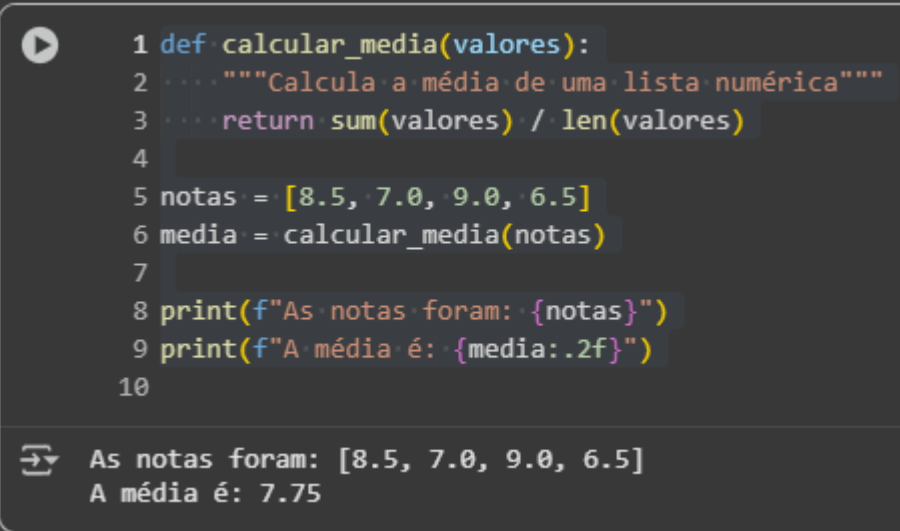


1)

```
def calcular_media(valores):  
    """Calcula a média de uma lista numérica"""  
  
    return sum(valores) / len(valores)  
  
notas = [8.5, 7.0, 9.0, 6.5]  
media = calcular_media(notas)  
  
print(f"As notas foram: {notas}")  
print(f"A média é: {media:.2f}")
```



```
1 def calcular_media(valores):  
2     """Calcula a média de uma lista numérica"""  
3     return sum(valores) / len(valores)  
4  
5 notas = [8.5, 7.0, 9.0, 6.5]  
6 media = calcular_media(notas)  
7  
8 print(f"As notas foram: {notas}")  
9 print(f"A média é: {media:.2f}")  
10
```

As notas foram: [8.5, 7.0, 9.0, 6.5]
A média é: 7.75

2)

```
def preco_com_desconto(preco, categoria):  
    """Aplica descontos conforme categoria"""  
    descontos = {"A": 0.9, "B": 0.85, "C": 0.8}  
    return preco * descontos.get(categoria, 1)  
  
produtos = [  
    (100, "A"),  
    (200, "B"),  
    (150, "C"),  
    (120, "D"), # Categoria inexistente → sem desconto  
]  
  
for preco, cat in produtos:  
    print(f"Preço original: R${preco:.2f} | Categoria: {cat} | Com desconto:  
    R${preco_com_desconto(preco, cat):.2f}")
```

```
1 def preco_com_desconto(preco, categoria):
2     """Aplica descontos conforme categoria"""
3     descontos = {"A": 0.9, "B": 0.85, "C": 0.8}
4     return preco * descontos.get(categoria, 1)
5
6 produtos = [
7     (100, "A"),
8     (200, "B"),
9     (150, "C"),
10    (120, "D"), # Categoria inexistente → sem desconto
11 ]
12
13 for preco, cat in produtos:
14     print(f"Preço original: R${preco:.2f} | Categoria: {cat} | Com desconto: R${preco_com_desconto(preco, cat):.2f}")
15
```

| | | |
|---------------------------|--------------|-------------------------|
| Preço original: R\$100.00 | Categoria: A | Com desconto: R\$90.00 |
| Preço original: R\$200.00 | Categoria: B | Com desconto: R\$170.00 |
| Preço original: R\$150.00 | Categoria: C | Com desconto: R\$120.00 |
| Preço original: R\$120.00 | Categoria: D | Com desconto: R\$120.00 |

3)

```
def calcular_total(itens):
    """Calcula total considerando categoria"""
    return sum(
        item["preco"] * (1.2 if item["categoria"] == "eletronico" else 1)
        for item in itens
    )
```

```
def processar_pedido(pedido):
    """Processa pedido e aplica descontos"""
    print(f"Iniciando pedido {pedido['id']}")
    total = calcular_total(pedido["itens"])
    print("Total:", total)
    if total > 1000:
        print("Aplicando desconto especial")
```

Exemplo de teste

```
pedido_exemplo = {
    "id": 101,
    "itens": [
        {"nome": "Notebook", "categoria": "eletronico", "preco": 800},
        {"nome": "Camisa", "categoria": "roupa", "preco": 150},
        {"nome": "Mouse", "categoria": "eletronico", "preco": 200}
    ]
}
```

```
processar_pedido(pedido_exemplo)
```

```
1 def calcular_total(itens):
2     """Calcula total considerando categoria"""
3     return sum(
4         item["preco"] * (1.2 if item["categoria"] == "eletronico" else 1)
5         for item in itens
6     )
7
8 def processar_pedido(pedido):
9     """Processa pedido e aplica descontos"""
10    print(f"Iniciando pedido {pedido['id']}")
11    total = calcular_total(pedido["itens"])
12    print("Total:", total)
13    if total > 1000:
14        print("Aplicando desconto especial")
15
16 # Exemplo de teste
17 pedido_exemplo = {
18     "id": 101,
19     "itens": [
20         {"nome": "Notebook", "categoria": "eletronico", "preco": 800},
21         {"nome": "Camisa", "categoria": "roupa", "preco": 150},
22         {"nome": "Mouse", "categoria": "eletronico", "preco": 200}
23     ]
24 }
25
26 processar_pedido(pedido_exemplo)
27
```

➡ Iniciando pedido 101
Total: 1350.0
Aplicando desconto especial

4)

```
def calcular_valor_total(preco, quantidade):
    """Retorna o valor total considerando produto e quantidade"""
    return preco * quantidade + (preco + quantidade)
```

Exemplo de teste:

```
print(calcular_valor_total(10, 5))
```

```
1 def calcular_valor_total(preco, quantidade):
2     """Retorna o valor total considerando produto e quantidade"""
3     return preco * quantidade + (preco + quantidade)
4
5 # Exemplo de teste:
6 print(calcular_valor_total(10, 5))
7
```

➡ 65

5)

```
idade = int(input("Digite sua idade: "))
```

```
# Estrutura simplificada com operador ternário
```

```
status = "menor" if idade < 18 else "maior"
```

```
# Saída de dados
```

```
print(f"Você é {status}.")
```

```
1 idade = int(input("Digite sua idade: "))
2
3 # Estrutura simplificada com operador ternário
4 status = "menor" if idade < 18 else "maior"
5
6 # Saída de dados
7 print(f"Você é {status}.")
8
```

➤ Digite sua idade: 20
Você é maior.

6)

```
etapas = ["Iniciando", "Executando tarefa", "Finalizando"]
```

```
# Estrutura de repetição (loop)
```

```
for etapa in etapas:
```

```
    print(f"{etapa}...")
```

```
1 etapas = ["Iniciando", "Executando tarefa", "Finalizando"]
2
3 # Estrutura de repetição (loop)
4 for etapa in etapas:
5     print(f"{etapa}...")
6
```

➤ Iniciando...
Executando tarefa...
Finalizando...

7)

```
TAXA_IMPOSTO = 0.07 # 7%
```

```
# Entrada de dados
```

```
preco = float(input("Digite o preço do produto: "))
```

```
# Cálculo do preço final com imposto
```

```
preco_final = preco * (1 + TAXA_IMPOSTO)
```

```
# Saída de dados
```

```
print(f"Preço final com imposto: R${preco_final:.2f}")
```

```
1 TAXA_IMPOSTO = 0.07 # 7%
2
3 # Entrada de dados
4 preco = float(input("Digite o preço do produto: "))
5
6 # Cálculo do preço final com imposto
7 preco_final = preco * (1 + TAXA_IMPOSTO)
8
9 # Saída de dados
10 print(f"Preço final com imposto: R${preco_final:.2f}")
```

```
✓ Digite o preço do produto: 10000
```

```
Preço final com imposto: R$10700.00
```

8)

```
def obter_taxa_bonus(cargo):
    taxas = {
        "gerente": 0.2,
        "analista": 0.1,
        "assistente": 0.05
    }
    # Retorna 0.05 como valor padrão caso o cargo não esteja no dicionário
    return taxas.get(cargo, 0.05)

# Função principal para calcular o bônus do funcionário
def calcular_bonus(funcionario):
    taxa = obter_taxa_bonus(funcionario["cargo"])
    return funcionario["salario"] * (1 + taxa)

# --- Teste prático ---
funcionarios = [
    {"nome": "Thiago", "cargo": "gerente", "salario": 5000},
    {"nome": "Porpa", "cargo": "analista", "salario": 4000},
    {"nome": "Lucas", "cargo": "assistente", "salario": 3000},
    {"nome": "Josefa", "cargo": "estagiario", "salario": 2000}
]

# Exibindo os resultados
for f in funcionarios:
    salario_com_bonus = calcular_bonus(f)
    print(f'{f["nome"]} ({f["cargo"]}): R${salario_com_bonus:.2f}')
```

```

1 def obter_taxa_bonus(cargo):
2     taxas = {
3         "gerente": 0.2,
4         "analista": 0.1,
5         "assistente": 0.05
6     }
7     # Retorna 0.05 como valor padrão caso o cargo não esteja no dicionário
8     return taxas.get(cargo, 0.05)
9
10 # Função principal para calcular o bônus do funcionário
11 def calcular_bonus(funcionario):
12     taxa = obter_taxa_bonus(funcionario["cargo"])
13     return funcionario["salario"] * (1 + taxa)
14
15 # --- Teste prático ---
16 funcionarios = [
17     {"nome": "Thiago", "cargo": "gerente", "salario": 5000},
18     {"nome": "Porpa", "cargo": "analista", "salario": 4000},
19     {"nome": "Lucas", "cargo": "assistente", "salario": 3000},
20     {"nome": "Josefa", "cargo": "estagiario", "salario": 2000}
21 ]
22
23 # Exibindo os resultados
24 for f in funcionarios:
25     salario_com_bonus = calcular_bonus(f)
26     print(f"{f['nome']} ({f['cargo']}): R${salario_com_bonus:.2f}")

```

```

Thiago (gerente): R$6000.00
Porpa (analista): R$4400.00
Lucas (assistente): R$3150.00
Josefa (estagiario): R$2100.00

```

9)

```

def test_soma():
    assert soma(15, 40) == 55

```

--- Etapa 2: Implementar a função para passar no teste ---

```

def soma(a, b):
    return a + b

```

try:

```

    test_soma()
    print("✅ Teste passou com sucesso!")

```

except AssertionError:

```

    print("❌ Teste falhou. A função precisa ser corrigida.")

```



```
1 def test_soma():
2     assert soma(15, 40) == 55
3
4 # --- Etapa 2: Implementar a função para passar no teste ---
5 def soma(a, b):
6     return a + b
7
8 try:
9     test_soma()
10    print("✅ Teste passou com sucesso!")
11 except AssertionError:
12    print("❌ Teste falhou. A função precisa ser corrigida.")
```



✅ Teste passou com sucesso!