

PROTOTYPING

QUANDO A MÁQUINA COMEÇA
A TOMAR DECISÕES

ANDRÉ DE FREITAS DAVID



02

LISTA DE FIGURAS

Figura 1 – A página do jogo <i>Diablo IV</i> exige que o visitante informe seu ano de nascimento	8
Figura 2 – Criação das variáveis dia, nome e saldo em um fluxograma.....	9
Figura 3 – Fluxograma representando desvio condicional simples	11
Figura 4 – Fluxograma representando o desvio condicional composto	12
Figura 5 – Fluxograma representando desvio condicional composto	13
Figura 6 – Realização de dois testes lógicos separados	22
Figura 7 – Realização de testes lógicos utilizando o or	22
Figura 8 – Realização de testes lógicos utilizando o and	23
Figura 9 – Execução do programa com erro no cupom	26

LISTA DE QUADROS

Quadro 1 – Tipos de dados básicos no Python	10
---------------------------------------------------	----

EXEMPLO

LISTA DE TABELAS

Tabela 1 – Tabela verdade operador or (ou)	23
Tabela 2 – Tabela verdade operador and (e).....	23

EMSE

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Criação das variáveis dia, nome e saldo em pseudocódigo.....	9
Código-fonte 2 – Criação das variáveis dia, nome e saldo em Python.....	9
Código-fonte 3 – Algoritmo com if simples em pseudocódigo	14
Código-fonte 4 – Sintaxe do if simples em Python.....	14
Código-fonte 5 – Algoritmo com if simples em Python.....	14
Código-fonte 6 – Algoritmo com dois if simples em pseudocódigo.....	15
Código-fonte 7 – Algoritmo com if composto em pseudocódigo	16
Código-fonte 8 – Sintaxe do if composto em Python	16
Código-fonte 9 – Algoritmo com if simples em Python.....	17
Código-fonte 10 – Algoritmo com if encadeado em pseudocódigo.....	18
Código-fonte 11 – Algoritmo com if encadeado em Python	19
Código-fonte 12 – Algoritmo da operadora com if encadeado em Python.....	20
Código-fonte 13 – Algoritmo da operadora com o uso do <i>elif</i> em Python.....	20
Código-fonte 14 – Solução do exercício 1 em Python	25
Código-fonte 15 – Solução do exercício 2 em Python	25
Código-fonte 16 – Utilização da função upper().....	26
Código-fonte 17 – Solução do exercício 2 em Python com tratamento de letras maiúsculas	27
Código-fonte 18 – Início da solução do exercício 3 em Python	27
Código-fonte 19 – Desvios da solução do exercício 3 com if encadeado em Python	28
Código-fonte 20 – Desvios da solução do exercício 3 com elif em Python.....	28
Código-fonte 21 – Solução do exercício 3 com elif em Python.....	29
Código-fonte 22 – Solução do exercício 3 com desvio condicional encadeado em Python	30

SUMÁRIO

1 QUANDO A MÁQUINA COMEÇA A TOMAR DECISÕES.....	7
1.1 Encontrando novos caminhos.....	7
1.1.1 Eu preciso lembrar do passado?.....	7
1.2 Você decide... computador!.....	10
1.2.1 If simples em Python.....	13
1.2.2 If composto em Python.....	15
1.2.3 If encadeado em Python.....	17
1.2.4 O Python e o poder do elif!.....	19
2 OPERANDO A LÓGICA!.....	22
2.1 Casar ou comprar uma bicicleta?.....	22
2.1.1 Tudo ou nada!.....	23
2.2 Hora da prática.....	24
2.2.1 Ajudando o doutor!.....	24
2.2.2 Com que roupa eu vou?.....	25
2.2.3 Essa eu sabia com maçãs.....	27
HORA DE TREINAR.....	31
REFERÊNCIAS.....	33

1 QUANDO A MÁQUINA COMEÇA A TOMAR DECISÕES

1.1 Encontrando novos caminhos

Você já é capaz de pensar em algoritmos e criar programas para realizar tarefas diversas, como calcular os juros de uma determinada compra ou fazer a divisão da conta do restaurante.

Mas e se começarmos a pensar em condições que mudam a forma como nosso algoritmo deve se comportar, como, por exemplo, uma taxa de juros diferenciada que dependa do número de parcelas?

É chegada a hora de aprendermos uma ferramenta de programação que nos auxiliará a desenvolver diferentes fluxos com base na análise de uma condição. São os *desvios condicionais*!

Ao longo deste capítulo, vamos aprender a propor algoritmos que levem em conta as variações que podem acontecer no cenário de aplicação! Vamos lá?

1.1.1 Eu preciso lembrar do passado?

Uma das características mais interessantes do estudo de algoritmos e programação é que o conteúdo vai se “acumulando” e as primeiras coisas que aprendemos oferecem o suporte para as demais.

É por isso que precisamos nos lembrar bem dos conceitos anteriores, e um dos mais importantes que você já aprendeu é a ideia de variável!

Quando construímos qualquer algoritmo, há dados que são temporários e, ainda assim, são importantes para a nossa solução. Esses dados são as variáveis.

Se tomarmos como exemplo a página de um game indicado para maiores de 18 anos, a data de nascimento do visitante do site é um dado **importante** (pois servirá como base para autorizar ou não o acesso do visitante) e **temporário** (pois não interessa armazenar esse dado por um prazo maior do que o da validação da idade).

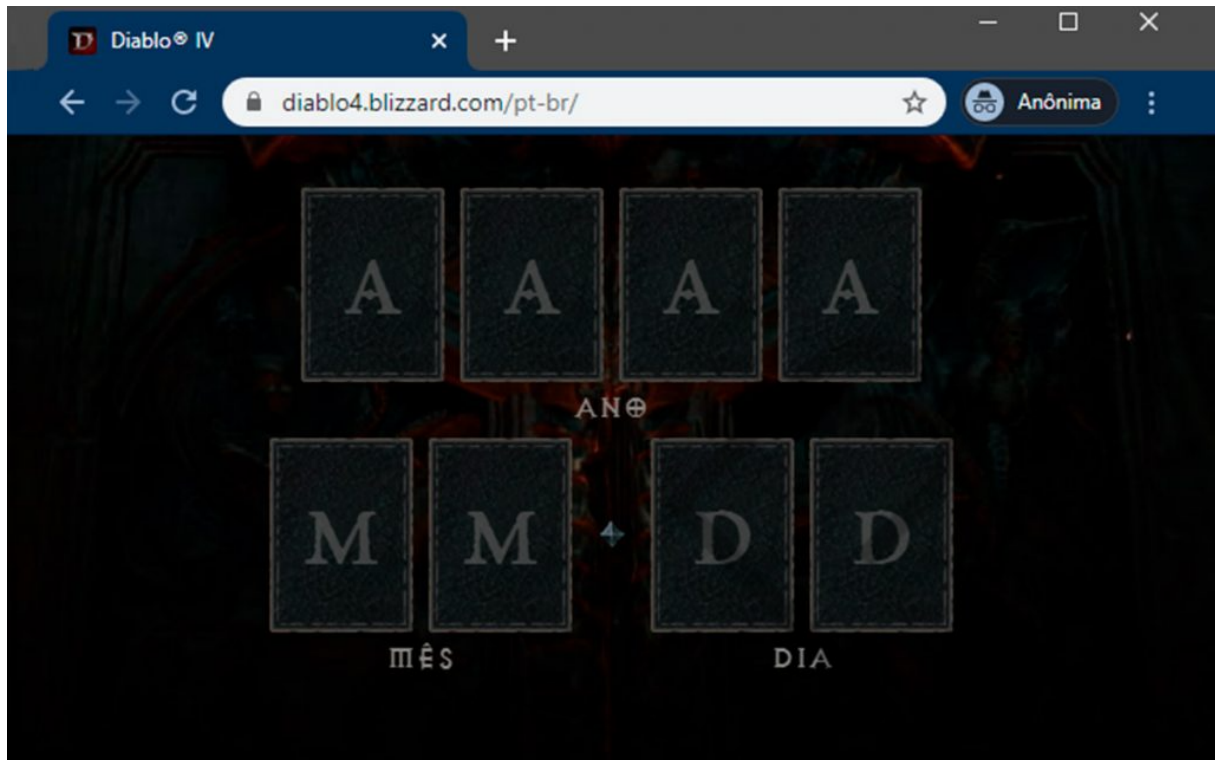


Figura 1 – A página do jogo *Diablo IV* exige que o visitante informe seu ano de nascimento
Fonte: Elaborado pelo autor (2020)

Um erro muito comum de quem começa a estudar programação é se esquecer de que as variáveis podem ser de **diferentes tipos**. Afinal, os dados podem ser de diferentes tipos, não é mesmo?

O **ano de nascimento** do nosso visitante é um número. E mais do que isso, é um número inteiro. Já seu **nome** é um texto e, portanto, outro tipo de dado. O **saldo em reais** que esse visitante tem na sua carteira virtual é, por sua vez, um número real.

Nos fluxogramas e pseudocódigos, a indicação do tipo de uma variável é feita no momento da criação delas – como ocorre na maior parte das linguagens de programação.

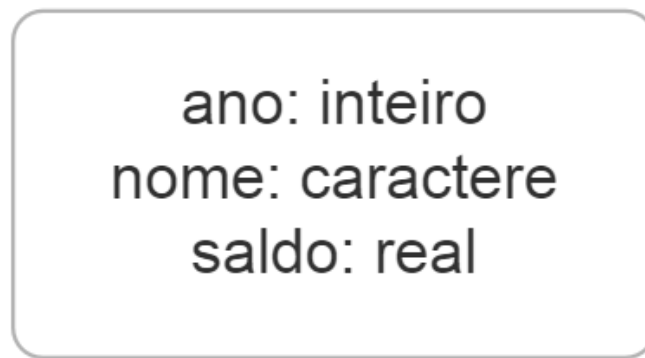


Figura 2 – Criação das variáveis dia, nome e saldo em um fluxograma
Fonte: Elaborado pelo autor (2020)

```
variáveis
ano: inteiro
nome: alfanumérico
saldo: real
```

Código-fonte 1 – Criação das variáveis dia, nome e saldo em pseudocódigo
Fonte: Elaborado pelo autor (2019)

No maravilhoso mundo do Python, tudo foi feito para facilitar a vida do programador. Dessa forma, ao atribuir um valor para uma variável, ela será criada e o tipo mais adequado será atribuído a ela.

Vamos fazer o teste de criar um script no PyCharm, chamado “tipos_variaveis.py”. O código-fonte a seguir serve para criar cada uma das variáveis, já atribuindo valores (como é característico do Python), e o comando *type* exibe o tipo dessas variáveis.

```
ano = 1989
nome = "Luke Skywalker"
saldo = 50.30
print(("O tipo da variável ano é {}".format(type(ano))))
print(("O tipo da variável nome é {}".format(type(nome))))
print(("O tipo da variável saldo é {}".format(type(saldo))))
```

Código-fonte 2 – Criação das variáveis dia, nome e saldo em Python
Fonte: Elaborado pelo autor (2020)

O resultado da execução desse programa indicará que o tipo da variável *ano* é **int** (números inteiros), enquanto a variável *nome* é do tipo **str** (sequência de caracteres alfanuméricos) e a variável *saldo* é do tipo **float** (números reais).

Tipos Básicos	Descrição	Exemplos
int	Números inteiros	1, 20, 1000
float	Números reais (ponto flutuante)	7.5, 2.07, 50.29
complex	Números complexos	4j, 5+2j, 15j
bool	Valores lógicos	True, False, 1, 0
string	Textos	"Bom dia", "x", "12", "f5"

Quadro 1 – Tipos de dados básicos no Python

Fonte: Elaborado pelo autor (2022)

Quando aprendemos uma nova linguagem de programação, é recomendável consultar a documentação para verificar quais são os tipos de variáveis disponíveis.

Agora que lembramos do conceito de variáveis, nosso desafio é: como fazer com que nosso programa tome decisões?

1.2 Você decide... computador!

É possível resolver muitos problemas com algoritmos que não precisam tomar decisões. Calcular a média de um aluno da FIAP, por exemplo, exige apenas que sejam informadas as notas do felizardo. Mas será que ele é mesmo um felizardo?

A tarefa de *olhar* para a média final do aluno e informar se ele pode celebrar ou se precisa ficar preocupado exige uma pergunta, ou melhor, uma condição!

Em algoritmos, nós chamamos de *desvio condicional* a estrutura que permite realizar um teste lógico e executar alguma ação dependendo do resultado do teste. Esse nome é fácil de entender: o algoritmo realizará um *desvio* na sua execução com base em uma *condição*.

Em um fluxograma, é fácil entender o funcionamento do desvio condicional:

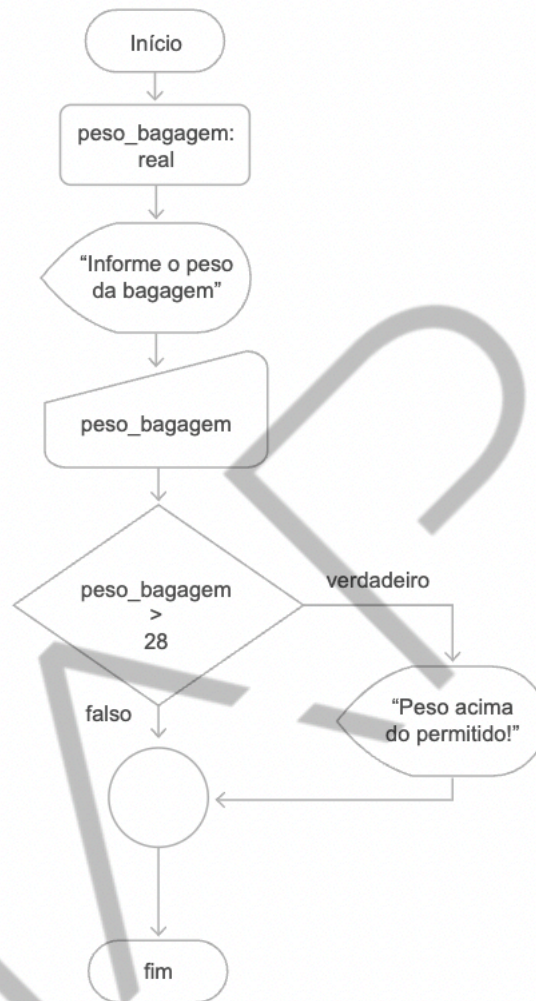


Figura 3 – Fluxograma representando desvio condicional simples
Fonte: Elaborado pelo autor (2020)

O algoritmo representado verifica uma *condição* (o peso da bagagem é maior que 28?) e toma um *desvio* (exibir a mensagem de peso excedido) caso o teste da condição seja verdadeiro.

Em algoritmos, chamamos esse tipo de desvio condicional de *desvio condicional simples*, pois ele apresenta um desvio apenas, se a condição testada for verdadeira; caso contrário, o programa segue o fluxo normal.

É possível que nos deparemos com problemas que exijam ações tanto para o caso de o teste ser verdadeiro quanto para o caso de o teste ser falso. Seguindo o

mesmo exemplo da passagem, vamos exibir uma mensagem caso o peso esteja em conformidade com os limites impostos pela companhia.



Figura 4 – Fluxograma representando o desvio condicional composto
Fonte: Elaborado pelo autor (2020)

Ao desvio condicional que é capaz de realizar uma ação para o caso de a condição ser verdadeira e outra ação para o caso de a condição ser falsa, damos o nome de *desvio condicional composto*.

Resta ainda um último cenário: o que acontece se, dependendo do resultado de uma condição, quisermos realizar um segundo teste? Em nosso caso, é possível que a companhia aérea permita aos clientes *premium* levarem até 32 kg de bagagem.

Para resolver esse problema, usamos uma estrutura chamada *desvio condicional encadeado*:

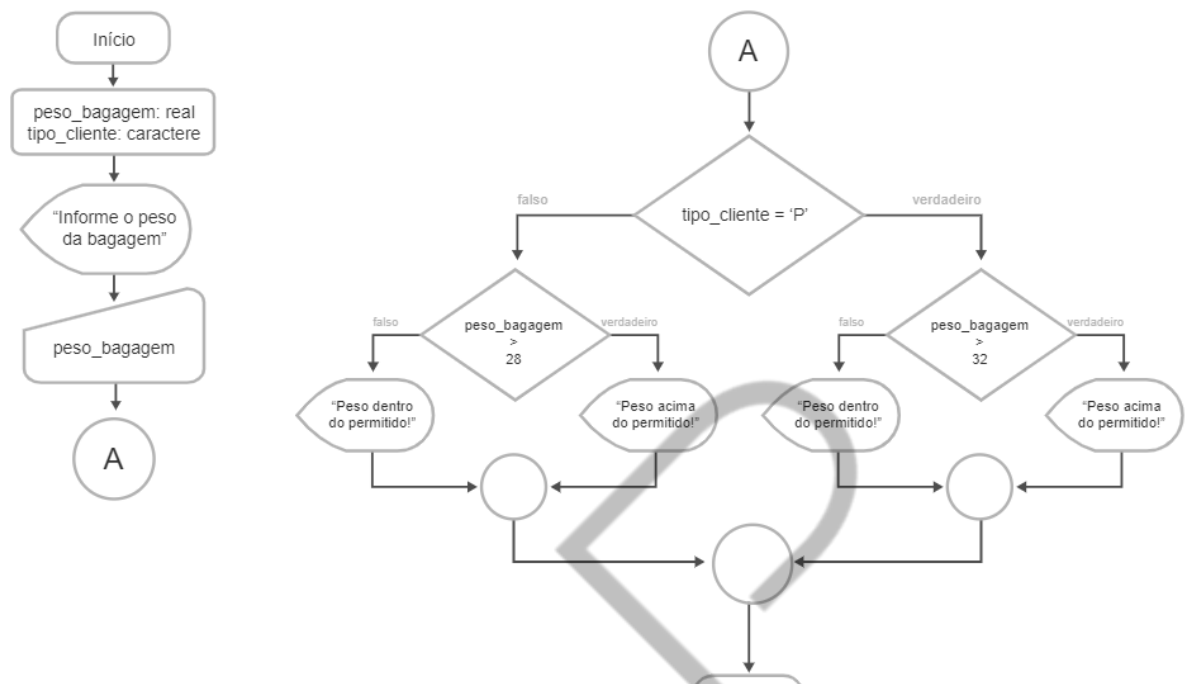


Figura 5 – Fluxograma representando desvio condicional composto
 Fonte: Elaborado pelo autor (2020)

É menos importante se preocupar com os **nomes** de cada uma das estruturas do que com as ocasiões em que devem ser usadas. Na sua jornada como programador, frequentemente os desvios condicionais serão referidos como *ifs*. Isso ocorre porque, na maior parte das linguagens de programação, essa é a palavra que designa um desvio.

Que tal conhecermos as aplicações de cada um deles em Python?

1.2.1 If simples em Python

Vamos imaginar que a FIAP esteja propondo a formação de um time de Esportes para representar a instituição e, para isso, realizará um campeonato interno no qual os alunos podem se inscrever. Porém, há uma condição: só podem ser inscritos alunos que são maiores de idade.

O algoritmo que implementa a solução para esse problema é simples:

```
variáveis
    idade: inteiro
    rm: alfanumérico
início
    Escreva "Por favor, digite seu RM"
    Leia rm
    Escreva "Por favor, digite sua idade"
    Leia idade
    Se idade >=18 então
        Escreva "Sua participação foi autorizada, aluno de RM",
rm
        Escreva "Mais instruções serão enviadas ao seu e-mail
cadastrado na FIAP!"
    Fim_se
Fim
```

Código-fonte 3 – Algoritmo com if simples em pseudocódigo
Fonte: Elaborado pelo autor (2020)

A lógica do problema já foi desenvolvida no pseudocódigo. Vamos escrever esse mesmo programa em um script em Python chamado *exemplo_if_simples.py*?

A sintaxe da estrutura condicional simples em Python é a seguinte:

```
if <condição>:
    <ação a ser realizada se a condição for verdadeira>
```

Código-fonte 4 – Sintaxe do if simples em Python
Fonte: Elaborado pelo autor (2020)

A estrutura do if exige uma atenção muito especial por parte do programador Python: esses **4 espaços em branco** que inserimos entre o início da linha e a ação que será realizada serve para que o interpretador entenda que aquele conteúdo está **dentro** do bloco verdadeiro do if. Sem eles o programa não funcionará.

O programa reescrito fica assim, portanto:

```
rm = input("Insira seu RM")
idade = input("Insira sua idade")

if int(idade) >= 18:
    print("Sua participação foi autorizada, aluno de RM
{}".format(rm))
    print("Mais informações serão enviadas para seu e-
mail cadastrado!")
```

Código-fonte 5 – Algoritmo com if simples em Python

Fonte: Elaborado pelo autor (2020)

Apesar de o programa atingir o objetivo proposto, ele possui algumas falhas no quesito *experiência do usuário*, não é mesmo? Afinal de contas, se alguém digitar que tem 17 anos, o programa não exibirá nada!

É por isso que vamos usar esse mesmo problema para o *if composto*!

1.2.2 If composto em Python

Se levarmos em conta o mesmo problema anterior, seria interessante exibir também uma mensagem para os alunos que são menores de idade.

Alguém mais apressado poderia escrever o seguinte algoritmo:

```
variáveis
    idade: inteiro
    rm: alfanumérico
início
    Escreva "Por favor, digite seu RM"
    Leia rm
    Escreva "Por favor, digite sua idade"
    Leia idade
    Se idade >=18 então
        Escreva "Sua participação foi autorizada, aluno de RM",
rm
        Escreva "Mais instruções serão enviadas ao seu e-mail
cadastrado na FIAP!"
        Fim_se
    Se idade <18 então
        Escreva "Sua participação não foi autorizada por causa
da sua idade"
        Fim_se
Fim
```

Código-fonte 6 – Algoritmo com dois if simples em pseudocódigo

Fonte: Elaborado pelo autor (2020)

Vamos analisar: caso um aluno tenha digitado que tem 17 anos, o primeiro desvio condicional não entrará na parte verdadeira (pois o teste idade >= 18 resultará **falso**) e o segundo desvio condicional executará a parte verdadeira (pois o teste idade <18 resultará **verdadeiro**).

Parece que o problema está resolvido, mas há um grande problema em termos de *uso dos recursos da máquina* nesse caso! Se o aluno digitar que tem 18 anos, o primeiro teste será verdadeiro e as mensagens serão exibidas. Então por que perguntaremos novamente se a idade é menor que 18 anos? Nós já sabemos que esse aluno é maior de idade!

Essa repetição de condições desnecessárias pode ser um fator determinante no desempenho de um programa. Daí a importância de saber usar cada um dos desvios no momento correto.

Portanto, vamos resolver nosso problema usando o desvio composto:

```
variáveis
    idade: inteiro
    rm: alfanumérico
início
    Escreva "Por favor, digite seu RM"
    Leia rm
    Escreva "Por favor, digite sua idade"
    Leia idade
    Se idade >=18 então
        Escreva "Sua participação foi autorizada, aluno
de RM", rm
        Escreva "Mais instruções serão enviadas ao seu
e-mail cadastrado na FIAP!"
    Senão
        Escreva "Sua participação não foi autorizada por
causa da sua idade"
    Fim_se
Fim
```

Código-fonte 7 – Algoritmo com if composto em pseudocódigo

Fonte: Elaborado pelo autor (2020)

A tarefa foi cumprida usando o bloco *senão*. No caso do Python, teremos a seguinte sintaxe para esse desvio condicional:

```
if <condição>:
    <ação a ser realizada se a condição for verdadeira>
else:
    <ação a ser realizada se a condição for falsa>
```

Código-fonte 8 – Sintaxe do if composto em Python

Fonte: Elaborado pelo autor (2020)

Se traduzirmos o algoritmo que foi escrito para a linguagem Python, chegaremos ao seguinte código-fonte no script *exemplo_if_composto.py*:


```
rm = input("Insira seu RM")
idade = input("Insira sua idade")

if int(idade) >= 18:
    print("Sua participação foi autorizada, aluno de RM
    {}".format(rm))
    print("Mais informações serão enviadas para seu e-
    mail cadastrado!")
else:
    print("Sua participação não foi autorizada por causa
    da sua idade")
```

Código-fonte 9 – Algoritmo com if simples em Python
Fonte: Elaborado pelo autor (2020)

O desvio condicional composto amplia bastante o leque de um programador! Seja pensando em jogos, aplicativos ou serviços, muitos problemas podem ser resolvidos com essa estrutura.

Como queremos sempre mais, vamos analisar o mesmo problema sob a perspectiva de um desvio condicional encadeado!

1.2.3 If encadeado em Python

Grande notícia: dado o elevado números de alunos com menos de 18 anos interessados em se inscrever no campeonato, a FIAP resolveu abrir vagas para aqueles que possuam autorização escrita dos responsáveis. Ou pelo menos é com esse exemplo que vamos trabalhar.

Se usarmos a lógica do desvio condicional encadeado (aquele em que colocamos um desvio dentro do outro), rapidamente chegamos a um algoritmo próximo do seguinte:

```
variáveis
    idade: inteiro
    rm: alfanumérico
    autorizado: caractere
início
    Escreva "Por favor, digite seu nome"
    Leia rm
    Escreva "Por favor, digite sua idade"
    Leia idade
    Se idade >=18 então
        Escreva "Sua participação foi autorizada, aluno
de RM", rm
        Escreva "Mais instruções serão enviadas ao seu
e-mail cadastrado na FIAP!"
        Senão
            Escreva "Você possui autorização dos
responsáveis para participar? S - SIM ou N - NÃO"
            Leia autorizado
            Se autorizado = "S" então
                Escreva "Sua participação foi autorizada,
aluno de RM", rm
                Escreva "Mais instruções serão enviadas ao
e-mail dos seus responsáveis"
                Senão
                    Escreva "Sua participação não foi autorizada
por causa da sua idade"
            Fim_se
        Fim_se
    Fim
```

Código-fonte 10 – Algoritmo com if encadeado em pseudocódigo
Fonte: Elaborado pelo autor (2020)

Como já conhecemos a sintaxe do if simples e do if composto em Python e já sabemos que o if encadeado é apenas a aplicação de um if dentro de outro, o script *exemplo_if_encadeado.py* pode ser desenvolvido da seguinte maneira:

```
rm = input("Insira seu RM")
idade = input("Insira sua idade")

if int(idade) >= 18:
    print("Sua participação foi autorizada, aluno de RM
{}".format(rm))
    print("Mais informações serão enviadas para seu e-
mail cadastrado!")
else:
    autorizado = input("Você possui autorização dos
responsáveis? S-SIM ou N-NÃO")
    if autorizado == 'S':
```

```
print("Sua participação foi autorizada, aluno de
RM {}".format(rm))
print("Mais informações serão enviadas para o e-
mail dos responsáveis!")
else:
    print("Sua participação não foi autorizada por
causa da sua idade")
```

Código-fonte 11 – Algoritmo com if encadeado em Python
Fonte: Elaborado pelo autor (2020)

Conhecendo a estrutura dos três tipos de desvio condicional, você será capaz de criar inúmeros programas diferentes. Esses tipos estão presentes em todas as linguagens que você vai estudar ao longo da sua formação.

Algumas linguagens de programação, porém, possuem outras formas condicionais, com suas vantagens e otimizações para aquela linguagem específica. No caso do Python, vamos conhecer um bloco muito interessante chamado **elif**!

1.2.4 O Python e o poder do elif!

Uma situação extremamente comum na vida de um programador é precisar testar uma série de condições em sequência.

Vamos tomar como exemplo uma operadora de celular que concede bônus em consumo da franquia de internet dependendo da pontuação dos clientes: clientes que fizerem mais de 1000 pontos recebem 3 GB adicionais em sua franquia, clientes que fizerem mais de 500 pontos recebem 1,5 GB adicionais em sua franquia, e clientes que fizerem mais de 200 pontos recebem 500 MB adicionais em sua franquia. Os demais não recebem nada.

Poderíamos resolver esse problema com um if encadeado e chegaríamos próximo à seguinte solução em Python:

```
pontuacao = input("Insira a pontuação do cliente: ")
pontuacao = int(pontuacao)
if pontuacao >= 1000:
    print("O cliente tem direito a receber mais 3 GB na
sua franquia de internet!")
else:
    if pontuacao >= 500:
        print("O cliente tem direito a receber mais 1,5
GB na sua franquia de internet!")
    else:
```

```
if pontuacao >=200:
    print("O cliente tem direito a receber mais
500 MB na sua franquia de internet!")
else:
    print("O cliente não receberá bônus.")
```

Código-fonte 12 – Algoritmo da operadora com if encadeado em Python
Fonte: Elaborado pelo autor (2020)

A solução está correta e funciona muito bem, mas podemos escrevê-la de outra maneira utilizando a estrutura *elif* que o Python possui.

O *elif* serve como substituto do bloco *else* quando queremos realizar uma nova verificação de condição dentro do *senão*. Em nosso caso, se você observar bem o código que foi escrito, o primeiro e o segundo *else* são seguidos imediatamente por uma verificação usando o *if*.

Com o uso do *elif*, teríamos o seguinte:

```
pontuacao = input("Insira a pontuação do cliente: ")
pontuacao = int(pontuacao)
if pontuacao >= 1000:
    print("O cliente tem direito a receber mais 3 GB na
sua franquia de internet!")
elif pontuacao >=500:
    print("O cliente tem direito a receber mais 1,5 GB
na sua franquia de internet!")
elif pontuacao >=200:
    print("O cliente tem direito a receber mais 500 MB
na sua franquia de internet!")
else:
    print("O cliente não receberá bônus.")
```

Código-fonte 13 – Algoritmo da operadora com o uso do *elif* em Python
Fonte: Elaborado pelo autor (2020)

Como utilizamos o *elif*, o programa passou a ter menos linhas e a ficar mais legível, uma vez que não foi necessário criar novas indentações para cada if encadeado.

Depois de conhecer esse “caminhão” de estruturas diversas, vamos resolver alguns exercícios que utilizam os diferentes *ifs*?

DICA: Enquanto estiver na fase de estudo, procure resolver um mesmo problema utilizando os diversos desvios condicionais para entender bem as diferenças entre eles!

EMSE

2 OPERANDO A LÓGICA!

Assim como utilizamos os operadores matemáticos para realizar operações aritméticas, existem operadores lógicos que nos auxiliam a realizar operações com testes lógicos.

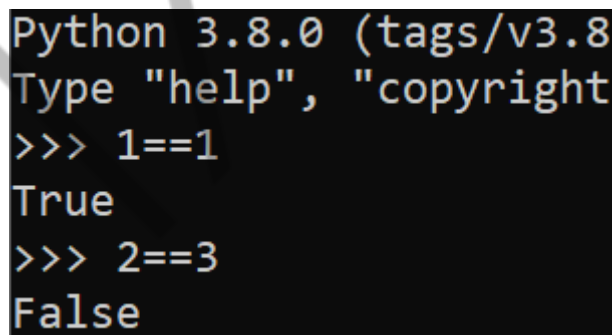
Vamos conhecer os mais importantes para todo programador: and e or.

2.1 Casar ou comprar uma bicicleta?

O operador lógico **or** tem uma tarefa muito importante: ele é a conexão entre dois testes lógicos e retornará verdadeiro caso qualquer um dos testes seja verdadeiro.

Para entender como ele funciona, abra o terminal do seu sistema operacional e execute o interpretador da linguagem Python (é só escrever Python, lembra?).

Vamos ver o resultado da execução das linhas “1==1” e depois “2==3”:

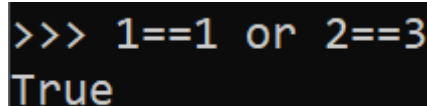


```
Python 3.8.0 (tags/v3.8
Type "help", "copyright
>>> 1==1
True
>>> 2==3
False
```

Figura 6 – Realização de dois testes lógicos separados
Fonte: Elaborado pelo autor (2020)

Fomos informados de que o teste 1==1 tem resultado True (verdadeiro), enquanto o teste 2==3 tem resultado False (falso).

Agora note o que acontece se escrevermos “1==1 or 2==3”:



```
>>> 1==1 or 2==3
True
```

Figura 7 – Realização de testes lógicos utilizando o or
Fonte: Elaborado pelo autor (2020)

Obtivemos um resultado verdadeiro! Isso ocorre porque o operador lógico **or** retorna resultado verdadeiro se pelo menos um dos dois testes que ele uniu for verdadeiro.

A	B	A or B
False	False	False
False	True	True
True	False	True
True	True	True

Tabela 1 – Tabela verdade operador or (ou)
Fonte: Elaborado pelo autor (2022)

2.1.1 Tudo ou nada!

O operador lógico **and** é um pouco mais exigente que o **or**: ele é a conexão entre dois testes lógicos e retornará verdadeiro apenas se ambos os testes retornarem verdadeiro.

A	B	A and B
False	False	False
False	True	False
True	False	False
True	True	True

Tabela 2 – Tabela verdade operador and (e)
Fonte: Elaborado pelo autor (2022)

Dessa maneira, se tentarmos realizar a união dos testes que fizemos anteriormente com o operador **and**, teremos o seguinte resultado:

```
>>> 1==1 and 2==3  
False
```

Figura 8 – Realização de testes lógicos utilizando o and
Fonte: Elaborado pelo autor (2020)

Isso ocorreu porque o primeiro teste ($1==1$) era verdadeiro, mas o segundo teste era falso.

O Python tem uma característica especial que **não está presente** na maior parte das outras linguagens de programação. Ele permite a avaliação de condições em cadeia.

Para comparar se a variável *a* é maior que a variável *b* e, ao mesmo tempo, a variável *b* é maior que a variável *c*, podemos escrever de duas formas: "*a>b and b>c*" ou "*a>b>c*".

É importante prestar atenção à linguagem com que se deseja trabalhar, para não tentar utilizar um recurso inexistente.

2.2 Hora da prática

Problemas envolvendo desvios condicionais não faltam! Vamos encarar alguns deles e entender como podemos utilizar o Python para facilitar nossa vida!

2.2.1 Ajudando o doutor!

O doutor Henry Jones Junior estabeleceu uma regra com seus alunos da disciplina de Arqueologia: todos os que obtiverem nota maior do que 8,5 na sua prova semestral serão convidados para uma visita de campo na América do Sul.

Nosso programa deve solicitar o e-mail e a nota do aluno, exibindo a mensagem "ENVIANDO CONVITE" caso a nota do aluno satisfaça a condição proposta.

Utilizando apenas um `if` simples, podemos resolver esse problema rapidamente! Basta solicitarmos a digitação dos dados, converter a nota para números reais (`float`) e verificar se ela atende à condição do professor Jones.

```
#solicitando os dados do aluno
email_aluno = input("Informe o e-mail do aluno")
nota_semestral = input("Informe a nota semestral do
aluno: ")
#convertendo a nota para o formato float
nota_semestral = float(nota_semestral)
#realizando o teste lógico
if nota_semestral > 8.5:
```



```
print("ENVIANDO E-MAIL PARA {}".format(email_aluno))
```

Código-fonte 14 – Solução do exercício 1 em Python

Fonte: Elaborado pelo autor (2020)

Vale lembrar que, por usarmos um número real, tanto na escrita do script como na hora de testar o programa, devemos utilizar um “.” (ponto) para separar as casas decimais, e não uma “,” (vírgula) como estamos acostumados.

2.2.2 Com que roupa eu vou?

A loja virtual FIAP Wear, que vende roupas personalizadas da instituição, disponibilizou no mês do seu aniversário o cupom NIVER10, que concede 10% de desconto no valor total de uma compra feita no site.

Caso o cliente digite o cupom corretamente, deverá ser informado do valor final da compra já com o desconto aplicado. Caso digite o cupom de maneira incorreta, deverá ser informado do valor da compra sem o desconto.

Esse problema claramente pede o uso de um if composto! Afinal de contas, temos uma única condição (o cupom digitado ser igual a “NIVER10”) e duas ações, sendo uma para cada resultado da condição.

Sabendo disso, podemos elaborar um script em Python desta forma:

```
#solicitando os dados do cliente
valor_compra = input("Informe o valor da compra realizada
")
cupom = input("Digite o cupom de desconto ")

#realizando o teste lógico
if cupom == "NIVER10":
    #cálculo de 10% de desconto
    valor_final = float(valor_compra) * 0.9
else:
    valor_final = float(valor_compra)
    print("CUPOM INVÁLIDO")
#exibindo o valor final da compra
print("O valor final da compra é {}".format(valor_final))
```

Código-fonte 15 – Solução do exercício 2 em Python

Fonte: Elaborado pelo autor (2020)

Ao executar o script, você pode ficar muito feliz ou muito triste... Tudo vai depender da forma como digitou o cupom. Veja o que acontece quando digitamos o cupom de qualquer forma que não seja com todas as letras maiúsculas:

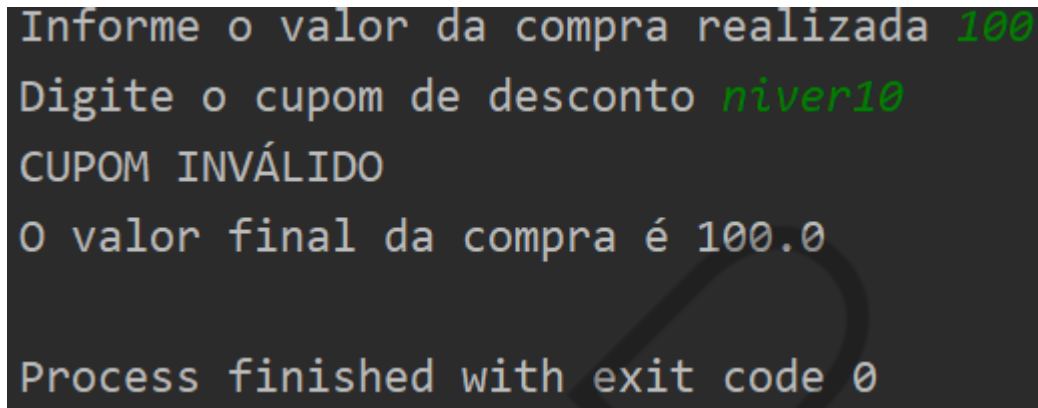
A terminal window with a dark background and light-colored text. The text shows the program's execution flow: it prompts for a purchase value (100), then a discount coupon (niver10). It then displays 'CUPOM INVÁLIDO' (Invalid Coupon) and 'O valor final da compra é 100.0' (The final purchase value is 100.0). The last line shows 'Process finished with exit code 0'.

Figura 9 – Execução do programa com erro no cupom
Fonte: Elaborado pelo autor (2020)

Isso ocorre porque o Python avalia a string “NIVER10” como diferente de “niver10”. Uma solução muito interessante para esse cenário, caso seja importante considerar ambos os casos, é converter tudo o que o usuário escreveu para letras maiúsculas.

Veja o que ocorre neste pequeno teste, feito em outro script:

```
minusculas = "mestre yoda"
#usamos a função upper para converter a string em letras
maiúsculas
maiusculas = minusculas.upper()
print(minusculas)
print(maiusculas)
```

Código-fonte 16 – Utilização da função upper()
Fonte: Elaborado pelo autor (2020)

O uso da função *upper()*, presente no Python, permite que uma determinada string seja convertida para caracteres maiúsculos.

Nosso programa original pode ser melhorado, portanto, da seguinte forma:

```
#solicitando os dados do cliente
valor_compra = input("Informe o valor da compra realizada
")
cupom = input("Digite o cupom de desconto ")

#realizando o teste lógico com o cupom em maiúsculas
```

```
if cupom.upper() == "NIVER10":
    #cálculo de 10% de desconto
    valor_final = float(valor_compra) * 0.9
else:
    valor_final = float(valor_compra)
    print("CUPOM INVÁLIDO")
#exibindo o valor final da compra
print("O valor final da compra é {}".format(valor_final))
```

Código-fonte 17 – Solução do exercício 2 em Python com tratamento de letras maiúsculas

Fonte: Elaborado pelo autor (2020)

2.2.3 Essa eu sabia com maçãs

Transporte-se no tempo e volte para a época de escola! Lembra do dia em que aprendeu a encontrar o valor de X nas equações de 2º grau? Aquelas que tinham uma carinha parecida com $Ax^2 + Bx + C = 0$?

Aquela fórmula que aprendemos como fórmula de Bhaskara (mas que não foi ele quem criou) deixou saudades (ou pesadelos terríveis).

Vamos dar um presente ao seu “eu” do passado e criar um programa no qual o usuário só tenha que escrever os valores de A, B e C e nosso programa vai se encarregar de fazer os cálculos.

A primeira etapa que aprendemos na escola é calcular o **delta** por meio da fórmula: $B^2 - 4 \cdot A \cdot C$. Depois, caso o delta seja positivo, existem dois valores para x. Caso seja zero, existe apenas um valor. E caso seja negativo, informamos que não há valor real para x.

Logo, teremos para a solicitação dos valores e o cálculo do delta:

```
#solicitando os valores de A, B e C
a = float(input("Informe o valor de A"))
b = float(input("Informe o valor de B"))
c = float(input("Informe o valor de C"))
#cálculo do delta
delta = b * b - 4 * a * c
```

Código-fonte 18 – Início da solução do exercício 3 em Python

Fonte: Elaborado pelo autor (2020)

Para fazer a verificação de cada uma das condições e, posteriormente, realizar os cálculos necessários, podemos utilizar um desvio condicional encadeado ou utilizar o *elif*. Vamos dar uma olhada em como ficaria a estrutura em cada um dos casos?

```
#solicitando os valores de A, B e C
a = float(input("Informe o valor de A"))
b = float(input("Informe o valor de B"))
c = float(input("Informe o valor de C"))
#cálculo do delta
delta = b * b - 4 * a * c
# verificação das condições com if encadeado
if delta > 0.0:
    # cálculo de 2 valores para x
else:
    if delta == 0.0:
        # cálculo de 1 valor para x
    else:
        # exibição da mensagem
```

Código-fonte 19 – Desvios da solução do exercício 3 com if encadeado em Python
Fonte: Elaborado pelo autor (2020)

```
#solicitando os valores de A, B e C
a = float(input("Informe o valor de A"))
b = float(input("Informe o valor de B"))
c = float(input("Informe o valor de C"))
#cálculo do delta
delta = b * b - 4 * a * c
#verificação das condições com elif
if delta>0.0:
    #cálculo de 2 valores para x
elif delta == 0.0:
    #cálculo de 1 valor para x
else:
    #exibição da mensagem
```

Código-fonte 20 – Desvios da solução do exercício 3 com elif em Python
Fonte: Elaborado pelo autor (2020)

Agora, dentro dos desvios, incluiremos os cálculos de x. A fórmula que aprendemos na escola é $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, na qual o que está dentro da raiz é o delta que já calculamos anteriormente.

E por falar em raiz, cada linguagem de programação possui sua função própria para realizar essa operação. No caso do Python, precisamos importar a classe **math** e depois usar a função **math.sqrt**.

Para importar a classe **math**, escrevemos na primeira linha do nosso script: `import math`.

Para calcular a raiz de algum valor depois de importarmos a classe math, devemos escrever `math.sqrt(valor)`.

Dessa forma, nossa solução completa com o `elif` fica assim:

```
import math
#solicitando os valores de A, B e C
a = float(input("Informe o valor de A"))
b = float(input("Informe o valor de B"))
c = float(input("Informe o valor de C"))
#cálculo do delta
delta = b * b - 4 * a * c
#verificação das condições com elif
if delta > 0.0:
    #cálculo de 2 valores para x
    x1 = (-b + math.sqrt(delta)) / (2 * a)
    x2 = (-b - math.sqrt(delta)) / (2 * a)
    print("Para a equação {}x² + {}x + {} = 0, obtivemos
os seguintes valores: x1 = {} e x2 = {}".format(a,b,c,x1,x2))
    elif delta == 0:
        #cálculo de 1 valor para x
        x = (-b + math.sqrt(delta)) / (2 * a)
        print("Para a equação {}x² + {}x + {} = 0, obtivemos
o seguinte valor: x = {}".format(a,b,c,x))

    else:
        #exibição da mensagem
        print("Para a equação {}x² + {}x + {} = 0, não existem
valores reais para x".format(a, b, c))
```

Código-fonte 21 – Solução do exercício 3 com `elif` em Python
Fonte: Elaborado pelo autor (2020)

Ou assim, se usarmos o desvio condicional encadeado:

```
import math
#solicitando os valores de A, B e C
a = float(input("Informe o valor de A"))
b = float(input("Informe o valor de B"))
c = float(input("Informe o valor de C"))
#cálculo do delta
delta = b * b - 4 * a * c
# verificação das condições com if encadeado
if delta > 0.0:
    # cálculo de 2 valores para x
    x1 = (-b + math.sqrt(delta)) / (2 * a)
    x2 = (-b - math.sqrt(delta)) / (2 * a)
    print("Para a equação {}x² + {}x + {} = 0, obtivemos
os seguintes valores: x1 = {} e x2 = {}".format(a, b, c, x1,
x2))
    else:
```

```
if delta == 0:
    # cálculo de 1 valor para x
    x = (-b + math.sqrt(delta)) / (2 * a)
    print("Para a equação {}x² + {}x + {} = 0,
obtivemos o seguinte valor: x = {}".format(a, b, c, x))
else:
    # exibição da mensagem
    print("Para a equação {}x² + {}x + {} = 0, não
existem valores reais para x".format(a, b, c))
```

Código-fonte 22 – Solução do exercício 3 com desvio condicional encadeado em Python

Fonte: Elaborado pelo autor (2020)

HORA DE TREINAR

1. Verificar se os batimentos cardíacos por minuto se encontram na faixa adequada. Para isso, você deve solicitar ao usuário que informe o seu número de BATIMENTOS POR MINUTO (BPM) e a IDADE. A partir disso, o script deve verificar e exibir uma mensagem informando se os batimentos do usuário encontram-se DENTRO da faixa adequada, ACIMA da faixa adequada ou ABAIXO da faixa adequada, de acordo com o site Tua Saúde (<https://www.tuasaude.com/frequencia-cardiaca/#:~:text=At%C3%A9%202%20anos%20de%20idade,idosos%3A%2050%20a%2060%20bpm>):

IDADE	BPM
Até 2 anos	120 a 140
De 8 anos até 17 anos	80 a 100
Adulto sedentário	70 a 80
Idosos	50 a 60

2. Viajar é bom demais! Uma agência de viagens está propondo uma estratégia para alavancar as vendas após os impactos da pandemia do coronavírus.

A empresa ofertará descontos progressivos na compra de pacotes, dependendo do número de viajantes que estão no mesmo grupo e moram na mesma residência.

Para ajudar a tornar esse projeto real, você deve criar um algoritmo que receba o VALOR BRUTO do pacote, a CATEGORIA DOS ASSENTOS no voo e a QUANTIDADE DE VIAJANTES que moram em uma mesma casa e calcule os descontos de acordo com a tabela abaixo:

Categoria	DESCONTOS	
Econômica	2 viajantes	3%
	3 viajantes	4%
	4 viajantes ou mais	5%
Executiva	2 viajantes	5%
	3 viajantes	7%
	4 viajantes ou mais	8%
Primeira classe	2 viajantes	10%
	3 viajantes	15%
	4 viajantes ou mais	20%

O programa deverá exibir o valor BRUTO DA VIAGEM (o mesmo que foi digitado), o VALOR DO DESCONTO, o VALOR LÍQUIDO DA VIAGEM (valor bruto menos os descontos) e o VALOR MÉDIO POR VIAJANTE.

3. Hora de decidir! Os colaboradores da sua equipe foram sorteados para ganhar um console de última geração, cada um, em razão do bom desempenho que tiveram nos últimos projetos. Por uma questão de logística, porém, a empresa pede que todos os cinco membros da equipe recebam o mesmo aparelho.

Crie um algoritmo em que o usuário possa digitar o voto de cada um dos 5 membros da equipe e, ao final, exiba qual foi o console escolhido e com quantos votos.

As opções são: PLAYSTATION, XBOX e NINTENDO.

CORREÇÕES DOS EXERCÍCIOS TREINO

REFERÊNCIAS

PIVA JÚNIOR, Dilermando et al. **Algoritmos e programação de computadores**. São Paulo: Elsevier, 2012.

PUGA, Sandra; RISSETTI, Gerson. **Lógica de Programação e Estrutura de Dados**. São Paulo: Pearson Prentice Hall, 2009.

RAMALHO, Luciano. **Python Fluente: Programação Clara, Concisa e Eficaz**. São Paulo: Novatec, 2015.

EMANIP