

Python I - Fundamentos

Caderno de Exercícios

SUMÁRIO

| | |
|---|--------------------------------------|
| Sumário | Erro! Indicador não definido. |
| Apresentação do curso | 5 |
| História do Python | 6 |
| Versões | 6 |
| Cronologia..... | 6 |
| PEP 8 - Guia de Estilo Para Python | 7 |
| Instalação | 7 |
| Interpretador ou Compilador | 7 |
| Ambiente de desenvolvimento (IDE)..... | 8 |
| CAPÍTULO 1 – Ambiente de desenvolvimento e Comandos básicos | 9 |
| Tópicos do capítulo | 9 |
| Atividade 1 – Preparar o ambiente de trabalho | 9 |
| Atividade 2 – Acessar o Terminal do Python | 11 |
| Atividade 3 – Configurar o VSCode | 12 |
| Atividade 4 – “Olá Mundo!” | 13 |
| Atividade 5 – Usando o Input | 14 |
| CAPÍTULO 2 – Variáveis e Operadores | 15 |
| Tópicos do capítulo | 15 |
| Atividade 1 – Somando Números | 15 |
| Atividade 2 – Área de um retângulo | 16 |
| Atividade 3 – Trabalhando como Textos | 17 |
| CAPÍTULO 3 – Estruturas Condicionais | 20 |
| Tópicos do capítulo | 20 |
| Atividade 1 – Verificar Número Par e Ímpar | 20 |
| Atividade 2 – Conversor de Medidas | 21 |
| Atividade 3 – Folha de Pagamento | 23 |
| Atividade Extra – Calcular IMC | 26 |
| CAPÍTULO 4 – Estruturas de repetição | 27 |
| Tópicos do capítulo | 27 |
| Atividade 1 – Tabuada | 27 |
| Atividade 2 – MultiTabuada | 28 |
| Atividade 3 – Verificar Número Primo | 29 |

| | |
|---|----|
| Atividade Extra – Menu | 31 |
| CAPÍTULO 5 – Listas e Tuplas | 32 |
| Tópicos do capítulo | 32 |
| Atividade 1 – Número por Extenso | 32 |
| Atividade 2 – Jogo: Papel, pedra e Tesoura | 34 |
| Atividade 3 – Lista de Cadastro | 36 |
| Atividade Extra – Contador de Jogadas..... | 37 |
| CAPÍTULO 6 – Funções, Tratamento de Erro e Módulos | 38 |
| Tópicos do capítulo | 38 |
| Atividade 1 – Criação de Funções | 38 |
| Atividade 2 – Tratamento de erros..... | 40 |
| Atividade 3 – Módulo Math..... | 42 |
| Atividade 4 – Função de Datas | 43 |
| Atividade Extra – Acertando o Símbolo | 44 |
| CAPÍTULO 7 – Criando uma classe | 46 |
| Tópicos do capítulo | 46 |
| Atividade 1 – Criando uma classe..... | 46 |
| CAPÍTULO 8 – Trabalhando com arquivos de texto | 49 |
| Tópicos do capítulo | 49 |
| Atividade 1 – Abrindo um arquivo csv | 49 |
| Atividade 2 – Criando um dicionário a partir de um csv | 52 |
| Atividade 3 – Criando um Relatório em csv | 53 |
| CAPÍTULO 9 – Extração de dados na Web | 55 |
| Tópicos do capítulo | 55 |
| Atividade Inicial – Instalando as Bibliotecas..... | 55 |
| Atividade 1 – Extração de dados com BeautifulSoup – Site UOL..... | 56 |
| Atividade 2 – Extração de dados com BeautifulSoup – Site IMDB | 58 |
| Atividade 3 – Extração de dados com BeautifulSoup – Site ChromeDriver | 61 |
| Atividade 4 – Download de arquivos e Unzip..... | 62 |
| Atividade 5 – Controlando o Chrome com o WebDriver | 64 |
| Códigos Completos | 69 |
| Capítulo 1 – Exercícios Resolvidos..... | 69 |
| Capítulo 2 – Exercícios Resolvidos..... | 70 |
| Capítulo 3 – Exercícios Resolvidos..... | 73 |

| | |
|---|-----|
| Capítulo 4 – Exercícios Resolvidos..... | 77 |
| Capítulo 5 – Exercícios Resolvidos..... | 80 |
| Capítulo 6 – Exercícios Resolvidos..... | 85 |
| Capítulo 7 – Exercícios Resolvidos..... | 90 |
| Capítulo 8 – Exercícios Resolvidos..... | 92 |
| Capítulo 9 – Exercícios Resolvidos..... | 96 |
| Anexos | 101 |
| Arquivo produtos.csv | 101 |
| Arquivo categoria.csv | 103 |

Apresentação do curso

Este curso tem como objetivo auxiliar na aprendizagem básica da programação em Python principalmente para os recursos de Web. Hoje a linguagem Python é uma das mais usadas no mundo e é considerada uma programação de alto nível, ou seja, possui uma sintaxe simplificada e de fácil compreensão.

O Python é uma linguagem interpretada e fracamente tipada podendo ser utilizada tanto para Web, Mobile e Desktop sendo multiplataforma. A utilização do Python é bem simples podendo ser executado de um computador através de um terminal de comando.

Como a linguagem é muito versátil ela pode ser executada como scripts para aplicações Web através de um servidor e expandindo funcionalidades de outros programas através de plugins.

O Python possui uma comunidade de desenvolvedores muito grande e atualmente tornando-se umas das mais populares linguagens de programação.

História do Python

O Python foi criado em 1989 por Guido Van Rossum. Ele trabalhava no Centrum Voor Wiskunde (CWI) no início dos anos 1980, e seu trabalho era implementar a linguagem de programação conhecida como ABC.

No final dos anos 1980, ele precisava de uma linguagem de script que tivesse sintaxe semelhante ao ABC e que tivesse acesso às chamadas de sistema Amoeba. Depois de procurar e não encontrar nenhuma linguagem que fizesse isto, Rossum decidiu criar uma linguagem de script simples. Em 1991 ele lançou a versão inicial desta linguagem de programação e que posteriormente foi chamada de “Python”.

O Python é uma linguagem de programação com suporte a orientação de objetos, uma linguagem intermediária, que será interpretada por uma máquina virtual, usa tipagem dinamicamente forte e multiplataforma funcionando em Windows, Linux, Unix e Mac.

Versões

O Python Software Foundation (PSF) é uma organização independente e sem fins lucrativos que foi criada para gerenciar os direitos autorais a partir da versão 2.1 do Python. A PSF tem como missão divulgar a utilização da tecnologia de código aberto relacionada à linguagem de programação Python. Para saber mais sobre a PSF acesse o site <https://www.python.org/psf/>

Cronologia

- Python 2.0.1 – 22 de junho de 2001
- Python 2.7.18 – 20 de abril de 2020
- Python 3.0.1 - 13 de fevereiro de 2009
- Python 3.9.7 – 30 de agosto de 2021

As versões 3 não é compatível com a versão 2 e por isto elas foram evoluindo em conjunto até que em 2020 a última atualização de segurança da versão 2.7.18 foi lançada e posteriormente descontinuada.

PEP 8 - Guia de Estilo Para Python

A PEP 8 possui vários detalhes interessantes sobre como programar em Python e é muito importante que o programa a leia pelo menos uma vez. Certos pontos são considerados indispensáveis por quase toda comunidade.

A grande maioria dos programadores Python acaba adotando grande parte da PEP-8 no seu dia a dia. Por isso, a sua leitura é fortemente recomendada.

Versão original em inglês da PEP 8

<https://www.python.org/dev/peps/pep-0008/>

Versão em português da PEP 8

<https://wiki.python.org.br/GuiaDeEstilo>

Instalação

Para utilizar o Python é necessário ter instalado um interpretador / compilador e utilizar um ambiente de desenvolvimento (IDE), existem várias opções de interpretadores e ambientes de desenvolvimento.

Interpretador ou Compilador

CPython é a implementação principal da linguagem de programação Python, escrita em Linguagem C. CPython é um interpretador de Bytecode.

O PyPy é um compilador Just in Time (JIT), ele usa uma técnica conhecida como meta-tracing, que é responsável por transformar um interpretador em um compilador JIT.

Jython é uma implementação da linguagem Python que gera bytecode para máquinas Java (JVM - Java Virtual Machine). Com ela é possível fazer o desenvolvimento de aplicações híbridas que unem código em Java e Python.

IronPython é uma implementação da linguagem de programação Python escrita em C#, para plataforma.NET e Mono, criada por Jim Hugunin.

Ambiente de desenvolvimento (IDE)

IDLE Python IDE – Este é o ambiente de desenvolvimento padrão do Python é bem simples de ser utilizado Python.

VSCode – Ambiente desenvolvimento pela Microsoft que apresenta diversos recursos avançados e é gratuito, no desenvolvimento do curso as atividades foram desenvolvidas nesta IDE.

Spyder – É uma poderosa IDE gratuita com diversos recursos avançados.

PyCharm – Uma das mais conhecidas IDEs, possui versão gratuita e paga para uso comercial é altamente utilizada para desenvolvimento WEB.

CAPÍTULO 1 – Ambiente de desenvolvimento e Comandos básicos

Neste capítulo você vai instalar os softwares para programar em Python e conferir se a instalação foi feita corretamente. Será necessário instalar o VSCode para escrever os códigos. E aprenderá os dois comandos mais básicos da linguagem, que é o **print** e **input**, e aprenderá a colocar comentários no código.

Objetivos:

- Instalar e preparar o ambiente de desenvolvimento.
- Criar comentários com #.
- Imprimir na tela com o comando **print**.
- Pedir informações para o usuário com o comando **input**.

Tópicos do capítulo

- Instalação do Python e da IDE.
- Comentários.
- Comandos: print e input.

Atividade 1 – Preparar o ambiente de trabalho

Objetivo: Preparar o ambiente de trabalho, instalar o interpretador do Python (CPython) e a IDE do VSCode em um ambiente Windows.

Instalando o Interpretador Python.

1. Primeiramente você irá instalar o interpretador do Python, abra o site:

<https://www.python.org/>



Existem diversas versões do interpretador Python, você vai usar o interpretador oficial da Python Software Foundation, outros interpretados podem ter um funcionamento um pouco diferente deste, mas os comandos do Python são os mesmos. Instalação de bibliotecas pode ser diferentes de um para o outro.

2. Clique no botão Downloads e escolha a versão mais atual do Python.
3. Após o download inicie a instalação.
4. Marque a opção **Add Python 3.X to Path** e clique em **Install Now**.

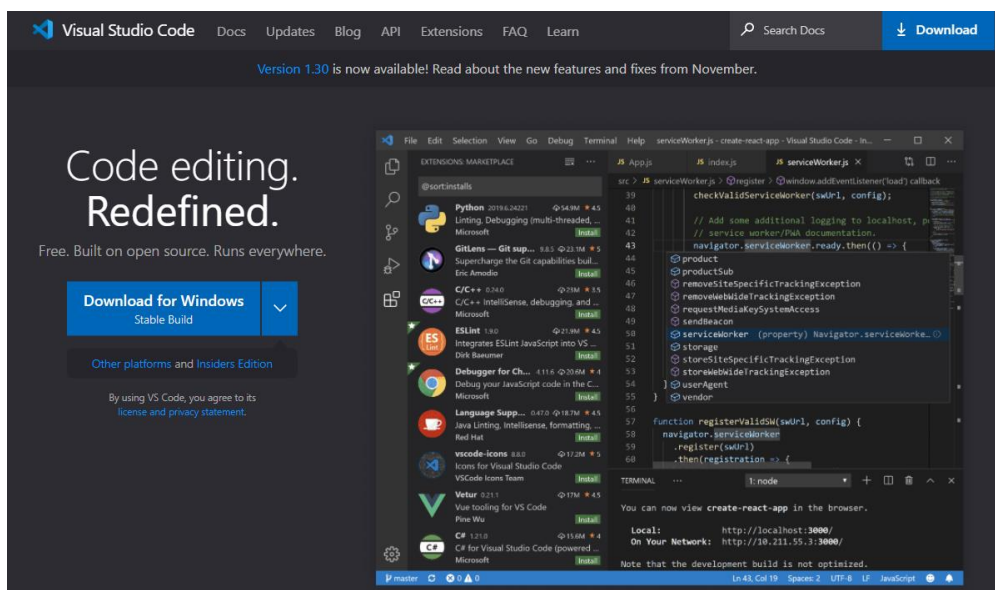


5. Pronto o interpretador já está instalado.

Instalando a IDE

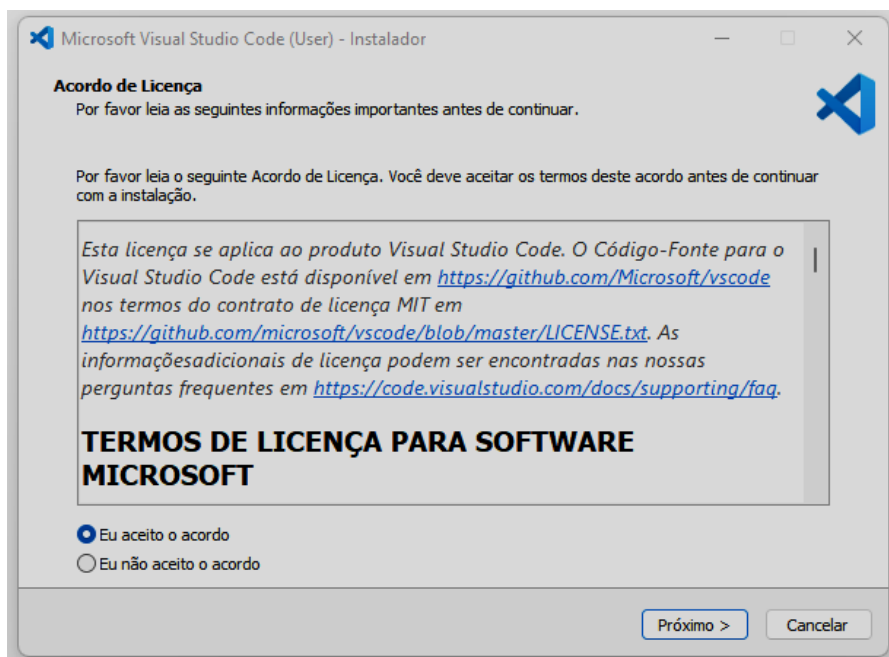
6. Agora é necessário instalar o Ambiente de Desenvolvimento que no caso será o VSCode, o VSCode é um software da Microsoft e é totalmente gratuito. abra o site:

<https://code.visualstudio.com/>



7. Clique do botão de Download.
8. Após o download e inicie a instalação.

9. Marque a opção “Eu aceito o acordo” e continue a instalação, nenhuma outra opção é necessário ser alterada.
10. A Instalação do software é feita em inglês, mas é possível instalar a extensão em português.



Atividade 2 – Acessar o Terminal do Python

Objetivo: Acessar o terminal do Python e verificar a versão instalada.

1. Abra um terminal de Prompt de Comando do Windows e digite **python**, aperte enter.
2. Será aberto o terminal de comandos do Python e será mostrada a versão atual instalada.



3. Para sair do terminal digite **exit()** ou aperte CTRL + Z.

Atividade 3 – Configurar o VSCode

Objetivo: Configurar o VSCode, instalar extensões para utilização do Python

As extensões implementam funcionalidades a mais para o VSCode, é pelas Extensões que podemos configurar o idioma do Software e colocar ferramentas para ajudar na linguagem de programação que utilizaremos no VSCode.

1. Abra o VSCode e na barra lateral procure o botão de Extensão ou pelo menu **Ver / Extensões** (View/Extension).

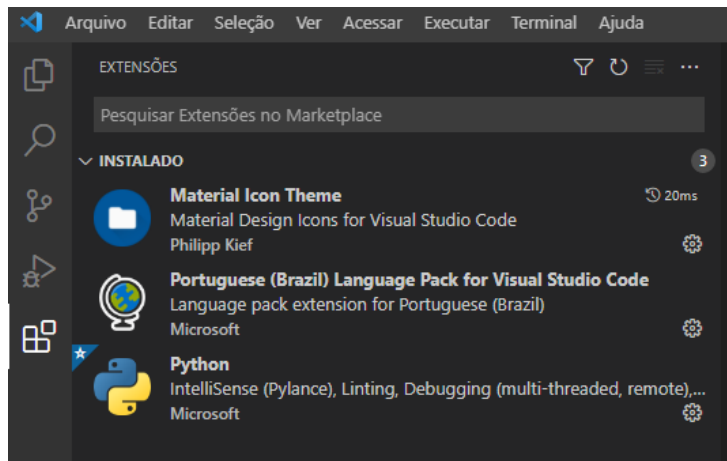


2. Para instalar as extensões basta realizar a pesquisa e clicar em instalar, faça a instalação destas 3 extensões:

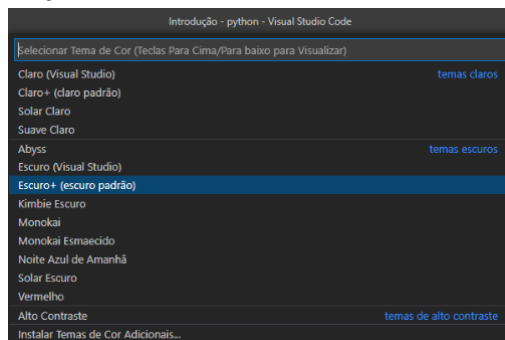
Portuguese (Brazil) Language Pack for Visual Studio Code,

Python e

Material Icon Theme



3. A extensão *Portuguese (Brazil) Language Pack for Visual Studio Code* altera o idioma do VSCode, será necessário reiniciar o programa para alterar o idioma.
4. A extensão *Python* incluir ferramentas para o desenvolvimento em Python.
5. A extensão *Material Icon Theme* altera os ícones na janela do Explorer, facilitando a identificação dos arquivos do projeto.
6. É possível alterar as cores do ambiente de desenvolvimento usando o menu **Arquivo / Preferências / Tema de Cores**.



Atividade 4 – “Olá Mundo!”

```
SAÍDA  TERMINAL  CONSOLE DE DEPUÇÃO  PROBLEMAS
Python + - [ ] [x]

PS D:\Sistemas\python> & C:/Users/laerc/AppData/Local/Programs/Python/Python310/python.exe
d:/Sistemas/python/cap01_atividade04.py
Olá mundo!
Meu primeiro código em Python
PS D:\Sistemas\python> 
```

Objetivo: Nesta atividade você irá imprimir na tela o texto “Olá mundo!” em uma linha e na linha abaixo o texto “Meu primeiro código em Python.”

Comandos utilizados: comentário e print

- # Símbolo de comentário, usado para descrever um código
- print Usado para “imprimir” uma informação no terminal

1. Abra um novo arquivo no VSCode e salve com o nome de `cap1_atividade04.py`.
2. Digite um comentário na primeira linha de código.

criando o meu primeiro programa em Python!!!

3. Agora digite o comando para imprimir no terminal.

```
print('Olá mundo!')
```

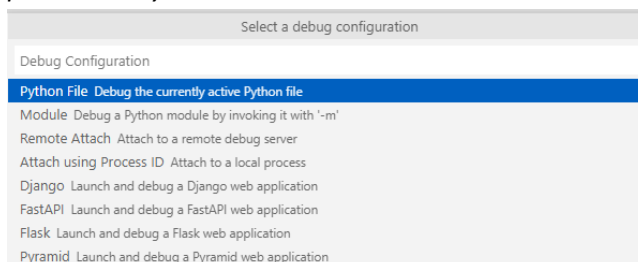
4. Execute o comando usando as teclas **CTRL+F5**.
5. No Terminal será mostrada a frase **Olá Mundo!**
6. Agora digite o comando print usando duas frases separadas por vírgula.

```
print('Meu primeiro código em', 'Python')
```

7. Para executar os comandos você tem algumas opções que são:
Apertar F5 ou menu Executar / Iniciar a Depuração.
Apertar CTRL + F5 ou menu Executar / Iniciar sem Depuração.
Clicar no botão Executar Arquivo no Terminal, que fica no canto superior direito da tela.

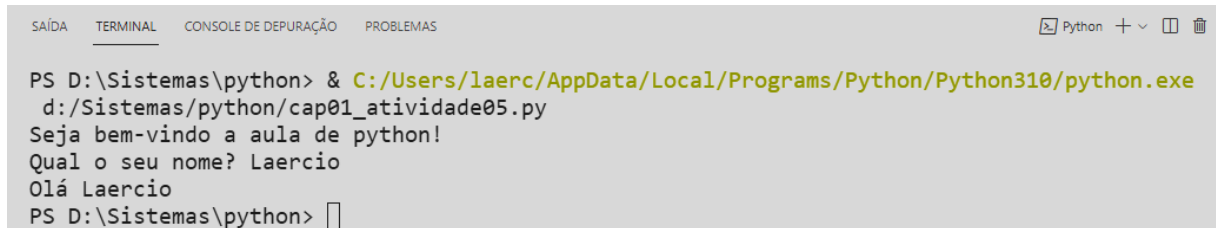


8. Ao usar F5 o VSCode pede um Depurador, pode usar a primeira opção que é o depurador padrão do Python.



9. Observe o resultado do comando no terminal que aparece na parte inferior da tela.
10. É mostrado no terminal o cominho do interpretador que está sendo utilizado, o nome do arquivo que foi executado e o resultado do comando print.

Atividade 5 – Usando o Input



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
Python  +  ▢  🗑
PS D:\Sistemas\python> & C:/Users/laerc/AppData/Local/Programs/Python/Python310/python.exe
d:/Sistemas/python/cap01_atividade05.py
Seja bem-vindo a aula de python!
Qual o seu nome? Laercio
Olá Laercio
PS D:\Sistemas\python> 
```

Objetivo: Nesta atividade você vai usar o comando input para interagir com o usuário pedindo uma informação e usará o comando print com format e posição de substituição.

Comandos utilizados: print, input

| | |
|-----------------|--|
| input | Usando para interagir com o usuário, o prompt fica esperando o usuário entrar com uma informação |
| print {} format | As {} indicam uma posição de substituição que será usada em conjunto com o comando format, conforme exemplo abaixo: <code>print('Exemplo de {}' . format('Posição de Substituição'))</code> |

1. Abra um novo arquivo no VSCode e salve com o nome de `cap2_exercicio05.py`.
2. Use o `print` para imprimir no terminal uma mensagem de bem-vindo, digite a mensagem separada usando o símbolo de `+` em vez de `,` como foi feito na atividade anterior.

```
print('Seja bem-vindo a aula de' + ' Python!')
```

3. Agora use o `input` para pedir o nome do usuário e o `print` para dizer um `olá`.

```
print('Olá {}' . format(input('Qual o seu nome? ')))
```

4. Execute os comandos usando as teclas **CTRL+F5**.
5. A primeira mensagem será impressa no terminal e a o prompt ficará aguardando que você escreva o seu nome, escreva o seu nome e aperte `Enter`.

CAPÍTULO 2 – Variáveis e Operadores

A variável é um dos recursos mais básicos de programação. Utilizada para armazenar valores em memória, elas permitem gravar e ler esses dados com facilidade a partir de um nome definido. Neste capítulo você vai aprender a declarar e atribuir valores a variáveis em Python e usar variáveis numéricas para realizar cálculos matemáticos e manipular variáveis de texto.

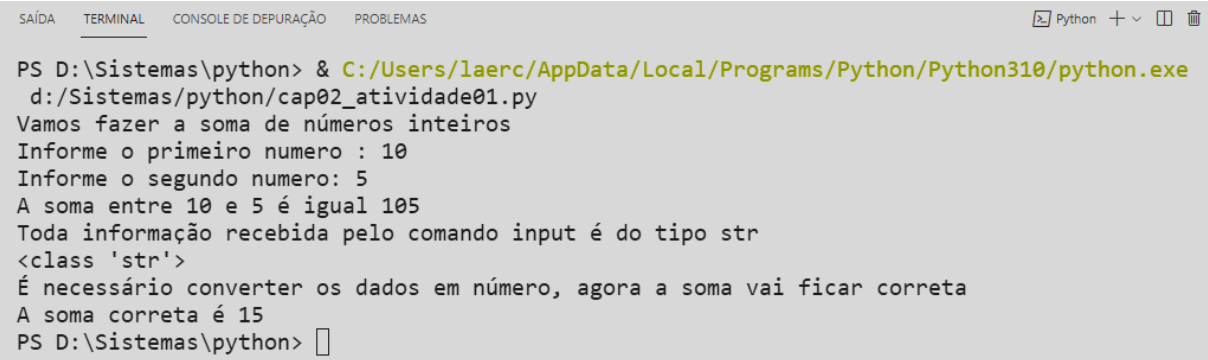
Objetivos:

- Entender o que são as variáveis.
- Trabalhar com os tipos de dados (número, texto).
- Utilizar operadores matemáticos básicos.
- Manipular textos.

Tópicos do capítulo

- Utilização de variáveis.
- Utilizando operadores matemáticos.
- Métodos para variáveis do tipo string.

Atividade 1 – Somando Números



```
SAÍDA  TERMINAL  CONSOLE DE DEPUÇÃO  PROBLEMAS  Python + - [ ] [X]
PS D:\Sistemas\python> & C:/Users/laerc/AppData/Local/Programs/Python/Python310/python.exe
d:/Sistemas/python/cap02_atividade01.py
Vamos fazer a soma de números inteiros
Informe o primeiro numero : 10
Informe o segundo numero: 5
A soma entre 10 e 5 é igual 105
Toda informação recebida pelo comando input é do tipo str
<class 'str'>
É necessário converter os dados em número, agora a soma vai ficar correta
A soma correta é 15
PS D:\Sistemas\python> [ ]
```

Objetivo: Nesta atividade você vai somar dois números usando variáveis e irá verificar os tipos de dados de uma variável e como converter uma variável em um tipo diferente.

Comandos utilizados: Variáveis, type, int, float

- | | |
|------|--|
| type | Retorna o tipo de dados de uma variável |
| int | Transforma uma variável no formato int (número inteiro) |
| + | Operador matemático para soma de números ou operador de concatenação para texto (String) |

1. Abra um novo arquivo do Python no VSCode e salve como `cap02_atividade01.py`.
2. A variável vai precisar receber um valor, para que o usuário passe este valor é necessário usar o comando `input` conforme o exemplo abaixo:

```
num1 = input('Informe o primeiro número : ')\nnum2 = input('Informe o segundo número: ')
```

3. Onde `num1` e `num2` são as variáveis e `input` permite que o usuário informe o valor desejado.
4. Para realizar a soma utilize uma terceira variável e use o símbolo de `+` para somar.

```
soma = num1 + num2
```

5. Para mostrar o resultado da variável utilize o comando `print / format`.

```
print('A soma entre {} e {} é igual {}'.format(num1, num2, soma))
```

6. Observe que não foi realizada a soma e sim a concatenação das informações, toda informação passada pelo `input` é do tipo `<str>`, para verificar o tipo de dados de uma variável utilize o comando `type`.

```
print(type(num1))
```

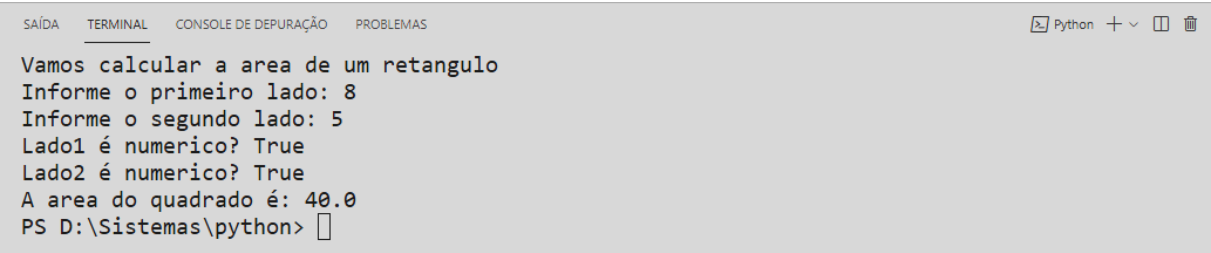
7. Para que a soma seja feita corretamente é necessário usar o comando `int`, o comando `int` vai transformar a variável em número inteiro, sem casas decimais, podemos usar este comando no momento da soma.

```
soma = int(num1) + int(num2)
```

8. Para finalizar imprima a soma de forma correta.

```
print('A soma correta é {}'.format(soma))
```

Atividade 2 – Área de um retângulo



```
SAÍDA  TERMINAL  CONSOLE DE DEPUÇÃO  PROBLEMAS\n\nVamos calcular a area de um retangulo\nInforme o primeiro lado: 8\nInforme o segundo lado: 5\nLado1 é numerico? True\nLado2 é numerico? True\nA area do quadrado é: 40.0\nPS D:\\Sistemas\\python>
```

Objetivo: Nesta atividade você vai calcular a área de um retângulo através de dois números fornecidos pelo usuário e fará também a verificação se os dados informados são realmente números.

Comandos utilizados: `Float` e método `isnumeric()`

- `float` Transforma uma variável no formato float (número com casas decimais)
- `isnumeric` Verifica se o valor de uma variável é numérico inteiro (`int`)
- `isdecimal` Verifica se o valor de uma variável é número com casas decimais (`float`)

1. Abra um novo arquivo no VSCode e salve com o nome `cap02_atividade02.py`.
2. Peça para o usuário os dois lados do retângulo, será necessário usar duas variáveis com o comando `input`.

```
lado1 = input('Informe o primeiro lado: ')
lado2 = input('Informe o segundo lado: ')
```

3. Antes de realizar o cálculo você vai mostrar para no terminal se o `lado1` é `numeric` e se o `lado2` é `decimal`, o retorno destes comandos é `True` ou `False`. Imprima esta informação na tela usando o `print`.

```
print('Lado1 é numérico?' , lado1.isnumeric())
print('Lado2 é numérico?' , lado2.isdecimal())
```

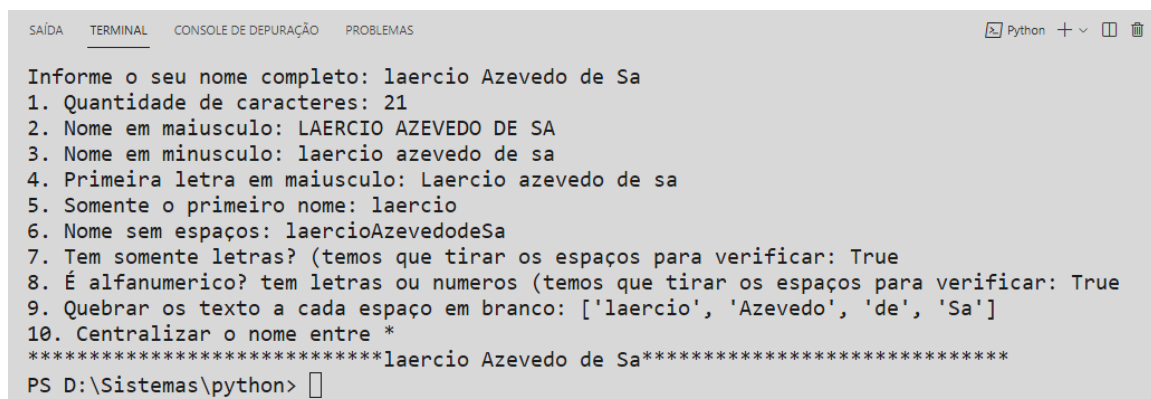
4. Para o cálculo da área do retângulo basta converter os valores em `float` e realizar a multiplicação, observe que é necessário realizar a conversão do número não basta apenas verificar se ele é numérico.

```
area = float(lado1) * float(lado2)
```

5. Para finalizar mostre o resultado do cálculo usando o comando `print / format`.

```
print('A área do quadrado é: {} ' . format(area))
```

Atividade 3 – Trabalhando como Textos



```
SAÍDA  TERMINAL  CONSOLE DE DEPURAÇÃO  PROBLEMAS
Python  +  -  [ ]  [X]

Informe o seu nome completo: laercio Azevedo de Sa
1. Quantidade de caracteres: 21
2. Nome em maiusculo: LAERCIO AZEVEDO DE SA
3. Nome em minusculo: laercio azevedo de sa
4. Primeira letra em maiusculo: Laercio azevedo de sa
5. Somente o primeiro nome: laercio
6. Nome sem espaços: laercioAzevedodeSa
7. Tem somente letras? (temos que tirar os espaços para verificar: True
8. É alfanumerico? tem letras ou numeros (temos que tirar os espaços para verificar: True
9. Quebrar os texto a cada espaço em branco: ['laercio', 'Azevedo', 'de', 'Sa']
10. Centralizar o nome entre *
*****laercio Azevedo de Sa*****
PS D:\Sistemas\python> [ ]
```

Objetivo: Nesta atividade você vai trabalhar com dados de texto, usando vários métodos para verificar e modificar o valor de uma variável. Aprenderá a usar a biblioteca `system` para limpar a tela do terminal.

Comandos utilizados: Biblioteca do sistema operacional `os` com a classe `system`, função `len` e os métodos `upper`, `lower`, `capitalize`, `find`, `replace`, `isalpha`, `isalnum`, `split`, `center` para variáveis de texto

`os`, `system` A biblioteca que permite acessar informações do sistema operacional é a `os`, `system` é classe que permite executar os comandos do sistema, você vai usar o comando `cls` que limpa o terminal do Windows em Linux é possível usar o comando `clear`

| | |
|-----------------|--|
| len | Conta o número de caracteres de um texto |
| upper | transforma um texto em maiúsculo |
| lower | transforma um texto em minúsculo |
| capitalize | somente a primeira letra da sentença em maiúsculo |
| find | Localizar a posição de um caractere |
| [início: final] | Contar o texto a partir de uma posição inicial e final |
| replace | Substitui um caractere por outro |
| isalnum | Verifica se o texto tem apenas letras e números |
| isalpha | Verifica se o texto tem apenas letras |
| split | Quebra o texto a partir de um caractere |
| center | Centraliza o texto no terminal |

1. Abra um novo arquivo no VSCode e salve com o nome de `cap02_atividade03.py`.
2. Neste exemplo você vai usar a biblioteca do sistema operacional para limpar a tela, as bibliotecas devem sempre ser declaradas no início do código.

```
from os import system
```

3. Para limpar o terminal é necessário usar o comando `cls`.

```
system('cls')
```

4. Agora peça para o usuário o seu nome completo guardando o valor em uma variável

```
nomecompleto = input('Informe o seu nome completo: ')
```

5. O primeiro comando que vai usar é o `len()`, nele é possível contar o número de caracteres de uma variável.

```
print('1. Quantidade de caracteres:', len(nomecompleto))
```

6. Os próximos métodos irão trabalhar com letras maiúsculas e minúsculas. O método `upper()` deixa todos os caracteres em maiúsculos o `lower()` em minúsculo e o `capitalize()` deixa a somente o primeiro caractere da sentença em maiúsculo.

```
print('2. Nome em maiúsculo:', nomecompleto.upper())
```

```
print('3. Nome em minúsculo:', nomecompleto.lower())
```

```
print('4. Primeira letra em maiúsculo:', nomecompleto.capitalize())
```

7. Agora você vai procurar o local onde se encontra o primeiro espaço do nome informado usando o método `find()` e com isto podemos pegar parte do texto usando a função de fatia [início: início + tamanho].

```
espaco = nomecompleto.find(' ')
```

```
print('5. Somente o primeiro nome:', nomecompleto[0:espaco])
```

8. O próximo método que vai usar permite de se faça a troca de um caractere por outro, este é o método `replace()`, faça isto para tirar os espaços em branco do nome.

```
print('6. Nome sem espaços:', nomecompleto.replace(' ', ''))
```

9. Existem dois métodos para verificar que o nome informado é composto de texto, temos o método `isalpha` e `isalnum`, no primeiro é verificado se o texto tem apenas letras e no segundo

se tem letras e números, mas para ambos você vai tirar o espaço em branco antes da verificação.

```
print('7. Tem somente letras? (temos que tirar os espaços para  
verificar:', nomecompleto.replace(' ', '').isalpha())  
print('8. É alfanumérico? tem letras ou números (temos que tirar os  
espaços para verificar:', nomecompleto.replace(' ', '').isalnum())
```

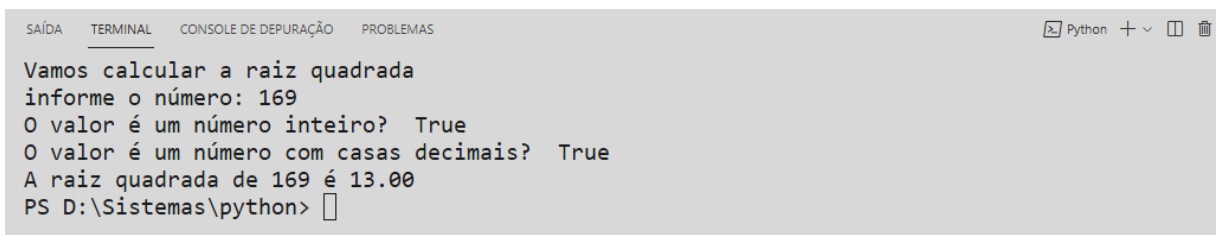
10. Existe outro método que permite quebrar o texto em partes específicas, aqui você pode quebrar o texto através do espaço em branco, isto é que o chamamos de lista (array), veremos isto com mais detalhes nos próximos capítulos.

```
print('9. Quebrar o texto a cada espaço em branco:', nomecompleto.split(" "))
```

*11. E para finalizar você vai centralizar o texto usando * para completar as laterais esquerda e direita.*

```
print('10. Centralizar o nome entre *')  
print(nomecompleto.center(80, "*"))
```

Atividade Extra – Calculando Raiz Quadrada



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS  
Python + - [ ] [X]  
Vamos calcular a raiz quadrada  
informe o número: 169  
O valor é um número inteiro? True  
O valor é um número com casas decimais? True  
A raiz quadrada de 169 é 13.00  
PS D:\Sistemas\python>
```

Objetivo: Nesta atividade você vai calcular a raiz quadrada a partir do operador matemático ****** (exponenciação). O valor será passado pelo usuário e verificaremos se este valor é numérico ou decimal.

Comandos utilizados: Variáveis, operador Exponenciação (******) e métodos **isnumeric** e **isdecimal**.

1. Crie um código para calcular a raiz quadrada de um número inteiro.
2. Peça número para o usuário.
3. Verifique se o número é numérico.
4. Faça o cálculo da raiz quadrada ($\text{número}^{1/2}$).
5. Mostre o resultado no terminal.

CAPÍTULO 3 – Estruturas Condicionais

Neste capítulo você vai aprender a usar estruturas condicionais para tomadas de decisão, toda tomada de decisão parte de uma condição lógica. No Python a decisão simples que seria apenas verdadeiro ou falso usamos o comando `if / else` e para quando precisamos de mais que duas opções de decisão podemos usar o `if / elif / else`.

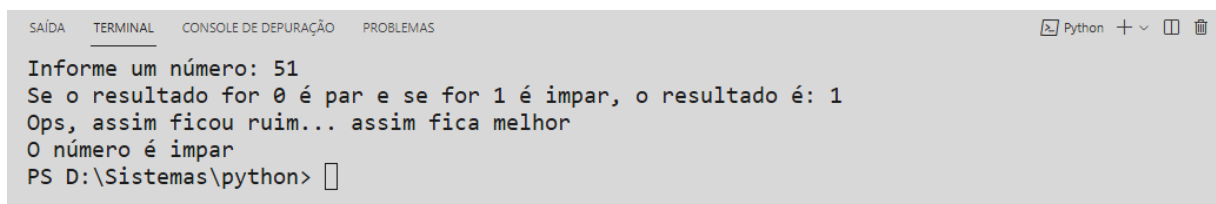
Objetivos:

- Criar estruturas condicionais de decisão simples e complexas usando `if / elif / else`.
- Conhecer os operadores básicos de comparação do teste lógico de um `if`.
- Utilizar um `if` aninhado.
- Saber construir um `if` ternário, `if` escrito em uma linha apenas.

Tópicos do capítulo

- Tomar decisões simples com `if / else`.
- Tomar decisões simples com `if / elif / else`.
- Usar `if` aninhados e ternário.

Atividade 1 – Verificar Número Par e Ímpar



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
Informe um número: 51
Se o resultado for 0 é par e se for 1 é ímpar, o resultado é: 1
Ops, assim ficou ruim... assim fica melhor
O número é ímpar
PS D:\Sistemas\python>
```

Objetivo: Nesta atividade você vai usar uma estrutura de decisão (`if / else`) para verificar se um número é par ou ímpar.

Comandos utilizados: Comando `if`, operador `%` (retorna o resto da divisão entre operandos)

`if / else`

`if` (teste lógico):

 #código para condição verdadeira do teste, tem que ser indentado.

`else:`

 #código para condição falsa do teste, tem que ser indentado.

O texto lógico normalmente é a comparação entre duas informações.

1. Abra um arquivo novo arquivo e salve como `cap03_atividade01.py`.
2. Peça um número com o comando `print` e guarde na variável `número` (observe que a variável no Python pode ter acento).

```
número = int(input('Informe um número: '))
```

3. Para saber se um número é par ou ímpar basta dividir o número por 2 e sobrar um resto na divisão o número é ímpar, esta função se chama MOD no Python podemos usar o símbolo de % para realizar o MOD.

```
resultado = int(número % 2)
```

4. Imprima o valor da variável `resultado` para visualizar como ficou.

```
print('Se o resultado for 0 é par e se for 1 é ímpar, o resultado é:',  
resultado)
```

5. Ao usar o comando `if/else` é possível personalizar a resposta se um número é par ou ímpar, a estrutura é feita da seguinte maneira `if` (teste lógico): resposta verdadeira do teste `else`: resposta falsa do teste, veja abaixo como seria o código.

```
if resultado == 0:  
    resultado = 'O número é par'  
else:  
    resultado = 'O número é ímpar'
```

Observe que a resposta final será armazenada em uma variável, os operadores mais comuns de teste lógico que são:

`==` igualdade

`!=` diferente

`>=` maior igual

`>` maior

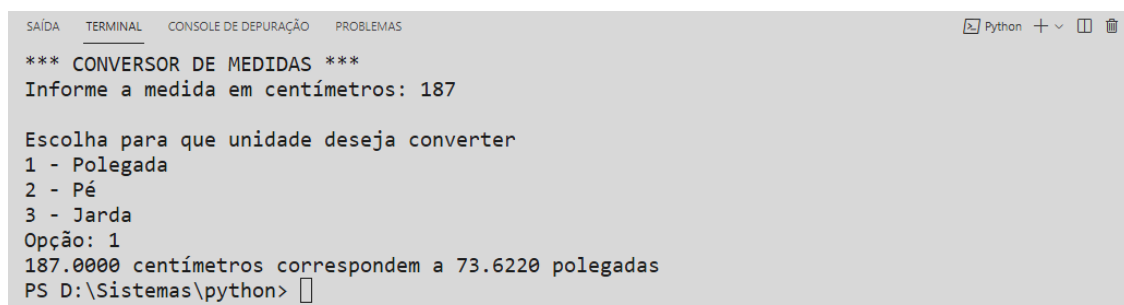
`<=` menor igual

`<` menor

6. Faça um `print` para mostrar o resultado no terminal.

```
print(resultado)
```

Atividade 2 – Conversor de Medidas



```
SAÍDA  TERMINAL  CONSOLE DE DEPUÇÃO  PROBLEMAS
*** CONVERSOR DE MEDIDAS ***
Informe a medida em centímetros: 187

Escolha para que unidade deseja converter
1 - Polegada
2 - Pé
3 - Jarda
Opção: 1
187.0000 centímetros correspondem a 73.6220 polegadas
PS D:\Sistemas\python>
```

Objetivo: Nesta atividade você vai converter um número em centímetros para Polegada, Pé ou Jarda. Será necessário usar o comando if / elif / else.

Comandos utilizados: If / elif / else, formatação de números com posição de substituição {:.4f}

If / elif / else If (teste lógico):

 #código para condição verdadeira do teste, tem que ser indentado.

elif (novo teste lógico):

 #código para condição verdadeira do teste, tem que ser indentado.

É possível utilizar quantos elif forem necessários.

else:

 #código para condição falsa dos testes, tem que ser indentado.

O texto lógico normalmente é a comparação entre duas informações.

{:.4f} Formatação de caractere de substituição, neste caso formatar número float com 4 casas decimais, para outras formações consulte.

https://www.w3schools.com/python/ref_string_format.asp

1. Abra um novo arquivo e salve como cap03_atividade02.py.
2. Limpe o terminal com a biblioteca os e imprima o nome do programa que está criando.

```
from os import system
system('cls')
print('*** CONVERSOR DE MEDIDAS ***')
```

3. Peça para o usuário informe a medida inicial em centímetros.

```
medida = float(input('Informe a medida em centímetros: '))
```

4. Crie um menu para o usuário saber como será feita a conversão de medida, use o \n dentro do print para quebrar linha no terminal.

```
print('\nEscolha para que unidade deseja converter')
print('1 - Polegada\n2 - Pé\n3 - Jarda')
```

5. Peça para o usuário informar a conversão desejada usando uma variável com o comando input.

```
menu = input('Opção: ')
```

6. Agora você vai precisar verificar qual foi a opção selecionada, neste caso temos 3 opções de escolha e o usuário ainda poderia indicar uma opção errada, para isto é necessário usar o if / elif / else, na estrutura do if crie uma variável para guardar o texto para o resultado que será impresso no terminal.

```
if menu=="1":
    polegadas = medida / 2.54
    resultado = '{:.4f} centímetros correspondem a {:.4f} polegadas' .
    format(medida, polegadas)
elif menu=="2":
    pes = medida / 30.48
```

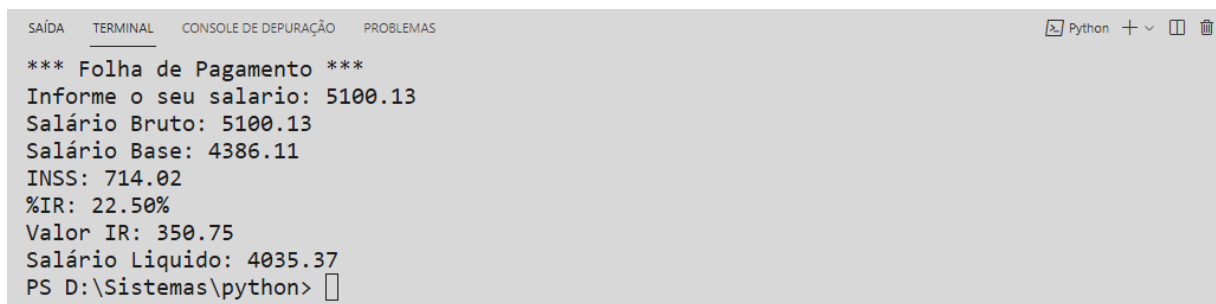
```

    resultado = '{:.4f} centímetros correspondem a {:.4f} pés' .
format(medida, pes)
elif menu=="3":
    jardas = medida / 91.44
    resultado = '{:.4f} centímetros correspondem a {:.4f} jardas' .
format(medida, jardas)
else:
    resultado = "Você não escolheu uma das opções..."

    Observe que os números precisam ser escritos com aspas ("" ) pois a variável input é sempre
    retornada como string.
7. Agora faça a impressão no terminal do resultado.
print(resultado)

```

Atividade 3 – Folha de Pagamento



```

SAÍDA  TERMINAL  CONSOLE DE DEPUÇÃO  PROBLEMAS
Python + v [ ] [ ]
*** Folha de Pagamento ***
Informe o seu salário: 5100.13
Salário Bruto: 5100.13
Salário Base: 4386.11
INSS: 714.02
%IR: 22.50%
Valor IR: 350.75
Salário Líquido: 4035.37
PS D:\Sistemas\python>

```

Objetivo: Nesta atividade você vai calcular uma folha de pagamento usando o comando if aninhado e mostrando o resultado dos valores de Imposto de Renda e INSS a serem pagos.

Comandos utilizados: Variáveis, if, operadores matemáticos e formatação com posição de substituição.

- | | |
|-------------|--|
| name | Classe da biblioteca os, retorna o nome do sistema operacional. |
| If ternário | <p>O if ternário é um if que é escrito em apenas uma linha, isto facilita quando é necessário verificar um valor e executar uma informação de uma maneira ou de outra, a sintaxe é:</p> <p>(código para o teste verdadeiro) if (teste lógico) else (código para o falso)</p> |

1. Abra um arquivo novo arquivo e salve como cap03_atividade03.py.
2. Coloque um cabeçalho inicial no código para limpar a tela e indicar o programa que estamos criando. Neste caso foi utilizado um if ternário, que nada mais é que um if realizado em apenas uma linha, foi utilizado desta forma para verificar se o sistema operacional é Windows ou Linux e o código ficar pronto para qualquer um deles. A classe name foi necessário para que seja possível saber qual o sistema operacional em uso.

```

from os import system, name
system('cls') if(name == 'nt') else system('clear')
print('*** Folha de Pagamento ***')

```

3. Peça o valor do salário para o usuário e já faça a conversão do valor em float (número com opção de casas decimais).

```
print('*** Folha de Pagamento ***')
```

4. Agora é necessário criar o cálculo do valor do INSS, a tabela atual do INSS é:

| Salário (de) | Salário (até) | Alíquota |
|--------------|---------------|----------|
| 0,00 | 1.100,00 | 7,5% |
| 1.100,01 | 2.203,48 | 9,0% |
| 2.203,49 | 3.305,22 | 12,0% |
| 3.305,23 | 6.433,57 | 14,0% |

Acima de R\$ 6.433,57 o valor é de 900.70 fixo.

5. Para o cálculo você pode usar um if aninhado para verificar se o valor é acima de 6433,57 e se for abaixo usar outro if para verificar a alíquota de % e assim multiplicar pelo salário.

```

# Calcular INSS
if salario > 6433.58:
    inss = 900.70
else:
    if salario > 3305.23:
        inssP = 0.14
    elif salario > 2203.49:
        inssP = 0.12
    elif salario > 1100.01:
        inssP = 0.09
    else:
        inssP = 0.075
    inss = salario * inssP

```

6. Agora você vai precisar calcular o valor de Imposto de Renda, o IR é calculado através do salário base que nada mais é do que o salário bruto - INSS. Faça este cálculo e guarde a informação em uma variável.

```
base = salario - inss
```

7. O IR é calculado através de faixa de valores, gerando um percentual de desconto e um valor a ser deduzido (descontado), a tabela atual do IR é calculada conforme a tabela a seguir:

| Base de Cálculo Mensal | Alíquota | Parcela a Deduzir |
|----------------------------------|----------|-------------------|
| Até R\$ 1.903,98 | isento | R\$ 0,00 |
| De R\$ 1.903,99 até R\$ 2.826,65 | 7,5 % | R\$ 142,80 |
| De R\$ 2.826,66 até R\$ 3.751,05 | 15 % | R\$ 354,80 |
| De R\$ 3.751,06 até R\$ 4.664,68 | 22,5 % | R\$ 636,13 |
| Acima de R\$ 4.664,68 | 27,5 % | R\$ 869,36 |

8. Na tabela do IR temos 5 valores diferentes e você pode usar o `if` neste caso também.

```
if base > 4664.68:
    irP = 0.275
    deducacao = 869.36
elif base > 3751.06:
    irP = 0.225
    deducacao = 636.13
elif base > 2826.66:
    irP = 0.15
    deducacao = 354.80
elif base > 1903.98:
    irP = 0.075
    deducacao = 142.80
else:
    irP = 0
    deducacao = 0
```

9. Para finalizar guarde o valor do IR em uma variável.

```
ir = (base * irP) - deducacao
```

10. E agora calcule o salário líquido subtraindo o salário base – ir

```
salarioLiquido = base - ir
```

11. Para finalizar imprimi no terminal as informações do Salário, Salário Base, INSS, %IR, Valor IR e Salário Líquido, todos formatados com dois dígitos decimais.

```
print('Salário Bruto: {:.2f}\nSalário Base: {:.2f}\nINSS: {:.2f}\n%IR: {:.2f}%\nValor IR: {:.2f}\nSalário Líquido: {:.2f}' . format(salario, base, inss, irP*100, ir, salarioLiquido))
```

Atividade Extra – Calcular IMC

```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
*** CALCULADORA DE IMC ***
Informe o seu peso em KG: 82
Agora a sua altura em centímetros: 176
O seu IMC é de 26.47
Sobrepeso
PS D:\Sistemas\python>
```

Objetivo: Nesta atividade você vai calcular o IMC a partir de um peso e uma altura, usará a comando if para mostrar o resultado do cálculo do IMC.

Comandos utilizados: Variáveis, if / elif / else.

1. Para esta atividade o desafio é criar uma calculadora para mostrar o índice de IMC.
2. Peça primeiramente o peso e depois a altura.
3. Para o cálculo do índice utilize a tabela abaixo:

| CLASSIFICAÇÃO | IMC |
|------------------------------|-------------------|
| Abaixo do Peso | Abaixo 18,5 |
| Peso Normal | 18,5 - 24,9 |
| Sobrepeso | 25 - 29,9 |
| Obesidade Grau I | 30 - 34,9 |
| Obesidade Grau II | 35 - 39,9 |
| Obesidade Grau III ou Móbida | Maior ou Igual 40 |

4. Ao final imprima as informações no terminal.

CAPÍTULO 4 – Estruturas de repetição

Neste capítulo você irá aprender a criar estruturas de repetição que são úteis para repetir uma operação determinadas vezes ou até que uma condição não seja mais verdadeira, no Python temos duas estruturas de repetição que podemos usar, comando for e o while.

Objetivos:

- Entender como funciona uma estrutura de loop for.
- Criar loops aninhados usando o for.
- Utilizar o while para criar um loop e o comando break para terminar a execução.
- Criar um menu com while para ser reutilizado em diversos códigos.

Tópicos do capítulo

- Loop usando For / range.
- Loop aninhado de For.
- Loop usando While.

Atividade 1 – Tabuada



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
Python  +  ▢  🗑️

*** Tabuada Simples ***
Informe o multiplicador 5
5 * 0 = 0
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
PS D:\Sistemas\python> ▢
```

Objetivo: Nesta atividade você vai montar uma Tabuada usando a estrutura de Loop do For e range.

Comandos utilizados: Comandos for e range

- For / range O for é utilizado para criar um loop, no for é indicado quantas vezes o loop será executado atrás do range.
- for i in range(10) #loop começando em 0 até 9 (10 vezes).
 - for i in range(2, 10) #loop começando em 2 até 9 (8 vezes).

1. Abra um novo arquivo e salve como `cap04_atividade01.py`.
2. Peça um número para o usuário e guarde numa variável.

```
multiplicador = int(input('Informe o multiplicador '))
```

3. Para criar a tabuada é necessário criar um loop com o comando `For / Range`.
4. O `range` indica quantas vezes o grupo de código será executado, o `range(11)` indica que o valor inicial começa em zero e vai até o 10.

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

5. Agora para imprimir a informação no terminal use o comando `print` e faça o cálculo da tabuada.

```
print('{} * {} = {}'.format(multiplicador, i, i*multiplicador))
```

Atividade 2 – MultiTabuada

```
*** MULTI TABUADA 1 ***
 1  2  3  4  5  6  7  8  9 10
 2  4  6  8 10 12 14 16 18 20
 3  6  9 12 15 18 21 24 27 30
 4  8 12 16 20 24 28 32 36 40
 5 10 15 20 25 30 35 40 45 50
 6 12 18 24 30 36 42 48 54 60
 7 14 21 28 35 42 49 56 63 70
 8 16 24 32 40 48 56 64 72 80
 9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100

*** MULTI TABUADA 2 ***
 1  2  3  4  5  6  7  8  9 10
 2  4  6  8 10 12 14 16 18 20
 3  6  9 12 15 18 21 24 27 30
 4  8 12 16 20 24 28 32 36 40
 5 10 15 20 25 30 35 40 45 50
 6 12 18 24 30 36 42 48 54 60
 7 14 21 28 35 42 49 56 63 70
 8 16 24 32 40 48 56 64 72 80
 9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

Objetivo: Nesta atividade você vai construir uma multitable de duas maneiras, na primeira usando `for` e na segunda usando `for` aninhado.

Comandos utilizados: Comandos `for` e `range`.

1. Abra um novo arquivo e salve como `cap04_atividade02.py`.
2. A `MultiTabuada` é composta de 10 colunas e 10 linhas começando pelo número 2 e a intersecção dos números gera a multiplicação.
3. Faça um `for` range de 1 até 11.

```
for i in range(1, 11):
```

4. E agora imprima na tela os valores formatados com espaços em branco para que a aparência fique alinhada.

```
print('{: >4} {: >4} {: >4} {: >4} {: >4} {: >4} {: >4} {: >4} {: >4} {: >4} {: >4}' . format(i, i*2, i*3, i*4, i*5, i*6, i*7, i*8, i*9, i*10))
```

Este exemplo é bem simples, mas gera uma dificuldade pois é necessário saber quantas colunas serão impressas, no caso das linhas é possível alterar facilmente, agora você vai criar outro for que o número de colunas poderá ser alterado facilmente.

5. No mesmo arquivo crie um for range de 1 até 11.

```
for i in range(1, 11):
```

6. Você vai precisar montar uma variável com a informação de cada linha a ser impressa, e começaremos com a informação da primeira coluna que é a variável i.

```
linha = '{: >4} ' . format(i)
```

7. Para as demais colunas da multilinha crie um for range de 2 a 11.

```
for ii in (range(2, 11)):
```

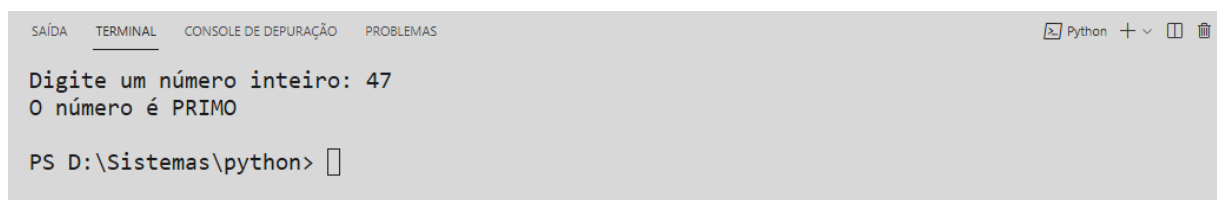
8. Agora incremente a variável linha com as informações da multiplicação da variável i com a ii.

```
linha+='{: >4} ' . format(ii*i)
```

9. Para finalizar volte a indentação e faça o print da variável linha.

```
print(linha)
```

Atividade 3 – Verificar Número Primo



Objetivo: Nesta atividade você vai verificar se um número é primo ou não e ainda indicar quem é o menor divisor deste número, usando o While para o realizar o loop.

Comandos utilizados: Comando f com variável de substituição, comando while e break.

| | |
|-------|---|
| While | While (condição) O loop While é um loop que será executado até que uma condição seja falsa, é necessário muito cuidado ao usar este loop pois pode acontecer um entrar em loop infinito. |
| break | Finaliza a execução de um while |

1. Abra um novo arquivo e salve como cap04_atividade03.py.

Os números primos são os números inteiros que podem ser divididos por apenas dois fatores: o número um e ele mesmo e com o for conseguiríamos fazer isto, mas neste exemplo usaremos o comando While para realizar o Loop.

2. Primeiramente peça ao usuário um número inteiro:

```
from os import system, name
system('cls') if(name == 'nt') else system('clear')
numero = int(input("Digite um número inteiro: "))
```

3. Será necessário criar uma variável para auxiliar o cálculo da divisão, esta variável deve ter o valor inicial de 2, o valor é 2 pois não iremos fazer verificação de números menores que 2.

```
i = 2
```

4. Agora você vai precisar de mais duas variáveis, uma para guardar o menor divisor do número informado quando ele não for primo e outra variável para mostrar a mensagem se o número é primo ou não, vamos iniciar a variável com o valor de 'O valor é menor que 2' para o caso de ser informado um número menor que 2.

```
divisor = 0
tipo = 'O número deve ser maior que 2'
```

5. O comando While é feito enquanto um teste lógico tem como resposta verdadeira, neste caso faremos da seguinte forma.

```
while i < numero:
```

6. Se o código entrar neste ponto mudaremos o valor da variável tipo para 'O número é PRIMO'.

```
tipo = 'O número é PRIMO'
```

7. Para verificar se um número é divisível por outra usaremos o operador %, se o resto for 0 significa que é divisível.

```
x = numero % i
```

8. Faça agora um if para verificar se a variável x é 0.

```
if x == 0:
```

9. Se o resto for zero quer dizer que o número não é primo e que o seu menor divisor é o valor da variável i, digite os seguintes comandos para fazer isto.

```
divisor = i
tipo = 'O número não é primo'
```

10. Como não é necessário continuar o Loop você pode usar o comando break para parar o While.

```
break
```

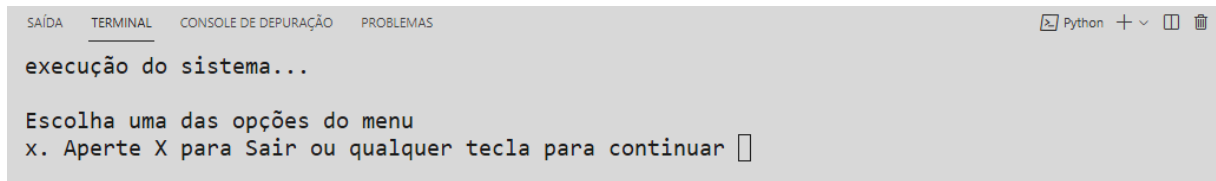
11. Se o resto da divisão não for zero, continue a aumentar a variável i com o mais 1.

```
i = i + 1
```

12. Para finalizar imprima na tela as mensagens se o número é primo ou não e qual é o seu menor divisor.

```
print(tipo)
print(f'O menor divisor é: {divisor}' if divisor > 0 else '')
```

Atividade Extra – Menu



The screenshot shows a terminal window with tabs for SAÍDA, TERMINAL, CONSOLE DE DEPURAÇÃO, and PROBLEMAS. The terminal output displays the text 'execução do sistema...' followed by a prompt 'Escolha uma das opções do menu' and a line 'x. Aperte X para Sair ou qualquer tecla para continuar' with a cursor.

```
SAÍDA  TERMINAL  CONSOLE DE DEPURAÇÃO  PROBLEMAS  Python + - [ ] [X]  
execução do sistema...  
  
Escolha uma das opções do menu  
x. Aperte X para Sair ou qualquer tecla para continuar █
```

Objetivo: Nesta atividade você vai criar um menu usando o While, este menu poderá ser utilizado em conjunto com outros códigos para que a execução do programa seja realizada até que o usuário deseje sair.

Comandos utilizados: Variáveis e while.

1. *Uma maneira fácil de criar um menu para repetir um código é utilizando o While, normalmente é pedido para o usuário apertar uma tecla para o programa continuar ou sair.*
2. *Podemos incorporar no menu também a opção de limpar a tela com o comando de sistema operacional (cls ou clear).*
3. *Faça o menu ficar repetindo até que o usuário aperte a tecla X.*

CAPÍTULO 5 – Listas e Tuplas

Neste capítulo você irá estudar a criação de listas e tuplas, em Python as listas e tuplas são sequências ou coleções ordenadas de valores, estes valores são chamados elementos ou itens. A diferença da Lista e Tuplas é que a tupla uma vez criada não pode ser modificada, ela é imutável, e a lista pode ter os valores incluídos, alteramos ou até mesmo excluídos. Ambas podem ser estruturas de um modo simples ou multidimensional.

Objetivos:

- Aprender a montar uma tupla ou lista.
- Listar os itens da tupla ou lista.
- Usar tupla / lista multidimensional.
- Manipular dados de uma lista.
- Contar a quantidade de itens de uma tupla / lista.

Tópicos do capítulo

- Criação de Tupla, utilize o símbolo ().
- Criação de Lista, utilize o símbolo [].
- Tupla ou Lista Multidimensional.

Atividade 1 – Número por Extenso



```
SAÍDA  TERMINAL  CONSOLE DE DEPUÇÃO  PROBLEMAS
Python + - []
Digite um número entre 0 e 99: 81
oitenta e um
PS D:\Sistemas\python> 
```

Objetivo: Nesta atividade você vai escrever um número por extenso, para isto usará uma tupla. A tupla é um array que contém dados que não podem ser alterados.

Comandos utilizados: Tupla, operadores / e %

- | | | |
|---------------|---|---|
| Criando tupla | a | A tupla é um array, e para criá-la use os () e dentro deles coloque os valores desejados, por exemplo. Tupla = ('valor1', 'valor2'). |
| Usando tupla | a | Para usar o valor de uma tupla indique a posição dela começando pelo valor zero, para mostrar o valor use os []. Print(Tupla[1]) |

1. Abra um novo arquivo e salve como `cap05_atividade01.py`.
2. A criação de uma biblioteca para transformar um número em seu formato por extenso pode ser feita de uma maneira simples usando tuplas ou listas (arrays), neste caso você irá usar a tupla (as informações não precisam ser alteradas).
3. Para a criação da tupla é usado os parênteses () para indicar o início e o fim dos dados, conforme exemplo abaixo:

```
numeros = ('zero', 'um', 'dois', 'três', 'quatro', 'cinco', 'seis',  
'sete', 'oito', 'nove')  
dez = ('dez', 'onze', 'doze', 'treze', 'quatorze', 'quinze', 'dezesseis',  
'dezesete', 'dezoito', 'dezenove')  
dezenas = ('', '', 'vinte', 'trinta', 'quarenta', 'cinquenta', 'sessenta',  
'setenta', 'oitenta', 'noventa')
```

Observe que foi criado uma tupla para os números de 0 a 10, outra para 10 a 19 e outra para as dezenas.

4. Tanto a tupla quanto a lista os itens começam em 0, neste caso para imprimir a tupla 3 das dezenas use o índice 2, conforme exemplo abaixo.

```
print(dezenas[2])  
#vinte
```

5. Agora peça para o usuário digitar um número inteiro de 0 a 99, para números maiores poderíamos complementar o código facilmente com novas tuplas.

```
pos = int(input('Digite um número entre 0 e 99: '))
```

6. Agora será necessário verificar com um if qual a faixa está o número informado, usaremos a faixa abaixo de 10, abaixo de 10, abaixo de 99 e se for informado um número acima podemos avisar o usuário que o número deve ser até 99.

```
if pos < 10:
```

```
elif pos < 20:
```

```
elif pos <= 99:
```

```
else:
```

7. Para cada faixa de valores você vai precisar indicar um código específico para mostrar o resultado do valor por extenso, comece pela faixa menor igual a 10, aqui você vai usar a tupla número.

```
extenso = numeros[pos]
```

8. Para a faixa de menor igual a 19 e consequentemente maior que 10 usará a tupla dez, será necessário subtrair 10 do valor para pegar a tupla correta.

```
extenso = dez[pos-10]
```

9. E para faixa entre 20 e 99 será necessário separar os números em duas partes, pegar a dezena e a numeral, para isto podemos usar os comandos de resto e divisão.

```
dezena = int(pos / 10)  
numeral = (pos % 10)
```

10. Ainda será necessário colocar um `if` para verificar se o número '0', pois aí não será necessário usar a palavra zero.

```
numero = ''
if (numeral!=0):
    numero = ' e ' + numeros[numeral]
extenso = dezenas[dezena] + numero
```

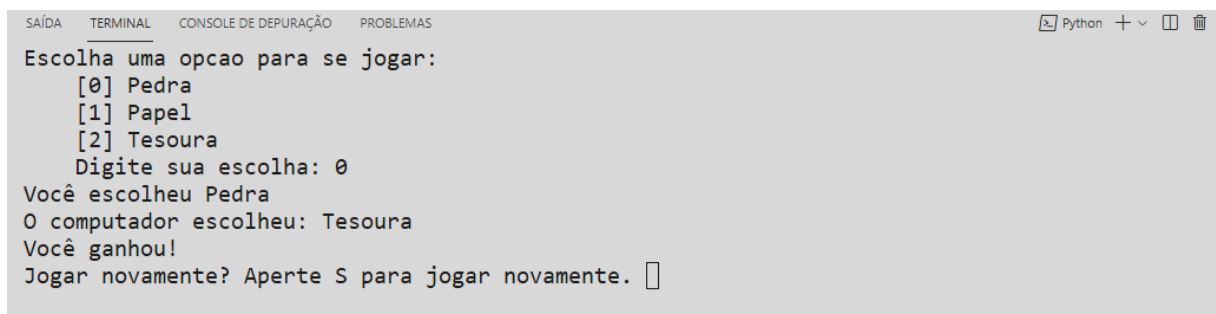
11. Para o `else` coloque uma mensagem que foi informado um número maior que 99.

```
extenso = 'Número maior que 100...'
```

12. Para finalizar imprima no terminal o valor por extenso.

```
print(extenso)
```

Atividade 2 – Jogo: Papel, pedra e Tesoura



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
Escolha uma opcao para se jogar:
[0] Pedra
[1] Papel
[2] Tesoura
Digite sua escolha: 0
Você escolheu Pedra
O computador escolheu: Tesoura
Você ganhou!
Jogar novamente? Aperte S para jogar novamente. □
```

Objetivo: Nesta atividade você vai criar um jogo usando tupla e tupla multidimensional.

Comandos utilizados: Tupla, Tupla Multidimensional, biblioteca `random` e `randint`.

Random A biblioteca `random` permite que você use números aleatórios

1. Abra um novo arquivo e salve com o nome `cap05_atividade02.py`.

2. Declare as bibliotecas no início do código.

```
from os import system, name
import random
```

3. Agora crie um menu com `while` para o programa seja repetido enquanto o usuário apertar `S`. aproveite e limpe o terminal assim que o `while` for executado.

```
opcao = 's'
while opcao.upper()=='S':
    system('cls') if(name == 'nt') else system('clear')
```

4. Para sortear a peça que o computador vai utilizar você vai usar a biblioteca `random` com o método `randint()`, no `randint` é possível indicar o valor inicial e o final do sorteio, neste caso será de 0 a 2.

```
computador = random.randint(0,2)
```

5. Peça para o usuário informar a peça que deseja utilizar, faça um menu informando o valor de cada peça de 0 até 2.

```
jogador = int(input('Escolha uma opção para se jogar:
[0] Pedra
[1] Papel
[2] Tesoura
Digite sua escolha: '))
```

6. Crie uma tupla com os itens que possam ser escolhidos, a tupla tem que seguir a mesma sequência do menu.

```
pecas = ("Pedra", "Papel", "Tesoura")
```

7. Imprima na tela as jogadas que foram feitas, tanto do usuário quanto do computador.

```
print('Você escolheu {}'.format(pecas[jogador]))
print('O computador escolheu: {}'.format(pecas[computador]))
```

8. Para sabermos quem ganhou e quem perdeu você pode criar uma tupla multidimensional, veja a tabela abaixo para ajudar a entender como ela funcionará.

| | | Jogador | | |
|------------|---------|------------------------|---------------------|------------------------|
| | | Pedra | Papel | Tesoura |
| Computador | Pedra | empate 0 | Vence Jogador 1 | Vence Computador -1 |
| | Papel | Vence Computador -1 | empate 0 | Vence Jogador 1 |
| | Tesoura | Vence Jogador 1 | Vence Jogador -1 | empate 0 |

9. Para montar a tupla multidimensional você pode fazer conforme o código abaixo:

```
tabela = ((0, 1, -1), (-1, 0, 1), (1, -1, 0))
```

10. E para pegar a jogada realizada basta indicar as duas dimensões da jogada, indicando primeiro a informação da linha e depois a informação da coluna.

```
jogada = tabela[computador][jogador]
```

11. Agora basta fazer uma verificação se a jogada foi 0, 1 ou -1. Utilize um if para isto.

```
if jogada == 0:
    resultado = "Deu empate vocês escolheram a mesma peça"
elif jogada == 1:
    resultado = "Você ganhou!"
else:
    resultado = "O computador ganhou"
```

12. Imprima no terminal o resultado do jogo e peça para o jogador apertar S para jogar novamente.

```
print(resultado)
opcao=input('Jogar novamente? Aperte S(sim) para jogar novamente. ')
```

Atividade 3 – Lista de Cadastro

```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
Informe o nome: Laercio
Informe o celular: (12) 99999-9999

Aperte X para Finalizar o cadastro ou qualquer tecla para continuar x
Nome: Laercio - Celular: (12) 99999-9999
PS D:\Sistemas\python> □
```

Objetivo: Nesta atividade você vai cadastrar o nome de pessoas e seu celular utilizando listas e ao final irá imprimir uma lista estes dados.

Comandos utilizados: Lista e método append, len para listas

| | |
|---------------|---|
| Criando lista | A lista é um array, e para cria-la use os [] e dentro deles coloque os valores desejados, diferente da tupla que é estática a lista pode ter índices adicionados, alterados ou excluídos. |
| Append(x) | Adiciona um item ao fim da lista |
| Remove(x) | Remove o primeiro item encontrado na lista cujo valor é igual a x |
| pop([i]) | Remove um item em uma dada posição na lista e o retorna. Se nenhum índice é especificado, a.pop() remove e devolve o último item da lista. |
| clear() | Remove todos os itens de uma lista. |

1. Abra um novo arquivo e salve com o nome `cap05_atividade03.py`.
2. Crie novamente um menu com `while` para que o usuário continue digitando informações até que aperte 'x', inicie uma variável chamada `opção` com o valor vazio de string e duas variáveis com o valor de uma lista vazia [].

```
from os import system, name
system('cls') if(name == 'nt') else system('clear')
opcao = ''
listaNome = []
listaCelular = []
while opcao!='x':
    #aqui vai o código do programa
    opcao=input('\nAperte X para Finalizar o cadastro ou qualquer tecla para
continuar ')

```

3. Peça para o usuário um nome e guarde na variável `nome` e faça a mesma coisa com o celular.

```
nome = input("Informe o nome: ")
celular = input("Informe o celular: ")

```

4. Agora será necessário acrescentar estes valores nas listas `listaNome` e `listaCelular`, para isto usaremos o método `append` da lista.

```
listaNome.append(nome)
listaCelular.append(celular)

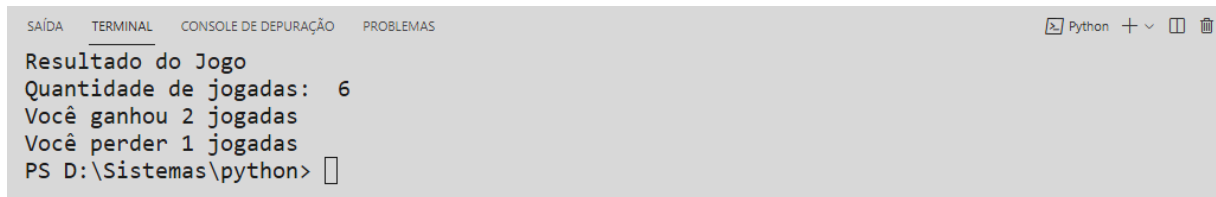
```

5. Pronto agora o loop vai permitir que você fique cadastrando quantos nomes e celulares quiser, até apertar x para parar.

6. Ao final use um print para mostrar os nomes e celulares cadastrados, neste caso use um for / range, o comando len(listaNome) conta quantos itens foram cadastrados.

```
for i in range(len(listaNome)):  
    print('Nome: ', listaNome[i], ' - Celular:' , listaCelular[i])
```

Atividade Extra – Contador de Jogadas



The screenshot shows a terminal window with tabs for SAÍDA, TERMINAL, CONSOLE DE DEPURACÃO, and PROBLEMAS. The output in the terminal is as follows:

```
Resultado do Jogo  
Quantidade de jogadas: 6  
Você ganhou 2 jogadas  
Você perder 1 jogadas  
PS D:\Sistemas\python>
```

Objetivo: Para executar esta atividade será necessário que a Atividade 2 (Pedra, Papel e Tesoura) esteja funcionando.

Comandos utilizados: Criação de Contador.

1. Abra um novo arquivo e salve com o nome `cap05_extra.py`.
2. Coloque um contador para contar as jogadas realizadas e mais dois contadores para contar quantas vezes você ganhou e quantas perdeu.

CAPÍTULO 6 – Funções, Tratamento de Erro e Módulos

Neste capítulo você vai criar funções usando o comando `def` dentro um código, o recurso de criar funções é muito usado em programação para reutilizar um código sempre que for necessário não precisando digitá-lo novamente. Outro recurso que irá estudar é o tratamento de erro usando o `try / except`, o tratamento de erros no Python é realizado pelas exceções que podem ser tratadas de uma maneira especial ou de forma genérica. Além disto aprenderá a usar funções dos módulos `math` e `datetime` do Python, o primeiro contém diversas funções matemáticas e o segundo permite a manipulação de variáveis no formato de data e hora.

Objetivos:

- Criar funções.
- Utilizar argumentos nas funções.
- Tratar erros no código usando `try except`.
- Usar módulos (bibliotecas) do Python com suas respectivas funções.
- Usar o módulo `math` e `datetime` com as suas funções.

Tópicos do capítulo

- Criação de funções.
- Tratamento de erros.
- Módulo `math`.
- Módulo `datetime`.

Atividade 1 – Criação de Funções



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
Python + v [ ] [ ]

PS D:\Sistemas\python> & C:/Users/laerc/AppData/Local/Programs/Python/Python310/python.exe
d:/Sistemas/python/cap06_atividade01.py
Primeiro numero: 14
Segundo numero: 8
1. Somar
2. Subtrair
3. Multiplicação
4. Divisão
Opção: 3
Multiplicacao:  112.0

Aperte 0 para Sair ou Enter para continuar
```

Objetivo: Nesta atividade você vai aprender a criar funções do Python, as funções são muito uteis para que você reaproveite código.

Comandos utilizados: Funções def, while

def Def função(argumentos)
A criação de uma função é realizada pelo comando def, a função pode ter um retorno através do comando return ou simplesmente através de um print. A função pode ter um ou mais argumentos indicados dentro dos parênteses.

7. Crie um arquivo e salve com o nome cap06_atividade01.py.
8. Para esta atividade você vai aprender a criar funções no Python, a função é utilizada principalmente quando você precisa reutilizar um mesmo código várias vezes.
9. Neste exemplo você irá criar uma função para as operações básicas de uma calculadora, soma, subtração, multiplicação e divisão.
10. Para criar a função utilize o comando def e o nome da função que quer usar, entre parênteses você coloca os argumentos que serão necessários, neste caso das operações básicas matemáticas precisaremos de dois argumentos, segue abaixo o código para criar a função soma e realizar um print da soma dos números.

```
def soma(x, y):  
    print("Soma: ", x+y)
```

11. Faça a mesma coisa para as demais funções de subtração, multiplicação e divisão.

```
def subtracao(x, y):  
    print("Subtracao: ", x-y)
```

```
def multiplicacao(x, y):  
    print("Multilicence: ", x*y)
```

```
def divisao(x, y):  
    print("Divisao: ", x/y)
```

12. Com as funções criadas podemos utilizá-las, crie um loop para que ao final o usuário possa iniciar a calculadora.

```
opcao=1  
while opcao:
```

13. Peça ao usuário primeiramente os dois números para fazer os cálculos matemáticos.

```
x = float(input("Primeiro número: "))  
y = float(input("Segundo número: "))
```

14. Informe as opções de cálculo matemático disponível usando um print e peça ao usuário qual operação deseja realizar.

```
print("1. Somar")  
print("2. Subtrair")  
print("3. Multiplicação")  
print("4. Divisão ")
```

15. Peça para o usuário qual cálculo deseja realizar.

```
operador = int(input("Opção: "))
```

16. Com o operador matemático escolhido utilize o comando `if` para executar a função correta, o `print` no terminal será realizado diretamente pela função.

```
if(operador==1):
    soma(x, y)
if(operador==2):
    subtracao(x, y)
if(operador==3):
    multiplicacao(x, y)
if(operador==4):
    divisao(x, y)
```

17. Peça para o usuário apertar 0 para finalizar ou qualquer tecla para continuar.

```
opcao = input("\nAperte 0 para Sair ou Enter para continuar")
if opcao=="0":
    opcao=int(opcao)
```

Atividade 2 – Tratamento de erros



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
PS D:\Sistemas\python> & C:/Users/laerc/AppData/Local/Programs/Python/Python310/python.exe
d:/Sistemas/python/cap06_atividade02.py
Primeiro numero: 10
Segundo numero: 0
1. Somar
2. Subtrair
3. Multiplicação
4. Divisão
Opção: 4
float division by zero

Aperte 0 para Sair ou Enter para continuar
```

Objetivo: Nesta atividade você vai criar a utilizar o tratamento de erro do Python, irá reutilizar o código da atividade anterior para fazer o tratamento de erros dentro das funções.

Comandos utilizados: Tratamento de erro Try / Except

Try / except

Na linguagem de programação erros podem acontecer e devem ser tratados, no Python os erros são tratados pelo comando Try / except.

| Exceção | Causa do erro |
|---------------|--|
| ImportError | Gerado quando o módulo importado não é encontrado. |
| IndexError | Gerado quando o índice de uma sequência está fora do intervalo. |
| MemoryError | Gerado quando uma operação fica sem memória. |
| OverflowError | Gerado quando o resultado de uma operação aritmética é muito grande para ser representado. |
| RuntimeError | Gerado quando um erro não se enquadra em nenhuma outra categoria. |

| | |
|--------------------------------|---|
| <code>TypeError</code> | Gerado quando uma função ou operação é aplicada a um objeto de tipo incorreto. |
| <code>ValueError</code> | Gerado quando uma função obtém um argumento do tipo correto, mas valor impróprio. |
| <code>ZeroDivisionError</code> | Gerado quando o segundo operando de divisão ou operação de módulo é zero. |

1. Utilize o código da atividade anterior (cap05_atividade01.py), nela você irá implementar o tratamento de erro na função divisão.
2. O tratamento de erro é colocado, normalmente, em situações que sabemos que um erro pode acontecer, no caso da divisão seria a divisão por zero.
3. O `try / except` é o comando utilizado para fazer o tratamento de erro, dentro do `try` colocamos os comandos que podem dar erro no `except` fazemos o tratamento do erro, conforme código abaixo.

```
def divisao(x, y):
    try:
        print("Divisao: ", float(x)/float(y))
    except:
        print("Ocorreu um erro")
```

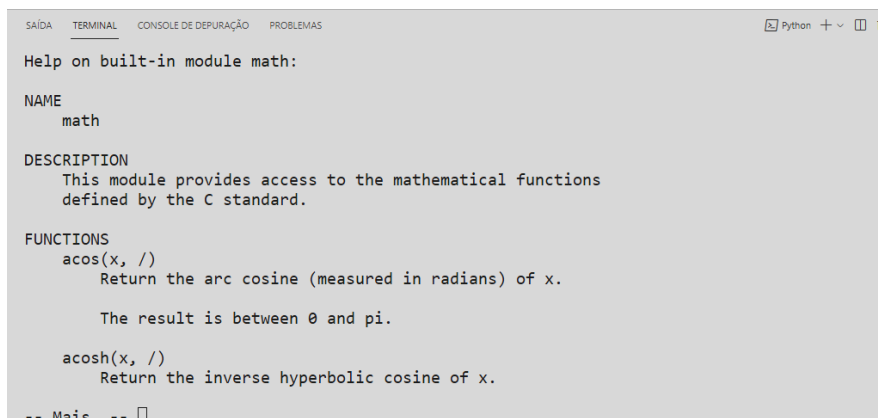
4. Foi usado um `except` geral, todos os erros irão passar por este `except`, que pode ser uma divisão por zero ou um valor de argumento que não seja numérico.
5. Agora você irá colocar um tratamento de erro específico para a divisão por Zero, que é o `except ZeroDivisionError`, a variável `erro` irá retornar a mensagem de erro em inglês de divisão por zero.

```
except ZeroDivisionError as erro:
    print(erro)
```

6. Faça o tratamento de erro para as demais funções, nos outros operadores não é necessário colocar a divisão por zero.

```
def soma(x, y):
    try:
        print("Soma: ", float(x)+float(y))
    except:
        print("Ocorreu um erro")
def subtracao(x, y):
    try:
        print("Subtracao: ", float(x)-float(y))
    except:
        print("Ocorreu um erro")
def multiplicacao(x, y):
    try:
        print("Multiplicacao: ", float(x)*float(y))
    except:
        print("Ocorreu um erro")
```

Atividade 3 – Módulo Math



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
Python  +  -  [ ]  [x]

Help on built-in module math:

NAME
  math

DESCRIPTION
  This module provides access to the mathematical functions
  defined by the C standard.

FUNCTIONS
  acos(x, /)
    Return the arc cosine (measured in radians) of x.

    The result is between 0 and pi.

  acosh(x, /)
    Return the inverse hyperbolic cosine of x.

-- Mais --
```

Objetivo: Nesta atividade você vai importar o módulo (bibliotecas) math, visualizar a ajuda deste módulo e das funções e aprender a usar algumas das principais funções.

Comandos utilizados: Módulo math, funções ceil, fabs, floor, fmod.

Math A biblioteca Math possui os operadores matemáticos mais utilizados. Use o comando `help(math)` para visualizar a ajuda desta biblioteca.

1. Abra um novo arquivo e salve como `cap06_atividade03.py`.
2. Para importar o módulo math utilize o comando `import`.

```
import math
```

3. Para visualizar a ajuda no módulo math é possível usar o comando `help`.

```
help(math)
```

4. Execute o código e será listado no terminal uma ajuda de todas as funções do módulo math.
5. Agora você utilizará alguns das principais funções, o primeiro será o `math.ceil()`, esta função retorna a parte inteira de um número.

```
print(math.ceil(4.2))
```

6. O `math.floor()` retorna o valor arredondado para cima.

```
print(math.floor(3.9))
```

7. O `math.fabs()` retorna o valor absoluto de um número (sem o símbolo de negativo).

```
print(math.fabs(-1))
```

8. O `math.fmod()` retorna o resto de uma divisão, mesma coisa que o operador %.

```
print(math.fmod(9, 4))
```

9. O `math.sqrt()` calcula a raiz quadrada de um número.

```
print(math.sqrt(36))
```

10. Para o arredondamento de um float não é necessário usar a biblioteca math, use o comando `round`.

```
print(round(3.988, 1))
```

Atividade 4 – Função de Datas

```
SAÍDA  TERMINAL  CONSOLE DE DEPUÇÃO  PROBLEMAS
Python + v  T

PS D:\Sistemas\python> & C:/Users/laerc/AppData/Local/Programs/Python/Python310/python.exe
d:/Sistemas/python/cap06_atividade04.py
2021-10-17 11:47:54.535737
<class 'datetime.datetime'>
<class 'datetime.datetime'>
17/10/2021 11:47
Informe a sua data de nascimento dd/mm/aaaa: 12/02/1975
Você tem: 46 anos
Você já viveu 17050 dias
PS D:\Sistemas\python> 
```

Objetivo: Nesta atividade você vai aprender a usar a biblioteca datetime do Python, com ela é possível manipular datas e converter uma string em data. E irá implementar um return numa função.

Comandos utilizados: Módulo datetime, métodos strptime, strftime e today. Função com return.

1. Abra um novo arquivo e salve com o nome cap06_atividade04.py.
2. Importe o módulo datetime usando a classe datetime.

```
from datetime import datetime
```

3. Primeiramente você irá imprimir no terminal a data e hora atual na tela para conhecer a função today().

```
print(datetime.now())
```

4. Agora guarde o valor da data e hora atual em uma variável e imprima no terminal o tipo desta variável.

```
atual = datetime.now()
print(type(atual))
```

5. Agora teste a criação de uma data a partir da classe datetime informando o ano, mês e dia nesta sequência e imprima no terminal o tipo da variável para conferir o tipo da variável.

```
montarData = datetime(2021, 10, 5)
print(type(montarData))
```

6. Para formatar uma variável do tipo datetime você pode usar a função strftime indicando a formatação desejada, neste caso usaremos o formato dia/mês/ano.

```
print(atual.strftime('%d/%m/%Y %H:%M'))
```

7. Peça para o usuário uma data no formato dia/mês/ano e guarde em uma variável.

```
dataNascimento = input('Informe a sua data de nascimento dd/mm/aaaa: ')
```

8. Agora você irá precisar transformar esta data que é uma string em uma variável datetime usando a função strptime.

```
dataNascimento = datetime.strptime(dataNascimento, '%d/%m/%Y')
```

9. Observe que o formato da string tem que ter o mesmo formato indicado na função `strptime`.
10. Agora você irá criar uma função para calcular a idade a partir da data informada, lembre-se que as funções devem ser criadas sempre no início do código, abaixo das declarações dos módulos.
11. Não existe uma função direta para calcular a idade então é necessário fazer da forma mostrada acima. O comando `return` é utilizado para retornar um valor da função, sempre prefira usar o `return` em vez do `print` dentro de uma função.

```
def idade(nascimento):
    today = datetime.today()
    return today.year - nascimento.year - ((today.month, today.day) <
(nascimento.month, nascimento.day))
```

12. Para usar esta função, na parte final do código coloque o seguinte comando.

```
print(f'Você tem: {idade(dataNascimento)} anos')
```

13. Agora crie uma outra função para mostrar quantos dias a pessoa já viveu. Para calcular dias corridos basta subtrair uma data de outra.

```
def dias(nascimento):
    today = datetime.today()
    return abs((nascimento - today).days)
```

14. E para usar esta função, novamente digite na parte final do código.

```
print(f'Você já viveu {dias(dataNascimento)} dias')
```

Atividade Extra – Acertando o Símbolo

```
SAÍDA  TERMINAL  CONSOLE DE DEPUAÇÃO  PROBLEMAS
Python  +  ▢  🗑

*****VOU ACERTAR O SIMBOLO QUE VC PENSAR*****
 1 ▶ | 2 !! | 3 ⚡ | 4 ♪ | 5 - | 6 ⚡ | 7 !! | 8 ▶ | 9 § | 10 ♪ |
11 ⚡ | 12 ♪ | 13 ◀ | 14 ◀ | 15 - | 16 ▶ | 17 ↑ | 18 § | 19 ◀ | 20 ⚡ |
21 ▶ | 22 ♪ | 23 ♪ | 24 ⚡ | 25 ♪ | 26 !! | 27 § | 28 ↑ | 29 ⚡ | 30 - |
31 ♪ | 32 ♪ | 33 !! | 34 ♪ | 35 ♪ | 36 § | 37 !! | 38 ♪ | 39 !! | 40 ↑ |
41 ▶ | 42 - | 43 ▶ | 44 ⚡ | 45 § | 46 ♪ | 47 - | 48 ◀ | 49 ⚡ | 50 ♪ |
51 ⚡ | 52 - | 53 ◀ | 54 § | 55 !! | 56 ♪ | 57 !! | 58 ⚡ | 59 ♪ | 60 - |
61 ♪ | 62 ⚡ | 63 § | 64 ⚡ | 65 !! | 66 ♪ | 67 ♪ | 68 ⚡ | 69 ↑ | 70 !! |
71 ◀ | 72 § | 73 !! | 74 ↑ | 75 ⚡ | 76 ♪ | 77 ⚡ | 78 - | 79 ⚡ | 80 ⚡ |
81 § | 82 ▶ | 83 ↑ | 84 - | 85 !! | 86 ◀ | 87 ⚡ | 88 - | 89 ⚡ | 90 § |
91 !! | 92 - | 93 ▶ | 94 !! | 95 - | 96 ▶ | 97 ▶ | 98 ⚡ | 99 § | 100 - |
*****
Pense em um numero de 11 a 99 e aperte Enter 13
Faça a soma dos digitos, exemplo o numero foi 22 = 2+2 e aperte Enter
Agora faça o numero menos a soma deles (exemplo 22 - 4 = 18) e aperte Enter
O simbolo é §
Opa o meu exemplo também deu certo!!
```

Objetivo: Nesta atividade você vai criar um jogo onde irá acertar um símbolo através de um cálculo matemático. Você vai usar todos os conceitos visto até aqui e basicamente será necessário criar uma lista como fizemos na atividade de tabuada.

Comandos utilizados: Variáveis, for, while e chr.

chr

O comando chr usa a tabela ascii para mostrar os caracteres especiais, no exemplo acima os caracteres usados foram do 16 ao 25, em alguns terminais estes caracteres podem não ser impressos corretamente, se não funcionar use a sequência de 179 a 190

1. *A lógica deste jogo é que será pedido para que o usuário pense em um número de 11 a 99 e aí peça para o usuário somar os dois dígitos do número e subtrair este resultado do número inicial.*
2. *A grande sacada deste cálculo é que sempre ele será múltiplo de 9, ou seja, para que o mistério do jogo de certo todos os múltiplos de 9 devem ter o mesmo símbolo.*
3. *Para mostrar símbolos diferentes na tela use o comando chr, procure pela tabela ASCII para mostrar estes símbolos, uma dica é usar o range de 16 a 25 da tabela.*
4. *Use o for para montar a tabela conforme a figura acima e garanta que os múltiplos de 9 sempre terão o mesmo símbolo.*

CAPÍTULO 7 – Criando uma classe

Neste capítulo você irá ver as informações básicas para a criação de Classes, uma classe permite uma forma de organizar dados e funcionalidades. Criar uma classe cria um tipo de objeto novo, permitindo que ele seja reaproveitamento inúmeras vezes. As classes podem ter métodos e propriedades.

Objetivos:

- Entender o que é uma classe.
- Criar uma classe.
- Criar métodos e propriedades dentro de uma classe.
- Consumir esta classe.

Tópicos do capítulo

- Criar a classe.
- Consumir a classe.

Atividade 1 – Criando uma classe



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
Python + - [ ] [X]

PS D:\Sistemas\python> & C:/Users/laerc/AppData/Local/Programs/Python/Python310/python.exe d:/Sis
temas/python/cap07_atividade01Consumir.py
desenvolvimento de sistemas em Python
*****Recibo*****

Recebemos de Laercio Azevedo de Sa a quantia de R$ 51.13 (cinquenta e um reais e treze centavos))

Referente desenvolvimento de sistemas em Python
PS D:\Sistemas\python> [ ]
```

Objetivo: Nesta atividade você vai criar uma classe com propriedades simples do Python. A Classe será um recibo onde é passado as informações de nome e valor e será retornado um recibo com estas informações.

Comandos utilizados: Class

| | |
|-------|---|
| class | A classe pode ter variáveis, funções que viram métodos ou propriedades, a classe precisa ser instanciada para ser usada, o Python usa todas as características mais modernas da linguagem de programação para a criação de classe. Nesta atividade veremos as informações básicas de uma classe |
|-------|---|

1. Para a trabalhar com classe você vai precisar de dois arquivos do Python, uma com a classe propriamente e outro arquivo para consumir esta classe.
2. Abra dois novos arquivos e salve eles com o nome de `cap07_atividade01Classe.py` e o outro com o nome `cap07_atividade01.Consumir.py`.
3. No arquivo `CLASSE` precisaremos iniciar a classe com o comando `class` e o nome da classe:

```
class recibo:
```

4. Você vai precisar indicar os valores iniciais da instancia da classe, para isto temos o construtor `__init__`, nesta classe você vai pedir inicialmente o nome da pessoa que aparecerá o recibo.

```
def __init__(self):
```

5. Todas as funções de uma classe têm como o primeiro argumento `self`, neste caso você irá usar um segundo argumento com a variável `nome`.

```
def __init__(self, nome):
    self.nome = nome
```

6. Crie uma função com o nome `descricao` e que pede um argumento.

```
def descricao(self, value):
    self._descricao = value
```

7. Para o valor do recibo você vai usar a construção de uma propriedade, neste caso precisaremos de duas funções uma para passar os dados e outra para retornar os dados.
8. Para retornar os dados é necessário identificar a função com o comando `@property`, conforme exemplo abaixo:

```
@property
def valor(self):
    return(self._valor)
```

9. Para passar os dados será criada outra função identificada com o comando `@valor.setter`.

```
@valor.setter
def valor(self, value):
    self._valor = value
```

10. Você vai criar também uma função que retornará o valor informado em extenso e para isto usaremos a biblioteca **`num2words`**.
11. É necessário que você instale esta biblioteca primeiramente indo em um terminal e digitando.
`pip install num2words`
12. Agora indique a biblioteca no início do código, antes da declaração `class`.

```
from num2words import num2words
```

13. Para a função do valor por extenso você vai precisar de uma nova função, conforme o código abaixo:

```
def extenso(self):
    vExtenso = num2words(self._valor, lang='pt_BR', to='currency')
    return vExtenso
```

14. A classe está quase pronta, para que finalize este código você vai criar um retorno da classe usando o construtor padrão `__str__` nele iremos imprimir um recibo com os dados informados para a classe:

```

def __str__(self):
    texto = 'Recebemos de {} a quantia de R$ {:.2f} ({}))' .
format(self.nome, self._valor, self.extenso())
    descricao = '\nReferente {}' . format(self._descricao) if
(self._descricao!='') else ''
    dados = '{}\n{}{}' . format('Recibo'.center(len(texto), '*'), texto,
descricao)
    return(dados)

```

15. Agora podemos iniciar os códigos do outro arquivo (CONSUMIR) para que possamos utilizar esta classe.

16. Primeiramente crie uma referência para o arquivo que está a classe.

```

from cap07_atividade01Classe import recibo

```

17. Para instanciar a classe use uma variável e passe o nome da pessoa que irá receber o recibo.

```

dados = recibo('Laercio Azevedo de Sa')

```

Quando é instanciada a classe as informações do construtor __init__ são utilizados.

18. Agora passe o valor do recibo através da propriedade valor.

```

dados.valor = 51.13

```

19. Passe a descrição que aparecerá no recibo usando o método descricao().

```

dados.descricao('desenvolvimento de sistemas em Python')

```

20. Para a descrição não foi criado um valor de retorno, mas é possível visualizar a valor informado para a descricao() usando a variável que recebeu o valor (_descricao).

```

print(dados._descricao)

```

21. E para mostrar no terminal o recibo pronto basta imprimir a classe com o comando print, será usado o construtor __str__.

```

print(dados)

```


CAPÍTULO 8 – Trabalhando com arquivos de texto

Neste capítulo você vai aprender a manipular arquivos do tipo texto/csv para poder ler, editar e criar. Você irá trabalhar com os dicionários do Python e Classes para simular um banco de dados.

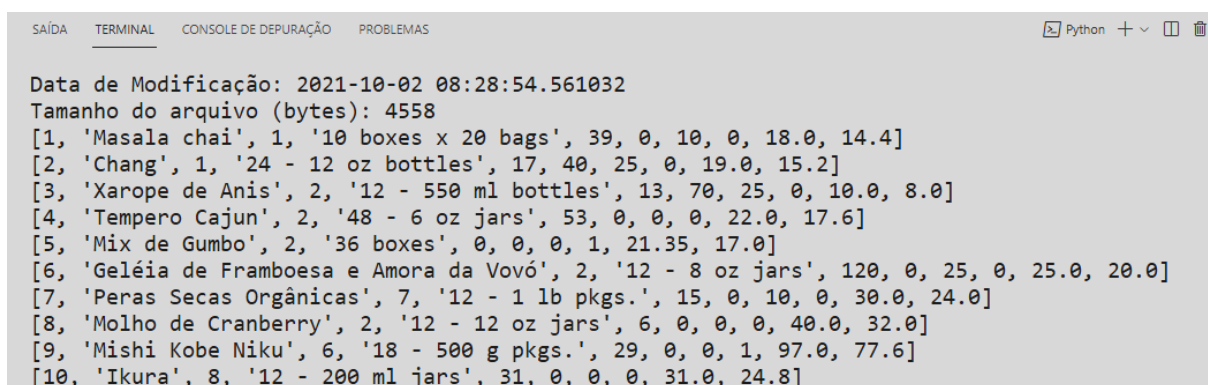
Objetivos:

- Ler arquivos texto, definir codificação do arquivo.
- Transformar um arquivo csv em uma lista.
- Criar arquivos em txt.
- Utilizar dicionários para simular uma base de dados relacional.

Tópicos do capítulo

- Abrir arquivos.
- Criar dicionários.
- Trabalhar com classes.
- Criar relatórios em formato texto.

Atividade 1 – Abrindo um arquivo csv



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS  Python + ~  [ ] [X]
Data de Modificação: 2021-10-02 08:28:54.561032
Tamanho do arquivo (bytes): 4558
[1, 'Masala chai', 1, '10 boxes x 20 bags', 39, 0, 10, 0, 18.0, 14.4]
[2, 'Chang', 1, '24 - 12 oz bottles', 17, 40, 25, 0, 19.0, 15.2]
[3, 'Xarope de Anis', 2, '12 - 550 ml bottles', 13, 70, 25, 0, 10.0, 8.0]
[4, 'Tempero Cajun', 2, '48 - 6 oz jars', 53, 0, 0, 0, 22.0, 17.6]
[5, 'Mix de Gumbo', 2, '36 boxes', 0, 0, 0, 1, 21.35, 17.0]
[6, 'Geléia de Framboesa e Amora da Vovó', 2, '12 - 8 oz jars', 120, 0, 25, 0, 25.0, 20.0]
[7, 'Peras Secas Orgânicas', 7, '12 - 1 lb pkgs.', 15, 0, 10, 0, 30.0, 24.0]
[8, 'Molho de Cranberry', 2, '12 - 12 oz jars', 6, 0, 0, 0, 40.0, 32.0]
[9, 'Mishi Kobe Niku', 6, '18 - 500 g pkgs.', 29, 0, 0, 1, 97.0, 77.6]
[10, 'Ikura', 8, '12 - 200 ml jars', 31, 0, 0, 0, 31.0, 24.8]
```

Objetivo: Nesta atividade você vai ler um arquivo no formato CSV, verificar as opções de encoding (codificação de caracteres) e separar o arquivo em uma lista para ler as informações de cada coluna. Para simplesmente ler um arquivo no formato de texto, que pode ser csv e txt não é necessário de nenhuma biblioteca, para arquivos PDF, Excel, Word são necessárias bibliotecas para fazer esta leitura e para arquivos no

formato JSON, XML e HTML é possível ler os arquivos como texto, mas para explorar os todos os recursos é recomendável usar alguma biblioteca para facilitar o trabalho de leitura.

Comandos utilizados: Biblioteca `os.path`, comando `open` mode “r”, `lista` e `for`.

| | |
|----------------------------------|---|
| <code>Os.path</code> | A biblioteca <code>os.path</code> permite pegar dados de um arquivo, alguns dos métodos que usaremos são: |
| <code>Os.path.isfile(x)</code> | Verifica se o arquivo existe. |
| <code>Os.path.isdir(x)</code> | Verifica se o diretório existe. |
| <code>os.path.getsize(x)</code> | Retorna o tamanho do arquivo |
| <code>os.path.getmtime(x)</code> | Retorna a data de modificação do arquivo |
| <code>os.listdir(path)</code> | Lista os arquivos de uma determinada pasta |
| <code>os.remove(x)</code> | Apaga um arquivo |
| Open mode “r” | Para abrir um arquivo no Python use o comando <code>open</code> e para ler este arquivo use o mode “r” |
| Diretório | Para descobrir o diretório do código atual: <code>os.path.dirname(os.path.realpath(__file__))</code> Para descobrir o diretório da pasta que executou o código atual: <code>os.getcwd()</code> |

1. Crie um arquivo chamado `cap08_atividade01.py`.
2. Importe a biblioteca `os.path` e `datetime` e “`os`”.

```
import os.path, datetime
from os import system, name
system('cls') if(name == 'nt') else system('clear')
```

Você vai precisar do arquivo `produtos.csv` (no capítulo Anexos você tem o conteúdo deste arquivo), declare uma variável com o nome do arquivo.

3. Verifique o arquivo está na mesma pasta que o arquivo `.py`.

```
if (os.path.isfile(arquivo)):
```

4. Abra o arquivo em mode ‘r’ e neste caso o arquivo está com codificação de caracteres UTF-8, é importante definir isto para não ter problemas com acentuação.

```
produtos = open(arquivo, 'r', encoding="utf-8")
```

5. Declare uma variável para guardar o tamanho do arquivo usando a método `getsize()`.

```
tamanho = os.path.getsize(arquivo)
```

6. Declare uma variável para guardar a data e hora da última modificação do arquivo usando o método `getmtime`. A data vem no formato `timestamp`.

```
modificacao = os.path.getmtime(arquivo)
```

7. Imprima no terminal estas informações, será necessário formatar a data usando o método `fromtimestamp` da biblioteca `datetime`.

```
print('Data de Modificação:',
datetime.datetime.fromtimestamp(modificacao))
print('Tamanho do arquivo (bytes):', tamanho)
```

8. Agora crie uma lista chamada `listaProdutos`.

```
listaProdutos = []
```

9. Para ler o arquivo csv você pode usar o comando `for`, o arquivo será quebrado por linha.

```
for line in produtos:
```

10. Para quebrar as colunas do arquivo csv podemos usar o comando `split`.

```
colunas = line.strip().split(";")
```

As colunas do arquivo são:

Código do Produto;

Nome do Produto;

Categoria do Produto;

Tipo de Embalagem;

Estoque;

Quantidade para separação de pedido;

Produto Inativo (0 ativo e 1 inativo);

Estoque mínimo;

Valor Venda;

Valor de Custo.

11. Para este caso esperamos no arquivo 10 colunas (de 0 a 9) será necessário converter algumas colunas pois no `split` todas as colunas ficam como `str`, as colunas 0, 2, 3, 4, 5 e 6 são do tipo `Int` e as colunas 8 e 9 do tipo `float`.

```
colunas[0]=int(colunas[0])
```

```
colunas[2]=int(colunas[2])
```

```
colunas[4]=int(colunas[4])
```

```
colunas[5]=int(colunas[5])
```

```
colunas[6]=int(colunas[6])
```

```
colunas[7]=int(colunas[7])
```

```
colunas[8]=float(colunas[8])
```

```
colunas[9]=float(colunas[9])
```

As colunas 1 e 3 são do tipo `str` e não precisam que sejam convertidas em algum tipo específico, para serem utilizadas da maneira correta.

12. Use o comando `append` para acrescentar os dados na lista `listaProdutos`.

```
listaProdutos.append(colunas)
```

13. Sempre que abrir um arquivo é importante usar o comando `close()` para fecha-lo.

```
produtos.close()
```

14. Para verificar se o arquivo está fechado é possível usar o comando `nome_arquivo.closed`, este comando retorna `true` ou `false`.

15. Para finalizar faça uma impressão dos dados da lista usando um `for`.

```
for prod in listaProdutos:
```

```
    print(prod)
```

Atividade 2 – Criando um dicionário a partir de um csv

```
SAÍDA  TERMINAL  CONSOLE DE DEPUÇÃO  PROBLEMAS
Python  + -  [ ]  [ ]
{'1': 'Bebidas', '2': 'Condimentos', '3': 'Confeitaria', '4': 'Lácteos', '5': 'Grãos / Cereais', '6': 'Carnes / Aves', '7': 'Processados', '8': 'Frutos do mar'}
PS D:\Sistemas\python> [ ]
```

Objetivo: Nesta atividade você vai criar um dicionário a partir de um arquivo csv, com o dicionário é possível simular uma base de dados e pesquisando dentro do dicionário a partir de um código chave.

Comandos utilizados: Dicionário.

Dicionário Para criar um dicionário no Python use {}.

1. Crie um arquivo com o nome `cap08_atividade02.csv`.
2. Crie uma variável para guardar o nome do arquivo que será aberto, nesta atividade você vai usar o arquivo `categoria.csv`.

```
arquivo = 'categorias.csv'
```

3. Abra o arquivo com a opção de encoding `utf-8`.

```
categorias = open(arquivo, 'r', encoding="utf-8")
```

4. Instancie uma variável para o dicionário de dados, para criar dicionários use os símbolos {}.

```
dicCategoria = {}
```

5. Use o comando `for` para ler o arquivo `categorias`. O arquivo possui 3 informações que são código da categoria, nome da categoria e descrição da categoria.

```
dicCategoria = {}
```

6. Use o comando `strip` para separar a informação pelo “;”.

```
colunas = line.strip().split(";")
```

7. Para criar um dicionário é necessário definir um valor de chave e um valor para os dados, use o código para a chave e crie uma lista com as informações do nome e descrição.

```
colunas = line.strip().split(";")
```

```
dados = [colunas[1], colunas[2]]
```

```
dicCategoria[colunas[0]] = dados
```

8. Feche o arquivo `categoria.csv`.

```
dicCategoria = {}
```

9. Imprima todos os dados do dicionário.

```
print(dicCategoria)
```

10. Imprima apenas as informações da categoria 3.

```
print(dicCategoria['3'])
```

11. Imprima apenas a descrição da categoria 3.

```
print(dicCategoria['3'][1])
```

Atividade 3 – Criando um Relatório em csv



```
relatorio.txt
1 ***** Categoria *****
2 BEBIDAS
3 Refrigerantes, cafés, chás, cervejas e cervejas
4
5 Produtos
6 1. Café branco ipoh 16 - 500 g tins | R$ 46.00
7 2. Cerveja alemã 24 - 0.5 l bottles | R$ 7.75
8 3. Cerveja australiana lager 24 - 355 ml bottles | R$ 15.00
9 4. Cerveja lager 24 - 12 oz bottles | R$ 14.00
10 5. Cerveja tipo ale 24 - 12 oz bottles | R$ 14.00
11 6. Cerveja tipo stout 24 - 12 oz bottles | R$ 18.00
12 7. Chang 24 - 12 oz bottles | R$ 19.00
13 8. Guaraná fantástica 12 - 355 ml cans | R$ 4.50
14 9. Licor de amora finlandes 500 ml | R$ 18.00
15 10. Licor francês verde 750 cc per bottle | R$ 18.00
16 11. Masala chai 10 boxes x 20 bags | R$ 18.00
17 12. Vinho de blaye 12 - 75 cl bottles | R$ 263.50
18
19
20 ***** Categoria *****
21 CONDIMENTOS
22 Molhos doces e salgados, condimentos, patês e temperos
23
```

Objetivo: Nesta atividade você vai criar um relatório texto com os dados dos arquivos de categoria e produtos, para fazer isto primeiramente vai transformar os exemplos das atividades 1 e 2 em uma classe e a partir daí consumir estas classes para gerar o relatório.

Comandos utilizados: open mode “w”.

Open mode “w” Para criar um arquivo no Python use o comando open com a opção mode “w”, assim o arquivo será criado.

1. Crie um primeiro arquivo com o nome `cap08_atividade03Classe.py`.
2. Neste arquivo você vai criar duas classes para serem consumidas no código que irá gerar o relatório.
3. A primeira classe do arquivo vai se chamar `tabCat`.

`class tabCat:`

4. Crie uma função com o nome de `dicCat`, esta função irá retornar um dicionário.

`def dicCat(self):`

5. Esta função irá usar o mesmo código da atividade02 do capítulo 08.

```
arquivo = 'categorias.csv'
categorias = open(arquivo, 'r', encoding="utf-8")
dicCategoria = {}
for line in categorias:
    colunas = line.strip().split(";")
    dados = [colunas[1], colunas[2]]
    dicCategoria[colunas[0]] = dados
categorias.close()
```

6. No final do código não utilize os prints, pode apagar eles e substituir por um return dicCategoria.

```
return dicCategoria
```

7. Agora você irá criar a segunda classe deste arquivo, defina o nome da classe como tabProd.

```
class tabProd:
```

8. Crie uma função com o nome de listProd, esta função irá retornar uma lista.

```
def listProd(self):
```

9. Esta função irá usar o mesmo código da atividade01 do capítulo 08.

```
arquivo = 'produtos.csv'
produtos = open(arquivo, 'r', encoding="utf-8")
listaProdutos = []
for line in produtos:
    colunas = line.strip().split(";")
    colunas[0]=int(colunas[0])
    colunas[2]=int(colunas[2])
    colunas[4]=int(colunas[4])
    colunas[5]=int(colunas[5])
    colunas[6]=int(colunas[6])
    colunas[7]=int(colunas[7])
    colunas[8]=float(colunas[8])
    colunas[9]=float(colunas[9])
    listaProdutos.append(colunas)
produtos.close()
```

10. No final do código não utilize os prints, pode apagar eles e substituir por um return listaProdutos.

```
return listaProdutos
```

CAPÍTULO 9 – Extração de dados na Web

Neste capítulo você vai aprender a fazer uma extração de dados na web (web scraping) no Python, este recurso ficou disponível a partir da versão 3. Será necessário baixar as bibliotecas de web scraping, as que você vai usar são: a BeautifulSoup e a biblioteca “**requests**”. Outra biblioteca que usará será a Selenium, que nada mais é, do que uma biblioteca que permite com que o Python abra o seu navegador para executar os comandos desejados e usará em conjunto dela o Chromedriver para ter total controle do browser Chrome.

Objetivos:

- Entender o que é e como fazer uma extração de dados.
- Instalar as principais bibliotecas para extração de dados.
- Extrair partes de um site.
- Realizar download de arquivos de um site.
- Utilizar o Selenium para manipular o Chrome.

Tópicos do capítulo

- Extrair dados.
- Realizar Download de arquivos.
- Controlar o Chrome e enviar mensagens pelo WhatsApp Web.

Atividade Inicial – Instalando as Bibliotecas

Objetivo: Nesta primeira atividade realizaremos a instalação das bibliotecas necessárias para a extração de dados.

Comandos utilizados: pip install.

1. Abra um novo terminal do sistema operacional (cmd ou powershell no windows).
2. Para a instalação do “**BeautifulSoup**” digite.
python -m pip install beautifulsoup4
3. Para a instalação do “**requests**” digite.
python -m pip install requests
4. Para a instalação do Selenium digite.
python -m pip install selenium

Atividade 1 – Extração de dados com BeautifulSoup – Site UOL

```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS
1
[<section class="currencies"></section>]
1
[<div class="info"><span class="name">IPCA</span><div class="numbers"><span class="data po
s">0,87</span> <span class="value">Ago.2021</span></div></div>]
0
QTD Class: 'subtituloGrafico subtituloGraficoValor': 4
R$ 5,455
R$ 5,455
R$6,329
R$ 6,329
PS D:\Sistemas\python> □
```

Objetivo: Nesta atividade você irá extrair dados da cotação de Dólar e Euro do site do UOL, você vai verificar que algumas vezes o conteúdo da página muda um pouco em relação ao que o “**BeautifulSoup**” acaba lendo.

Comandos utilizados: Biblioteca “**requests**” e “**BeautifulSoup**”.

1. Crie um arquivo com o nome `cap09_atividade01.py`.
2. As informações que você vai pegar no site são estas: Dólar. e Euro.

| DÓLAR COM. | PESO | EURO |
|------------------|------------------|------------------|
| -1,11% R\$ 5,455 | -1,20% R\$ 0,055 | -1,05% R\$ 6,329 |

3. Aqui temos o código HTML da área que ficam as informações de cotação.

```
<section class="currencies">
  <div class="info">
    <a href="https://economia.uol.com.br/cotacoes/cambio/dolar-comercial-estados-unidos/">
      <span class="name"> Dólar com.</span>
      <div class="numbers">
        <span class="data neg">...</span>
        <span class="value bra">
          ::before
          "5,455"
        </span>
      </div>
    </a>
  </div>
  <div class="info">...</div>
  <div class="info">...</div> == $0
  <div class="info hidden-md">...</div>
</section>
```

4. Importe as bibliotecas do “**requests**” e “**bs4**”. Utilize também a biblioteca do “**os**” para limpar o terminal.

```
from requests import get
from bs4 import BeautifulSoup
from os import system, name
system('cls') if(name == 'nt') else system('clear')
```


5. Crie uma variável para guardar o endereço do site que fará a extração de dados, neste caso será usado o site de Economia do UOL.

```
url = 'https://economia.uol.com.br/'
```

6. Agora é necessário passar a informação do endereço do site para a biblioteca do “requests”.

```
response = get(url)
```

7. O “BeautifulSoup” não lê diretamente um endereço apenas o conteúdo em HTML do site, por isto que precisamos usar a biblioteca “requests”, faça a extração dos dados da página com a biblioteca do “bs4”.

```
html_soup = BeautifulSoup(response.text, 'html.parser')
```

8. Nesta página você precisa pegar a informação da cotação do Dólar e do Euro, é necessário ter um conhecimento em HTML para conseguir usar os comandos do “bs4”, primeiramente você irá procurar a seção que tem a class:currencies.

```
secaoDinheiro = html_soup.find_all('section', class_ = 'currencies')
```

9. Faça um print da quantidade de itens que achou e dos dados encontrados.

```
print(len(secaoDinheiro))  
print(secaoDinheiro)
```

10. Observe que foi encontrado uma seção, mas nenhuma informação de texto somente a tag e a classe, então você vai tentar procurar a div que tem a class:info.

```
info = html_soup.find_all('div', class_ = 'info')
```

11. Faça um print da quantidade de itens que achou e dos dados encontrados, use o método text para visualizar apenas o texto sem a tag.

```
print(len(info))  
print(info)  
print(info[0].text)
```

12. Novamente foi retornado apenas uma seção e neste caso foi retornado um texto sobre o IPCA.

13. Ainda não tem as informações dos valores do dólar e do Euro, para uma última tentativa procure os dados da tag span class: value bra.

```
infoValor = html_soup.find_all('span', class_ = 'value bra')
```

14. Imprima a quantidade de itens encontrados.

```
print(len(infoValor))
```

15. Desta forma não foi possível encontra a informação desejada, precisaremos salvar o arquivo HTML e analisar em qual lugar os dados estão.

```
relatorio = open("uol.txt", mode="w", encoding="utf-8")  
relatorio.write(html_soup.prettify())
```

16. Analise o arquivo salvo, verifique os valores de Dólar e Euro estão dentro de uma tag a com a class: 'subtituloGrafico subtituloGraficoValor'. Faça a pesquisa desta classe.

```
valores = html_soup.find_all('a', class_ = 'subtituloGrafico  
subtituloGraficoValor')
```

17. Imprima a quantidade de itens encontrados.

```
print("QTD Class:'subtituloGrafico subtituloGraficoValor':", len(valores))
```

18. Faça um for para mostrar o texto dos itens encontrados.

```
for valor in valores:
```

```
    print(valor.text)
```

19. Pronto agora você conseguiu pegar as informações que desejava. Observe que as vezes pode ser bem trabalhoso conseguir extrair uma informação de um site.

Atividade 2 – Extração de dados com “BeautifulSoup” – Site IMDB


```
SAÍDA    TERMINAL    CONSOLE DE DEPUÇÃO    PROBLEMAS
Python  +  -  [ ]  [X]

50
1 One On One (2020) 10.0 7
2 Haiti Speaks (2020) 10.0 6
3 Do Creation Originals (2020- ) 10.0 5
4 Captain of Desert (2020) 10.0 6
5 A World Away (Remix) (2020 Video) 10.0 9
6 Podpah (2020- ) 10.0 5
7 Memories of Love (2020- ) 10.0 12
8 The Prequel of Gold Convoyers: A Battle of Desert (2020) 10.0 7
9 Are We Developing Artificial Intelligence to Support Humanity? (2020) 10.0 6
10 Em busca da Casa Automática (2013- ) 10.0 12 - Episódio: A Fornalha Infinita
PS D:\Sistemas\python> [ ]
```

Objetivo: Nesta atividade você vai extrair dados de uma página Web, para isto o Python tem duas bibliotecas que é a “BeautifulSoup” e “requests”, é necessário um conhecimento básico de HTML para realizar esta raspagem de dados. Para este exemplo foi escolhido o site https://www.imdb.com/search/title/?release_date=2020-01-01,2020-12-31&sort=user_rating,desc nele você vai pegar os 10 filmes mais bem ranqueados.

Comandos utilizados: Biblioteca “requests” e “BeautifulSoup”.

1. Primeiramente crie um arquivo com o nome de cap09_atividade02.py.
2. As informações que você vai extrair são estas.



1. One On One (2020)
14 min | Short, Drama
★ 10,0 ☆ Rate this
A guy comes to the sea to drown himself. But he can not do it. He sees a handicapped girl who fights for being able-bodied person. Then he decides to vent his anger on the handicapped girl.
Director: Maria Solovyova | Stars: Ivan Mojeyko, Vasilisa Stefanidi, Maria Solovyova
Votes: 7

3. O HTML que contém estas informações é este:

```
.. <div class="lister-item mode-advanced"> == $0
  ><div class="lister-top-right">...</div>
  ><div class="lister-item-image float-left">...</div>
  ><div class="lister-item-content">
    ><h3 class="lister-item-header">
      ><span class="lister-item-index unbold text-primary">1.</span>
      ><a href="/title/tt6883844/?ref=adv_li_tt">One On One</a>
      ><span class="lister-item-year text-muted unbold">(2020)</span>
    </h3>
    ><p class="text-muted">...</p>
    ><div class="ratings-bar">
      ><div class="inline-block ratings-imdb-rating" name="ir" data-value="10">
        ><span class="global-sprite rating-star imdb-rating"></span>
        ><strong>10,0</strong>
      </div>
      ><div class="inline-block ratings-user-rating">...</div>
    </div>
    ><p class="text-muted">...</p>
    ><p class="sort-num_votes-visible">
      ><span class="text-muted">Votes:</span>
      ><span name="nv" data-value="7">7</span>
    </p>
  </div>
</div>
```

4. Importe as bibliotecas do “requests”, “bs4” e “os”.

```
from requests import get
from bs4 import BeautifulSoup
from os import system, name
system('cls') if(name == 'nt') else system('clear')
```

5. Crie uma variável para guardar a url que iremos extrair dados.

```
url = 'https://www.imdb.com/search/title/?release_date=2020-01-01,2020-12-31&sort=user_rating,desc'
```

6. Crie uma variável para pegar os dados da página para usar com a “bs4”.

```
response = get(url)
```

7. Leia a página com o “bs4”.

```
html_soup = BeautifulSoup(response.text, 'html.parser')
```

8. Faça a procura pela div com a class: 'lister-item mode-advanced'.

```
filmes = html_soup.find_all('div', class_ = 'lister-item mode-advanced')
```

9. Imprima a quantidade de dados que foram encontrados.

```
print(len(filmes))
```

10. Se foi encontrado 50 está correto, a página lista 50 filmes, faça um for para ler os primeiros 10 filmes.

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

11. Crie uma variável para guardar os dados encontrados.

```
filme_dados = filmes[i]
```

```
nome = filme_dados.h3.a.text
```

```
lancamento = filme_dados.h3.find('span', class_ = 'lister-item-year
text-muted unbold')
```

```
votos = filme_dados.find('span', attrs = {'name': 'nv'})
```

```

<h3 class="list-item-header">
  <span class="list-item-index unbold text-primary">10.</span>
  <a href="/title/tt12423408/?ref=adv_li_tt"> Em busca da Casa Automática</a>
  <span class="list-item-year text-muted unbold">(2013- )</span>
  <br>
  <small class="text-primary unbold">Episode:</small>
  <a href="/title/tt15152428/?ref=adv_li_tt">A Fornoalha Infinita</a> == $0
  <span class="list-item-year text-muted unbold">(2020)</span>
</h3>
<p class="text-muted">...</p>
<div class="ratings-bar">...</div>
<p class="text-muted">...</p>
<p class="text-muted">...</p>
<p class="text-muted">...</p>
<p class="text-muted">...</p>
</div>

```

```
episodio = filme dados.h3.find('small', 'text-primary unbold')
```

X= " "

```
if episodio is not None:
```

```
ep = filme_dados.find_all('a')
x = '- Episodio: ' + ep[2].text
```

```
print('{} - {} {} \nPontuação: {} - Votos: {}'.format(i + 1, nome,
lançamento.text, filme_dados.strong.text, votos.text))
```

60

Atividade 3 – Extração de dados com “BeautifulSoup” – Site ChromeDriver

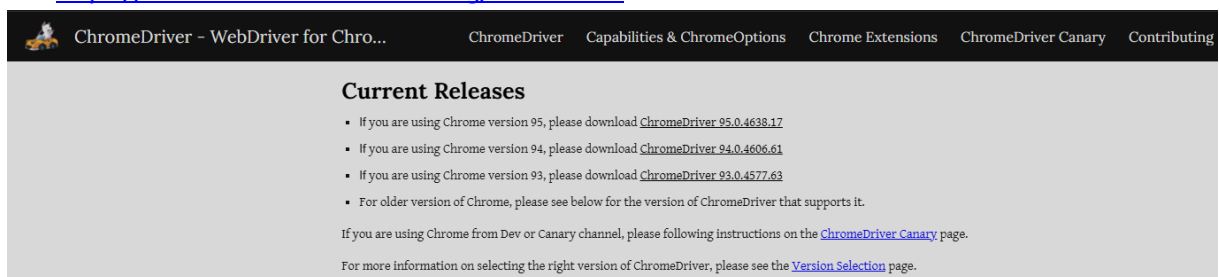
```
SAÍDA  TERMINAL  CONSOLE DE DEPUÇÃO  PROBLEMAS
200
If you are using Chrome version 94, please download ChromeDriver 94.0.4606.61
94.0.4606.61
https://chromedriver.storage.googleapis.com/94.0.4606.61/chromedriver_win32.zip
PS D:\Sistemas\python> □
```

Objetivo: Nesta atividade você vai procurar uma informação específica do site <https://chromedriver.chromium.org/downloads> neste site temos a versão mais atualizada de uma biblioteca que usaremos para controlar o browser Chrome.

Comandos utilizados: Biblioteca “requests” e “BeautifulSoup”.

1. Crie um arquivo com o nome `cap09_atividade03.py`.
2. Os dados que vai pegar esta neste site.

<https://chromedriver.chromium.org/downloads>



3. Precisamos pegar o código da versão do segundo link, o primeiro indica a versão beta, o segundo a versão estável e o terceiro a versão anterior, com esta informação vamos redirecionar o site para a página de download do arquivo desta biblioteca.
4. Importe as bibliotecas do “requests”, “bs4” e “os”.

```
from requests import get
from bs4 import BeautifulSoup
from os import system, name
system('cls') if(name == 'nt') else system('clear')
```

5. Crie uma variável para guardar a url que iremos extrair dados.
- ```
url = 'https://chromedriver.chromium.org/downloads'
```

6. Crie uma variável para pegar os dados da página para usar com a “bs4”.
- ```
response = get(url)
```

7. Imprima o status da página para ver se ela esta funcionando, tem que retornar 200.
- ```
print(response.status_code)
```

8. Leia a página com o “bs4”.
- ```
html_soup = BeautifulSoup(response.text, 'html.parser')
```

9. Faça a procura pela tag `ul`.

```
tags = html_soup.find_all('ul')
```

10. Utilize um for para varrer as informações encontradas.

```
for i in range(len(tags)):
```

11. Verifique se o texto da ul contém 'you are using Chrome version', foi tirada a palavra if pois ela esta separada do restante do texto por outra tag.

```
if (tags[i].text.find('you are using Chrome version')>0):
```

12. Armazene uma nova pesquisa pelo tag li.

```
listas = tags[i].find_all('li')
```

13. Verifique se existe mais que dois itens da lista da pesquisa de li.

```
if (len(listas)>=1):
```

14. Se tiver imprima o text do segundo item deste item.

```
print(listas[1].text)
```

15. Imprima também os últimos 12 caracteres.

```
print(listas[1].text[-12:])
```

16. Crie um link para o arquivo que deverá ser baixado.

```
urldownload = 'https://chromedriver.storage.googleapis.com/' +  
listas[1].text[-12:] + '/chromedriver_win32.zip'  
print(urldownload)
```

Atividade 4 – Download de arquivos e Unzip



```
SAÍDA  TERMINAL  CONSOLE DE DEPURACÃO  PROBLEMAS  
PS D:\Sistemas\python> & C:/Users/laerc/AppData/Local/Programs/Python/Python310/python.exe  
d:/Sistemas/python/cap09_atividade04.py  
Download realizado na página https://chromedriver.storage.googleapis.com/94.0.4606.61/chro  
medriver_win32.zip  
Tamanho do arquivo 5993847  
Arquivos extraídos  
PS D:\Sistemas\python> □
```

Objetivo: Nesta atividade você vai realizar o download de um arquivo zipado em uma determinada pasta e descompactar este arquivo.

Comandos utilizados: Biblioteca zipfile, download de arquivos da web.

1. Crie um arquivo com o nome de cap09_atividade04.py.

2. Importe as bibliotecas necessárias "io", "sys", "os" e "zipfile".

```
import io  
import sys  
import os  
import zipfile
```

```
import urllib.request as request
```

3. Crie uma variável para armazenar a url do arquivo que fará o download.

```
url =  
'https://chromedriver.storage.googleapis.com/94.0.4606.61/chromedriver_win32.zip'
```

4. Use o “requests” para abrir a url.

```
response = request.urlopen(url)
```

5. Crie uma variável para guardar o caminho da pasta do arquivo de código.

```
pasta = os.path.abspath(os.getcwd())
```

6. Crie um arquivo, este será o nome e local do arquivo ao final do download.

```
out_file = io.FileIO(f"{pasta}\chromedriver_win32.zip", mode="w")  
download(response, out_file)
```

7. Para realizar o download você irá criar uma função, no início do código crie uma função com o nome download e com os argumentos response e output. É necessário definir uma variável com um valor em bytes do tamanho dos pacotes de download use 1024.

```
BUFF_SIZE = 1024
```

```
def download(response, output):
```

8. Crie uma variável para somar o tamanho do download do arquivo, inicie com zero.

```
total_downloaded = 0
```

9. Faça um while para ir baixando o arquivo, este while será continuo e só irá parar quando o arquivo for totalmente baixado.

```
while True:
```

10. Crie uma variável para iniciar a leitura do arquivo para download.

```
data = response.read(BUFF_SIZE)
```

11. Some a quantidade de bytes que já foram baixados.

```
total_downloaded += len(data)
```

12. Se não existirem mais dados para baixar realize um break para finalizar o while.

```
if not data:  
    break
```

13. Escreva o arquivo na pasta local.

```
output.write(data)
```

14. Ao final do download faça a impressão do tamanho do arquivo.

```
print('Tamanho do arquivo {bytes}'.format(bytes=total_downloaded))
```

15. Volte ao código e chame a função de download.

```
download(response, out_file)
```

16. Agora iremos criar uma função para descompactar o arquivo que foi feito o download. Crie uma função com o nome zip logo abaixo da função download.

```
def zip(path):
```

17. Faça uma verificação se o arquivo não existe, e no else colocaremos o código para descompactar o arquivo.

```
if not os.path.exists(path):  
    print("Arquivo {} não existe".format(path))  
    sys.exit(-1)  
else:
```

18. Para descompactar o arquivo usaremos a biblioteca zip, o comando extractall descompacta o arquivo.

```
zfile = zipfile.ZipFile(path)  
zfile.extractall()  
print("Arquivos extraídos")
```

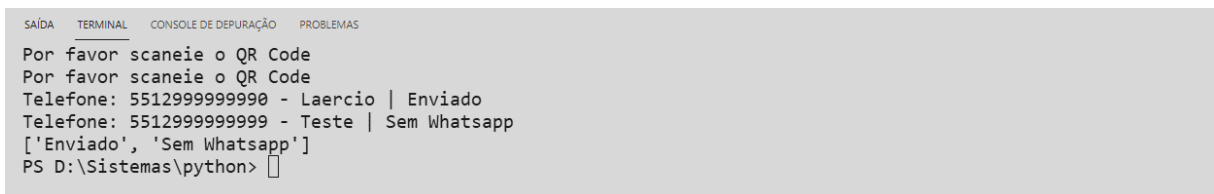
19. Volte ao código e após a chamada de uso da função download chame a função zip.

```
zip(f"{pasta}\chromedriver_win32.zip")
```

20. Imprima no terminal uma informação de que o download foi realizado.

```
print(f'Download realizado na pagina {url}')
```

Atividade 5 – Controlando o Chrome com o WebDriver



Objetivo: Nesta atividade você vai controlar o Chrome e enviar mensagens usando o WhatsApp Web. Utilizando o Selenium.

Comandos utilizados: Biblioteca Selenium e Chromedriver.

1. Abra um novo arquivo e salve com o nome `cap09_atividade05.py`.
2. Para esta atividade é necessário instalar a biblioteca selenium através do terminal.

```
pip install selenium
```

3. Declare as variáveis para utilizar a biblioteca selenium.

```
from selenium import webdriver  
from selenium.webdriver.common.keys import Keys  
from selenium.webdriver.common.by import By
```

4. É necessário também usar um timer para executar as ações, pois algumas podem demorar alguns segundos e precisar sincronizar este tempo, para isto use a classe `sleep` da biblioteca `time`.

```
from time import sleep
```


5. Primeiramente crie uma variável para guardar o endereço da página inicial do WhatsApp web.

```
WP_LINK = 'https://web.whatsapp.com'
```

6. Para controlar o Chrome uma das maneiras mais fáceis de determinar o que deseja fazer é pegar o XPath de um elemento HTML, para este exemplo usaremos 3 elementos da tela, caixa de novo chat, a caixa de mensagem para enviar um texto e o botão de envio, crie uma variável para cada elemento.

```
NEW_CHAT = '//*[@id="side"]/header/div[2]/div/span/div[2]/div'
```

```
MESSAGE_BOX =
```

```
'//*[@id="main"]/footer/div[1]/div/div/div[2]/div[1]/div/div[2]'
```

```
SEND = '//*[@id="main"]/footer/div[1]/div/div/div[2]/div[2]'
```

7. Para a conexão do selenium com o Chrome é necessário o arquivo do WebDriver da mesma versão do Chrome. Baixe o arquivo do ChromeWebDriver em:

<https://chromedriver.chromium.org/home>

8. Depois do download do arquivo descompacte-o na mesma pasta que esteja o arquivo de código .py.

9. Crie a instancia do webdriver.Chrome.

```
driver = webdriver.Chrome()
```

10. Indique com o comando get a pagina que deseja abrir, neste caso use a variável WP_LINK.

```
driver.get(WP_LINK)
```

11. Para este exemplo você vai criar duas listas, uma com telefones e outra com nomes estas listas serão usadas para o envio das mensagens.

```
listaTelefone = ['5512981256440', '5512999999999']
```

```
listaNome = ['Laercio', 'Teste']
```

12. Crie uma variável com o valor True, usaremos ela para verificar se já foi escaneado o QRCode do WhatsApp Web.

```
contador = True
```

13. Inicie o um loop com esta variável.

```
while (contador):
```

14. Agora coloque um sleep de 3 segundos, este sleep é necessário para que a execução do código permita que as páginas sejam abertas e executadas, este tempo pode ser aumentado ou diminuído dependendo da velocidade do celular e da internet.

```
sleep(3)
```

15. Inicie um try.

```
try:
```

16. Coloque o código para encontrar o elemento NEW_CHAT na página através de XPath (find_element).

```
driver.find_element(By.XPATH, NEW_CHAT)
```

17. Se encontrar mude a variável para False.

```
contador = False
```

18. Agora faça o tratamento do erro com o except.

except:

19. No tratamento você vai apenas indicar uma mensagem para o usuário scanear o QRCode.

```
print("Por favor scaneie o QR Code")
contador = True
```

20. Crie uma lista vazia para guardar a informação de qual mensagem foi enviada.

```
envio = []
```

21. Inicie um for com o comando ZIP para pegar as informações da lista de Nome e Telefone ao mesmo tempo.

```
for nome, telefone in zip(listaNome, listaTelefone):
```

22. Inicie um tratamento de erro try.

try:

23. Coloque um novo sleep de 3 segundos.

```
sleep(3)
```

24. Crie uma variável para indicar a página do WhatsApp Web da variável telefone.

```
url = f"https://web.whatsapp.com/send?phone={telefone}"
```

25. Abra a nova página.

```
driver.get(url)
```

26. Crie um sleep de 5 segundos para que a página possa ser carregada.

```
sleep(5)
```

27. Coloque um comentário neste ponto, será criado uma função para o envio da mensagem, mas primeiro você deve terminar o except deste código.

28. Crie um except para tratar algum possível erro.

except Exception as e:

29. Coloque um código para parar os scripts da página.

```
driver.execute_script("window.stop();")
status = 'Erro ao enviar'
```

30. Coloque um print para informar que não foi possível enviar a mensagem.

```
print(f"Telefone: {telefone} - {nome} | ERRO ao Enviar")
```

31. Volte a indentação Faça um append na lista envio indicando que houve um erro.

```
envio.append(status)
```

32. Agora você vai criar a função.

33. No início do código após as importações das bibliotecas cria a função enviar com o parâmetro nome.

```
def enviar(nome):
```

34. Inicie um tratamento de erro try.

try:

35. Declare uma variável com o elemento XPath, MESSAGE_BOX.

```
mensagem = driver.find_element(By.XPATH, MESSAGE_BOX)
```

36. Execute um clique com o método click().

```
mensagem.click()
```

37. Para escrever um texto na caixa use o comando send_keys.

```
mensagem.send_keys(f"Olá *{nome}*, tudo bem?")
```

38. Para pular de linha na caixa de mensagem use o comando abaixo.

```
mensagem.send_keys(Keys.CONTROL, Keys.RETURN)
```

39. Escreva uma segunda linha de mensagem.

```
mensagem.send_keys("Esta é uma mensagem de *Teste*.")
```

40. Use o sleep para fazer uma pausa de 3 segundos.

```
sleep(3)
```

41. Inicie outra variável para o XPath SEND.

```
botao = driver.find_element(By.XPATH, SEND)
```

42. Envie o método click para o botão SEND.

```
botao.click()
```

43. Retorne o texto 'Enviado'.

```
return 'Enviado'
```

44. Crie o except para o caso de um erro, aqui o erro provavelmente será porque o numero informado não tem WhatsApp.

except Exception as e:

45. Use um return 'Não possui WhatsApp'.

```
return 'Sem WhatsApp'
```

46. Volte para o código onde colocou o comentário e chame a função passando como argumento o nome da lista e armazene isto em uma variável.

```
status = enviar(nome)
```

47. Passe o valor desta variável para a lista de envio.

```
print(f"Telefone: {telefone} - {nome} | {status}")
```

48. Volte toda a indentação e faça um print da lista de envio.

```
print(envio)
```

49. Coloque um sleep de 5 segundos.

```
sleep(5)
```

50. Feche o Chrome usando um close da variável do webdriver.

```
driver.close()
```

Python I - Fundamentos

Código Completo das atividades

Códigos Completos

Capítulo 1 – Exercícios Resolvidos

```
"""
    Cap01 - Atividade 04
    "Olá Mundo!"

    Objetivos:
    Nesta atividade você irá imprimir na tela o texto "Olá mundo!" em uma
    linha e na linha abaixo o texto "Meu primeiro código em Python"

    Comandos utilizados:
    Comentário e print
"""

# criando o meu primeiro programa em python!!!
# o comando print escreve no terminal
print('Olá mundo!')
# é possível colocar dois ou mais textos separados por vírgula
print('Meu primeiro código em', 'Python')
```

```
"""
    Cap01 - Atividade 05
    Usando o input

    Objetivos:
    Nesta atividade você vai usar o comando input para interagir com o
    usuário pedindo uma informação e usará o print de outra forma

    Comandos utilizados:
    print, input, format e posição de substituição
"""

# Usando o print
print('Seja bem-vindo a aula de' + ' Python!')
# podemos usar o comando format para concatenar texto
print('Olá {}' . format(input('Qual o seu nome? ')))
```

Capítulo 2 – Exercícios Resolvidos

```
"""
    Cap02 - Atividade 01
    Somando Números

    Objetivos:
    Nesta atividade você vai somar dois numeros usando variáveis e irá
    verificar os tipos de dados de uma variável e como converter uma variável
    em um tipo diferente.

    Comandos utilizados:
    Variáveis, type, int, float
    """

# função de soma
print('Vamos fazer a soma de números inteiros')
num1 = input('Informe o primeiro número : ')
num2 = input('Informe o segundo número: ')
soma = num1 + num2
print('A soma entre {} e {} é igual {}'.format(num1, num2, soma))
print('Toda informação recebida pelo comando input é do tipo str')
# o comando type retorna o tipo da variável
print(type(num1))
print('É necessário converter os dados em número, agora a soma vai ficar
correta')
# use int para numeros inteiros
soma = int(num1) + int(num2)
print('A soma correta é {}'.format(soma))
```

```
"""
    Cap02 - Atividade 02
    Área de um retângulo

    Objetivos:
    Nesta atividade você vai calcular a área de um retângulo através de dois
    números fornecidos pelo usuário e fará também a verificação se os dados
    informados são realmente números.

    Comandos utilizados:
    Float e método isnumeric()
    """
```

```
# neste exemplo iremos usar a biblioteca do sistema operacional para
limpar o terminal
print('Vamos calcular a area de um retangulo')
lado1 = input('Informe o primeiro lado: ')
lado2 = input('Informe o segundo lado: ')
# o método isnumeric valida se uma variável é um numero inteiro
print('Lado1 é numerico?' , lado1.isnumeric())
print('Lado2 é numerico?' , lado2.isdecimal())
area = float(lado1) * float(lado2)
print('A area do quadrado é: {} ' . format(area))
```

```
"""
    Cap02 - Atividade 03
    Trabalhando como Textos

    Objetivos:
    Nesta atividade você vai trabalhar com dados de texto, usando vários
    métodos para verificar e modificar o valor de uma variável.

    Comandos utilizados:
    Função len e os métodos upper, lower, capitalize, find, replace,
    isalpha, isalnum, split, center
    """
from os import system
system('cls')
nomecompleto = input('Informe o seu nome completo: ')
# função len retorno a quantidade de caracteres de uma variável
print('1. Quantidade de caracteres:', len(nomecompleto))
# metodos utilizados:
# upper = transforma um texto em maiusculo
# lower = transforma um texto em minusculo
# capitalize = somente a primeira letra em maiusculo
print('2. Nome em maiusculo:', nomecompleto.upper())
print('3. Nome em minusculo:', nomecompleto.lower())
print('4. Primeira letra em maiusculo:', nomecompleto.capitalize())
# metodo find sendo usado para localizar o primeiro espaço em branco
espaco = nomecompleto.find(' ')
# usando a primeira posição de espaco para separar o texto usando a
notação [x:x]
print('5. Somente o primeiro nome:', nomecompleto[0:espaco])
# metodo replace para tirar todos os espacos em branco
```

```

print('6. Nome sem espaços:', nomecompleto.replace(' ', ''))
# metodo isaplha para verificar se tem somente letras na palavra
print('7. Tem somente letras? (temos que tirar os espaços para
verificar:', nomecompleto.replace(' ', '').isalpha())
# metodo isaplha para verificar se tem somente letras ou numeros na
palavra
print('8. É alfanumerico? tem letras ou numeros (temos que tirar os
espaços para verificar:', nomecompleto.replace(' ', '').isalnum())
# metodo split para criar uma lista com os nomes usando o espaço em branco
como quebra
print('9. Quebrar os texto a cada espaço em branco:', nomecompleto.split("
"))
# metodo center para centralizar o texto em 80 colunas usando o *
print('10. Centralizar o nome entre *')
print(nomecompleto.center(80,"*"))

```

```

"""
Cap02 - Atividade Extra
Calculando Raiz Quadrada

Objetivos:
Nesta atividade você vai calcular a raiz quadrada a partir do operador
matemático ** (exponenciação). O valor será passado pelo usuário e
verificaremos se este valor é numerico ou decimal.

Comandos utilizados:
Variáveis, operador Exponenciação (**) e métodos isnumeric e isdecimal.
"""
from os import system
system('cls')
print('Vamos calcular a raiz quadrada')
n = input('informe o número: ')
print('O valor é um número inteiro? ', n.isnumeric())
print('O valor é um número com casas decimais? ', n.isdecimal())
# ** = Exponenciação - retorna um número elevado a potência de outro,
para raiz quadrada use 1/2
resultado = int(n) ** (1/2)
print('A raiz quadrada de {} é {:.2f}'.format(n, resultado))

```


Capítulo 3 – Exercícios Resolvidos

```
"""
    Cap03 - Atividade 01
    Verificar Número Par e Impar

    Objetivos:
    Nesta atividade você vai usar uma estrutura de decisão (if / else) para
    verificar se um número é par ou ímpar.

    Comandos utilizados:
    If, operador % (retorna o resto da divisão entre operandos)
    """
from os import system, name
system('cls') if(name == 'nt') else system('clear')
número = int(input('Informe um número: '))
# operador % restorna o restante de uma divisão
resultado = int(número % 2)
print('Se o resultado for 0 é par e se for 1 é ímpar, o resultado é:',
      resultado)
input('Ops, assim ficou ruim... assim fica melhor')
# if é usado para tomada de decisão
# comparação de igualdade no if sempre com ==
if resultado == 0:
    resultado = '0 número é par'
else:
    resultado = '0 número é ímpar'
print(resultado)
```

```
"""
    Cap03 - Atividade 02
    Conversor de Medidas

    Objetivos:
    Nesta atividade você vai converter um número em centímetros para
    Polegada, Pé ou Jarda. Será necessário usar o comando if / elif / else.

    Comandos utilizados:
    If / elif / else, formatação de números com posição de substituição
    {:.4f}
    """
```

```

from os import system
system('cls')
print('*** CONVERSOR DE MEDIDAS ***')
# utilize o \n para quebra de linha
medida = float(input('Informe a medida em centímetros: '))
print('\nEscolha para que unidade deseja converter')
print('1 - Polegada\n2 - Pé\n3 - Jarda')
menu = input('Opção: ')
if menu=="1":
    polegadas = medida / 2.54
    resultado = '{:.4f} centímetros correspondem a {:.4f} polegadas' .
    format(medida, polegadas)
elif menu=="2":
    pes = medida / 30.48
    resultado = '{:.4f} centímetros correspondem a {:.4f} pés' .
    format(medida, pes)
elif menu=="3":
    jardas = medida / 91.44
    resultado = '{:.4f} centímetros correspondem a {:.4f} jardas' .
    format(medida, jardas)
else:
    resultado = "Você não escolheu uma das opções..."
print(resultado)

```

```

"""
Cap03 - Atividade 03
Folha de Pagamento

Objetivos:
Nesta atividade você vai calcular uma folha de pagamento usando o
comando if aninhado e mostrando o resultado dos valores de Imposto de
Renda e INSS a serem pagos.

Comandos utilizados:
Variáveis, if, operadores matemáticos e formatação com posição de
substituição.
"""
from os import system, name
system('cls') if(name == 'nt') else system('clear')
print('*** Folha de Pagamento ***')
salario = float(input('Informe o seu salario: '))

```

```

# Calcular INSS
if salario > 6433.58:
    inss = 900.70
else:
    if salario > 3305.23:
        inssP = 0.14
    elif salario > 2203.49:
        inssP = 0.12
    elif salario > 1100.01:
        inssP = 0.09
    else:
        inssP = 0.075
    inss = salario * inssP
base = salario - inss

# Calcular IR
if base > 4664.68:
    irP = 0.275
    deducacao = 869.36
elif base > 3751.06:
    irP = 0.225
    deducacao = 636.13
elif base > 2826.66:
    irP = 0.15
    deducacao = 354.80
elif base > 1903.98:
    irP = 0.075
    deducacao = 142.80
else:
    irP = 0
    deducacao = 0

ir = (base * irP) - deducacao
salarioLiquido = base - ir

print('Salário Bruto: {:.2f}\nSalário Base: {:.2f}\nINSS: {:.2f}\nIR: {:.2f}%\nValor IR: {:.2f}\nSalário Liquido: {:.2f}' . format(salario, base, inss, irP*100, ir, salarioLiquido))

```

```

"""
Cap03 - Atividade Extra
Calcular IMC

Objetivos:
Nesta atividade você vai calcular o IMC a partir de um peso e uma
altura, usará a comando if para mostrar o resultado do calculo do IMC.

Comandos utilizados:
Variáveis, if / elif / else
"""
from os import system, name
system('cls') if(name == 'nt') else system('clear')
print('*** CALCULADORA DE IMC ***')
peso = int(input('Informe o seu peso em KG: '))
altura = float(input('Agora a sua altura em centímetros: '))/100
imc = peso / (altura * altura)
print("O seu IMC é de {:.2f}".format(imc))
if imc < 18.5:
    resultado = 'Abaixo do Peso'
elif imc < 25:
    resultado = 'Peso Normal'
elif imc < 30:
    resultado = 'Sobrepeso'
elif imc < 35:
    resultado = 'Obesidade Grau I'
elif imc < 40:
    resultado = 'Obesidade Grau II'
else:
    resultado = 'Obesidade Grau III'
print(resultado)

```

Capítulo 4 – Exercícios Resolvidos

```
"""
    Cap04 - Atividade 01
    Tabuada

    Objetivos:
    Nesta atividade você vai montar uma Tabuada usando a estrutura de Loop
    do For e range.

    Comandos utilizados:
    Comandos for e range
    """
from os import system, name
system('cls') if(name == 'nt') else system('clear')
print('*** Tabuada Simples ***')
multiplicador = int(input('Informe o multiplicador '))
# para criar o loop usaremos a função for e range
for i in range(10):
    print('{} * {} = {}'.format(multiplicador, i, i*multiplicador))
```

```
"""
    Cap04 - Atividade 02
    MultiTabuada

    Objetivos:
    Nesta atividade você vai construir uma multitabuada de duas maneiras, na
    primeira usando for e na segunda usando for aninhado

    Comandos utilizados:
    Comandos for range
    """
from os import system, name
system('cls') if(name == 'nt') else system('clear')
print('\n*** MULTI TABUADA 1 ***')
for i in range(1, 11):
    print('{: >4} {: >4} {: >4} {: >4} {: >4} {: >4} {: >4} {: >4} {: >4} {: >4}'
          .format(i, i*2, i*3, i*4, i*5, i*6, i*7, i*8, i*9, i*10))
print('\n*** MULTI TABUADA 2 ***')
# usando um for dentro de outro
for i in range(1, 11):
```

```

linha = '{: >4} ' . format(i)
for ii in (range(2, 11)):
    linha+='{: >4} ' . format(ii*i)
print(linha)

```

"""

Cap04 - Atividade 03
Verificar Número Primo

Objetivos:

Nesta atividade você vai verificar se um número é primo ou não e ainda indicar quem é o menor divisor deste número, usando o While para o realizar o loop.

Comandos utilizados:

Comando f com variável de substituição, comando while e break e if ternário

"""

```

from os import system, name
system('cls') if(name == 'nt') else system('clear')
numero = int(input("Digite um número inteiro: "))
i = 2
divisor = 0
tipo = '0 número deve ser maior que 2'
# usaremos o while para criar o loop
while i < numero:
    tipo = '0 número é PRIMO'
    x = numero % i
    if x == 0:
        divisor = i
        tipo = '0 número não é primo'
        # comando break finaliza um loop
        break
    i = i + 1
# observe que aqui usamos a função f e o nome da variável dentro das {} em vez do comando format
print(tipo)
print(f'0 menor divisor é: {divisor}' if divisor > 0 else '')

```

```
"""
    Cap04 - Atividade Extra
    Menu

    Objetivos:
    Nesta atividade você vai criar um menu usando o While, este menu poderá
    ser utilizado em conjunto com outros códigos para que a execução do
    programa seja realizado até que o usuário deseje sair.

    Comandos utilizados:
    Variáveis, while
    """
from os import system, name
system('cls') if(name == 'nt') else system('clear')

opcao = ''
while opcao!='x':
    print('execução do sistema...\n')
    opcao=input('Escolha uma das opções do menu\nx. Aperte X para Sair ou
    qualquer tecla para continuar ')
    system('cls') if(name == 'nt') else system('clear')
```

Capítulo 5 – Exercícios Resolvidos

```
"""
    Cap05 - Atividade 01
    Número por Extenso

    Objetivos:
    Nesta atividade você vai escrever um número por extenso, para isto usará
    uma tupla. A tupla é um array que contem dados que não pode ser alterados.

    Comandos utilizados:
    Tupla, operadores / e %
    """

from os import system, name
system('cls') if(name == 'nt') else system('clear')
# tupla é criada com () e não pode ter seus dados alterados
numeros = ('zero', 'um', 'dois', 'três', 'quatro', 'cinco', 'seis',
'sete', 'oito', 'nove')
dez = ('dez', 'onze', 'doze', 'treze', 'quatorze', 'quinze', 'dezesesseis',
'dezessete', 'dezoito', 'dezenove')
dezenas = ('', '', 'vinte', 'trinta', 'quarenta', 'cinquenta', 'sessenta',
'setenta', 'oitenta', 'noventa')
print(dezenas[2])
pos = int(input('Digite um número entre 0 e 99: '))
if pos < 10:
    extenso = numeros[pos]
elif pos < 20:
    extenso = dez[pos-10]
elif pos <= 99:
    dezena = int(pos / 10) # outra forma de usar seria assim:
int(str(pos)[0:1])
    numeral = (pos % 10) # outra forma de usar seria assim:
int(str(pos)[1:2])
    numero = ''
    if (numeral!=0):
        numero = ' e ' + numeros[numeral]
    extenso = dezenas[dezena] + numero
else:
    extenso = 'Número maior que 100...'
print(extenso)
```



```

"""
Cap05 - Atividade 02
Jogo: Papel, pedra e Tesoura

Objetivos:
Nesta atividade você vai criar um jogo usando tupla e tupla multi-
dimensional.

Comandos utilizados:
Tupla, Tupla Multi-Dimensional, biblioteca random e randint
"""
from os import system, name

# biblioteca random
import random

opcao = 's'

while opcao.upper()=='S':
    system('cls') if(name == 'nt') else system('clear')

    computador = random.randint(0,2)
    jogador = int(input('Escolha uma opcao para se jogar:
[0] Pedra
[1] Papel
[2] Tesoura
Digite sua escolha: '))

    pecas = ("Pedra", "Papel", "Tesoura")

    # lista multi-dimensional
    # nesta lista quando o resultado for -1 o computador ganha, 1 o jogador
    ganha e 0 deu empate

    if (jogador > 3):
        resultado = 'Você não escolheu um item correto!!!'
    else:
        print('Você escolheu {}'.format(pecas[jogador]))
        print('O computador escolheu: {}'.format(pecas[computador]))
        tabela = ((0, 1, -1),(-1, 0, 1),(1, -1, 0))
        jogada = tabela[computador][jogador]
        if jogada == 0:
            resultado = "Deu empate vocês escolheram a mesma peça"

```

```

elif jogada == 1:
    resultado = "Você ganhou!"
else:
    resultado = "O computador ganhou"

print(resultado)
opcao=input('Jogar novamente? Aperte S(sim) para jogar novamente. ')

```

```

"""
Cap05 - Atividade 03
Lista de Cadastro

Objetivos:
Nesta atividade você vai cadastrar o nome de pessoas e seu celular
utilizando listas e ao final irá imprimir uma lista estes dados.

Comandos utilizados:
Lista e método append
"""

from os import system, name
system('cls') if(name == 'nt') else system('clear')
opcao = ''
listaNome = []
listaCelular = []

while opcao!='x':
    nome = input("Informe o nome: ")
    celular = input("Informe o celular: ")

    listaNome.append(nome)

    listaCelular.append(celular)
    opcao=input('\nAperte X para Finalizar o cadastro ou qualquer tecla para
continuar ')

for i in range(len(listaNome)):
    print('Nome: ', listaNome[i], ' - Celular:' , listaCelular[i])

```

```

"""
Cap05 - Atividade Extra
Jogo: Papel, pedra e Tesoura

Objetivos:
Nesta atividade você vai criar um jogo usando tupla e tupla multi-
dimensional.

Comandos utilizados:
Tupla, Tupla Multi-Dimensional, biblioteca random e randint
"""
from os import system, name
# biblioteca random
import random

opcao = 's'
contadorJogadas = 0
contadorJogador = 0
contadorComputador = 0
while opcao.upper()=='S':
    system('cls') if(name == 'nt') else system('clear')

    computador = random.randint(0,2)
    jogador = int(input('Escolha uma opcao para se jogar:
    [0] Pedra
    [1] Papel
    [2] Tesoura
    Digite sua escolha: '))
    contadorJogadas+=1
    pecas = ("Pedra", "Papel", "Tesoura")
    # lista multi-dimensional
    # nesta lista quando o resultado for -1 o computador ganha, 1 o jogador
    # ganha e 0 deu empate

    if (jogador > 3):
        resultado = 'Você não escolheu um item correto!!!'
    else:
        print('Você escolheu {}'.format(pecas[jogador]))
        print('O computador escolheu: {}'.format(pecas[computador]))
        tabela = ((0, 1, -1),(-1, 0, 1),(1, -1, 0))
        jogada = tabela[computador][jogador]
        if jogada == 0:
            resultado = "Deu empate vocês escolheram a mesma peça"

```

```
elif jogada == 1:
    resultado = "Você ganhou!"
    contadorJogador +=1
else:
    resultado = "O computador ganhou"
    contadorComputador +=1

print(resultado)
opcao=input('Jogar novamente? Aperte S para jogar novamente. ')
system('cls') if(name == 'nt') else system('clear')
print('Resultado do Jogo')
print('Quantidade de jogadas: ', contadorJogadas)
print(f'Você ganhou {contadorJogador} jogadas ')
print(f'Você perder {contadorComputador} jogadas ')
```

Capítulo 6 – Exercícios Resolvidos

```
"""
    Cap06 - Atividade 01
    Criação de Funções

    Objetivos:
    Nesta atividade você vai aprender a criar funções do python, as funções
    são muito uteis para que você reaproveite código.

    Comandos utilizados:
    Funções def, while
    """

# def = função
def soma(x, y):
    print("Soma: ", x+y)

def subtracao(x, y):
    print("Subtracao: ", x-y)

def multiplicacao(x, y):
    print("Multiplicacao: ", x*y)

def divisao(x, y):
    print("Divisao: ", x/y)

opcao=1

while opcao:

    x = float(input("Primeiro numero: "))
    y = float(input("Segundo numero: "))

    print("1. Somar")
    print("2. Subtrair")
    print("3. Multiplicação")
    print("4. Divisão ")

    operador = int(input("Opção: "))
    if(operador==1):
        soma(x, y)
```

```

if(operador==2):
    subtracao(x, y)
if(operador==3):
    multiplicacao(x, y)
if(operador==4):
    divisao(x, y)

opcao = input("\nAperte 0 para Sair ou Enter para continuar")

if opcao=="0":
    opcao=int(opcao)

```

```

"""
Cap06 - Atividade 02
Tratamento de erros

Objetivos:
Nesta atividade você vai criar a utilizar o tratamento de erro do
python, irá reutilizar o código da atividade anterior para fazer o
tratamento de erros dentro das funções.

Comandos utilizados:
Tratamento de erro Try / Except
"""
# def = função
def soma(x, y):
    try:
        print("Soma: ", float(x)+float(y))
    except:
        print("Ocorreu um erro")

# def = função
def subtracao(x, y):
    try:
        print("Subtracao: ", float(x)-float(y))
    except:
        print("Ocorreu um erro")

def multiplicacao(x, y):
    try:
        print("Multiplicacao: ", float(x)*float(y))

```

```

except:
    print("Ocorreu um erro")

def divisao(x, y):
    try:
        print("Divisao: ", float(x)/float(y))
    except ZeroDivisionError as erro:
        print(erro)
    except:
        print("Ocorreu um erro")

opcao=1

while opcao:
    x = float(input("Primeiro numero: "))
    y = float(input("Segundo numero: "))

    print("1. Somar")
    print("2. Subtrair")
    print("3. Multiplicação")
    print("4. Divisão ")

    operador = int(input("Opção: "))

    if(operador==1):
        soma(x, y)
    if(operador==2):
        subtracao(x, y)
    if(operador==3):
        multiplicacao(x, y)
    if(operador==4):
        divisao(x, y)

    opcao = input("\nAperte 0 para Sair ou Enter para continuar")

    if opcao=="0":
        opcao=int(opcao)
    else:
        opcao=1

```

```
"""
Cap06 - Atividade 03
Biblioteca Math

Objetivos:
Nesta atividade você vai importar a biblioteca math, visualizar a ajuda
da biblioteca e usar algumas das principais funções.

Comandos utilizados:
Biblioteca math, funções ceil, fabs, floor, fmod, trunc
"""
import math
# use o comando help(biblioteca) para visualizar a sua ajuda
help(math)
# inteiro para cima
print(math.ceil(4.2))
# inteiro para baixo
print(math.floor(3.9))
# número absoluto
print(math.fabs(-1))
# mod = restante de uma divisão
print(math.fmod(9, 4))
# raiz quadrada
print(math.sqrt(36))
# não faz parte da biblioteca math, é usado para arredondar um número.
print(round(3.988, 1))
```

```
"""
Cap06 - Atividade 04
Função de Datas

Objetivos:
Nesta atividade você vai aprender a usar a biblioteca datetime do
python, com ela é possível manipular datas e converter uma string em data.
E irá implementar um return numa função.

Comandos utilizados:
Biblioteca datetime, metodos strptime, strftime e today. Função com
return
"""
from datetime import datetime
```



```

def idade(nascimento):
    today = datetime.today()
    return today.year - nascimento.year - ((today.month, today.day) <
(nascimento.month, nascimento.day))

def dias(nascimento):
    today = datetime.today()
    return abs((nascimento - today).days)

print(datetime.now())
atual = datetime.now()
print(type(atual))
montarData = datetime(2021, 10, 5)
print(type(montarData))
print(atual.strftime('%d/%m/%Y %H:%M'))
dataNascimento = input('Informe a sua data de nascimento dd/mm/aaaa: ')
dataNascimento = datetime.strptime(dataNascimento, '%d/%m/%Y')
print(f'Você tem: {idade(dataNascimento)} anos')
print(f'Você já viveu {dias(dataNascimento)} dias')

```

Capítulo 7 – Exercícios Resolvidos

```
"""
    Cap07 - Atividade 01
    Criar um Classe

    Objetivos:
    Nesta atividade você vai criar uma objeto de classe, aprendendo sobre os
    metodos e propriedades e usar metodos especiais. Irá também aprender a
    consumir esta classe.

    Comandos utilizados:
    Classe, __init__, __str__, Propriedades e Metodos de uma classe
    """
from num2words import num2words
class recibo:

    def __init__(self, nome):
        self.nome = nome
        self._valor = 0
        self._descricao = ''

    def __str__(self):
        texto = 'Recebemos de {} a quantia de R$ {:.2f} ({}))' .
        format(self.nome, self._valor, self.extenso())
        descricao = '\nReferente {}' . format(self._descricao) if
        (self._descricao!='') else ''
        dados = '{}\n{}}' . format('Recibo'.center(len(texto), '*'), texto,
        descricao)
        return(dados)

    def descricao(self, value):
        self._descricao = value

    @property
    def valor(self):
        return(self._valor)

    @valor.setter
    def valor(self, value):
        self._valor = value
```

```
def extenso(self ):
    vExtenso = num2words(self._valor, lang='pt_BR', to='currency')
    return vExtenso
```

```
"""
```

```
    Cap07 - Atividade 01
    Criar um Classe
```

```
    Objetivos:
```

```
    Nesta atividade você vai criar uma objeto de classe, aprendendo sobre os
    metodos e propriedades e usar metodos especiais. Irá também aprender a
    consumir esta classe.
```

```
    Comandos utilizados:
```

```
    Classe, __init__, __str__, Propriedades e Metodos de uma classe
```

```
"""
```

```
from cap07_atividade01Classe import recibo
dados = recibo('Laercio Azevedo de Sa')
dados.valor = 51.13
dados.descricao('desenvolvimento de sistemas em Python')
print(dados._descricao)
print(dados)
```

Capítulo 8 – Exercícios Resolvidos

```
"""
    Cap08 - Atividade 01
    Abrir um arquivo

    Objetivos:
    Nesta atividade você vai ler um arquivo no formato CSV, verificar as
    opções de encoding (codificação de caracteres) e separar os arquivo em
    uma lista para ler as informações de cada coluna.

    Comandos utilizados:
    Biblioteca os, comando open, path, lista e for
    """
import os.path, datetime
from os import system, name
system('cls') if(name == 'nt') else system('clear')

arquivo = 'produtos.csv'

if (os.path.isfile(arquivo)):
    produtos = open(arquivo, 'r', encoding="utf-8")
    tamanho = os.path.getsize(arquivo)
    modificacao = os.path.getmtime(arquivo)
    print('Data de Modificação:',
datetime.datetime.fromtimestamp(modificacao))
    print('Tamanho do arquivo (bytes):', tamanho)
    listaProdutos = []
    for line in produtos:
        colunas = line.strip().split(";")
        colunas[0]=int(colunas[0])
        colunas[2]=int(colunas[2])
        colunas[4]=int(colunas[4])
        colunas[5]=int(colunas[5])
        colunas[6]=int(colunas[6])
        colunas[7]=int(colunas[7])
        colunas[8]=float(colunas[8])
        colunas[9]=float(colunas[9])
        listaProdutos.append(colunas)
    produtos.close()
    for prod in listaProdutos:
        print(prod)
```

```

"""
Cap08 - Atividade 02
Criando um Dicionário

Objetivos:
    Nesta atividade você vai abrir um arquivo e criar um dicionário, o
    dicionário no Python é muito utilizado quando queremos armazenar dados de
    forma organizada e que possuam identificação única como num banco de
    dados.

    Comandos utilizados:
    Dicionário {}, Biblioteca OS, Metodos de string strip() e split()
"""

from os import system, name
import os.path
system('cls') if(name == 'nt') else system('clear')

arquivo = 'categorias.csv'
categorias = open(arquivo, 'r', encoding="utf-8")

dicCategoria = {}

for line in categorias:
    colunas = line.strip().split(";")
    # criar uma lista para as informações do dicionário
    dados = [colunas[1], colunas[2]]
    dicCategoria[colunas[0]] = dados
categorias.close()

# imprimir todo o dicionário
print(dicCategoria)

# imprimir apenas o conteúdo da chave 3 do dicionário
print(dicCategoria['3'])

# imprimir a descrição da conteúdo da chave 3 do dicionário
print(dicCategoria['3'][1])

```

```

"""
Cap08 - Atividade 03
Criando uma Classe

Objetivos:
Nesta atividade você vai abrir criar uma classe com os dados de um
arquivo CSV e retornar um Dicionário.

Comandos utilizados:
Classe, Abrir Arquivo, Dicionário
"""
class tabCat:
    def dicCat(self):
        arquivo = 'categorias.csv'
        categorias = open(arquivo, 'r', encoding="utf-8")
        dicCategoria = {}
        for line in categorias:
            colunas = line.strip().split(";")
            dados = [colunas[1], colunas[2]]
            dicCategoria[colunas[0]] = dados
        categorias.close()
        return dicCategoria

class tabProd:
    def listProd(self):
        arquivo = 'produtos.csv'
        produtos = open(arquivo, 'r', encoding="utf-8")
        listaProdutos = []
        for line in produtos:
            colunas = line.strip().split(";")
            colunas[0]=int(colunas[0])
            colunas[2]=int(colunas[2])
            colunas[4]=int(colunas[4])
            colunas[5]=int(colunas[5])
            colunas[6]=int(colunas[6])
            colunas[7]=int(colunas[7])
            colunas[8]=float(colunas[8])
            colunas[9]=float(colunas[9])
            listaProdutos.append(colunas)
        produtos.close()
        return listaProdutos

```

```

"""
Cap08 - Atividade 03 Consumir
Criar um Relatório

Objetivos:
Nesta atividade você vai salvar um arquivo como um relatório em texto,
com as informações das classes realizadas anteriormente.
Comandos utilizados:
Bibliotecas, Classe, salvar um Arquivo, For
"""

import io
from cap08_atividade03Classe import tabCat
from cap08_atividade03Classe import tabProd
cat = tabCat()
prod = tabProd()
dicCategoria = cat.dicCat()
listaProduto = prod.listProd()
relatorio = open("relatorio.txt", mode="w", encoding="utf-8")
for e in dicCategoria:
    titulo = '{}\n{}\n\nProdutos\n' .format(dicCategoria[e][0].upper(),
dicCategoria[e][1])
    prod = []
    for p in listaProduto:
        if str(p[2]) == e:
            valor = '{:.2f}' . format(p[8])
            produto = '{:<60} | R$ {:>8}' . format(str(p[1].capitalize()) + ' ' +
p[3])[0:60], valor)
            prod.append(produto)
    prod.sort()
    lista = ''
    i = 1
    for p in prod:
        lista += '{:>4}. {} \n' . format(i, p)
        i+=1
    divisor = ' Categoria '.center(80,"*")
    relatorio.write(divisor + '\n' + titulo + lista + '\n\n')
categorias = 'Total de Categorias: {:>4}' . format(len(dicCategoria))
produtos = 'Total de Produtos: {:>4}' . format(len(listaProduto))
resumo = ' Resumo '.center(80,"*")
final = "{}\n\n{:>80}\n{:>80}" . format(resumo, categorias, produtos)
relatorio.write(final)
relatorio.close()

```

Capítulo 9 – Exercícios Resolvidos

```
"""
    Cap09 - Atividade 01
    Extrair dados do Uol Economia
"""

from requests import get
from bs4 import BeautifulSoup
from os import system, name
system('cls') if(name == 'nt') else system('clear')

url = 'https://economia.uol.com.br/'
response = get(url)
html_soup = BeautifulSoup(response.text, 'html.parser')
secaoDinheiro = html_soup.find_all('section', class_ = 'currencies')
print(len(secaoDinheiro))
print(secaoDinheiro)
info = html_soup.find_all('div', class_ = 'info')
print(len(info))
print(info)
print(info[0].text)
infoValor = html_soup.find_all('span', class_ = 'value bra')
print(len(infoValor))
relatorio = open("uol.txt", mode="w", encoding="utf-8")
relatorio.write(html_soup.prettify())

valores = html_soup.find_all('a', class_ = 'subtituloGrafico
subtituloGraficoValor')
print("QTD Class:'subtituloGrafico subtituloGraficoValor':", len(valores))
for valor in valores:
    print(valor.text)
```

```
"""
    Cap09 - Atividade 02
    Extrair dados de filmes do site IMDB
"""

from requests import get
from bs4 import BeautifulSoup
from os import system, name
system('cls') if(name == 'nt') else system('clear')
```



```

url = 'https://www.imdb.com/search/title/?release_date=2020-01-01,2020-12-31&sort=user_rating,desc'
response = get(url)
html_soup = BeautifulSoup(response.text, 'html.parser')
filmes = html_soup.find_all('div', class_ = 'lister-item mode-advanced')
print(len(filmes))
for i in range(10):
    filme_dados = filmes[i]
    nome = filme_dados.h3.a.text
    lancamento = filme_dados.h3.find('span', class_ = 'lister-item-year text-muted unbold')
    votos = filme_dados.find('span', attrs = {'name':'nv'})
    episodio = filme_dados.h3.find('small', 'text-primary unbold')
    x=""
    if episodio is not None:
        ep = filme_dados.find_all('a')
        x = '- Episodio: ' + ep[2].text
        # print(episodio)
    print('{} - {} {} \nPontuação: {} - Votos: {}'.format(i + 1, nome , lancamento.text, filme_dados.strong.text, votos.text))

```

```

"""
    Cap09 - Atividade 01
    Extrair dados do site Chromedriver
"""

from requests import get
from bs4 import BeautifulSoup
from os import system, name
system('cls') if(name == 'nt') else system('clear')

url = 'https://chromedriver.chromium.org/downloads'
response = get(url)
print(response.status_code)

html_soup = BeautifulSoup(response.content, 'html.parser')
tags = html_soup.find_all('ul')

for i in range(len(tags)):
    if (tags[i].text.find('you are using Chrome version')>0):
        listas = tags[i].find_all('li')

```

```

if (len(listas)>=1):
    print(listas[1].text)
    print(listas[1].text[-12:])
    urldownload = 'https://chromedriver.storage.googleapis.com/' +
listas[1].text[-12:] + '/chromedriver_win32.zip'
    print(urldownload)

```

```

"""
    Cap09 - Atividade 04
    Realizar um download e descompactar um arquivo
"""

import io
import sys
import os
import zipfile
import urllib.request as request

BUFF_SIZE = 1024
def download(response, output):

    total_downloaded = 0
    while True:
        data = response.read(BUFF_SIZE)
        total_downloaded += len(data)
        if not data:
            break
        output.write(data)
    print('Tamanho do arquivo {bytes}'.format(bytes=total_downloaded))

def zip(path):

    if not os.path.exists(path):
        print("Arquivo {} não existe".format(path))
        sys.exit(-1)
    else:
        zfile = zipfile.ZipFile(path)
        zfile.extractall()
        print("Arquivos extraídos")

```

```

url =
'https://chromedriver.storage.googleapis.com/94.0.4606.61/chromedriver_win
32.zip'
response = request.urlopen(url)
pasta = os.path.abspath(os.getcwd())
out_file = io.FileIO(f"{pasta}\chromedriver_win32.zip", mode="w")
download(response, out_file)
zip(f"{pasta}\chromedriver_win32.zip")
print(f'Download realizado na pagina {url}')

```

```

"""
    Cap09 - Atividade 05
    Selenium e Webdriver
"""
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from time import sleep

def enviar(nome):
    try:
        mensagem = driver.find_element(By.XPATH, MESSAGE_BOX)
        mensagem.click()
        mensagem.send_keys(f"Olá *{nome}*, tudo bem?")
        mensagem.send_keys(Keys.CONTROL, Keys.RETURN)
        mensagem.send_keys("Esta é uma mensagem de *Teste*.")
        sleep(3)
        botao = driver.find_element(By.XPATH, SEND)
        botao.click()
        return 'Enviado'
    except Exception as e:
        return 'Sem WhatsApp'

WP_LINK = 'https://web.WhatsApp.com'
## XPATHS
SEND = '//*[@id="main"]/footer/div[1]/div/div/div[2]/div[2]'
MESSAGE_BOX =
'//*[@id="main"]/footer/div[1]/div/div/div[2]/div[1]/div/div[2]'
NEW_CHAT = '//*[@id="side"]/header/div[2]/div/span/div[2]/div'

driver = webdriver.Chrome()
driver.get(WP_LINK)

```

```

listaTelefone = ['5512981256440', '5512999999999']
listaNome = ['Laercio', 'Teste']

contador = True
while (contador):
    sleep(3)
    try:
        driver.find_element(By.XPATH, NEW_CHAT)
        contador = False
    except:
        print("Por favor scaneie o QR Code")
        contador = True

envio = []
for nome, telefone in zip(listaNome, listaTelefone):
    try:
        sleep(3)
        url = f"https://web.WhatsApp.com/send?phone={telefone}"
        driver.get(url)
        sleep(5)
        status = enviar(nome)
        print(f"Telefone: {telefone} - {nome} | {status}")
    except Exception as e:
        driver.execute_script("window.stop();")
        print(f"Telefone: {telefone} - {nome} | ERRO ao Enviar")
        status = 'Erro ao enviar'

    envio.append(status)

print(envio)
sleep(5)
driver.close()

```

Anexos

Arquivo produtos.csv

As colunas deste arquivo são:

Código do Produto; Nome do Produto; Categoria do Produto; Tipo de Embalagem; Estoque; Quantidade para separação de pedido; Estoque mínimo; Produto Inativo; Valor Venda; Valor de Custo

```
1;Masala chai;1;10 boxes x 20 bags;39;0;10;0;18.00;14.40
2;Chang;1;24 - 12 oz bottles;17;40;25;0;19.00;15.20
3;Xarope de Anis;2;12 - 550 ml bottles;13;70;25;0;10.00;8.00
4;Tempero Cajun;2;48 - 6 oz jars;53;0;0;0;22.00;17.60
5;Mix de Gumbo;2;36 boxes;0;0;0;1;21.35;17.00
6;Geléia de Framboesa e Amora da Vovó;2;12 - 8 oz jars;120;0;25;0;25.00;20.00
7;Peras Secas Orgânicas;7;12 - 1 lb pkgs.;15;0;10;0;30.00;24.00
8;Molho de Cranberry;2;12 - 12 oz jars;6;0;0;0;40.00;32.00
9;Mishi Kobe Niku;6;18 - 500 g pkgs.;29;0;0;1;97.00;77.60
10;Ikura;8;12 - 200 ml jars;31;0;0;0;31.00;24.80
11;Queijo de Cabra;4;1 kg pkg.;22;30;30;0;21.00;14.00
12;Queijo de Ovelha;4;10 - 500 g pkgs.;86;0;0;0;38.00;30.40
13;Alga Marinha Konbu;8;2 kg box;24;0;5;0;6.00;4.80
14;Tofu;7;40 - 100 g pkgs.;35;0;0;0;23.25;18.60
15;Shouyu;2;24 - 250 ml bottles;39;0;5;0;15.50;12.40
16;Pavlova;3;32 - 500 g boxes;29;0;10;0;17.45;13.90
17;Carneiro;6;20 - 1 kg tins;0;0;0;1;39.00;31.20
18;Tubarão Tigre;8;16 kg pkg.;42;0;0;0;62.50;50.00
19;Biscoitos de chocolate;3;10 boxes x 12 pieces;25;0;5;0;9.20;7.30
20;Marmelada;3;30 gift boxes;40;0;0;0;81.00;64.80
21;Biscoito de Aveia;3;24 pkgs. x 4 pieces;3;40;5;0;10.00;8.00
22;Pão Sueco Crocante;5;24 - 500 g pkgs.;104;0;25;0;21.00;16.80
23;Pão Sueco de Panqueca;5;12 - 250 g pkgs.;61;0;25;0;9.00;7.20
24;Guaraná Fantástica;1;12 - 355 ml cans;20;0;0;1;4.50;3.60
25;Creme de chocolate;3;20 - 450 g glasses;76;0;30;0;14.00;11.20
26;Bala de goma;3;100 - 250 g bags;15;0;0;0;31.23;24.90
27;Chocolate Amargo;3;100 - 100 g pieces;49;0;30;0;43.90;35.10
```

28;Chucrute;7;25 - 825 g cans;26;0;0;1;45.60;36.40
 29;Salsicha Alemã;6;50 bags x 30 sausgs.;0;0;0;1;123.79;99.00
 30;Arenque cru;8;10 - 200 g glasses;10;0;15;0;25.89;20.70
 31;Gorgonzola;4;12 - 100 g pkgs;0;70;20;0;12.50;10.00
 32;Mascarpone;4;24 - 200 g pkgs.;9;40;25;0;32.00;25.60
 33;Queijo norueguês;4;500 g;112;0;20;0;2.50;2.00
 34;Cerveja Tipo Ale;1;24 - 12 oz bottles;111;0;15;0;14.00;11.20
 35;Cerveja Tipo Stout;1;24 - 12 oz bottles;20;0;15;0;18.00;14.40
 36;Arenque curado;8;24 - 250 g jars;112;0;20;0;19.00;15.20
 37;Salmão cru marinado;8;12 - 500 g pkgs.;11;50;25;0;26.00;20.80
 38;Vinho de Blaye;1;12 - 75 cl bottles;17;0;15;0;263.50;210.80
 39;Licor Francês Verde;1;750 cc per bottle;69;0;5;0;18.00;14.40
 40;Carne de Caranguejo;8;24 - 4 oz tins;123;0;30;0;18.40;14.70
 41;sopa de amêijoas;8;12 - 12 oz cans;85;0;10;0;9.65;7.70
 42;Hokkien mee;5;32 - 1 kg pkgs.;26;0;0;1;14.00;9.80
 43;café branco Ipoh;1;16 - 500 g tins;17;10;25;0;46.00;36.80
 44;Açúcar de palma;2;20 - 2 kg bags;27;0;15;0;19.45;15.50
 45;Arenque do Atlântico;8;1k pkg.;5;70;15;0;9.50;7.60
 46;Arenque Marinado;8;4 - 450 g glasses;95;0;0;0;12.00;9.60
 47;biscoito holandês stroopwafel;3;10 - 4 oz boxes;36;0;0;0;9.50;7.60
 48;Chocolate;3;10 pkgs.;15;70;25;0;12.75;10.20
 49;Maxilaku;3;24 - 50 g pkgs.;10;60;15;0;20.00;16.00
 50;chocolate branco;3;12 - 100 g bars;65;0;30;0;16.25;13.00
 51;Maça Desidratada;7;50 - 300 g pkgs.;20;0;10;0;53.00;42.40
 52;Massa Filo;5;16 - 2 kg boxes;38;0;25;0;7.00;5.60
 53;Empanada do Reino Unido;6;48 pieces;0;0;0;1;32.80;26.20
 54;torta de carne canadense;6;16 pies;21;0;10;0;7.45;5.90
 55;torta canadense / francesa;6;24 boxes x 2 pies;115;0;20;0;24.00;19.20
 56;Nhoque;5;24 - 250 g pkgs.;21;10;30;0;38.00;30.40
 57;Ravioli;5;24 - 250 g pkgs.;36;0;20;0;19.50;15.60
 58;Escargot;8;24 pieces;62;0;20;0;13.25;10.60
 59;Queijo Raclette;4;5 kg pkg.;79;0;0;0;55.00;44.00
 60;Queijo Camembert;4;15 - 300 g rounds;19;0;0;0;34.00;27.20
 61;xarope de bordo;2;24 - 500 ml bottles;113;0;25;0;28.50;22.80
 62;Torta doce;3;48 pies;17;0;0;0;49.30;39.40
 63;Torta de vegetais;2;15 - 625 g jars;24;0;5;0;43.90;35.10
 64;Pão redondo da Áustria;5;20 bags x 4 pieces;22;80;30;0;33.25;26.60

65;Molho Picante;2;32 - 8 oz bottles;76;0;0;0;21.05;16.80
 66;Molho Picante de Quiabo;2;24 - 8 oz jars;4;100;20;0;17.00;13.60
 67;Cerveja Lager;1;24 - 12 oz bottles;52;0;10;0;14.00;11.20
 68;Baguete Escocesa;3;10 boxes x 8 pieces;6;10;15;0;12.50;10.00
 69;Manteiga;4;10 kg pkg.;26;0;15;0;36.00;28.80
 70;Cerveja Australiana Lager;1;24 - 355 ml bottles;15;10;30;0;15.00;12.00
 71;"queijo marrom ";4;10 - 500 g pkgs.;26;0;0;0;21.50;17.20
 72;Mozzarella;4;24 - 200 g pkgs.;14;0;0;0;34.80;27.80
 73;Caviar vermelho;8;24 - 150 g jars;101;0;5;0;15.00;12.00
 74;Tofu Liquido;7;5 kg pkg.;4;20;5;0;10.00;8.00
 75;Cerveja Alemã;1;24 - 0.5 l bottles;125;0;25;0;7.75;6.20
 76;Licor de Amora Finlandes;1;500 ml;57;0;20;0;18.00;14.40
 77;Salsa Verde;2;12 boxes;32;0;15;0;13.00;10.40

Arquivo categoria.csv

As colunas deste arquivo são:

Código da Categoria; Nome da Categoria; Descrição da Categoria.

1;Bebidas;Refrigerantes, cafés, chás, cervejas e cervejas
 2;Condimentos;Molhos doces e salgados, condimentos, patês e temperos
 3;Confeitaria;Doces, sobremesas, doces e pães doces
 4;Lácteos;Leites e Queijos
 5;Grãos / Cereais;Pães, biscoitos, massas e cereais
 6;Carnes / Aves;Preparadas de carnes
 7;Processados;Frutas secas e coalhada de feijão
 8;Frutos do mar;Algas e peixes