UFCD 5417: Lab. 5

Estruturas de controlo

v1.0

Nelson Santos
nelson.santos.0001376@edu.atec.pt

ATEC — 25 de novembro de 2024

Índice

1 Introdução

Neste laboratório, vamos focar-nos no estudo e aplicação de estruturas de controlo de fluxo em `Python`, como estruturas condicionais e ciclos. Através de uma série de exercícios práticos, será possível consolidar os conhecimentos sobre como tomar decisões no código, repetir operações de forma eficiente e criar interações dinâmicas com um utilizador.

Os problemas propostos foram selecionados para demonstrar a importância e a utilidade das estruturas de controlo no desenvolvimento de soluções para diferentes situações. A implementação desses exercícios permitirá praticar conceitos fundamentais, como instruções `if`, `for` e `while`, juntamente com operadores lógicos e matemáticos.

Os exercícios incluem:

- Adivinha o número (`Guess the Number`)
- Calculadora de juros (`Interest Calculator`)
- Leis por Idade (`Laws by Age`)
- Treinador de matemática (`Math Trainer`)
- Sistema de login (`Login System`)
- Verificador de ano bissexto (`Leap Year`)

Cada exercício explora um aspeto diferente das estruturas de controlo, sendo descrito detalhadamente nas secções seguintes.

2 Estruturas de Controlo no Python

As estruturas de controlo em `Python` permitem-nos manobrar o fluxo de execução de um programa, decidindo quais instruções devem ser executadas com base em determinadas condições ou repetições. Estas estruturas incluem as condicionais, ciclos e instruções de controlo de fluxo.

2.1 Estruturas Condicionais

As estruturas condicionais são usadas para tomar decisões no código, baseadas em testes lógicos.

- `if`, `elif`, `else`: Estas palavras-chave permitem criar condições que definem quais os blocos de código que devem ser executados.

```
x = 10
if x > 5:
    print("x maior que 5")
elif x == 5:
    print("x igual a 5")
else:
    print("x menor que 5")
```

Neste exemplo, o programa verifica a condição (`x > 5`) e executa o bloco de código correspondente. Se a condição não for verdadeira, a próxima condição (`elif`) é verificada, e, se nenhuma das condições for satisfeita, o bloco `else` será executado.

2.2 Ciclos (Loops)

Os ciclos permitem a repetição de blocos de código até que uma condição seja satisfeita. Existem dois tipos principais de ciclos em Python:

- `for`: Usado para iterar sobre uma sequência (como uma lista, tuplo ou string) ou para repetir uma operação um número específico de vezes.

```
for i in range(5):
    print(i)
```

Neste exemplo, o ciclo `for` itera sobre os números de 0 a 4, imprimindo cada valor. A função `range()` gera uma sequência de números.

- `while`: Executa um bloco de código enquanto uma condição for verdadeira.

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

Neste exemplo, o ciclo `while` continua a correr enquanto a variável `x` for menor que 5. Em cada iteração, `x` é incrementado até que a condição deixe de ser verdadeira.

2.3 Instruções de Controlo de Fluxo

O `Python` oferece ainda instruções especiais para modificar o fluxo de execução dentro de ciclos ou blocos de código:

- `break`: Interrompe a execução de um ciclo antes de este terminar
- `continue`: Salta a execução do código restante dentro do ciclo e passa para a próxima iteração
- `pass`: Um `placeholder` que não executa nada. É útil quando uma sintaxe requer um bloco de código, mas este ainda não está implementado, por exemplo

```
for i in range(10):
    if i == 5:
        break  # Interrompe o ciclo quando i for igual a 5
    print(i)
```

Neste exemplo, o ciclo `for` é interrompido quando `i` atinge o valor 5, devido ao uso da instrução `break`.

```
for i in range(5):
    if i == 3:
        continue  # Salta o valor 3
    print(i)
```

Aqui, o valor 3 é "ultrapassado"devido à instrução `continue`, e o ciclo prossegue com os restantes valores.

2.4 Operadores Lógicos e Comparativos

Para tomar decisões em estruturas de controlo, utilizamos operadores comparativos e lógicos que nos permitem combinar condições.

- **Operadores de Comparação**: `==`, `!=`, `>`, `<`, `>=`, `<=`
- **Operadores Lógicos**: `and`, `or`, `not`

```
x = 10
y = 5
if x > y and x < 20:
    print("x entre 5 e 20")
```

Neste exemplo, a condição combina dois testes lógicos usando o operador `and`, o que significa que ambos têm de ser verdadeiros para que o bloco de código seja executado.

3 Exercício 1: Adivinha o Número (`Guess the Number`)

Neste exercício, o utilizador deve adivinhar um número gerado aleatoriamente pelo programa.

3.1 Instruções

1. Gere um número aleatório entre 1 e 100 usando a função `random.randint()`
2. Solicite ao utilizador para adivinhar o número e forneça dicas como "Muito alto!"ou "Muito baixo!"até que ele adivinhe corretamente
3. Ao acertar, mostre o número de tentativas que o utilizador precisou para acertar

3.2 Perguntas

- De que forma a função `random.randint()` gera números aleatórios?
- O que aconteceria se o utilizador introduzisse um valor inválido (não numérico)? Como poderia tratar essa situação?

4 Exercício 2: Calculadora de juros (`Interest Calculator`)

Desenvolva uma calculadora simples de juros composta e simples.

4.1 Instruções

1. Peça ao utilizador para inserir o montante inicial (capital), a taxa de juro anual e o número de anos
2. Implemente duas funções: uma para calcular os juros simples e outra para calcular os juros compostos
3. Exiba os resultados para ambos os cálculos

4.2 Perguntas

- Qual é a fórmula para o cálculo de juros simples? E para juros compostos?
- Qual seria a melhor opção de investimento dependendo da taxa de juro e do número de anos?

5 Exercício 3: Leis por idade (`Laws by Age`)

Este exercício foca-se em fornecer informações baseadas na idade do utilizador.

5.1 Instruções

1. Solicite ao utilizador para introduzir a sua idade
2. Dependendo da idade, apresente mensagens apropriadas, como "Pode votar", "Pode conduzir", "Pode beber álcool", entre outras regras baseadas em faixas etárias
3. Utilize estruturas condicionais para implementar as diferentes regras de idade

5.2 Perguntas

- Que estrutura condicional seria mais eficiente para implementar este exercício: múltiplos `if-else` ou outra abordagem?
- Que regras podem variar de país para país? Como poderia adaptar o programa para suportar diferentes legislações?

6 Exercício 4: Treinador de matemática (`Math Trainer`)

Crie um treinador de matemática que gera operações aleatórias para o utilizador resolver.

6.1 Instruções

1. Construa uma operação matemática aleatória (soma, subtração, multiplicação ou divisão) entre dois números utilizando a função `random`.
2. Peça ao utilizador para inserir a resposta à operação e diga se está correta ou não
3. Continue a gerar novas operações até o utilizador decidir parar

6.2 Perguntas

- Como pode garantir que as operações geradas são equilibradas, ou seja, que não são demasiado fáceis ou difíceis?
- Como pode o programa fornecer feedback ao utilizador sobre o seu progresso ou desempenho?

7 Exercício 5: Sistema de Login (`Login System`)

Neste exercício, vai criar um sistema simples de login que verifica se o utilizador está registado.

7.1 Instruções

1. Crie uma lista de utilizadores e palavras-passe predefinidas
2. Peça ao utilizador para introduzir o seu nome de utilizador e palavra-passe
3. Verifique se o nome de utilizador e a palavra-passe estão corretos. Se estiverem, exiba uma mensagem de boas-vindas; caso contrário, exiba um erro

7.2 Perguntas

- Como pode guardar os utilizadores de forma segura, evitando o uso direto de palavras-passe em texto simples?
- Como poderia melhorar este sistema de login para suportar funcionalidades como a recuperação de senha ou múltiplas tentativas?

8 Exercício 6: Verificador de Ano Bissexto (`Leap Year`)

Implemente um programa que verifica se um determinado ano é bissexto.

8.1 Instruções

1. Peça ao utilizador para inserir um ano
2. Verifique se o ano inserido é bissexto de acordo com as regras: o ano deve ser divisível por 4, mas não por 100, exceto se também for divisível por 400
3. Exiba uma mensagem indicando se o ano é bissexto ou não

8.2 Perguntas

- Porque é que existem anos bissextos? Qual a importância dessa correção no calendário?
- Como poderia adaptar este programa para verificar um intervalo de anos?