Linguagem de Programação Python

Comandos de Repetição

Professor: Ritomar Torquato

1

Repetições

Objetivos: Identificar a necessidade de uso de comandos de repetição.

2

Estruturas para Algoritmos

Estruturas básicas de controle dos algoritmos:

1. Sequenciação
2. Decisão
3. Repetição

3

Estrutura Sequencial ou Sequenciação

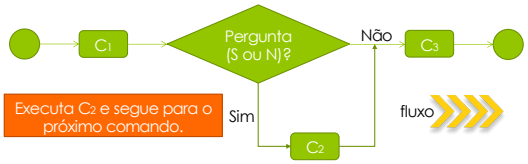
Todo algoritmo é uma sequência. A sequenciação é aplicada quando a solução do problema é decomposta em passos individuais.

A ordem na qual os comandos se encontram é relevante, pois serão executados um de cada vez, estritamente, de acordo com essa ordem.

4

Estrutura Condicional ou Decisão/Seleção

Um ou mais comandos só serão executados se uma condição for verdadeira.

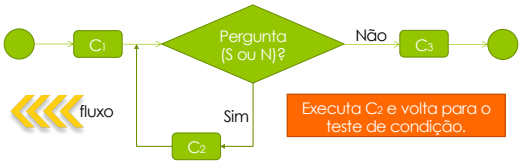


Usada quando há a necessidade de testar alguma condição e em função da mesma tomar uma atitude.

5

Estrutura de Repetição ou Iteração

Também é conhecida por "looping" ou laço, permite que comandos sejam repetidos um número **determinado** ou **indeterminado** de vezes.



A diferença fundamental da condição é o **sentido** do fluxo.

6

Estrutura de Repetição ou Iteração

- Repetir conjunto de comandos

comer.malhar.dormir.repetir

7

Pela definição...

algoritmo

Um algoritmos deve ser finito.

substantivo masculino

1. MATEMÁTICA
sequência finita de regras, raciocínios ou operações que, aplicada a um número finito de dados, permite solucionar classes semelhantes de problemas.
2. INFORMÁTICA
conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas.

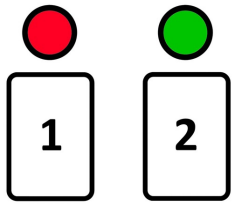
Origem

ETIM lat.medv. *algorismus*, com infl. do gr. *arithmós* 'número'

Conjunto **finito** e ordenado de comandos que levam à solução de um problema.

8

Condição de Parada



não faça isso

faça isso

9

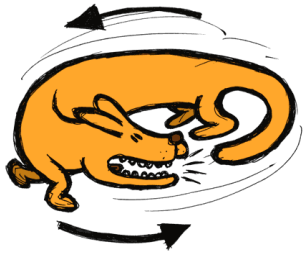
Estrutura de Repetição ou Iteração

Desviar o um programa para mais distante do final por um número **indeterminado** de vezes torna possível ele não chegar ao fim.

Se a condição não for atingida o programa pode não ter fim.

10

Loop Infinito



11

Estrutura de Repetição ou Iteração

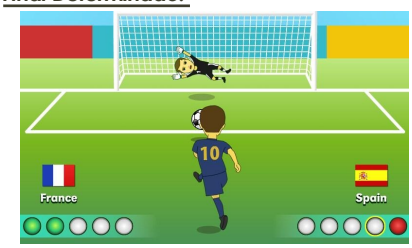
- **Final Determinado**: a quantidade de vezes que um bloco de comandos será repetida é conhecida antes do início.

for

12

Estrutura de Repetição ou Iteração

- **Final Determinado:**



13

Estrutura de Repetição ou Iteração

- **Final Determinado**: a atitude de "Bater penalti." será repetida um número determinado de vezes (cinco).

```
for i in range(5):
    print('Bater penalti.')
print('Cobrança finalizada.')
```

Atenção para a endentação e blocos de comandos.

Apenas `print('Bater penalti.')` faz parte da repetição.

O comando `print('Cobrança finalizada.')` é executado apenas **uma vez após a repetição** ser finalizada.

14

Comando for

- **Final Determinado**: Imprime as letras da frase, uma por linha.

```
frase = "Curso de Python"
for letra in frase:
    print(letra)
```

15

Comando for

- **Final Determinado**: Imprimir os números de 0 a 9.

```
for i in range(10):
    print(i)
```

16

Comando for

- **Final Determinado**: Imprimir os números de 1 a 10.

```
for i in range(10):
    print(i+1)
```

17

Comando for

- **Final Determinado**: Imprimir os números de 10 a 19.

```
for i in range(10, 20):
    print(i)
```

18

Comando `for`

- **Final Determinado**: Imprime os múltiplos de 3 entre 0 e 9.

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

19

Exemplo 1 de 2

- Escreva um programa que leia um número de **início** e uma **quantidade**. Mostre a **quantidade** de números pares maiores que o número de **início**. Por exemplo:
  - Início: 15; Quantidade 5; Deve mostrar:
    - 16 18 20 22 24

    (5 números pares maiores que 15)
  - Início: 28; Quantidade 7; Deve mostrar:
    - 30 32 34 36 38 40 42

    (7 números pares maiores que 28)

20

```
def e_par(n):
    return n % 2 == 0

def numeros_pares(inicio, quantidade):
    if e_par(inicio):
        proximo = inicio + 2
    else:
        proximo = inicio + 1

    numeros_pares = ''
    for n in range(quantidade):
        numeros_pares += str(proximo) + ' '
        proximo += 2

    # .strip() remove espaço o espaço em branco final
    return numeros_pares.strip()

def main():
    num_inicio = int(input('Início do intervalo: '))
    qtd = int(input('Quantidade de números pares: '))
    print(f'{qtd} pares após {num_inicio}:')
    print(numeros_pares(num_inicio, qtd))

if __name__ == '__main__':
    main()
```

21

```
def numeros_pares(inicio, quantidade):
    if e_par(inicio):
        inicio += 2
    else:
        inicio += 1

    numeros_pares = ''
    for n in range(inicio, inicio + (quantidade * 2), 2):
        numeros_pares += str(n) + ' '

    # .strip() remove espaço o espaço em branco final
    return numeros_pares.strip()

def main():
    num_inicio = int(input('Início do intervalo: '))
    qtd = int(input('Quantidade de números pares: '))
    print(f'{qtd} pares após {num_inicio}:')
    print(numeros_pares(num_inicio, qtd))

if __name__ == '__main__':
    main()
```

22

Exemplo 2 de 2

- Escreva um programa que leia o nome e a nota de 10 alunos e mostre o nome e a nota do melhor aluno da sala.

23

```
def melhor_de(n):
    nota_melhor = 0
    nome_melhor = ''

    for k in range(n):
        nome = input('Digite o nome do aluno: ')
        nota = float(input('Nota do aluno: '))
        if nota > nota_melhor:
            nome_melhor = nome
            nota_melhor = nota
    return nome_melhor, nota_melhor

def main():
    nome, nota = melhor_de(10);
    print(f'Melhor aluno: {nome}')
    print(f'Nota: {nota:.2f}')

if __name__ == '__main__':
    main()
```

Inicialização de variáveis auxiliares.

24