histórico ao arquivo Excel
def adicionar_historico_excel():
    try:
        # Nome do arquivo Excel
        arquivo_excel = "Orcamento_de_Obra.xlsx"

        # Verifica se o arquivo Excel existe
        if not os.path.exists(arquivo_excel):
            # Se não existir, cria um novo arquivo Excel e adiciona cabeçalho
            print("Arquivo Excel não encontrado. Criando um novo.")
            workbook = openpyxl.Workbook()
```

# SCRIPT CÓDIGO 5

```
print("Arquivo Excel não encontrado. Criando um novo.")
workbook = openpyxl.Workbook()
sheet = workbook.active
sheet.append(["Material", "Custo"])
else:
    # Se o arquivo existir, carrega o workbook existente
    workbook = openpyxl.load_workbook(arquivo_excel)
    # Verifica se a planilha 'Sheet' já existe, senão cria uma nova
    sheet = workbook.active if "Sheet" in workbook.sheetnames else workbook.create_sheet("Sheet")

# Itera sobre os cálculos no histórico
for calculo in historico_calculos:
    # Divide a string do cálculo em material e custo
    material, custo_str = calculo.split(',')
    # Converte a string do custo para float, mantendo apenas dígitos e o ponto decimal
    custo = float(''.join(filter(str.isdigit, custo_str)))
    # Adiciona as informações à planilha
    sheet.append([material.split(':')[1], custo])

# Salva as alterações no arquivo Excel
workbook.save(arquivo_excel)

# Mensagem de sucesso
print("Histórico adicionado ao Excel com sucesso.")
print("Caminho absoluto do arquivo Excel:", os.path.abspath(arquivo_excel))
except Exception as e:
    # Em caso de erro, exibe uma mensagem de erro
    print(f"Erro ao salvar no Excel: {e}")

# Configuração da interface gráfica
root = tk.Tk() # Cria a janela principal
```

# SCRIPT CÓDIGO 6

```
nome_entry = ttk.Entry(frame1) # Entrada de texto para o nome do material
preco_unitario_entry = ttk.Entry(frame1) # Entrada de texto para o preço unitário
quantidade_entry = ttk.Entry(frame1) # Entrada de texto para a quantidade

# Posições das entradas de texto
nome_entry.grid(row=0, column=1, sticky=tk.W)
preco_unitario_entry.grid(row=1, column=1, sticky=tk.W)
quantidade_entry.grid(row=2, column=1, sticky=tk.W)

# Botão para calcular o custo
calcular_button = ttk.Button(
    frame1, text="Calcular Custo", command=calcular_custo)
calcular_button.grid(row=3, column=1)

# Botão para adicionar ao histórico no Excel
adicionar_excel_button = ttk.Button(frame1, text="Adicionar ao Excel", command=adicionar_historico_excel)
adicionar_excel_button.grid(row=7, column=1)

resultado_label = ttk.Label(frame1, text="") # Rótulo para exibir o resultado
resultado_label.grid(row=4, column=1, sticky=tk.W) # Posição do rótulo de resultado

historico_label = ttk.Label(frame1, text="Histórico de Cálculos:") # Rótulo para o histórico
historico_label.grid(row=5, column=0, columnspan=2, sticky=tk.W) # Posição do rótulo de histórico

historico_text = tk.Text(frame1, state=tk.DISABLED,
    wrap=tk.WORD, height=5, width=40) # Área de texto para o histórico
historico_text.grid(row=6, column=0, columnspan=2) # Posição da área de texto

root.mainloop() # Inicia o loop principal da interface gráfica
```

# SCRIPT CALCULADORA

Calculadora de Custo de Materiais de C...

Nome do material:

Preço unitário (R\$):

Quantidade:

Calcular Custo

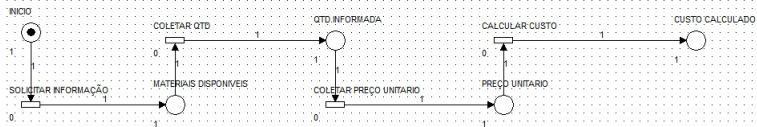
Resultado: O custo total do material TIJOLOS é de R\$ 1000.00

Histórico de Cálculos:

Material: CIMENTO, Custo: R\$ 17250.00  
Material: GESSO, Custo: R\$ 7700.00  
Material: TIJOLOS, Custo: R\$ 1000.00

Adicionar ao Excel

# SCRIPT REDE DE PETRI





# CONCLUSÃO

Este projeto exemplifica a aplicação prática da tecnologia da informação na construção civil por meio de uma "Calculadora de Custo de Materiais de Construção" em Python. Destaca a importância da pesquisa para identificar tendências e desafios nessa integração da TI na indústria. A tecnologia demonstrada simplifica o cálculo de custos, economizando tempo e reduzindo erros. Em resumo, ressalta a relevância da TI na modernização e melhoria da eficiência na construção civil.

# REFERÊNCIAS

Como Sair do ZERO no Python em APENAS UMA AULA.

Disponível em:

<https://www.youtube.com/watch?v=GQpQha2Mfpg>. Acesso em: 10/09/23

Como Criar uma Tela em Python Para Seus Códigos - [Interface Gráfica Intuitiva com Tkinter]. Disponível em:

[https://youtu.be/AiBC01p58ol?si=AUXfUH\\_a-ZkIaw9X](https://youtu.be/AiBC01p58ol?si=AUXfUH_a-ZkIaw9X). Acesso em : 14/10/23

Integração entre Python e Excel usando Pandas e o Openpyxl. Disponível

em: <https://youtu.be/IT7zPluDADk?si=hINH9nBtrBP8JtZY>. Acesso em 29/11/23

FEREGUETTI, Larissa. O que é a language Pmython e como ela está presente na engenharia?. [S. l.], 12 maio 2019. Disponível em: <https://engenharia360.com/linguagem-python-na-engenharia/>.

Acesso em: 05/10/23.

COUTINHO, Thiago. Como o Python pode ajudar um engenheiro e quais as suas aplicações na engenharia?. [S. l.], 18 mar. 2021.

Disponível em:

<https://www.voitto.com.br/blog/artigo/python-para-engenheiros>.

Acesso em: 23/09/23.