

Algoritmos e Lógica de Programação

Prof. Flavius Gorgônio

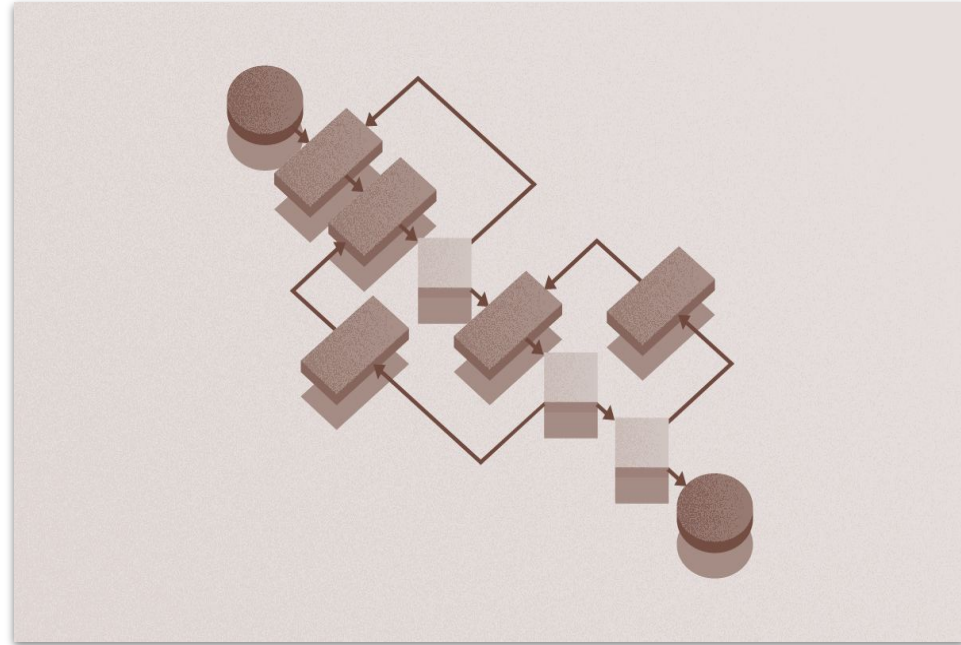


Semana 7

Estruturas de controle de repetição: A estrutura WHILE

Agenda

- Motivação
- Estruturas de controle de fluxo
 - Estruturas de controle de decisão
 - Estruturas de controle de repetição
- Estruturas de controle de repetição
 - Repetição com teste lógico
 - Repetição com contador
- Aplicações e exemplos

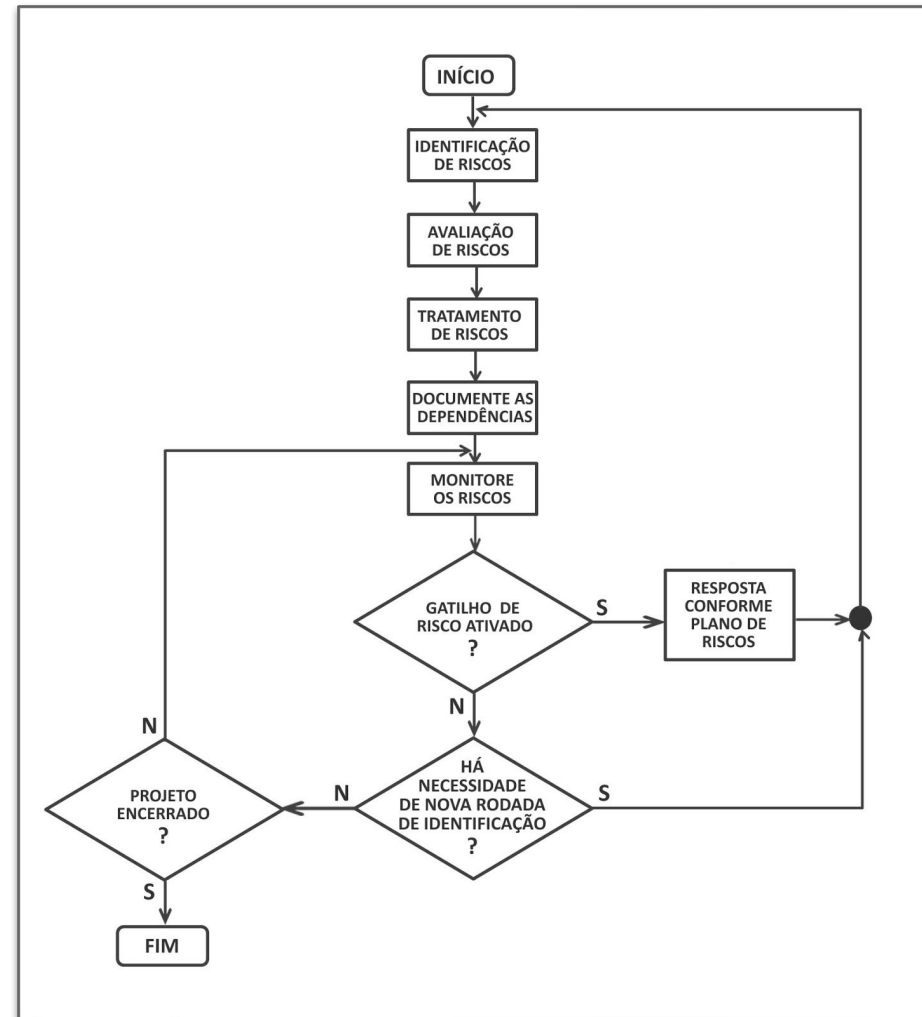


Motivação

Suponha que você precisa implementar o fluxo de verificação de riscos em projetos apresentado na imagem ao lado através de um programa de computador

Perceba que pode ser necessário que o **programa** REPITA ALGUNS TRECHOS em determinados momentos do fluxo

Alguns setas direcionam o fluxo para trechos que já foram executados antes



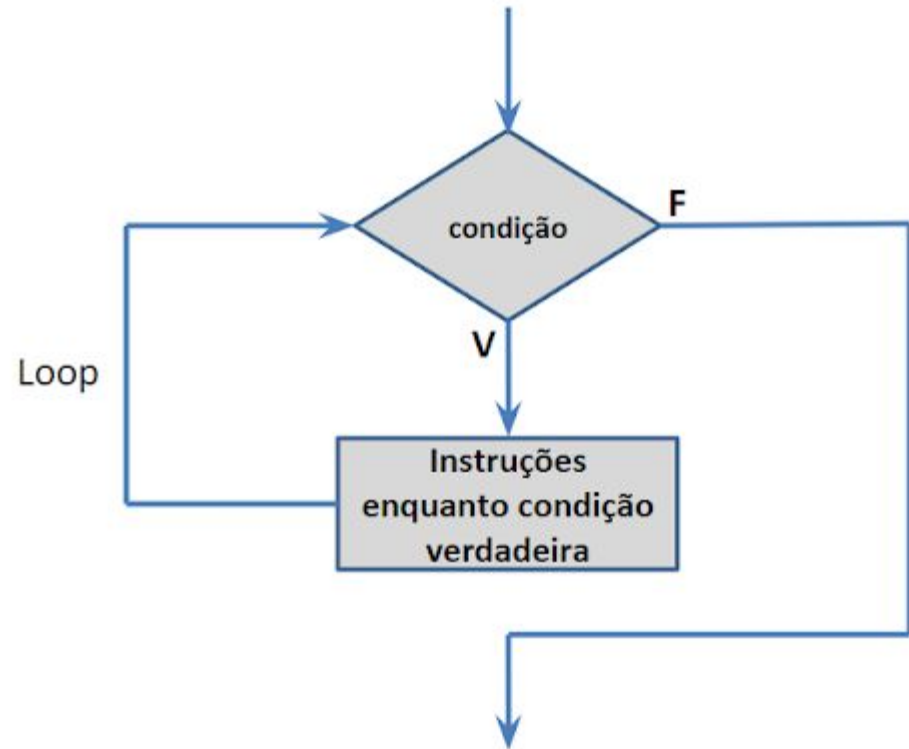
A estrutura **enquanto ... fim**

Uma estrutura de repetição do tipo **enquanto ... fim** permite executar, zero ou mais vezes, um ou mais comandos (bloco de comandos)

Enquanto a expressão resultar em VERDADEIRO, o comando (ou bloco de comandos) subordinado à estrutura é executado

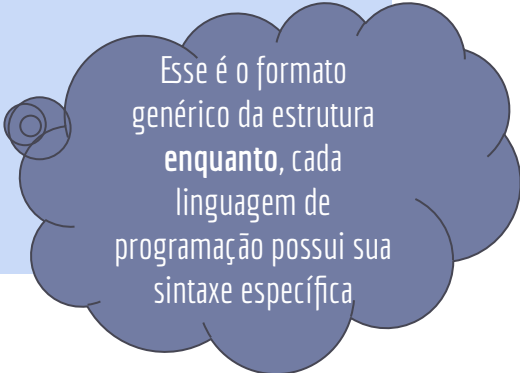
Se resultar em FALSO, o comando (ou bloco de comandos) subordinado à estrutura não é executado

Em ambos os casos, a execução continua com os comandos subsequentes à estrutura de decisão



A forma genérica da estrutura **enquanto ... fim**

```
...  
enquanto <condição> início  
    instrução_1  
    instrução_2  
    ...  
    instrução_n  
fim  
...
```



Esse é o formato genérico da estrutura **enquanto**, cada linguagem de programação possui sua sintaxe específica

A condição (cor vermelha) é verificada e, no caso de ser verdadeira, o bloco de comandos subordinado à estrutura (cor azul) será executado

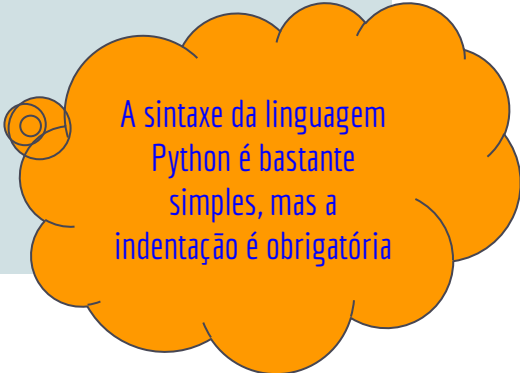
Ao final da execução do bloco, o controle do programa retorna ao teste e a condição é novamente verificada

Enquanto a condição for verdadeira, o bloco de comandos (conhecido como laço) continuará a ser executado

Se a condição falhar, o bloco azul não é executado e o fluxo continua após o término do laço

A estrutura **while** no Python

```
...  
while <condição>:  
    instrução_1  
    instrução_2  
    ...  
    instrução_n  
...
```



A sintaxe da linguagem Python é bastante simples, mas a indentação é obrigatória

Na linguagem Python, a estrutura enquanto é nomeada de while

O delimitador de início de bloco é o símbolo de dois pontos (:)

Não há delimitador de final de bloco, o fim da indentação delimita o término do bloco

O bloco deve, obrigatoriamente, estar indentado mais à direita em relação à indentação do while

Exemplo de uso da estrutura while

```
print("Contando de 1 até 10")

i = 1                                # Inicialização

while i <= 10:                        # Teste

    print(i)

    i = i + 1                        # Atualização

print("Fim do programa!")
```

É necessário usar um critério de parada na condição teste do while (#Teste)

A variável utilizada no critério de parada deve ser inicializada antes do while (#Inicialização)

A variável utilizada no critério de parada deve ser atualizada dentro do while (#Atualização)

O alinhamento do código demarca o bloco subordinado ao while

Calculando a média

Escreva um programa no Python que leia um conjunto de 3 notas, calcule a média aritmética dessas notas e determine se o(a) aluno(a) foi APROVADO/A ou REPROVADO/A

Para fins de aprovação, considere que a média mínima é 7.0 (sete)

```
#####  
## Programa Cálculo da Média de Aluno com Várias Notas  
## UFRN/CERES/DCT/DCT1101 - Alg e Log de Programação  
## by @flgorgonio - 04/09/2017  
#####  
  
n1 = int(input("Informe a 1a. nota: "))  
n2 = int(input("Informe a 2a. nota: "))  
n3 = int(input("Informe a 3a. nota: "))  
media = (n1 + n2 + n3) / 3  
print("Sua média foi: %.1f"%media)  
if media >= 7.0:  
    print("Parabéns! Você foi aprovado!")  
else:  
    print("Pena, você foi reprovado!")  
print("Fim do Programa")
```

Esta versão só funciona para
médias com 3 notas

Usando o while

A versão a seguir utiliza a estrutura while, para repetir a entrada de dados várias vezes

Utiliza-se menos variáveis do que no exemplo anterior

Utiliza-se uma variável do tipo acumulador (somaNota) para armazenar a soma das entradas

```
#####  
## Programa Cálculo da Média de Aluno com Várias Notas  
## UFRN/CERES/DCT/DCT1101 - Alg e Log de Programação  
## by @flgorgonio - 04/09/2017  
#####  
soma = 0  
i = 1  
while i <= 3:  
    nota = float(input("Informe a nota %d: "%i))  
    soma = soma + nota  
    i = i + 1  
media = soma / 3  
print("Sua média foi: %.1f"%media)  
if media >= 7.0:  
    print("Parabéns! Você foi aprovado!")  
else:  
    print("Pena, você foi reprovado!")  
print("Fim do Programa")
```

Esta versão também
funciona apenas para
médias com 3 notas

Melhorando o código

É possível generalizar o código anterior, de forma que ele funcione para médias com **n** notas

O valor de **n** precisa ser definido de alguma forma, por exemplo, pode ser fornecido pelo usuário

Esta versão funciona para médias com qualquer quantidade de notas

```
#####  
## Programa Cálculo da Média de Aluno com Várias Notas  
## UFRN/CERES/DCT/DCT1101 - Alg e Log de Programação  
## by @flgorgonio - 04/09/2017  
#####  
i = 1  
soma = 0  
n = int(input("Quantas notas? "))  
while i <= n:  
    nota = float(input("Informe a nota: "))  
    soma = soma + nota  
    i = i + 1  
media = soma / n  
print("Sua média foi: %.1f"%media)  
if media >= 7.0:  
    print("Parabéns! Você foi aprovado!")  
else:  
    print("Pena, você foi reprovado!")  
print("Fim do Programa")
```

Cara ou coroa



Implemente um jogo de Cara ou Coroa entre um jogador humano e o computador

- O usuário deverá escolher entre Cara ou Coroa
- O computador será responsável por jogar a moeda (sorteia valor entre 1 e 2)
- Se o valor sorteado for igual ao valor que o usuário escolheu, ele ganha o jogo. Se for diferente, ele perde
- Adicione um laço (estrutura de repetição) que permita ao usuário jogar várias vezes

```
###  
### Programa Cara ou Coroa  
###
```

```
from random import randint
```

```
jog = int(input("Escolha 1 p/ Cara ou 2 p/ Coroa: "))  
moeda = randint(1,2)  
if (jog == 1) and (moeda == 1):  
    print("Deu Cara, você ganhou!")  
elif (jog == 1) and (moeda == 2):  
    print("Deu Coroa, você perdeu!")  
elif (jog == 2) and (moeda == 2):  
    print("Deu Coroa, você ganhou!")  
else:  
    print("Deu Cara, você perdeu!")  
print("Fim do Programa")
```

Este é o programa
base, modifique-o!

Cara ou coroa modificado

Dicas:

- Pergunte ao usuário quantas partidas ele quer jogar
- Adicione um contador para as partidas
- Use uma estrutura de repetição para controlar o número de repetições
- Incremente o contador a cada execução do laço

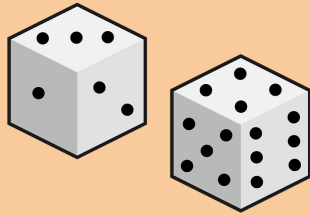
Você pensou em outra solução? Qual?

```
# Parte inicial do código aqui
...
p = int(input('Quantas partidas você quer jogar? '))
i = 1
while i <= p:
    print("Partida no.", i)
    jog = int(input("Escolha 1 p/ Cara ou 2 p/ Coroa: "))
    moeda = randint(1,2)
    if (jog == 1) and (moeda == 1):
        print("Deu Cara, você ganhou!")
    elif (jog == 1) and (moeda == 2):
        print("Deu Coroa, você perdeu!")
    elif (jog == 2) and (moeda == 2):
        print("Deu Coroa, você ganhou!")
    else:
        print("Deu Cara, você perdeu!")
    i = i + 1
    print()
print("Fim do Programa")
```



Lançando a sorte com dados

```
###  
### Programa Jogo de Dados  
###  
  
import random  
  
jog = random.randint(1,6)  
comp = random.randint(1,6)  
print("Jogador   : ", jog)  
print("Computador: ", comp)  
if jog > comp:  
    print("Jogador Ganhou!")  
elif comp > jog:  
    print("Computador Ganhou!")  
else:  
    print("Empate!")  
print("Fim do Jogo!")
```



Programa base

Escreva um programa em Python que simule um jogo de dados entre um jogador humano e um computador. O programa deve lançar dados eletrônicos, simulados por software

A cada partida, o programa deverá exibir o resultado de cada um dos dados e indicar quem foi o vencedor ou se houve empate

Ao final da partida, o programa deve perguntar ao usuário se ele deseja jogar novamente. Caso ele opte por continuar o jogo, este deve ser executado novamente a partir do início

Caso ele deseje encerrar a disputa, o programa deverá exibir o número de vitórias do jogador, o número de vitórias do computador e o número de empates

Incluindo a estrutura de repetição

Dicas:

- Use uma variável do tipo string para controlar o laço
- Inicialize com uma pergunta
- Repita a pergunta ao final do bloco

Inicialização

Teste

Atualização

```
# Parte inicial do código aqui
...
resp = input("Deseja jogar (S/N)? ")
while resp == "S" or resp == "s":
    jog = random.randint(1,6)
    comp = random.randint(1,6)
    print("Jogador   : ", jog)
    print("Computador: ", comp)
    if jog > comp:
        print("Jogador Ganhou!")
    elif comp > jog:
        print("Computador Ganhou!")
    else:
        print("Empate!")
    resp = input("Deseja jogar novamente (S/N)? ")
print("Fim do Jogo!")
```

Como contar os pontos?

- Defina uma variável para contar os pontos de cada participante
 - ➔ ptsJog
 - ➔ ptsComp
 - ➔ ptsEmp
- Cada variável deve ser inicializada com valor zero (zerada), antes do **while**

```
ptsJog = 0  
ptsComp = 0  
ptsEmp = 0
```

```
import random  
ptsJog = 0  
ptsComp = 0  
ptsEmp = 0  
resp = input("Deseja jogar (S/N)? ")  
while resp.upper() == "S":  
    jog = random.randint(1,6)  
    comp = random.randint(1,6)  
    print("Jogador    : ", jog)  
    print("Computador: ", comp)  
    ...  
  
# fim do while
```


Como contar os pontos?

- A cada vitória, do jogador ou do computador, as respectivas variáveis de contagem de pontos devem ser alteradas
- A variáveis que computa os empates também deve ser alterada, quando isso ocorrer
- Use a estrutura de decisão já existente no programa para definir qual variável será alterada

```
while resp.upper() == "S":  
    ...  
    if jog > comp:  
        print("Jogador Ganhou!")  
        ptsJog = ptsJog + 1  
    elif comp > jog:  
        print("Computador Ganhou!")  
        ptsComp = ptsComp + 1  
    else:  
        print("Empate!")  
        ptsEmp = ptsEmp + 1  
    ...  
...
```

Como contar os pontos?

- Após o encerramento do jogo, exibir o resultado final
- As variáveis irão conter os pontos de cada participante e a quantidade de empates

Melhore seu programa:

- Você consegue exibir o número de partidas que foram disputadas?
- Que tal exibir a quantidade de partidas vencidas por cada jogador em valores percentuais?

```
...
    resp = input("Jogar novamente (S/N)? ")
print()
print("Pontuação Final")
print("-----")
print("Jogador      :", ptsJog)
print("Computador:", ptsComp)
print("Empates      :", ptsEmp)
print("Fim do Programa")
```

Jogo de Dados 7 ou 11

- Escreva um programa no Python que simule um jogo de dados disputado entre um jogador humano e um computador, onde dois dados eletrônicos (simulados por software, através de valores aleatórios) devem ser lançados simultaneamente. O jogador vence se a soma dos pontos dos dois dados for 7 ou 11, caso contrário vence o computador
- Ao final da partida, o programa deverá perguntar ao usuário se o mesmo deseja jogar novamente. O programa deverá permitir uma nova partida, caso a resposta seja afirmativa ou encerrar em caso negativo.

Jogo Zerinho ou Um

- Escreva um programa no Python que simule o tradicional jogo do ZERINHO ou UM. Nesse jogo, deve haver pelo menos três jogadores, onde o primeiro é um jogador humano e os demais são simulados pelo computador. Para jogar, eles devem escolher um dos valores: 0 (ZERO) ou 1 (UM). Vence aquele que apresentar um valor distinto de todos os outros. Se todos escolherem números iguais, a partida está empatada. O programa deve indicar se houve um vencedor ou se houve empate.
- Permita ao jogador repetir o jogo, caso deseje.

Pedra, Papel e Tesoura

- Escreva um programa no Python que simule o jogo PEDRA, PAPEL e TESOURA, a ser disputado entre um jogador humano e o computador.
- O jogador humano deverá escolher entre uma das três opções e a escolha do computador deverá ser feita de forma aleatória.
- O programa deverá realizar o julgamento e definir quem venceu o jogo, lembrando que PEDRA vence TESOURA, TESOURA vence PAPEL e PAPEL vence PEDRA. Considere a possibilidade de haver empate.
- O jogador deverá poder jogar novamente, caso deseje.

Estratégias de repetição

No programa **Cara ou Coroa**, a quantidade de repetições era definida pelo usuário, no início do programa:

```
...  
  
p = int(input('Qtas partidas quer jogar? '))  
  
i = 1  
  
while i <= p:  
    print("Partida no.", i)  
    ...  
    i = i + 1  
  
...
```

No programa **Jogo de Dados**, a cada execução do laço, o usuário deveria informar se queria ou não repetir o jogo:

```
...  
  
resp = input("Deseja jogar (S/N)? ")  
  
while resp.upper() == "S":  
    jog = random.randint(1,6)  
    ...  
    resp = input("Jogar novamente (S/N)? ")  
  
...
```

Outras estratégias de repetição

Experimente usar outras estratégias de repetição:

- a. Disputar 5 partidas e identificar quantas partidas cada jogador venceu, apresentando inclusive o número de empates
- b. Disputar várias partidas até que um dos jogadores obtenha 5 vitórias
- c. Disputar várias partidas até que um dos jogadores obtenha 3 vitórias seguidas. No caso de ocorrer um empate, deve-se zerar a contagem



Adivinhe o Número

- Escreva um programa no Python que simule um jogo de adivinhação, onde o computador sorteará um valor entre 1 e 9 e o jogador terá três chances para acertar o número.
- Caso o usuário acerte na primeira tentativa, o programa deverá exibir a mensagem “VOCÊ TEVE MUITA SORTE” e, em seguida, encerrar o programa.
- Se errar, o programa deverá fornecer uma primeira dica, dizendo “DIGITE UM NÚMERO MENOR” ou “DIGITE UM NÚMERO MAIOR”, de acordo com o valor fornecido.

Adivinhe o Número

- Caso o usuário acerte na segunda tentativa, o programa deverá exibir a mensagem “VOCÊ JOGA BEM, MAS AINDA CONTOU SORTE” e, em seguida, encerrar o programa.
- Se errar, o programa deverá fornecer uma última dica, dizendo “DIGITE UM NÚMERO MENOR” ou “DIGITE UM NÚMERO MAIOR”, de acordo com o valor fornecido.

Adivinhe o Número

- Caso o usuário acerte na terceira e última tentativa, o programa deverá exibir a mensagem “VOCÊ É UM EXCELENTE ESTRATEGISTA” e, se errar, deverá fornecer a mensagem “ANALISE MELHOR SUA ESTRATÉGIA ANTES DE JOGAR NOVAMENTE”.
- Ao final, o usuário deverá ter a possibilidade de repetir o jogo ou de encerrar a disputa.

Adivinhe o Número

Modifique o programa anterior, progressivamente, de forma a atender os seguintes requisitos:

- O número sorteado deve estar entre 1 e 100
- Ao final do jogo, o programa deverá informar quantos palpites foram necessários para que o jogador acertasse o número
- Limite a partida a, no máximo, 10 tentativas
- Valide os palpites, de forma que o jogador seja desclassificado se fornecer um palpite fora dos limites indicados. Os limites devem ser atualizados a cada palpite fornecido
- Crie uma escala de pontuação baseada na quantidade de palpites

Sugestões de bibliografia

Básica

- MENEZES, Nilo Ney Coutinho. Introdução à programação com Python: algoritmos e lógica de programação para iniciantes. 2a ed. São Paulo: Novatec, 2010. 328p. ISBN: 9788575224083
- FARRELL, Joyce. Lógica e design de programação: introdução. São Paulo: Cengage Learning, 2010. xiv, 416p. ISBN: 9788522107575.
- SOUZA, Marco; GOMES, Marcelo; SOARES, Márcio; CONCILIO, Ricardo. Algoritmos e Lógica de Programação. 2a ed. São Paulo: Cengage Learning, 2013. 240 p. ISBN: 9788522111299

Complementar

- ASCENCIO, Ana Fernanda; CAMPOS, Edilene Aparecida. Fundamentos da programação de computadores: Algoritmos, Pascal, C/C++(Padrão Ansi) e Java. 3a ed. São Paulo: Pearson Prentice Hall. 569 p. ISBN: 9788564574168
- ZELLE, John M. Python programming: an introduction to computer science. 2nd ed. Sherwood, Or.: Franklin, Beedle & Associates, c2010. xiv, 514 p. ISBN: 9781590282410.
- LUTZ, Mark; ASCHER, David. Learning Python. 4nd ed. Sebastopol, CA: O Reilly, c2009. xlv, 1162 p. ISBN: 978059615064.



Copyright

Este material é para uso exclusivo durante as aulas da disciplina DCT1101 - ALGORITMOS E LÓGICA DE PROGRAMAÇÃO, do curso de Bacharelado em Sistemas de Informação da Universidade Federal do Rio Grande do Norte, não estando autorizada a sua publicação, compartilhamento, divulgação ou utilização em outros contextos diferentes dos aqui apresentados.

Todos os textos, imagens, exemplo de código e demais materiais utilizados nesses slides são apenas para fins didáticos, o autor e a instituição não permitem a sua utilização sem autorização expressa, assim como não se responsabilizam pelo seu uso indevido ou por danos causados pelos mesmos.