

Aula 1: O que é Programação

1) O que significa programar?

- a) Criar aplicativos sem precisar seguir regras
 - b) Dar instruções lógicas e detalhadas para o computador executar tarefas
 - c) Apenas digitar códigos em uma linguagem específica
 - d) Montar computadores
-

2) Qual comparação foi usada para explicar programação?

- a) Jogar videogame
 - b) Resolver uma equação matemática
 - c) Seguir uma receita de bolo
 - d) Dirigir um carro
-

3) Qual dessas opções NÃO é uma razão para a programação ser importante?

- a) Automatizar tarefas
 - b) Resolver problemas
 - c) Criar ideias e projetos
 - d) Substituir completamente o pensamento humano
-

4) Qual é o papel do programador segundo os slides?

- a) Apenas copiar códigos prontos
 - b) Somente decorar comandos
 - c) Pensar logicamente para resolver problemas e traduzir soluções em código
 - d) Criar programas sem planejamento
-

5) Qual é a sequência do processo de criação do programador?

- a) Planejar → Testar → Codar → Entender o problema
- b) Entender → Planejar → Codificar → Testar e Melhorar
- c) Codificar → Testar → Planejar → Entender
- d) Escrever código direto e ajustar depois

Aula 2: Variáveis e Tipos de Dados

1) O que é uma variável na programação?

- a) Um bloco de código que sempre executa a mesma coisa
 - b) Um espaço na memória que armazena um valor e pode mudar durante o programa
 - c) Um erro que o computador corrige sozinho
 - d) Um tipo de dado fixo que nunca pode alterar seu valor
-

2) Qual tipo de dado é adequado para armazenar o valor 3.14?

- a) int
 - b) str
 - c) float
 - d) bool
-

3) Qual dos exemplos abaixo representa um valor booleano?

- a) "Verdadeiro"
 - b) 150.00
 - c) True
 - d) 25
-

4) Qual das opções abaixo representa uma boa prática ao nomear variáveis?

- a) Usar acentos e espaços para facilitar a leitura
 - b) Começar com número
 - c) Usar nomes claros e que façam sentido, sem caracteres especiais
 - d) Usar palavras reservadas da linguagem para facilitar
-

5) Por que os tipos de dados são importantes na programação?

- a) Porque o computador só aceita números
- b) Para organizar informações, evitar erros e usar a memória de forma eficiente
- c) Porque deixam o código mais bonito
- d) Para que o programa rode apenas em Python

Aula 3: Estruturas Condicionais

1) Para que serve uma estrutura condicional?

- a) Para repetir um bloco de código
 - b) Para armazenar informações
 - c) Para tomar decisões no código com base em uma condição
 - d) Para criar funções
-

2) Qual palavra-chave representa o "senão" em Python?

- a) else
 - b) if
 - c) elif
 - d) end
-

3) Em qual situação usamos **elif**?

- a) Quando queremos sempre executar um bloco específico
 - b) Quando queremos testar várias condições diferentes
 - c) Para comparar valores apenas com números
 - d) Quando a condição inicial sempre será verdadeira
-

4) Qual operador é utilizado para comparar igualdade em Python?

- a) =
- b) ==
- c) !=
- d) equals

5) O que acontece se esquecermos a indentação correta após um `if` em Python?

- a) O Python ignora o comando
- b) O programa roda normalmente
- c) O Python gera um erro, pois a indentação é obrigatória
- d) O bloco é executado mesmo assim

Aula 4: Laços de Repetição

1) Para que servem os laços de repetição?

- a) Para criar variáveis automaticamente
 - b) Para repetir ações de forma automática
 - c) Para comparar valores lógicos
 - d) Para definir funções no código
-

2) Quando geralmente usamos o laço **for** em Python?

- a) Quando não sabemos quantas vezes vamos repetir
- b) Quando queremos repetir uma ação até que o usuário pare o programa
- c) Quando sabemos a quantidade de repetições ou queremos percorrer uma sequência
- d) Apenas quando trabalhamos com números decimais

Resposta correta: C

3) O que o comando **range(0, 10, 2)** gera?

- a) 1,2,3,4,5
 - b) 0,1,2,3,4,5,6,7,8,9
 - c) 0,2,4,6,8
 - d) 2,4,6,8,10
-

4) Quando usamos **while** em Python?

- a) Quando queremos percorrer uma lista
 - b) Quando queremos repetir enquanto uma condição for verdadeira
 - c) Apenas quando sabemos quantas vezes repetir
 - d) Para substituir o comando **for** sempre
-

5) O que pode causar um loop infinito?

- a) Declarar uma variável dentro do laço
- b) Esquecer de atualizar a condição dentro do loop
- c) Usar a função `range()`
- d) Utilizar o comando `break`

Aula 5: Funções

1) O que é uma função na programação?

- a) Um tipo de variável que armazena textos
 - b) Um bloco de código que executa uma tarefa específica e pode ser reutilizado
 - c) Uma estrutura usada apenas para repetir comandos
 - d) Um comando usado para comparar valores
-

2) Qual palavra-chave usamos para definir uma função em Python?

- a) func
 - b) def
 - c) function
 - d) return
-

3) O que são parâmetros em uma função?

- a) Os nomes das funções
 - b) Comentários dentro do código
 - c) Valores que a função recebe para usar dentro dela
 - d) Resultados retornados pela função
-

4) Para que serve a palavra-chave **return** em uma função?

- a) Para encerrar o programa
- b) Para chamar outra função
- c) Para armazenar um valor dentro de uma variável
- d) Para devolver um resultado gerado pela função

5) Qual é uma boa prática ao criar funções?

- a) Criar uma função que faz várias tarefas diferentes
- b) Usar nomes curtos e sem sentido
- c) Fazer cada função realizar apenas uma tarefa específica e usar nomes claros
- d) Nunca testar a função antes de usá-la

– NÍVEL MÉDIO –

Aula 2 — Variáveis e Tipos de Dados

1)

```
idade = 20  
  
nome = "Carlos"  
  
print(nome + " tem " + str(idade) + " anos.")
```

O que será exibido na tela?

- a) Carlos tem idade anos.
 - b) Carlos tem 20 anos.
 - c) Erro de tipo.
 - d) nome tem 20 anos.
-

2)

```
x = 10  
  
y = 2.5  
  
resultado = x + y  
  
print(type(resultado))
```

Qual será o tipo do valor exibido?

- a) <class 'int'>
 - b) <class 'float'>
 - c) <class 'str'>
 - d) <class 'bool'>
-

3)

```
a = True  
  
b = False  
  
print(a and not b)
```

Qual será o resultado impresso?

- a) True
 - b) False
 - c) 1
 - d) 0
-

4)

```
nome = "Matheus"  
  
idade = 18  
  
print(f"Olá, {nome}. Você tem {idade} anos.")
```

Qual é a vantagem de usar o `f` antes das aspas?

- a) Ele converte tudo em texto automaticamente.
- b) Permite inserir variáveis dentro da string de forma prática.
- c) Evita erros de indentação.
- d) Faz o programa rodar mais rápido.

5)

```
numero = "25"
```

```
print(int(numero) + 5)
```

O que acontece se retirarmos o `int()`?

- a) O programa continua funcionando normalmente.
- b) O resultado será "255".
- c) O programa exibirá um erro de tipo.
- d) O número será convertido em float.

Aula 3 — Estruturas Condicionais

1)

```
nota = 8
if nota >= 7:
    print("Aprovado")
else:
    print("Reprovado")
```

O que o programa exibirá?

- a) Reprovado
 - b) Aprovado
 - c) Erro de sintaxe
 - d) Nada será exibido
-

2)

```
idade = 17
if idade >= 18:
    print("Maior de idade")
elif idade >= 16:
    print("Pode votar, mas não dirigir")
else:
    print("Menor de idade")
```

Qual será a saída?

- a) Maior de idade
- b) Pode votar, mas não dirigir
- c) Menor de idade
- d) Erro lógico

3)

```
x = 10
y = 5
if x > y and y > 0:
    print("Condição verdadeira")
else:
    print("Condição falsa")
```

Qual será o resultado?

- a) Condição falsa
 - b) Erro de comparação
 - c) Condição verdadeira
 - d) Nenhum resultado
-

4)

```
senha = "1234"
entrada = "4321"

if entrada == senha:
    print("Acesso permitido")
else:
    print("Senha incorreta")
```

O que será exibido?

- a) Erro de comparação
 - b) Acesso permitido
 - c) Senha incorreta
 - d) Nenhuma saída
-

5)

```
n = 4
if n % 2 == 0:
    print("Número par")
else:
    print("Número ímpar")
```

Qual é a saída?

- a) Número ímpar
- b) Número par
- c) Erro de módulo
- d) Nenhum resultado

Aula 4 — Laços de Repetição

1)

```
for i in range(3):  
    print("Olá!")
```

Quantas vezes a palavra “Olá!” será exibida?

- a) 2 vezes
 - b) 3 vezes
 - c) 4 vezes
 - d) 5 vezes
-

2)

```
for i in range(1, 6):  
    print(i)
```

Qual será a saída do programa?

- a) 0 1 2 3 4 5
 - b) 1 2 3 4 5
 - c) 1 2 3 4 5 6
 - d) 0 1 2 3 4
-

3)

```
contador = 1  
while contador <= 3:  
    print("Contagem:", contador)  
    contador += 1
```

O que será exibido na tela?

- a) Contagem: 0 / Contagem: 1 / Contagem: 2
 - b) Contagem: 1 / Contagem: 2 / Contagem: 3
 - c) Contagem: 1 / Contagem: 2
 - d) Loop infinito
-

4)

```
for n in range(0, 10, 3):  
    print(n)
```

O que o programa imprime?

- a) 0 3 6 9
 - b) 0 3 6 9 12
 - c) 0 2 4 6 8
 - d) 3 6 9
-

5)

```
x = 0  
while x < 5:  
    print(x)  
    x += 2
```

Qual é a saída?

- a) 0 1 2 3 4
- b) 0 2 4
- c) 0 2 4 6
- d) 1 3 5

– NÍVEL DIFÍCIL –

Aula 2 — Variáveis e Tipos de Dados

1)

```
x = 10
y = "5"
print(x + int(y))
```

O que será exibido?

- a) 15
 - b) 105
 - c) Erro de tipo
 - d) 5
-

2)

```
a = 3
b = 2
a = a + b
b = a - b
print(a, b)
```

Saída final:

- a) 3 2
 - b) 5 3
 - c) 2 3
 - d) 5 2
-

3)

```
x = 5  
x += 3 * 2  
print(x)
```

Resultado:

- a) 16
 - b) 11
 - c) 10
 - d) 6
-

4)

```
texto = "Python"  
print(texto[1:4])
```

Saída:

- a) Pyt
 - b) yth
 - c) ytho
 - d) yth
-

5)

```
valor = 7.8  
print(int(valor) + bool(valor))
```

Saída:

- a) 7
- b) 8
- c) 9
- d) Erro

Aula 3 — Estruturas Condicionais

1)

```
a = 10
b = 5
if a > b*2:
    print("Maior")
elif a == b*2:
    print("Igual")
else:
    print("Menor")
```

Saída:

- a) Maior
 - b) Igual
 - c) Menor
 - d) Erro
-

2)

```
x = 15
if x % 3 == 0 and x % 5 == 0:
    print("Divisível por 3 e 5")
elif x % 3 == 0:
    print("Divisível por 3")
else:
    print("Outro caso")
```

Saída:

- a) Divisível por 3
- b) Divisível por 3 e 5
- c) Outro caso
- d) Nenhum

3)

```
idade = 20
tem_carteira = False

if idade >= 18:
    if tem_carteira:
        print("Pode dirigir")
    else:
        print("Maior de idade, mas sem carteira")
else:
    print("Menor de idade")
```

Saída:

- a) Pode dirigir
- b) Maior de idade, mas sem carteira
- c) Menor de idade
- d) Erro

4)

```
n = 10
if n % 2 == 0:
    if n % 5 == 0:
        print("Par e múltiplo de 5")
    else:
        print("Par")
else:
    print("Ímpar")
```

Saída:

- a) Ímpar
 - b) Par e múltiplo de 5
 - c) Par
 - d) Nenhum
-

5)

```
a, b, c = 3, 7, 5
if a < b > c:
    print("b é o maior")
else:
    print("Outra condição")
```

Saída:

- a) b é o maior
 - b) Outra condição
 - c) Erro lógico
 - d) Nada
-

Aula 4 — Laços de Repetição

1)

```
for i in range(2, 10, 3):  
    print(i, end=" ")
```

Saída:

- a) 2 5 8
 - b) 2 3 4
 - c) 3 6 9
 - d) 2 5 7 9
-

2)

```
soma = 0  
for i in range(1, 5):  
    soma += i  
print(soma)
```

Resultado:

- a) 10
 - b) 15
 - c) 5
 - d) 4
-

3)

```
x = 0
while x < 5:
    x += 1
    if x == 3:
        continue
    print(x)
```

Saída:

- a) 1 2 3 4 5
 - b) 1 2 4 5
 - c) 0 1 2 3 4
 - d) Erro
-

4)

```
for i in range(3):
    for j in range(2):
        print(i, j)
```

Quantas vezes o print será executado?

- a) 3
 - b) 6
 - c) 9
 - d) 2
-

5)

```
x = 10
while x > 0:
    print(x, end=" ")
    x -= 3
```

Saída:

- a) 10 9 8 7 6 5 4 3 2 1
 - b) 10 7 4 1
 - c) 10 7 4
 - d) 10 8 6 4 2
-

Aula 5 — Funções

1)

```
def soma(a, b=2):  
    return a + b  
  
print(soma(3))
```

Saída:

- a) 3
 - b) 5
 - c) Erro
 - d) None
-

2)

```
def misterio(x):  
    if x % 2 == 0:  
        return x / 2  
    else:  
        return x * 3 + 1  
  
print(misterio(5))
```

Saída:

- a) 2.5
 - b) 10
 - c) 16
 - d) 0
-

3)

```
def calc(a, b):  
    return a + b, a * b  
  
soma, produto = calc(2, 4)  
print(soma + produto)
```

Saída:

- a) 6
 - b) 8
 - c) 14
 - d) 12
-

4)

```
def f(x):  
    return x**2  
  
def g(y):  
    return f(y) + 2  
  
print(g(3))
```

Saída:

- a) 9
 - b) 11
 - c) 5
 - d) 6
-

5)

```
def repetir(msg, vezes):  
    for i in range(vezes):  
        print(msg)  
  
repetir("Oi", 3)
```

Saída:

- a) Oi
- b) Oi Oi
- c) Oi Oi Oi
- d) Erro de indentação