

Questão 1

Observe o trecho de código abaixo:

int INDICE = 13,

SOMA = 0,

K = 0;

enquanto K < INDICE faça { K = K + 1; SOMA = SOMA + K; }

imprimir(SOMA);

Ao final do processamento, qual será o valor da variável SOMA?

Um loop está somando os números de 1 até um determinado índice, neste caso, 13.

Explicação linha por linha:

- **int INDICE = 13, SOMA = 0, K = 0;** Isso declara três variáveis inteiras: **INDICE**, **SOMA** e **K**, e as inicializa com os valores 13, 0 e 0, respectivamente.
- **enquanto K < INDICE faça:** Este é um loop enquanto. Enquanto a condição **K < INDICE** for verdadeira, o bloco de código dentro do loop será executado.
- **{**: Início do bloco de código do loop.
- **K = K + 1;** Incrementa o valor de **K** em 1 a cada iteração do loop.
- **SOMA = SOMA + K;** Adiciona o valor de **K** à variável **SOMA** em cada iteração do loop. Isso acumula a soma dos números de 1 até o valor de **INDICE**.
- **}**: Fim do bloco de código do loop.

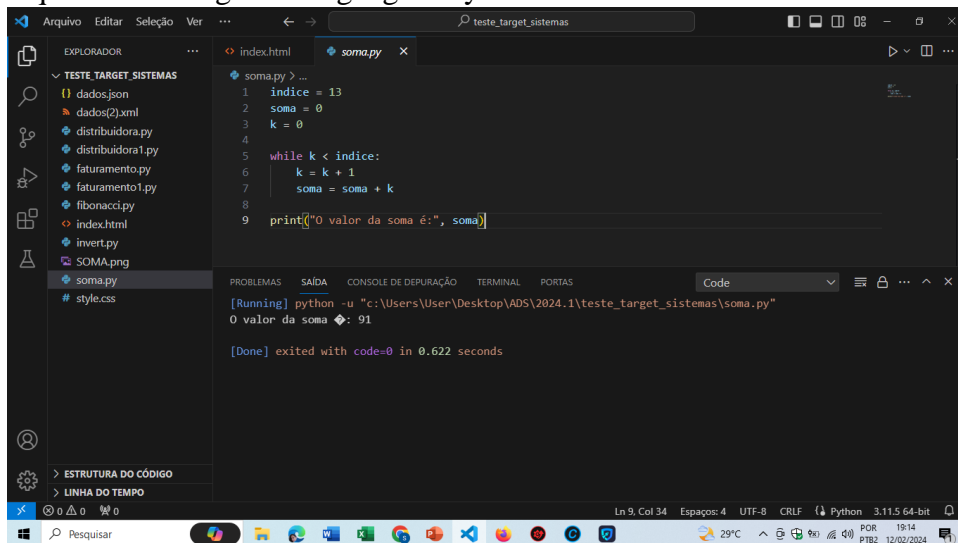
Neste código, **K** é uma variável de controle do loop que vai de 1 até 13, e **SOMA** é uma variável que acumula a soma dos números de 1 até 13. No final da execução do loop, **SOMA** conterá o valor da soma dos números de 1 até 13.

Para determinar o valor da soma, vamos executar o código linha por linha:

- Inicialmente, **K** é 0 e **SOMA** é 0.
- No primeiro ciclo do loop (**K = 0 + 1 = 1**), **SOMA** é **0 + 1 = 1**.
- No segundo ciclo do loop (**K = 1 + 1 = 2**), **SOMA** é **1 + 2 = 3**.
- No terceiro ciclo do loop (**K = 2 + 1 = 3**), **SOMA** é **3 + 3 = 6**.
- No quarto ciclo do loop (**K = 3 + 1 = 4**), **SOMA** é **6 + 4 = 10**.
- No quinto ciclo do loop (**K = 4 + 1 = 5**), **SOMA** é **10 + 5 = 15**.
- No sexto ciclo do loop (**K = 5 + 1 = 6**), **SOMA** é **15 + 6 = 21**.
- No sétimo ciclo do loop (**K = 6 + 1 = 7**), **SOMA** é **21 + 7 = 28**.
- No oitavo ciclo do loop (**K = 7 + 1 = 8**), **SOMA** é **28 + 8 = 36**.
- No nono ciclo do loop (**K = 8 + 1 = 9**), **SOMA** é **36 + 9 = 45**.
- No décimo ciclo do loop (**K = 9 + 1 = 10**), **SOMA** é **45 + 10 = 55**.
- No décimo primeiro ciclo do loop (**K = 10 + 1 = 11**), **SOMA** é **55 + 11 = 66**.
- No décimo segundo ciclo do loop (**K = 11 + 1 = 12**), **SOMA** é **66 + 12 = 78**.
- No décimo terceiro ciclo do loop (**K = 12 + 1 = 13**), **SOMA** é **78 + 13 = 91**.

Portanto, o valor da soma é 91.

Aqui está o código em linguagem Python:



```
1 indice = 13
2 soma = 0
3 k = 0
4
5 while k < indice:
6     k = k + 1
7     soma = soma + k
8
9 print(f"0 valor da soma é:", soma)
```

PROBLEMAS SAÍDA CONSOLE DE DEPUAÇÃO TERMINAL PORTAS Code

[Running] python -u "c:\Users\User\Desktop\ADS\2024.1\teste_target_sistemas\soma.py"

0 valor da soma é: 91

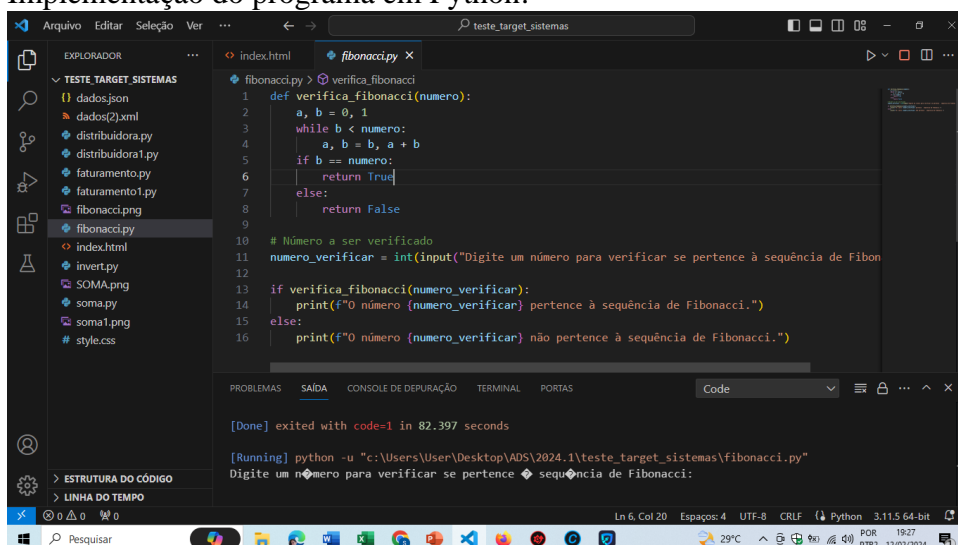
[Done] exited with code=0 in 0.622 seconds

Este código produzirá a mesma saída, com o valor da soma sendo exibido como 91.

Questão 2

Dado a sequência de Fibonacci, onde se inicia por 0 e 1 e o próximo valor sempre será a soma dos 2 valores anteriores (exemplo: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34...), escreva um programa na linguagem que desejar onde, informado um número, ele calcule a sequência de Fibonacci e retorne uma mensagem avisando se o número informado pertence ou não a sequência. **IMPORTANTE:** Esse número pode ser informado através de qualquer entrada de sua preferência ou pode ser previamente definido no código;

Implementação do programa em Python:



```
1 def verifica_fibonacci(numero):
2     a, b = 0, 1
3     while b < numero:
4         a, b = b, a + b
5     if b == numero:
6         return True
7     else:
8         return False
9
10 # Número a ser verificado
11 numero_verificar = int(input("Digite um número para verificar se pertence à sequência de Fibonacci: "))
12
13 if verifica_fibonacci(numero_verificar):
14     print(f"0 número {numero_verificar} pertence à sequência de Fibonacci.")
15 else:
16     print(f"0 número {numero_verificar} não pertence à sequência de Fibonacci.")
```

PROBLEMAS SAÍDA CONSOLE DE DEPUAÇÃO TERMINAL PORTAS Code

[Done] exited with code=1 in 82.397 seconds

[Running] python -u "c:\Users\User\Desktop\ADS\2024.1\teste_target_sistemas\fibonacci.py"

Digite um número para verificar se pertence à sequência de Fibonacci: 1

sequência de Fibonacci

Neste código, a função **verifica_fibonacci** analisa se um número pertence à sequência de Fibonacci. A função itera através da sequência de Fibonacci até encontrar um número maior

ou igual ao número dado. Se o número dado for encontrado na sequência, a função retorna **True**, caso contrário, retorna **False**. Em seguida, pedimos ao usuário para digitar um número e verificamos se ele pertence à sequência de Fibonacci utilizando essa função.

A implementação dessa função chamada **verifica_fibonacci** determina se um número dado pertence à sequência de Fibonacci. A sequência de Fibonacci é uma sequência de números onde cada número subsequente é a soma dos dois números anteriores na sequência. A sequência começa com os números 0 e 1.

A função **verifica_fibonacci** recebe um número como entrada e utiliza duas variáveis, **a** e **b**, para representar os dois números mais recentes da sequência de Fibonacci. Inicialmente, **a** é atribuído como 0 e **b** como 1. Em seguida, um loop **while** é usado para calcular os números da sequência de Fibonacci até que **b** se torne maior ou igual ao número dado como entrada.

Dentro do loop, a técnica de atribuição múltipla é utilizada para atualizar os valores de **a** e **b**, onde **a** recebe o valor de **b** e **b** recebe a soma dos valores antigos de **a** e **b**. Isso é feito repetidamente até que **b** seja maior ou igual ao número dado.

Após sair do loop, a função verifica se **b** é igual ao número fornecido. Se for o caso, isso significa que o número está presente na sequência de Fibonacci e a função retorna **True**. Caso contrário, a função retorna **False**, indicando que o número não está na sequência de Fibonacci.

Finalmente, o código solicita ao usuário que insira um número para verificar se ele pertence à sequência de Fibonacci. Dependendo do resultado da função **verifica_fibonacci**, uma mensagem apropriada é exibida informando se o número pertence ou não à sequência de Fibonacci.

Questão 3

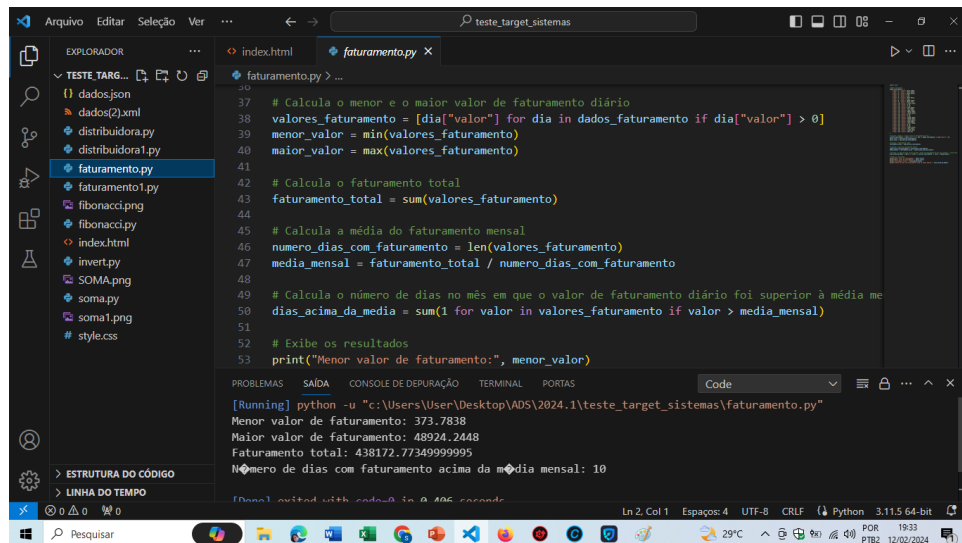
Dado um vetor que guarda o valor de faturamento diário de uma distribuidora, faça um programa, na linguagem que desejar, que calcule e retorne:

- O menor valor de faturamento ocorrido em um dia do mês;
- O maior valor de faturamento ocorrido em um dia do mês;
- Número de dias no mês em que o valor de faturamento diário foi superior à média mensal.

IMPORTANTE:

- a) Usar o json ou xml disponível como fonte dos dados do faturamento mensal;
- b) Podem existir dias sem faturamento, como nos finais de semana e feriados. Estes dias devem ser ignorados no cálculo da média;

-----Utilizando JSON-----



```
37 # Calcula o menor e o maior valor de faturamento diário
38 valores_faturamento = [dia["valor"] for dia in dados_faturamento if dia["valor"] > 0]
39 menor_valor = min(valores_faturamento)
40 maior_valor = max(valores_faturamento)
41
42 # Calcula o faturamento total
43 faturamento_total = sum(valores_faturamento)
44
45 # Calcula a média do faturamento mensal
46 numero_dias_com_faturamento = len(valores_faturamento)
47 media_mensal = faturamento_total / numero_dias_com_faturamento
48
49 # Calcula o número de dias no mês em que o valor de faturamento diário foi superior à média mensal
50 dias_acima_da_media = sum(1 for valor in valores_faturamento if valor > media_mensal)
51
52 # Exibe os resultados
53 print("Menor valor de faturamento:", menor_valor)
54
55 [Running] python -u "c:\Users\User\Desktop\ADS\2024.1\teste_target_sistemas\faturamento.py"
Menor valor de faturamento: 373.7838
Maior valor de faturamento: 48924.2448
Faturamento total: 438172.77349999995
Número de dias com faturamento acima da média mensal: 10
```

O código em Python utiliza dados de faturamento representados em formato JSON para calcular e exibir diversas métricas relacionadas ao faturamento diário de uma distribuidora ao longo de um mês.

- **Dados Fornecidos:**

- Os dados de faturamento são fornecidos como uma lista de dicionários em Python.
- Cada dicionário na lista representa um dia do mês, com a chave "dia" indicando o número do dia e a chave "valor" indicando o valor do faturamento para aquele dia.

- **Cálculo do Menor e Maior Valor de Faturamento Diário:**

- O código utiliza compreensão de lista para criar uma lista apenas com os valores de faturamento maiores que zero.
- O menor valor de faturamento diário é calculado utilizando a função **min()** e a lista de valores filtrados.
- O maior valor de faturamento diário é calculado utilizando a função **max()** e a mesma lista de valores filtrados.

- **Cálculo do Faturamento Total:**

- O faturamento total é calculado somando todos os valores de faturamento diário presentes na lista de valores filtrados.

- **Cálculo da Média do Faturamento Mensal:**

- A média do faturamento mensal é calculada dividindo o faturamento total pelo número de dias com faturamento.
- O número de dias com faturamento é determinado pelo comprimento da lista de valores filtrados.

- **Cálculo do Número de Dias com Faturamento Acima da Média Mensal:**

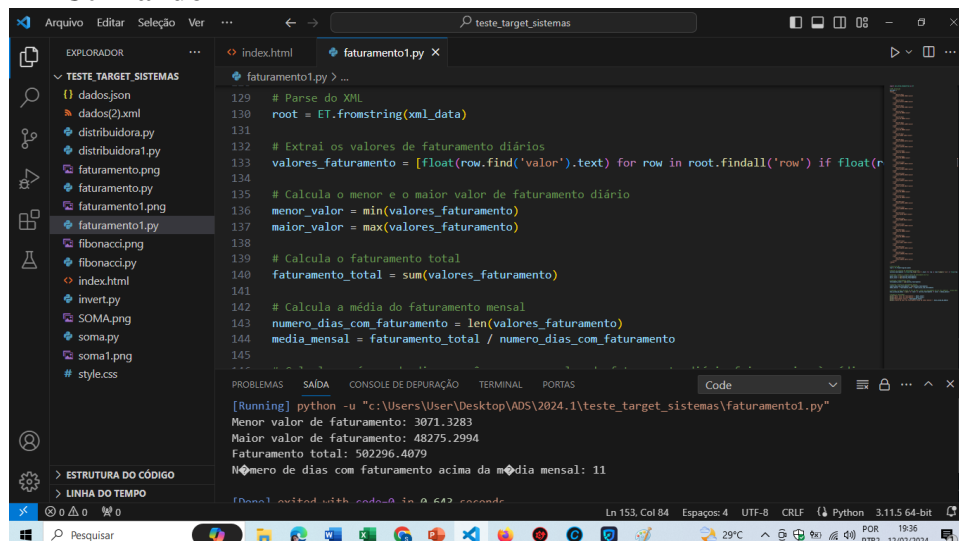
- Utilizando uma expressão geradora, o código conta quantos valores de faturamento diário são maiores que a média mensal calculada anteriormente.

- **Exibição dos Resultados:**

- Por fim, o código imprime na tela os resultados calculados, incluindo o menor valor de faturamento, o maior valor de faturamento, o faturamento total e o número de dias com faturamento acima da média mensal.

Essencialmente, o código automatiza o processo de análise dos dados de faturamento, fornecendo informações importantes para a distribuidora sobre o desempenho ao longo do mês.

-----Utilizando XML-----



```
Arquivo Editar Seleção Ver ... teste_target_sistemas
EXPLORADOR
TESTE_TARGET_SISTEMAS
  dados.json
  dados(2).xml
  distribuidora.py
  distribuidora1.py
  faturamento.png
  faturamento.py
  faturamento1.png
  faturamento1.py
  fibonacci.png
  fibonacci.py
  index.html
  invert.py
  SOMA.png
  soma.py
  soma1.png
  style.css
ESTRUTURA DO CÓDIGO
LINHA DO TEMPO
faturamento1.py
129 # Parse do XML
130 root = ET.fromstring(xml_data)
131
132 # Extraí os valores de faturamento diários
133 valores_faturamento = [float(row.find('valor').text) for row in root.findall('row') if float(r
134
135 # Calcula o menor e o maior valor de faturamento diário
136 menor_valor = min(valores_faturamento)
137 maior_valor = max(valores_faturamento)
138
139 # Calcula o faturamento total
140 faturamento_total = sum(valores_faturamento)
141
142 # Calcula a média do faturamento mensal
143 numero_dias_com_faturamento = len(valores_faturamento)
144 media_mensal = faturamento_total / numero_dias_com_faturamento
145
PROBLEMAS SAÍDA CONSOLE DE DEPUÇÃO TERMINAL PORTAS
[Running] python -u "C:\Users\User\Desktop\ADS\2024.1\teste_target_sistemas\faturamento1.py"
Menor valor de faturamento: 3071.3283
Maior valor de faturamento: 48275.2994
Faturamento total: 502296.4079
Número de dias com faturamento acima da média mensal: 11
Ln 153, Col 84 Espaços: 4 UTF-8 CRLF Python 3.11.5 64-bit
```

O código em Python utiliza a biblioteca **xml.etree.ElementTree** para processar dados fornecidos em formato XML.

- **XML Fornecido:**

- Os dados de faturamento são fornecidos como uma string contendo dados em formato XML.
- Cada nó **<row>** representa um dia do mês, com subelementos **<dia>** e **<valor>** indicando o número do dia e o valor do faturamento para aquele dia, respectivamente.

- **Parse do XML:**

- Utilizando **ET.fromstring()**, o código faz o parse da string XML e cria uma estrutura de dados hierárquica que representa o XML.

- **Extração dos Valores de Faturamento Diários:**

- O código utiliza uma expressão de compreensão de lista para iterar sobre todos os elementos **<row>** do XML.
- Para cada elemento **<row>**, é verificado se o valor de faturamento é maior que zero (considerando apenas dias com faturamento).
- Os valores de faturamento maiores que zero são armazenados em uma lista.

- **Cálculo do Menor e Maior Valor de Faturamento Diário:**

- Utilizando as funções **min()** e **max()**, o código calcula o menor e o maior valor de faturamento presentes na lista de valores filtrados.

- **Cálculo do Faturamento Total:**

- O faturamento total é calculado somando todos os valores de faturamento diário presentes na lista de valores filtrados.

- **Cálculo da Média do Faturamento Mensal:**

- A média do faturamento mensal é calculada dividindo o faturamento total pelo número de dias com faturamento.
- O número de dias com faturamento é determinado pelo comprimento da lista de valores filtrados.

- **Cálculo do Número de Dias com Faturamento Acima da Média Mensal:**

- Utilizando uma expressão geradora, o código conta quantos valores de faturamento diário são maiores que a média mensal calculada anteriormente.

- **Exibição dos Resultados:**

- Por fim, o código imprime na tela os resultados calculados, incluindo o menor valor de faturamento, o maior valor de faturamento, o faturamento total e o número de dias com faturamento acima da média mensal.

Este código automatiza a análise dos dados de faturamento presentes no XML, fornecendo informações importantes sobre o desempenho ao longo do mês.

Questão 4

Dado o valor de faturamento mensal de uma distribuidora, detalhado por estado:

SP – R\$67.836,43

RJ – R\$36.678,66

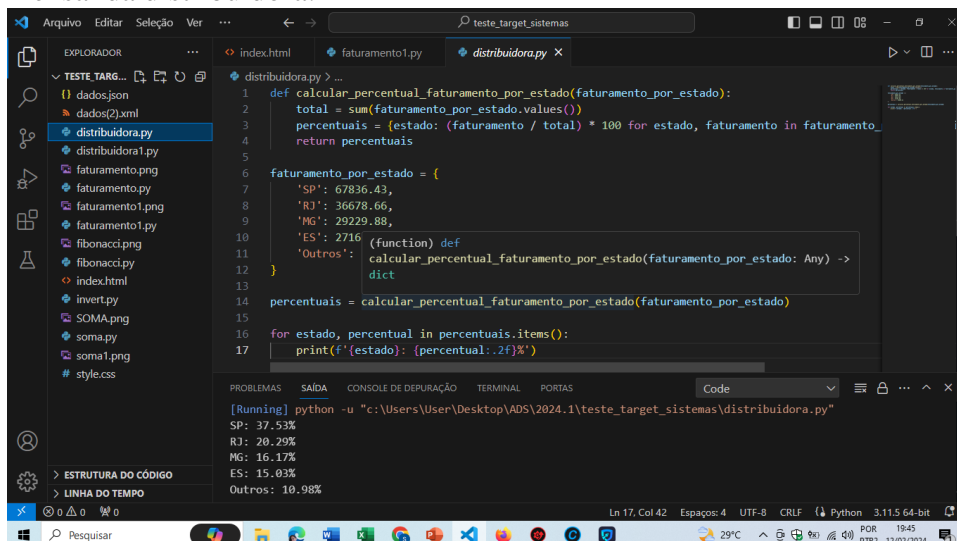
MG – R\$29.229,88

ES – R\$27.165,48

Outros – R\$19.849,53

Escreva um programa na linguagem que desejar onde calcule o percentual de representação que cada estado teve dentro do valor total mensal da distribuidora.

Programa em Python que calcula o percentual de representação de cada estado no valor total mensal da distribuidora:



```
1 def calcular_percentual_faturamento_por_estado(faturamento_por_estado):
2     total = sum(faturamento_por_estado.values())
3     percentuais = {estado: (faturamento / total) * 100 for estado, faturamento in faturamento_por_estado.items()}
4     return percentuais
5
6 faturamento_por_estado = {
7     'SP': 67836.43,
8     'RJ': 36678.66,
9     'MG': 29229.88,
10    'ES': 27165.48,
11    'Outros': 19849.53
12 }
13
14 percentuais = calcular_percentual_faturamento_por_estado(faturamento_por_estado)
15
16 for estado, percentual in percentuais.items():
17     print(f'{estado}: {percentual:.2f}%')
```

PROBLEMAS SAÍDA CONSOLE DE DEPUÇÃO TERMINAL PORTAS Code

[Running] python -u "c:\Users\User\Desktop\ADS\2024.1\teste_target_sistemas\distribuidora.py"

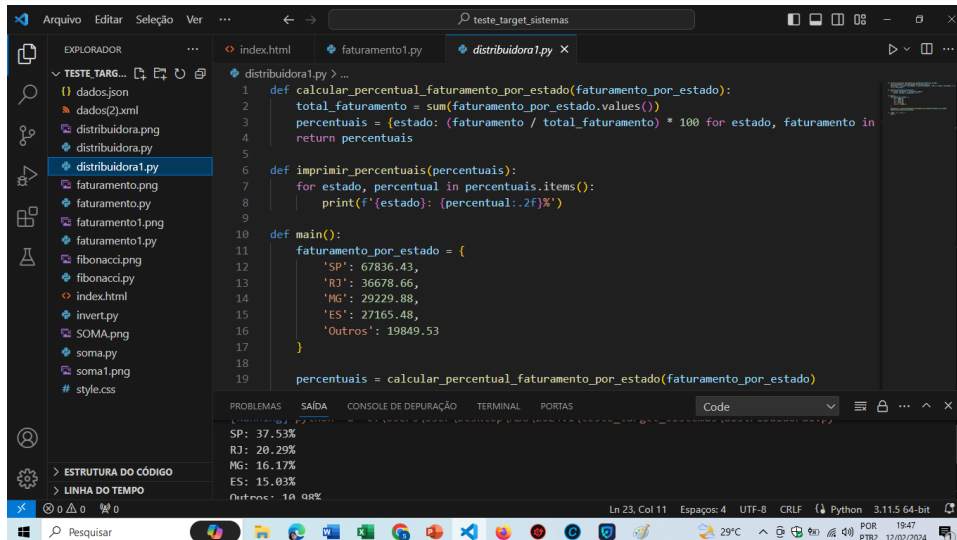
SP: 37.53%
RJ: 20.29%
MG: 16.17%
ES: 15.03%
Outros: 10.98%

Este código define uma função **calcular_percentual_faturamento_por_estado** que recebe um dicionário contendo o faturamento de cada estado como entrada e retorna um novo dicionário com os percentuais de representação de cada estado no faturamento total.

Em seguida, definimos o faturamento por estado em um dicionário e chamamos a função **calcular_percentual_faturamento_por_estado** com este dicionário. Por fim, iteramos sobre o dicionário de percentuais resultante e imprimimos os resultados formatados.

O programa foi otimizado para organizar o código de forma a seguir boas práticas de programação e melhorar a estrutura para torná-lo mais claro e legível:

Neste código:



```
1 def calcular_percentual_faturamento_por_estado(faturamento_por_estado):
2     total_faturamento = sum(faturamento_por_estado.values())
3     percentuais = {estado: (faturamento / total_faturamento) * 100 for estado, faturamento in
4                       faturamento_por_estado.items()}
5     return percentuais
6
7 def imprimir_percentuais(percentuais):
8     for estado, percentual in percentuais.items():
9         print(f'{estado}: {percentual:.2f}%')
10
11 def main():
12     faturamento_por_estado = {
13         'SP': 67836.43,
14         'RJ': 36678.66,
15         'MG': 29229.88,
16         'ES': 27165.48,
17         'Outros': 19849.53
18     }
19     percentuais = calcular_percentual_faturamento_por_estado(faturamento_por_estado)
20     imprimir_percentuais(percentuais)
```

- A função **calcular_percentual_faturamento_por_estado** calcula os percentuais de faturamento de cada estado em relação ao total.
 - A função **imprimir_percentuais** imprime os resultados formatados.
 - A função **main** é a função principal do programa, onde definimos o faturamento por estado, calculamos os percentuais e os imprimimos.
 - Utilizamos a condição **if __name__ == "__main__":** para garantir que o código dentro de **main()** só seja executado se o script for executado diretamente, não se for importado como um módulo em outro script. Isso é uma boa prática em Python.
- Essas mudanças tornam o código mais modular e fácil de entender, facilitando a manutenção e a leitura por outros programadores.

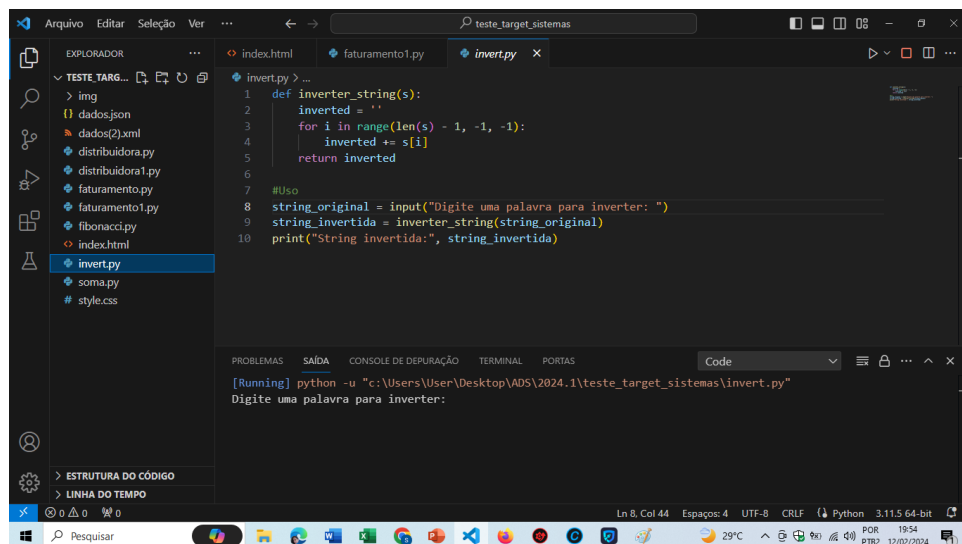
Questão 5

Escreva um programa que inverta os caracteres de um string.

IMPORTANTE:

- a) Essa string pode ser informada através de qualquer entrada de sua preferência ou pode ser previamente definida no código;
- b) Evite usar funções prontas, como, por exemplo, **reverse**;

Programa em Python que inverte os caracteres de uma string sem usar funções prontas como **reverse**:



```
Arquivo  Editar  Seleção  Ver  ...  teste_target_sistemas

EXPLORADOR
TESTE_TARG...
  > img
  {} dados.json
  > dados[2].xml
  distribuidora.py
  distribuidora1.py
  faturamento.py
  faturamento1.py
  fibonacci.py
  index.html
  invert.py
  soma.py
  style.css

index.html
faturamento1.py
invert.py
soma.py
style.css

1 def inverter_string(s):
2     inverted = ''
3     for i in range(len(s) - 1, -1, -1):
4         inverted += s[i]
5     return inverted
6
7 #Uso
8 string_original = input("Digite uma palavra para inverter: ")
9 string_invertida = inverter_string(string_original)
10 print("String invertida:", string_invertida)

PROBLEMAS  SAÍDA  CONSOLE DE DEPUÇÃO  TERMINAL  PORTAS
[Running] python -u "c:\Users\User\Desktop\ADS\2024.1\teste_target_sistemas\invert.py"
Digite uma palavra para inverter:

Ln 8, Col 44  Espaços: 4  UTF-8  CRLF  Python  3.11.5 64-bit  19:54 12/02/2024
```

Este programa define uma função **inverter_string** que recebe uma string como entrada e retorna a string invertida. Ele itera através dos caracteres da string original de trás para frente e os adiciona a uma nova string, resultando na string invertida. Em seguida, solicita ao usuário que insira uma string e imprime a string invertida.