Lista de Exercícios 1 - Fundamentos de Programação em Python

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Março 2025

---

1. Conceitos Básicos

1.1 - Explique a diferença entre um programa e um algoritmo. Como Python facilita a implementação de algoritmos?

1.2 - Considere o seguinte código Python:

```
print("Bem-vindo ao curso de Python!")
print(2 + 3 * 4)
print("Resultado:", 2 ** 3)
```

Qual será a saída do programa? Justifique sua resposta.

1.3 A programação de computadores é uma atividade fundamental na área da tecnologia, permitindo a criação de sistemas e aplicações para diversas finalidades. Sobre os conceitos fundamentais da programação, avalie as afirmativas a seguir:

I. Um algoritmo é uma sequência finita e ordenada de passos que levam à solução de um problema.

II. Uma linguagem de programação compilada converte todo o código-fonte para código de máquina antes da execução, enquanto uma linguagem interpretada traduz e executa linha por linha.

III. A estrutura de dados define a organização, o armazenamento e a manipulação de informações em um programa, sendo listas, pilhas e filas exemplos comuns.

IV. Em linguagens de programação, a estrutura de controle "if-else" é utilizada exclusivamente para repetir instruções enquanto uma condição for verdadeira.

É correto apenas o que se afirma em:

A) I e II.
B) I, II e III.
C) I, III e IV.
D) II, III e IV.
E) I, II, III e IV.
F) Nenhuma das alternativas.

2. Variáveis e Tipos de Dados

2.1 O que é uma variável? Explique a diferença entre os tipos de dados `int`, `float`, `bool` e `str` em Python, fornecendo exemplos.

2.2 - O código abaixo apresenta um erro. Qual é o erro e como corrigí-lo?

```python
numero = input("Digite um número: ")
dobro = numero * 2
print("O dobro do número é:", dobro)
```

Justifique sua resposta.

2.3 - Escreva um programa que recebe dois números do usuário e exibe a soma, a subtração, a multiplicação e a divisão entre eles.

2.4 Considere o seguinte trecho de código Python:

```python
x = "10"
y = 5
z = x + y
print(z)
```

Sobre a execução desse código, analise as afirmativas a seguir:

I. O código apresentará um erro de `TypeError`, pois a variável `x` é do tipo `str` e a variável `y` é do tipo `int`, tornando a operação `x + y` inválida.

II. Para que o código funcione corretamente e exiba a soma de 10 e 5, seria necessário converter `x` para um número utilizando `int(x)`.

III. Em Python, as variáveis não possuem um tipo fixo, ou seja, um mesmo identificador pode armazenar valores de diferentes tipos ao longo da execução do programa.

IV. A instrução `print(z)` imprimirá 105, pois Python realiza automaticamente a conversão de tipos quando necessário.

É correto apenas o que se afirma em:

A) I e II.
B) II e III.
C) I, II e III.
D) I, III e IV.
E) II, III e IV.
F) Nenhuma das alternativas.

2.5 Em Python, operadores booleanos seguem uma ordem de precedência, o que pode influenciar o resultado final da avaliação de uma expressão lógica. Excute o código Python e responda: Qual será o valor final da variável `resultado`? Justifique sua resposta utilizando a ordem de avaliação dos operadores.

```python
resultado = not (True or False) and (False or True)
```

2.6 Considere o seguinte código Python:

```python
a = 10
b = 5
c = 2
resultado = (a > b) and (b < c or not (a > c)) and (c * b > a)
print(resultado)
```

Explique detalhadamente como a expressão booleana na variável `resultado` é avaliada.

3. Estruturas Condicionais (`if-elif-else`)

3.1 - O que é uma estrutura condicional? Explique como funciona o `if`, `elif` e `else` em Python. Exemplifique com um fluxograma.

3.2 - O que será impresso pelo seguinte código?

```
idade = 18
if idade >= 18:
    print("Maior de idade")
else:
    print("Menor de idade")
```

3.3 - Considere o seguinte código Python. Esse código apresenta um erro lógico. Qual é o erro e como corrigí-lo?

```
nota = 75

if nota >= 90:
    conceito = "A"
if nota >= 80:
    conceito = "B"
if nota >= 70:
    conceito = "C"
if nota >= 60:
    conceito = "D"
else:
    conceito = "F"

print(conceito)
```

3.4 - Escreva um programa que recebe um número do usuário e verifica se ele é positivo, negativo ou zero, exibindo uma mensagem apropriada.

4. Estruturas de Repetição (`while` e `for`)

4.1 Qual a diferença entre um loop `for` e um loop `while`? Dê exemplos de quando seria mais adequado usar cada um.

4.2 - Produza uma tabela incluíndo os valores da variável `contador` e a saída produzida pelo código para cada iteração do loop `while` abaixo.

```
contador = 1
while contador <= 5:
    print(contador)
    contador += 1
```

4.3 - O código abaixo apresenta um erro lógico que pode causar um problema de execução. Qual é o problema e como corrigí-lo?

```
n = 10
while n > 0:
    print(n)
```

4.4 - Escreva um programa que exibe todos os números primos de 1 a 20 utilizando um loop `for`.

4.5 - Escreva um programa que solicita um número do usuário e exibe sua tabuada de 1 a 10.

4.6 - Sobre a utilização de estruturas de repetição (loops) em Python, analise as afirmativas a seguir:

I. O loop for é ideal para percorrer elementos de uma lista ou iterar sobre um intervalo definido.

II. O loop while executa um bloco de código repetidamente enquanto uma condição especificada for verdadeira.

III. A palavra-chave break interrompe um loop imediatamente, enquanto continue faz com que a execução passe para a próxima iteração sem concluir o restante do bloco atual.

IV. Todo loop while pode ser substituído por um loop for sem alterar a lógica do programa.

É correto apenas o que se afirma em:

A) I e II.
B) I, II e III.
C) II, III e IV.
D) I, III e IV.
E) I, II, III e IV.
F) Nenhuma das alternativas.

5. Strings

5.1 - O que é uma string em Python? Como strings podem ser manipuladas usando indexação e fatiamento (`slicing`)?

5.2 - O que será impresso pelo seguinte código?

```
texto = "Python"
print(texto[0])
print(texto[-1])
print(texto[1:4])
```

5.3 - Considere o seguinte código Python:

```
frase = "Python é incrível"
frase[0] = "J"
print(frase)
```

Esse código apresenta um erro. Explique por que o erro ocorre e como ele pode ser corrigido.

5.4 - Escreva um programa que recebe uma string do usuário e imprime essa string ao contrário. Note que há várias formas de fazer a inversão da string.

5.5 - Considere o seguinte código Python:

```
texto = "Programação em Python"
print(texto.upper())
print(texto.replace("Python", "Java"))
print(len(texto))
print(texto[0:5])
```

Sobre a execução desse código, analise as afirmativas:

(I) A função `upper()` converterá todos os caracteres da string para maiúsculas, imprimindo `"PROGRAMAÇÃO EM PYTHON"`.
(II) O método `replace()` substituirá a palavra `"Python"` por `"Java"`, resultando na saída `"Programação em Java"`.
(III) A função `len(texto)` retornará o número total de caracteres na string, incluindo espaços em branco.
(IV) A instrução `texto[0:5]` imprimirá `"Progr"`, pois realiza um fatiamento (slicing) da string, pegando os caracteres do índice 0 ao 4.

É correto apenas o que se afirma em:

A) I e II.
B) II e III.
C) I, II e III.
D) I, III e IV.
E) I, II, III e IV.
F) Nenhuma das alternativas.