Educação Profissional Paulista

Técnico em
Desenvolvimento de Sistemas

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Estrutura de seleção

Comandos condicionais: *SWITCH*

Aula 1

Código da aula: [SIS]ANO1C1B2S13A1

Exposição

Objetivo da Aula

- Compreender as principais diferenças entre a estrutura *SWITCH* e os comandos tradicionais if-else.

Competências da Unidade (Técnicas e Socioemocionais)

- Desenvolver sistemas computacionais, utilizando ambiente de desenvolvimento.
- Trabalhar a criatividade e a resolução de problemas computacionais.

Recursos Didáticos

- Recurso audiovisual para exibição de vídeos e imagens.
- Folhas sulfite, canetas coloridas e lápis.

Duração da Aula

50 minutos

Exposição

Introdução ao comando *SWITCH*

Este tema aborda os **fundamentos do comando SWITCH, essencial para entender a lógica de programação condicional**. É a base para construir estruturas de decisão mais complexas.

- O que é o comando SWITCH e como difere do IF-ELSE.
- Sintaxe básica do SWITCH em diferentes linguagens de programação.
- Exemplos simples de uso do SWITCH.

Exposição

O que é o comando *SWITCH* e como difere do *IF-ELSE*

Comando SWITCH

Definição: o SWITCH é uma estrutura de controle usada para selecionar entre múltiplas opções baseadas no valor de uma variável. Ele é comumente usado quando se tem muitas condições a serem verificadas.

Vantagens: mais limpo e mais legível do que múltiplos IF-ELSE, quando lidando com várias condições.

Funcionamento: baseia-se no valor de uma expressão ou variável, executando o bloco de código correspondente ao valor correspondente (caso, ou *case*).

Exposição

O que é o comando *SWITCH* e como difere do *IF-ELSE*

IF-ELSE:

Definição: IF-ELSE é uma estrutura condicional básica que executa um bloco de código se uma condição for verdadeira, e outro bloco se for falsa.

Uso: mais adequado para situações com um número menor de condições ou quando as condições são complexas, e não apenas baseadas em um valor de variável.

Diferenças-chave:

Estrutura: SWITCH é mais estruturado para múltiplas escolhas baseadas em um único valor de variável, enquanto IF-ELSE é mais flexível para avaliar condições variadas e complexas.

Legibilidade: em casos com muitas condições, SWITCH tende a ser mais limpo e legível.

Aplicabilidade: IF-ELSE pode ser usado em praticamente qualquer situação, enquanto SWITCH é limitado a cenários nos quais se compara uma variável com valores constantes.

Vídeo

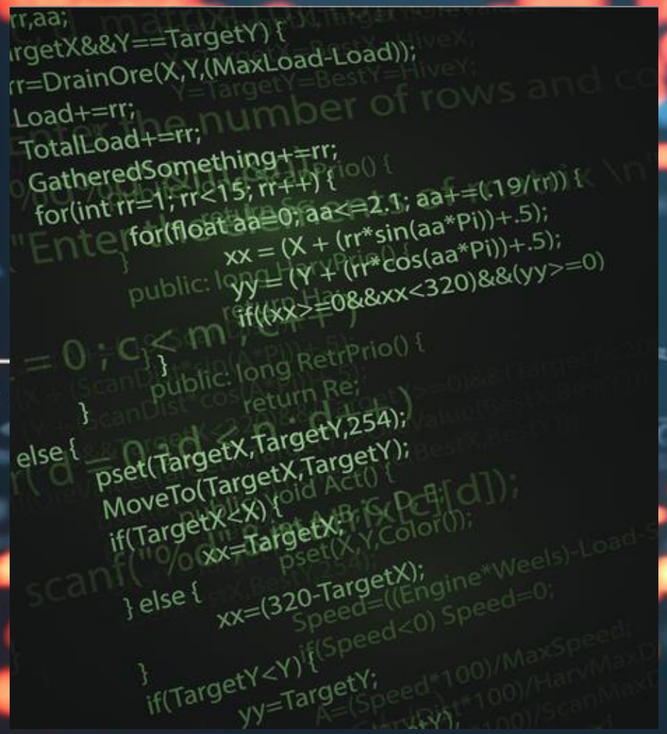
© Getty Images

Exposição

Que tal relembrarmos a aplicação do if-else em Python?

ALURA. *Python*: começando com a linguagem. Comparando variáveis. Disponível em: https://cursos.alura.com.br/course/python-introducao-a-linguagem/task/22779. Acesso em: 21 mar. 2024.

Exposição



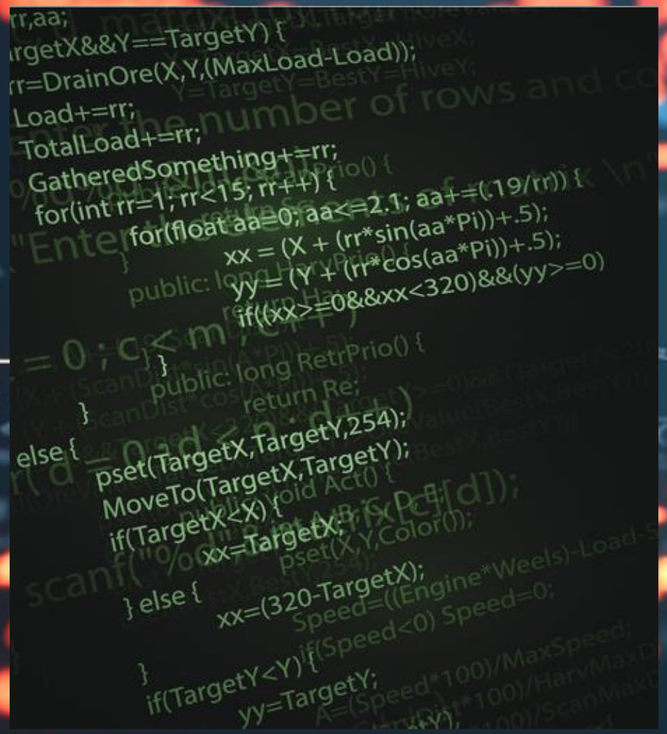
Sintaxe básica do *SWITCH* em diferentes linguagens

Python não tem um comando **SWITCH** nativo, mas pode-se **simular seu comportamento com dicionários ou if-elif-else.** Vejamos a sintaxe em outras linguagens:

C#

```
switch (variavel) {
    case valor1:
        // código
        break;
    case valor2:
        // código
        break;
    default:
        // código padrão
}
```

Exposição



© Getty Images

Exemplo em *Python* (Simulando *SWITCH*)

Python não dispõe de um SWITCH nativo, mas você pode usar um dicionário para simular esse comportamento:

```
def switch_dia(dia):
    dias = {
        1: "Segunda-feira",
        2: "Terça-feira",
        3: "Quarta-feira",
        # Restante dos dias...
    }
    return dias.get(dia, "Dia inválido")

dia_selecionado = switch_dia(3)
print("O dia é:", dia_selecionado)
```

Exposição

Exemplos de mercado

Seleção de configurações: em um software de configurações, em que cada opção leva a um conjunto diferente de parâmetros, o SWITCH pode ser usado para selecionar a configuração apropriada.

Processamento de comandos: em um sistema de comando de voz ou texto, *em que* cada comando (como "abrir", "salvar", "editar") requer ações diferentes, o SWITCH é ideal para direcionar para a função correta.

Interfaces de usuário: em interfaces gráficas, ao lidar com eventos de diferentes botões ou ações (como menus), o SWITCH pode ser usado para determinar a ação correspondente a cada elemento de interface.

Exposição

Exemplo prático

Suponha que temos um aplicativo de automação residencial que recebe comandos de voz ou texto para controlar diferentes dispositivos na casa. **Cada comando requer uma ação específica para um dispositivo correspondente.**

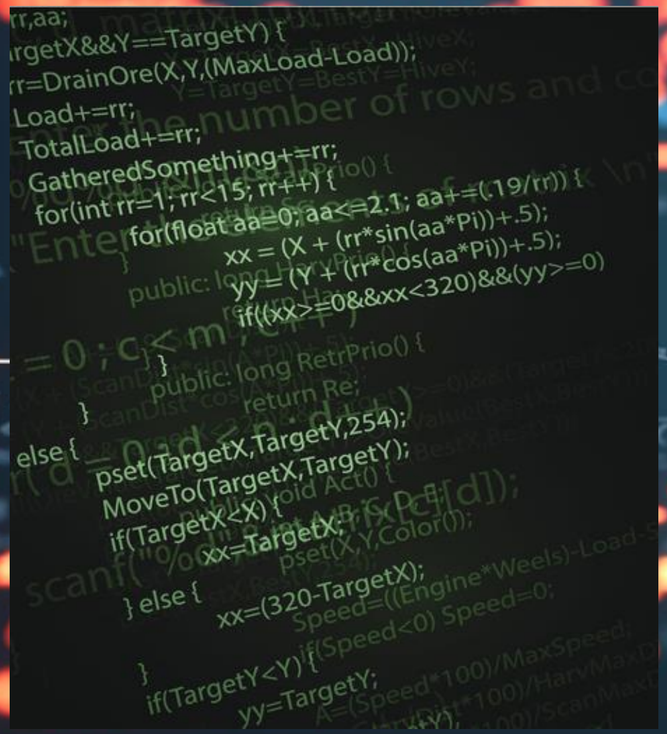
Por que usar um método tipo SWITCH?

Clareza e manutenibilidade: diferentes comandos e suas ações associadas podem ser claramente mapeados, tornando o código mais legível e fácil de manter.

Eficiência: quando comparado a múltiplos if-elif-else, um dicionário (simulando SWITCH) pode oferecer acesso mais rápido aos comandos, especialmente quando há muitos deles.

Flexibilidade para expansão: adicionar novos comandos é tão simples quanto adicionar uma nova entrada no dicionário, sem a necessidade de modificar uma cadeia longa de if-elif-else.

Exposição



Exemplo de implementação

```
def acender_luzes():
    return "Luzes acesas"

def ajustar_termostato(temperatura):
    return f"Termostato ajustado para {temperatura} graus"

def tocar_musica(musica):
    return f"Tocando {musica}"

def comando_nao_encontrado():
    return "Comando não reconhecido"

# Dicionário simulando SWITCH
comandos = {
    "acender luzes": acender_luzes,
    "ajustar termostato": ajustar_termostato,
    "tocar música": tocar_musica
}

# Função para processar comandos
def processar_comando(comando, *args):
    acao = comandos.get(comando, comando_nao_encontrado)
    return acao(*args)

# Exemplo de uso
print(processar_comando("acender luzes"))
print(processar_comando("ajustar termostato", 22))
print(processar_comando("tocar música", "Beethoven"))
print(processar_comando("abrir portas"))
```

Neste exemplo, **cada comando é mapeado para uma função específica. Quando um comando é recebido, a função processar_comando busca a ação correspondente no dicionário.** Se o comando não existir, uma função padrão é chamada, indicando que o comando não é reconhecido. **Isso torna o sistema extensível e fácil de manter.**

Vamos fazer um **quiz**

Teste de conhecimento

Em Python, qual é a melhor maneira de implementar uma estrutura de controle que simule o comportamento de um comando SWITCH presente em outras linguagens de programação?

- Usando uma série de if-elif-else.
- Utilizando um dicionário para mapear valores a funções.
- Aplicando um loop for.
- Utilizando a declaração switch-case.

Vamos fazer um **quiz**

Resposta da atividade

✗ Usando uma série de if-elif-else.

RESPOSTA ERRADA! Pois esquecer o *break* não causa um erro de compilação em Java; é uma questão de lógica.

✓ Utilizando um dicionário para mapear valores a funções.

RESPOSTA CORRETA! Porque, sem break, o Java continua a execução nos casos subsequentes até encontrar um break ou o fim do SWITCH.

✗ Aplicando um loop for.

RESPOSTA ERRADA! Pois o *SWITCH* não termina automaticamente; ele continua para os casos seguintes.

✗ Utilizando a declaração switch-case.

RESPOSTA ERRADA! Pois a execução continua sequencialmente nos casos seguintes, não em todos os casos do SWITCH.

Registro

Vamos fazer um **quiz**

Teste de conhecimento

Qual é o resultado de usar o método .get() em um dicionário em Python, quando a chave especificada não existe?

- Retorna um erro.
- Retorna None.
- Cria uma nova chave com valor padrão.
- Retorna o último valor do dicionário.

Vamos fazer um **quiz**

Resposta da atividade

✗ Retorna um erro.

RESPOSTA ERRADA! Pois o número de iterações em um *loop for* depende do tamanho da lista, não é fixo.

✓ Retorna None.

RESPOSTA CORRETA! Porque um *loop for* em *Python* itera sobre cada elemento da lista, começando pelo primeiro e terminando no último.

✗ Cria uma nova chave com valor padrão.

RESPOSTA ERRADA! Pois o *loop for* não discrimina entre elementos ímpares e pares; ele itera sobre todos os elementos.

✗ Retorna o último valor do dicionário.

RESPOSTA ERRADA! Pois o *loop for* não termina automaticamente na metade da lista; ele continua até o final, a menos que seja interrompido por uma condição ou comando explícito.

Vamos fazer um **quiz**

Teste de conhecimento

Como você pode simular a funcionalidade de um comando SWITCH em Python, considerando que a linguagem não apresenta essa estrutura nativamente?

- Usando uma série de comandos if-elif-else.
- Utilizando uma lista e iterando sobre ela com um loop for.
- Empregando um dicionário para mapear chaves a funções específicas.
- Implementando uma série de funções recursivas para cada caso.

Vamos fazer um **quiz**

Resposta da atividade

✗ Usando uma série de comandos if-elif-else..

RESPOSTA ERRADA! Pois, embora if-elif-else seja uma forma de lidar com múltiplas condições, não simula eficientemente a estrutura de um SWITCH.

✗ Utilizando uma lista e iterando sobre ela com um loop for.

RESPOSTA ERRADA! Porque uma lista com um *loop for* não simula a funcionalidade de um SWITCH, que é fundamentada em selecionar ações com base em chaves específicas.

✓ Empregando um dicionário para mapear chaves a funções específicas.

RESPOSTA CORRETA! Pois utilizar um dicionário em Python, no qual as chaves representam os casos do SWITCH e os valores são funções correspondentes a cada caso, é uma maneira eficaz de simular a estrutura SWITCH.

✗ Implementando uma série de funções recursivas para cada caso

RESPOSTA ERRADA! Pois funções recursivas não são apropriadas para simular a funcionalidade de um SWITCH, pois o SWITCH envolve escolher entre diversas opções baseadas em uma chave, não a repetição de chamadas de função.

© Getty Images

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1. Os conhecimentos sobre **o que é o comando SWITCH e como difere do IF-ELSE.**

2. A aprendizagem sobre **a sintaxe básica do SWITCH em diferentes linguagens de programação**.

3. Uma **análise de exemplos simples de uso do SWITCH e um estudo de caso que reforça sua importância.**

Saiba mais

Durante esta aula, **aprendemos conceitos utilizando a linguagem de programação Python**, mas muitos exercícios podem ser resolvidos também com o C#. Vamos ver?

CANAL DO OVIDIO. *MVC – ASP.Net MVC na Prática – Aula 1*. Disponível em: https://www.youtube.com/watch?v=-ciXTC3JZ9U. Acesso em: 21 mar. 2024.

Referências da aula

ALURA. *Python*: começando com a linguagem. Comparando variáveis. Disponível em: https://cursos.alura.com.br/course/python-introducao-a-linguagem/task/22779 . Acesso em: 21 mar. 2024.

CANAL DO OVIDIO. *MVC – ASP.Net MVC na Prática – Aula 1*. Disponível em: https://www.youtube.com/watch?v=-ciXTC3JZ9U. Acesso em: 21 mar. 2024.

O MATEMÁTICO COMPUTACIONAL. *Programação em C*: Estruturas de Seleção. Disponível em: https://www.youtube.com/watch?v=MWS8KSfLkmI. Acesso em: 21 mar. 2024.

VALE, J. C. S. *Estruturas de Seleção em Python – #07*. DEV, 2022. Disponível em: https://dev.to/jcarlosvale/estruturas-de-selecao-em-python-07-2mac. Acesso em: 21 mar. 2024.

Identidade visual: Imagens © Getty Images.

Educação Profissional Paulista

Técnico em
Desenvolvimento de Sistemas

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO