

Introdução à Análise de Dados

Cap.2 - Fundamentos de Python



Prof. MSc. Renzo P. Mesquita

Objetivos

- Conhecer detalhes de funcionamento da linguagem de programação Python;
- Compreender a sintaxe e recursos primordiais dessa linguagem de programação;
- Entender quais as facilidades oferecidas pela IDE de desenvolvimento PyCharm;



Capítulo 2

Fundamentos de Python

- 2.1. Por que usar Python?*
- 2.2. Instalando o Python;*
- 2.3. Começando com Python;*
- 2.4. A IDE de desenvolvimento PyCharm;*
- 2.5. Tipos de Dados;*
- 2.6. Operadores Aritméticos;*
- 2.7. Módulos e Pacotes em Python;*
- 2.8. Manipulando cadeias de Caracteres;*
- 2.9. Estruturas de Decisão;*
- 2.10. Laços de Repetição;*



2.1. Por que usar Python?

Com tantas linguagens de programação disponíveis no mercado, por que usar Python?



- É uma **linguagem de propósito geral**, ou seja, você pode usar Python para criar qualquer tipo de programa;
- É considerada uma das **linguagens mais fáceis e intuitivas** já criadas (comandos muito simples);
- É uma linguagem **multiplataforma** (um mesmo programa roda em diferentes tipos de sistemas operacionais e dispositivos);
- Vem com um **conjunto de bibliotecas muito completa**;
- É um **software livre** (não é necessário pagar para usá-lo ou distribuí-lo);
- É **MUITO organizada!** Ela força o programador a deixar o código organizado;
- É **ORIENTADA A OBJETOS**, ou seja, faz uso do paradigma de programação mais popular do mercado;

2.2. Instalando o Python

Atualmente existem duas versões disponíveis do Interpretador Python. Qual escolher?



- Independente da versão, uma notícia boa: **ambos são Python**. Aprender a programar em Python em uma versão pouco vai interferir no jeito de programar na outra;
- A versão 3 veio para corrigir erros da versão 2, por isso, para quem está começando agora a versão 3 é a mais indicada;
- **Um código em Python 2 vai rodar perfeitamente no Python 3? Pode ser que NÃO.** Por conta das pequenas modificações que aconteceram, pode acontecer erros;
- O Python 2 apesar de ainda ser mais popular (por existir a mais tempo), está perdendo posição cada vez mais para o Python 3. A versão 2 em breve será descontinuada.

2.2. Instalando o Python

O sistema operacional que utilizaremos nessa disciplina será o Windows. Como vimos anteriormente, dos sistemas operacionais mais populares, ele é um dos poucos que não vem com o Python instalado, por isso, precisaremos instalá-lo.

1. Primeiramente, verifique se já existe alguma versão do Python instalado no seu PC. Para isso, abra o terminal (Prompt de Comando) e digite: *python --version*
2. Caso nenhuma versão do Python esteja instalada, aparecerá uma seguinte mensagem do tipo: '*python' não é reconhecido como um comando...*
3. Se a versão Python 3.x já tiver instalada em sua máquina, já está tudo certo. Senão, você poderá fazer o download do interpretador Python no seguinte website: <https://www.python.org/getit/>



2.2. Instalando o Python

4. Uma vez que o Python 3 esteja devidamente instalado em seu computador, faça o seguinte teste para verificar se está tudo certinho:

- No prompt, digite: *python*
- Em seguida, aparecerá a versão do seu python e três setinhas >>> indicando que você já pode digitar um comando;
- Digite primeiramente: *print('Olá Python!')*
- Após ele mostrar a mensagem na tela, digite: *7 + 7*
- Caso ele mostre o resultado corretamente, existe grandes chances do seu Python estar instalado corretamente :)

IMPORTANTE: Caso o Python ainda não esteja devidamente instalado no seu computador, faça download do instalador no website informado e escolha a versão correta para seu computador (versão 32 bits ou 64 bits).

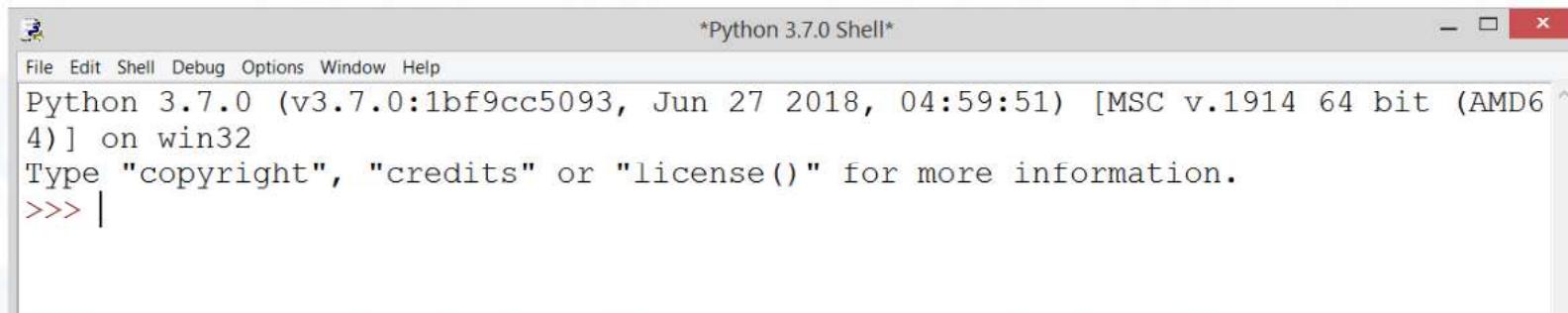
Ao executar o instalador, não deixe de marcar a opção "Add Python 3.x to PATH". Isso permite que você use o Python diretamente no prompt de comando do Windows. ;)



2.3. Começando com Python

Modo Interativo em Python

- O **IDLE (Integrated Development and Learning Environment)** é um programa escrito em Python para ser utilizado como um ambiente de execução simples para execução de códigos Python;
- Ele vem junto com as instalações Python e serve para proporcionar uma **maneira rápida de executar códigos Python** sem precisar abrir o prompt de comando;
- Para abri-lo, execute o **IDLE (Python 3.x)** dentro da pasta de instalação do seu Python;



- A fim de verificar o funcionamento do IDLE, digite os comandos que utilizamos anteriormente;
- Conforme formos trabalhando com Python, vamos aprendendo mais sobre essa ferramenta :)

2.3. Começando com Python

Todo comando em Python 3 é uma Função

- **Função** nada mais é que uma espécie de comando que é escrito dentro de um **bloco que começa com (e termina com)**;
- **Para escrevermos algo na tela, usamos a função print**, que deve ter seu conteúdo colocado dentro do (). Caso o conteúdo seja uma mensagem, então ela também deve estar circundada com aspas simples ou duplas;

Ex: print ("Olá Python!")

- Caso o conteúdo seja um número ou uma operação matemática, não é necessário colocar as aspas, mas sim o número ou a operação em si dentro do parêntesis;

Ex: print (7 + 7)

- Caso queira colocar **mensagens juntamente com números ou operações de forma conjunta, basta utilizar da , (vírgula)**;

Ex: print('O resultado de 7 + 7 é', 7+7)

Veja que para juntar coisas diferentes usamos a , (vírgula).

2.3. Começando com Python

Variáveis permitem armazenar valores

- Variáveis são espaços de memória capazes de armazenar informações de diferentes tipos. Para criar uma variável em Python basta escrever o nome dela, seguido do operador = (recebe) e do valor que deseja armazenar;

Exemplo: `nome = 'Vincent'`

`idade = 30`

`peso = 83.5`

`print(nome,idade,peso)`

- O Python é uma linguagem de Tipagem Dinâmica. Isso significa que o próprio interpretador do Python infere o tipo do dado de uma variável, sem a necessidade do usuário deixar isso explícito (linguagens como C, C++, Java, entre outras, são de Tipagem Estática).

2.3. Começando com Python

E se quisermos perguntar algo ao usuário?

- Para solicitarmos alguma informação ao usuário podemos fazer que a variável receba o resultado de uma função input, responsável por capturar algo digitado após uma mensagem;

Exemplo: `nome = input('Qual o seu nome?')`
`idade = input('Qual sua idade?')`
`peso = input('Qual o seu peso?')`

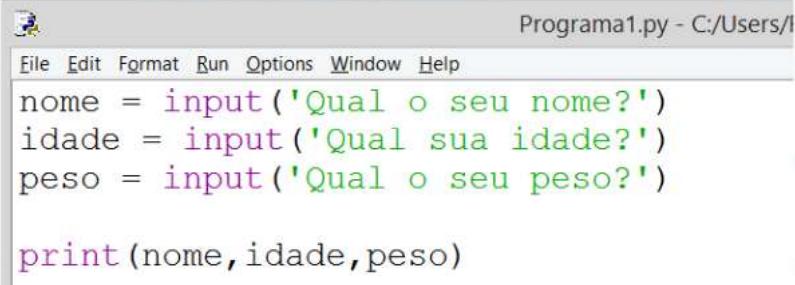


- Vale ressaltar que o **conteúdo inserido** pelo usuário será guardado naturalmente no **formato de texto**. Em breve veremos como transformá-lo rapidamente em um tipo específico de dados para que, por exemplo, possamos fazer operações matemáticas com ele.

2.3. Começando com Python

Achei muito prático o modo interativo do Python por meio do IDLE. Mas se eu fechá-lo eu perco tudo?

- Da forma que utilizamos, sim. O modo interativo é utilizado para realização de testes e execução de comandos rápidos;
- Para armazenar todo o nosso código para que possamos executá-lo novamente mais tarde, precisamos criar um Script em Python;
- Para que isso seja possível, vá em File -> New File para criar um novo arquivo em branco;
- Neste novo arquivo agora é possível colocar todos os comandos que criarmos, um embaixo do outro e de forma sequencial para que possamos salvá-lo e executá-lo mais tarde;



```
Programa1.py - C:/Users/I  
File Edit Format Run Options Window Help  
nome = input('Qual o seu nome?')  
idade = input('Qual sua idade?')  
peso = input('Qual o seu peso?')  
  
print(nome, idade, peso)
```

Depois de dar um nome e salvar o seu arquivo, vá em Run -> Run Module ou só aperte F5.

2.4. A IDE de desenvolvimento PyCharm

Uma IDE (Integrated Development Environment) de desenvolvimento nada mais é que uma ferramenta que facilita a vida do desenvolvedor na hora de utilizar uma linguagem de programação.

- Nesta disciplina, utilizaremos o PyCharm, IDE de desenvolvimento da empresa JetBrains, para desenvolver e continuar nossos estudos de Python;
- Dentre as várias vantagens oferecidas por essa IDE, pode-se destacar:
 - *Editor intuitivo e de fácil manuseio;*
 - *Função auto-complete de códigos;*
 - *Auto identação;*
 - *Vários atalhos para diferentes funções;*
 - *Possibilidade de adicionar e organizar plugins de forma simplificada;*
 - *Geração de código automático, e muito mais ;).*



2.4. A IDE de desenvolvimento PyCharm

Para criar um novo projeto no PyCharm é muito simples: depois de abri-lo, vá em File -> New Project.

- Vale ressaltar que um projeto pode ter uma série de arquivos Python dentro dele. Futuramente vamos compreender melhor a vantagem de criar vários arquivos dentro de um mesmo projeto;
- Para criar pelo menos um arquivo Python no seu projeto e começar a programar, clique com o botão direito em cima do seu projeto -> New -> Python File;
- O PyCharm oferece muitas facilidades e recursos. Caso queira customizar a IDE pra ficar mais 'a sua cara' (como deixar a letra maior, etc.), vá em File -> Settings;
- Para executar um programa em um arquivo específico, clique com o botão direito sobre seu código -> Run (setinha verde);



2.5. Tipos de Dados

Em todas as linguagens de programação existem os tipos básicos de dados, também chamados de tipos primitivos.

Os tipos primitivos mais básicos que existem são:

- *int*: usado para representar números inteiros (Ex: 7, -5, 0, 999);
- *float*: usado para representar números reais (Ex: 4.5, 0.75, -15.5);
- *bool*: só aceita dois valores (Ex: True e False);
- *str*: usado para representar palavras (Ex: 'Oi', 'Python é legal');

Para saber qual é o tipo de uma determinada variável em Python, digite: *print(type(variavel))*

Mas como mostrar pro Python que uma variável será int, outra float, e assim por diante?

Na hora de inserir o valor em uma variável, basta explicitar o tipo da seguinte forma:

num = int(input('Entre com um número:'))

num = float(input('Entre com um número:'))

e assim por diante..

2.5. Tipos de Dados

Agora que já sabe como explicitar o tipo de uma variável em Python, que tal fazer o Exercício 3 da nossa última aula?

Crie um script que leia dois números e tente mostrar a soma deles.

Na hora de mostrar os valores, sem dúvida você utilizará do seguinte formato: *Ex: print('A soma de',n1,'com',n2,'é: ',soma)*

Porém, existe uma forma mais elegante de apresentar os resultados, neste caso, fazendo uso do recurso `format()`.

print('A soma de {} com {} é: {}'.format(n1,n2,soma))

Observe que, no meio do seu texto, o `{}` representa cada uma das variáveis que foram colocadas dentro do método `format()`, levando em consideração sua ordem da esquerda para a direita.

E aí? animado para mais algumas facilidades com os tipos primitivos? Vamos lá!

2.6. Operadores Aritméticos

Os Operadores Aritméticos, originários da Matemática são de suma importância na programação. É por meio deles que se torna possível a realização dos cálculos.

- No Python, os operadores aritméticos mais populares são:

- | | |
|--|---|
| <ul style="list-style-type: none">• + (<i>soma</i>);• - (<i>subtração</i>);• * (<i>multiplicação</i>);• / (<i>divisão</i>); | <ul style="list-style-type: none">• == (<i>igualdade</i>);• ** (<i>potenciação</i>);• // (<i>divisão inteira</i>);• % (<i>resto da divisão</i>); |
|--|---|

IMPORTANTE:

- Da mesma forma que acontece na matemática, devemos nos atentar com as precedência dos operadores na hora da montagem de expressões;
- O == é diferente do =. O == significa 'igual a' e o = significa 'recebe';
- No Python, é possível "multiplicarmos" uma String, como por exemplo:

```
var = input('Entre com uma palavra')
var2 = var*5;
print(var2)
```



*A palavra inserida
será impressa 5x.*

2.7. Módulos e Pacotes em Python

Os programas em Python inicialmente operam com um conjunto básico de recursos, porém, caso haja necessidade, podemos IMPORTAR novas bibliotecas ou módulos a fim de adicionarmos mais recursos.

Existem duas formas de importar bibliotecas externas no Python, são elas:

- `import novaBiblioteca` -> importa uma biblioteca inteira de recursos;
- `from novaBiblioteca import parteDaNovaBiblioteca` -> importa apenas um recurso específico de uma biblioteca;

Dentre as bibliotecas mais conhecidas, podemos citar a `math`, que oferece ao programador uma série de funções matemáticas já prontas para serem utilizadas, como por exemplo:

- `ceil` -> arredonda um número para cima;
- `floor` -> arredonda um número para baixo;
- `trunc` -> pega apenas a parte inteira de um número;
- `pow` -> potência (funciona igual ao operador `**`);
- `sqrt` -> retorna a raiz quadrada de um número;
- `factorial` -> retorna o fatorial de um número;

```
import math  
num = 2.5  
print(math.trunc(num))
```



Será impressa a parte inteira do número.

2.7. Módulos e Pacotes em Python

Se afogue nas bibliotecas do Python! Elas foram criadas para facilitar a sua vida :)

Apesar de termos visto como importar novas bibliotecas dentro dos nossos programas e também ter dado uma olhadinha na biblioteca math, podemos ver e entender muitas outras no seguinte endereço:

<https://docs.python.org/3/library/index.html>



As bibliotecas acima já vêm instaladas junto ao nosso interpretador Python, e quando queremos colocá-las dentro dos nossos programas, podemos simplesmente utilizar do import novaBiblioteca para isso.

Professor e se eu quiser utilizar de outras bibliotecas (libs) que não vieram instaladas com o meu interpretador?

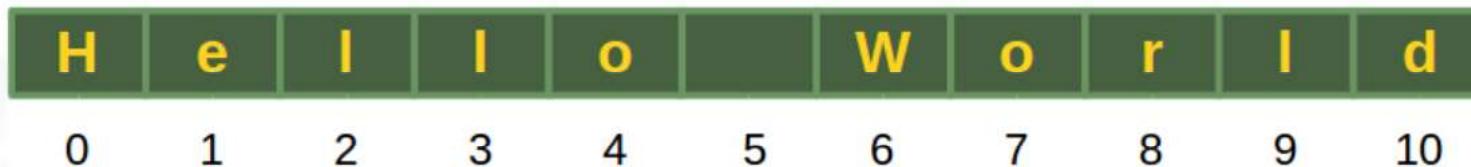
Um ÓTIMO lugar para já pegar bibliotecas prontas para resolver diversos tipos de problemas é o website: <https://pypi.org/>

Visite o website, procure por algo e se impressione!

2.8. Manipulando cadeias de Caracteres

Qualquer conteúdo que seja colocado dentro de '' ou "" é chamado de string (str), que nada mais é que uma cadeia de caracteres.

Ao ser criada uma String, o Python a organiza da seguinte forma dentro da memória, dando um índice para cada um de seus caracteres:



Graças a esse comportamento do Python e de outras linguagens de programação, é possível realizar várias operações de forma simplificada sobre uma cadeia de caracteres, por exemplo:

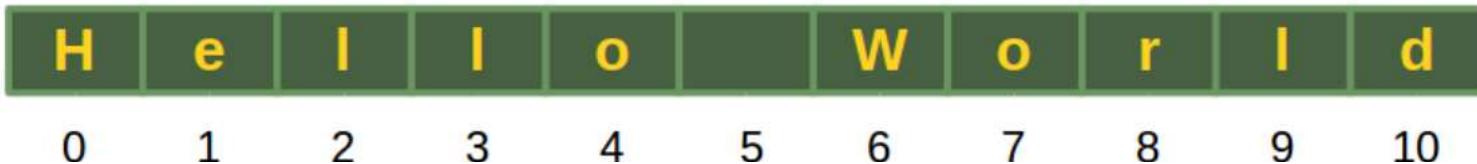
- *var[6] -> captura a letra W da nossa String exemplo;*
- *var[:5] -> captura a palavra Hello;*
- *var[6:11] -> captura a palavra World; (6 inclusive e 11 exclusive);*
- *var[6:] -> também captura a palavra World;*
- *var[0:10:2] -> mostra HloWrd (ou seja, pula de 2 em 2);*
- *var[6::2] -> mostra Wl (dentro da palavra World, pula de 3 em 3);*

```
var = 'Hello World'  
print(var[:5])
```



Hello

2.8. Manipulando cadeias de Caracteres



Outras formas muito legais para manipular Strings:

- *len(var) -> retorna o número de caracteres em uma string;*
- *var.count('o') -> conta o número de o's da nossa palavra;*
- *var.count('l',0,5) -> conta quantos l's tem dentro de Hello;*
- *var.find('lo') -> indica a posição de onde começa lo;*
- *"World" in var -> retorna true se a palavra World estiver dentro de var;*
- *var.replace('World','Python') -> Troca a palavra World po Python;*
- *var.upper() -> troca todas as letras para MAIÚSCULAS;*
- *var.lower() -> troca todas as letras para minúsculoas;*
- *var.split() -> quebra a frase em partes;*
entre muitas outras possibilidades..

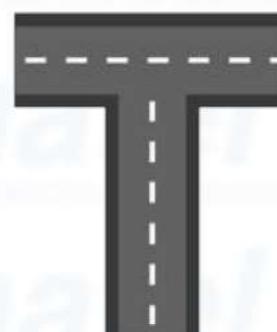
Para outras possiblidade acesse a documentação oficial do Python 3 em:

<https://docs.python.org/3/>

2.9. Estruturas de Decisão

Até o momento, todos os nossos scripts em Python foram criados de forma sequencial. Porém, podemos permitir que o software tome diferentes caminhos em determinados momentos.

*Nossos programas
até o momento*



*Nossos programas
daqui pra frente*

Para que isso seja possível, deveremos utilizar das chamadas **ESTRUTURAS DE DECISÃO**, ou seja, **blocos de comandos que podem mudar o fluxo de execução de um software dependendo do resultado de uma variável ou de uma condição**. A estrutura de decisão mais popular na programação é o:

if-else (se-senão)

2.9. Estruturas de Decisão

No Python, a Identação (organização correta) do código é algo ESSENCIAL para que seus comandos funcionem corretamente.

No bloco if-else isso não seria diferente. A Identação é ESSENCIAL!

Exemplo:

```
1 idade = int(input('Entre com sua idade: '))
2
3 if idade < 18:
4     print('Você é menor de idade!')
5 else:
6     print('Você é maior de idade!')
```

- Observe a identação e sintaxe do if-else. Além de deixar seu código mais organizado, permite que seu programa execute corretamente no Python;
- Todo comando que estiver ao lado esquerdo da tela, ele será executado SEMPRE. Já os comandos dentro do if-else, vão executar dependendo do resultado da condição;

Mas e se for possível tomar mais de uma decisão?

2.9. Estruturas de Decisão

O if não necessariamente precisa do else, mas as vezes, previsamos de vários else's para dar várias opções de execuções aos nossos programas.

Exemplo:

```
1     nota1 = float(input('Entre com sua nota 1: '))
2     nota2 = float(input('Entre com sua nota 2: '))
3
4     notaFinal = (nota1 + nota2)/2
5
6     if notaFinal < 30:
7         print('Você foi reprovado :(!')
8     elif notaFinal >= 60:
9         print('Você foi aprovado :))')
10    else:
11        print('Você pegou prova final!')
```

- Veja o papel do elif, ele permite adicionar mais condições ao seu bloco if-else;

2.10. Laços de Repetição

Até o momento vimos como desviar códigos, mas agora, vamos ver como repetí-los em Python.

Em algumas situações é comum que uma mesma instrução (ou conjunto delas) precise ser executada várias vezes seguidas. Nesses casos, normalmente utilizamos um loop (ou laço de repetição) que permite executar o mesmo bloco de código enquanto uma condição é atendida.

Em Python, podemos fazer isso com as estruturas de repetição for e while.

- A sintaxe da estrutura for é a seguinte:

```
for c in range(valorInicial, valorFinal):  
    meu(s) comando(s)
```

Exemplo: como seria possível mostrar 10x na tela que Python é legal?

```
1     for c in range(0, 10):  
2         print('Python é legal! :)')
```

Vamos ver mais alguns exemplos legais?

2.10. Laços de Repetição

Exemplo 2: como seria mostrar números dentro de um intervalo?

```
1 for c in range(0, 10):  
2     print(c)
```



Mostrará os números de 0 a 9

Exemplo 3: como seria mostrar números dentro de um intervalo mas em ordem decrescente?

```
1 for c in range(10, 0, -1):  
2     print(c)
```



Mostrará os números de 10 a 1

IMPORTANTE:

- Observe que na hora de mostrar os valores, o fim de um laço não conta (exclusive);
- Veja que podemos configurar os números de passos da repetição pelo terceiro parâmetro do range;

Exemplo 4: mostre quais números dentro de um determinado intervalo são pares.

```
1 inicio = int(input('Entre com o inicio:'))  
2 fim = int(input('Entre com o fim:'))  
3  
4 for c in range(inicio, fim):  
5     if c % 2 == 0:  
6         print(c)
```

2.10. Laços de Repetição

Na estrutura for, é obrigatório que coloquemos um limite para o laço. Já no while (enquanto), a repetição acontece até que uma determinada condição seja atendida.

- A sintaxe da estrutura while é a seguinte no Python:

while condição:

meu(s) comando(s)

Exemplo 1: como seria mostrar números na tela até que uma variável assumisse o valor 5?

```
1     var = 1
2      while var < 5:
3         print(var)
4          var += 1
```

Exemplo 2: como seria pedir para o usuário entrar com uma senha correta?

```
1     senha = ''
2     while senha != 'python123':
3         senha = input('Entre com a senha correta: ')
4     print('Bem-vindo ao sistema! :))')
```

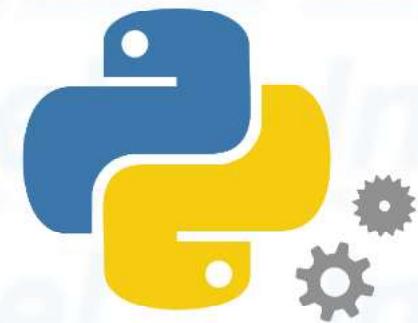
Agora que já sabemos como estes operadores operam no Python, que tal praticarmos um pouco?

2.10. Laços de Repetição

Exercícios Propostos

Baseado nos comandos que vimos até o momento, crie scripts em Python que resolvam os seguintes problemas:

1. Crie um programa que leia seu nome completo e mostre: seu nome com todas as letras maiúsculas, seu nome com todas as letras minúsculas, quantas letras ao todo tem seu nome (contando os espaços) e como seria se trocássemos seu último nome para "do Inatel";
2. Mostre a tabuada de um número que o usuário escolher dentro de um intervalo específico também escolhido por ele;
3. Faça um programa que leia o sexo de uma pessoa e diga se ela é homem (caso seja digitado M) ou mulher (caso seja digitado F). Caso seja digitado algo inválido, continue perguntando até que o usuário entre com um sexo válido.



FIM DO CAPÍTULO 2



Próximo Capítulo
Coleções