Modulo 4

Contents

1 Criando datas com modulo datetime

```
    # Criando datas com modulo datetime
# datetime(ano, mes, dia)
# datetime(ano, mes, dia, horas, minutos, segundos)
# datetime.strptime('DATA', 'FORMATO')
# datetime.now()
# https://pt.wikipedia.org/wiki/Era_Unix
# datetime.fromtimestamp(Unix Timestamp)
# https://docs.python.org/3/library/datetime.html
# Para timezones
# https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
# Instalando o pytz
# pip install pytz types-pytz
from datetime import datetime

data_str_data = '2022/04/20 07:49:23'
data_str_data = '20/04/2022'
data_str_fmt = '%d/%m/%Y'

# data = datetime(2022, 4, 20, 7, 49, 23)
data = datetime.strptime(data_str_data, data_str_fmt)
print(data)
```

1.1 Data e hora atual (now), com Unix Timestamp e Timezone diferente (pytz)

```
    # pip install pytz types-pytz
    from datetime import datetime

    data_str_data = '2022/04/20 07:49:23'
    data_str_data = '20/04/2022'
    data_str_fmt = '%d/%m/%Y'
    # from pytz import timezone

    # data = datetime(2022, 4, 20, 7, 49, 23)
    data = datetime.strptime(data_str_data, data_str_fmt)
    print(data)
    data = datetime.now()
    print(data.timestamp())  # Isso está na base de dados
    print(datetime.fromtimestamp(1670849077))
    # data_str_data = '2022/04/20 07:49:23'
    # data_str_data = '20/04/2022'
    # data_str_fmt = '%d/%m/%Y'

    # data = datetime(2022, 4, 20, 7, 49, 23, tzinfo=timezone('Asia/Tokyo'))
    # data = datetime.strptime(data_str_data, data_str_fmt)
```

1.2 datetime.timedelta e dateutil.relativetimedelta (calculando datas)

```
    # datetime.timedelta e dateutil.relativetimedelta (calculando datas)
# Docs:
# https://dateutil.readthedocs.io/en/stable/relativedelta.html
# https://docs.python.org/3/library/datetime.html#timedelta-objects
from datetime import datetime

from dateutil.relativedelta import relativedelta

fmt = '%d/%m/%Y %H:%M:%S'
data_inicio = datetime.strptime('20/04/1987 09:30:30', fmt)
data_fim = datetime.strptime('12/12/2022 08:20:20', fmt)
# delta = timedelta(days=10, hours=2)
delta = relativedelta(data_fim, data_inicio)
print(delta.days, delta.years)
# print(data_fim - delta)
# print(data_fim)
# print(data_fim + relativedelta(seconds=60, minutes=10))

# delta = data_fim - data_inicio
# print(delta.days, delta.seconds, delta.microseconds)
# print(delta.total_seconds())
# print(data_fim > data_inicio)
# print(data_fim < data_inicio)
# print(data_fim == data_inicio)
```

1.3 Formatando datas do datetime com strftime no Python main

```
    Formatando datas do datetime
    # datetime.strftime('DATA', 'FORMATO')
    # https://docs.python.org/3/library/datetime.html
```

```
    from datetime import datetime

    # data = datetime(2022, 12, 13, 7, 59, 23)
    data = datetime.strptime('2022-12-13 07:59:23', '%Y-%m-%d %H:%M:%S')
    print(data.strftime('%d/%m/%Y'))
    print(data.strftime('%d/%m/%Y %H:%M'))
    print(data.strftime('%d/%m/%Y %H:%M:%S'))
    print(data.strftime('%Y'), data.year)
    print(data.strftime('%d'), data.day)
    print(data.strftime('%m'), data.month)
    print(data.strftime('%H'), data.hour)
    print(data.strftime('%M'), data.minute)
    print(data.strftime('%S'), data.second)
```

2 Usando calendar para calendários e datas

```
    # Usando calendar para calendários e datas
# https://docs.python.org/3/library/calendar.html
# calendar e usado para coisas genericas de calendarios e datas.
# Com calendar, voce pode saber coisas como:
# - Qual o ultimo dia do mes (ex.: monthrange)
# - Qual o nome e numero do dia de determinada data (ex.: weekday)
# - Criar um calendário em si (ex.: monthcalendar)
# - Trabalhar com coisas especificas de calendarios (ex.: calendar, month)
# Por padrao dia da semana começa em 0 ate 6
# 0 = segunda-feira | 6 = domingo
import calendar

# print(calendar.calendar(2022))
# print(calendar.month(2022, 12))
# numero_primeiro_dia, ultimo_dia = calendar.monthrange(2022, 12)
# print(list(enumerate(calendar.day_name)))
# print(calendar.day_name[numero_primeiro_dia])
# print(calendar.day_name[calendar.weekday(2022, 12, ultimo_dia)])
for week in calendar.monthcalendar(2022, 12):
    for day in week:
        if day == 0:
            continue
        print(day)
```

3 locale para internacionalização (tradução)

```
    # locale para internacionalização (tradução)
# https://docs.python.org/3/library/locale.html
# https://learn.microsoft.com/fr-fr/powershell/module/international/get-winsystemlocale?
   view=windowsserver2022-ps&viewFallbackFrom=win10-ps
import calendar
import locale

locale.setlocale(locale.LC_ALL, '')

print(calendar.calendar(2022))
```

4 O módulo os para interação com o sistema

```
# Doc: https://docs.python.org/3/library/os.html
# O modulo `os` fornece funcoes para interagir com o sistema operacional.
# Por exemplo, o modulo os.path contem funesco para trabalhar com caminhos de
# arquivos e a função os.listdir() pode ser usada para listar os arquivos em um
# diretorio. O metodo os.system() permite executar comandos do sistema
# operacional a partir do seu codigo Python.
# Windows 11 (PowerShell), Linux, Mac = clear
# Windows (antigo, cmd) = cls
import os

os.system('clear')
os.system('echo "Hello world"')

print('a' * 80)
print('a' * 80)
print('a' * 80)
print('a' * 80)
print('a' * 80)
print('a' * 80)
```

4.1 os.path trabalha com caminhos em Windows, Linux e Mac

```
# Doc: https://docs.python.org/3/library/os.path.html#module-os.path
# os.path e um modulo que fornece funcoes para trabalhar com caminhos de
# arquivos em Windows, Mac ou Linux sem precisar se preocupar com as diferenças
# entre esses sistemas.
# Exemplos do os.path:
# os.path.join: junta strings em um unico caminho. Desse modo,
# os.path.join('pasta1', 'pasta2', 'arquivo.txt') retornaria
# 'pasta1/pasta2/arquivo.txt' no Linux ou Mac, e
# 'pasta1\pasta2\arquivo.txt' no Windows.
# os.path.split: divide um caminho uma tupla (diretorio, arquivo).
# Por exemplo, os.path.split('/home/user/arquivo.txt')
# retornaria ('/home/user', 'arquivo.txt').
# os.path.exists: verifica se um caminho especificado existe.
# os.path so trabalha com caminhos de arquivos e nao faz nenhuma
# operacao de entrada/saida (I/O) com arquivos em si.
import os

caminho = os.path.join('Desktop', 'curso', 'arquivo.txt')
# print(caminho)
diretorio, arquivo = os.path.split(caminho)
nome_arquivo, extensao_arquivo = os.path.splitext(arquivo)
# print(nome_arquivo, extensao_arquivo)
# print(os.path.exists('/Users/luizotavio/Desktop/curso-python-rep'))
# print(os.path.abspath('.'))
print(caminho)
print(os.path.basename(caminho))
print(os.path.basename(diretorio))
print(os.path.dirname(caminho))
```

4.2 os.listdir para navegar em caminhos

```
# /Users/luizotavio/Desktop/EXEMPLO
# C:\Users\luizotavio\Desktop\EXEMPLO
# caminho = r'C:\\Users\\luizotavio\\Desktop\\EXEMPLO'
import os

caminho = os.path.join('/Users', 'luizotavio', 'Desktop', 'EXEMPLO')

for pasta in os.listdir(caminho):
    caminho_completo_pasta = os.path.join(caminho, pasta)
    print(pasta)

    if not os.path.isdir(caminho_completo_pasta):
        continue

    for imagem in os.listdir(caminho_completo_pasta):
        print('  ', imagem)
```

4.3 os.walk para navegar de caminhos de forma recursiva

```
# os.walk e uma funcao que permite percorrer uma estrutura de diretorios de
# maneira recursiva. Ela gera uma sequencia de tuplas, onde cada tupla possui
# tres elementos: o diretorio atual (root), uma lista de subdiretorios (dirs)
# e uma lista dos arquivos do diretorio atual (files).
import os
from itertools import count

caminho = os.path.join('/Users', 'luizotavio', 'Desktop', 'EXEMPLO')
counter = count()

for root, dirs, files in os.walk(caminho):
    the_counter = next(counter)
    print(the_counter, 'Pasta atual', root)

    for dir_ in dirs:
        print('  ', the_counter, 'Dir:', dir_)

    for file_ in files:
        caminho_completo_arquivo = os.path.join(root, file_)
        print('  ', the_counter, 'FILE:', caminho_completo_arquivo)
        # NAO FACA ISSO (VAI APAGAR TUDO DA PASTA)
        # os.unlink(caminho_completo_arquivo)
```

4.4 os.path.getsize e os.stat para dados dos arquivos (tamanho em bytes)

```
import math
import os
from itertools import count


def formata_tamanho(tamanho_em_bytes: int, base: int = 1000) -> str:
    """Formata um tamanho, de bytes para o tamanho apropriado"""

    # Original:
    # https://stackoverflow.com/questions/5194057/better-way-to-convert-file-sizes-in-
       python

    # Se o tamanho for menor ou igual a 0, 0B.
    if tamanho_em_bytes <= 0:
        return "0B"

    # Tupla com os tamanhos
    #                       0    1     2     3     4     5
    abreviacao_tamanhos = "B", "KB", "MB", "GB", "TB", "PB"
    # Logaritmo -> https://brasilescola.uol.com.br/matematica/logaritmo.htm
    # math.log vai retornar o logaritmo do tamanho_em_bytes
    # com a base (1000 por padrão), isso deve bater
    # com o nosso indice na abreviação dos tamanhos
    indice_abreviacao_tamanhos = int(math.log(tamanho_em_bytes, base))
    # Por quanto nosso tamanho deve ser dividido para
    # gerar o tamanho correto.
    potencia = base ** indice_abreviacao_tamanhos
    # Nosso tamanho final
    tamanho_final = tamanho_em_bytes / potencia
    # A abreviação que queremos
    abreviacao_tamanho = abreviacao_tamanhos[indice_abreviacao_tamanhos]
    return f'{tamanho_final:.2f} {abreviacao_tamanho}'


caminho = os.path.join('/Users', 'luizotavio', 'Desktop', 'EXEMPLO')
counter = count()

for root, dirs, files in os.walk(caminho):
    the_counter = next(counter)
    print(the_counter, 'Pasta atual', root)

    for dir_ in dirs:
        print('  ', the_counter, 'Dir:', dir_)

    for file_ in files:
        caminho_completo_arquivo = os.path.join(root, file_)
        # tamanho = os.path.getsize(caminho_completo_arquivo)
        stats = os.stat(caminho_completo_arquivo)
        tamanho = stats.st_size
        print('  ', the_counter, 'FILE:', file_, formata_tamanho(tamanho))
        # NAO FACA ISSO (VAI APAGAR TUDO DA PASTA)
        # os.unlink(caminho_completo_arquivo)
```

4.5 os + shutil - Copiando arquivos com Python

```
# Vamos copiar arquivos de uma pasta para outra.
# Copiar -> shutil.copy
import os
import shutil

HOME = os.path.expanduser('~')
DESKTOP = os.path.join(HOME, 'Desktop')
PASTA_ORIGINAL = os.path.join(DESKTOP, 'EXEMPLO')
NOVA_PASTA = os.path.join(DESKTOP, 'NOVA_PASTA')

os.makedirs(NOVA_PASTA, exist_ok=True)

for root, dirs, files in os.walk(PASTA_ORIGINAL):
    for dir_ in dirs:
        caminnho_novo_diretorio = os.path.join(
            root.replace(PASTA_ORIGINAL, NOVA_PASTA), dir_
        )
        os.makedirs(caminnho_novo_diretorio, exist_ok=True)

    for file in files:
        caminho_arquivo = os.path.join(root, file)
        caminnho_novo_arquivo = os.path.join(
```

```
            root.replace(PASTA_ORIGINAL, NOVA_PASTA), file
        )
        shutil.copy(caminho_arquivo, caminnho_novo_arquivo)
```

4.6 os + shutil - Apagando, copiando, movendo e renomeando pastas com Python

```
# Vamos copiar arquivos de uma pasta para outra.
# Copiar -> shutil.copy
# Copiar Arvore recursivamente -> shutil.copytree
# Apagar Arvore recursivamente -> shutil.rmtree
# Apagar arquivos -> os.unlink
# Renomear/Mover -> shutil.move ou os.rename
import os
import shutil

HOME = os.path.expanduser('~')
DESKTOP = os.path.join(HOME, 'Desktop')
PASTA_ORIGINAL = os.path.join(DESKTOP, 'EXEMPLO')
NOVA_PASTA = os.path.join(DESKTOP, 'NOVA_PASTA')

shutil.rmtree(NOVA_PASTA, ignore_errors=True)
shutil.copytree(PASTA_ORIGINAL, NOVA_PASTA)
# shutil.move(NOVA_PASTA, NOVA_PASTA + '_EITA')
shutil.rmtree(NOVA_PASTA, ignore_errors=True)

# os.makedirs(NOVA_PASTA, exist_ok=True)

# for root, dirs, files in os.walk(PASTA_ORIGINAL):
#     for dir_ in dirs:
#         caminnho_novo_diretorio = os.path.join(
#             root.replace(PASTA_ORIGINAL, NOVA_PASTA), dir_
#         )
#         os.makedirs(caminnho_novo_diretorio, exist_ok=True)

#     for file in files:
#         caminho_arquivo = os.path.join(root, file)
#         caminnho_novo_arquivo = os.path.join(
#             root.replace(PASTA_ORIGINAL, NOVA_PASTA), file
#         )
#         shutil.copy(caminho_arquivo, caminnho_novo_arquivo)
```

5 JSON

```
    "python.analysis.diagnosticSeverityOverrides": {},
  // "python.defaultInterpreterPath": "./venv/bin/python",
  "python.analysis.typeCheckingMode": "basic",
  "cSpell.enabled": false
  "cSpell.enabled": true
```

```
    {
  "title": "O Senhor dos Aneis: A Sociedade do Anel",
  "original_title": "The Lord of the Rings: The Fellowship of the Ring",
  "is_movie": true,
  "imdb_rating": 8.8,
  "year": 2001,
  "characters": ["Frodo", "Sam", "Gandalf", "Legolas", "Boromir"],
  "budget": null
}
```

```
    # O que e JSON - JavaScript Object Notation
# JSON - JavaScript Object Notation (extensão .json)
# E uma estrutura de dados que permite a serializacao
# de objetos em texto simples para facilitar a transmissão de
# dados atraves da rede, APIs web ou outros meios de comunicacao.
# O JSON suporta os seguintes tipos de dados:
# Numeros: podem ser inteiros ou com ponto flutuante, como 42 ou 3.14
# Strings: são cadeias de caracteres, como "Olá, mundo!" ou "12345"
#   As strings devem ser envolvidas por aspas duplas
# Booleanos: são os valores verdadeiro (true) ou falso (false)
# Arrays: são listas ordenadas de valores, como [1, 2, 3] ou
#   ["Oi", "Olá", "Bom dia"]
# Objetos: são conjuntos de pares chave/valor -> {"nome": "João", "idade": 30}
# null: e um valor especial que representa ausencia de valor
#
# Ao converter de Python para JSON:
# Python        JSON
# dict          object
# list, tuple   array
# str           string
```

```
# int, float    number
# True          true
# False         false
# None          null
```

5.1 json.dumps e json.loads com strings + typing.TypedDict

```
    "python.testing.pytestEnabled": true,
  "python.analysis.diagnosticSeverityOverrides": {},
  // "python.defaultInterpreterPath": "./venv/bin/python",
  "python.analysis.typeCheckingMode": "basic",
  "python.analysis.typeCheckingMode": "strict",
  "cSpell.enabled": true
```

```
    # json.dumps e json.loads com strings + typing.TypedDict
# Ao converter de Python para JSON:
# Python        JSON
# dict          object
# list, tuple   array
# str           string
# int, float    number
# True          true
# False         false
# None          null
import json
# from pprint import pprint
from typing import TypedDict


class Movie(TypedDict):
    title: str
    original_title: str
    is_movie: bool
    imdb_rating: float
    year: int
    characters: list[str]
    budget: None | float


string_json = '''
{
  "title": "O Senhor dos Aneis: A Sociedade do Anel",
  "original_title": "The Lord of the Rings: The Fellowship of the Ring",
  "is_movie": true,
  "imdb_rating": 8.8,
  "year": 2001,
  "characters": ["Frodo", "Sam", "Gandalf", "Legolas", "Boromir"],
  "budget": null
}
'''
filme: Movie = json.loads(string_json)
# pprint(filme, width=40)
# print(filme['title'])
# print(filme['characters'][0])
# print(filme['year'] + 10)

json_string = json.dumps(filme, ensure_ascii=False, indent=2)
print(json_string)
```

5.2 json.dump e json.load com arquivos

```
    {
  "title": "O Senhor dos Aneis: A Sociedade do Anel",
  "original_title": "The Lord of the Rings: The Fellowship of the Ring",
  "is_movie": true,
  "imdb_rating": 8.8,
  "year": 2001,
  "characters": [
    "Frodo",
    "Sam",
    "Gandalf",
    "Legolas",
    "Boromir"
  ],
  "budget": null
}
```

```
    # json.dump e json.load com arquivos
import json
import os
```

```
NOME_ARQUIVO = 'aula177.json'
CAMINHO_ABSOLUTO_ARQUIVO = os.path.abspath(
    os.path.join(
        os.path.dirname(__file__),
        NOME_ARQUIVO
    )
)

filme = {
    'title': 'O Senhor dos Aneis: A Sociedade do Anel',
    'original_title': 'The Lord of the Rings: The Fellowship of the Ring',

    'is_movie': True,
    'imdb_rating': 8.8,
    'year': 2001,
    'characters': ['Frodo', 'Sam', 'Gandalf', 'Legolas', 'Boromir'],
    'budget': None
}

with open(CAMINHO_ABSOLUTO_ARQUIVO, 'w') as arquivo:
    json.dump(filme, arquivo, ensure_ascii=False, indent=2)

with open(CAMINHO_ABSOLUTO_ARQUIVO, 'r') as arquivo:
    filme_do_json = json.load(arquivo)
    print(filme_do_json)
```

6 CSV (Comma Separated Values - Valores separados por vírgulas) main

```
    Nome,Idade,Endereço
    Luiz Otávio,32,"Av Brasil, 21, Centro"
    João da Silva,55,"Rua 22, 44, Nova Era"

    Nome,Idade,Endereço
    Luiz Otávio,32,"Av Brasil, 21, ""Centro"""
    João da Silva,55,"Rua 22, 44, Nova Era"

    # CSV (Comma Separated Values - Valores separados por virgulas)
    # E um formato de arquivo que armazena dados em forma de tabela, onde cada
    # linha representa uma linha da tabela e as colunas sao separadas por virgulas.
    # Ele e amplamente utilizado para transferir dados entre sistemas de diferentes
    # plataformas, como por exemplo, para importar ou exportar dados para uma
    # planilha (Google Sheets, Excel, LibreOffice Calc) ou para uma base de dados.
    # Um arquivo CSV geralmente tem a extensão ".csv" e pode ser aberto em um
    # editor de texto ou em uma planilha eletronica.
    # Um exemplo de um arquivo CSV pode ser:
    # Nome,Idade,Endereço
    # Luiz Otávio,32,"Av Brasil, 21, Centro"
    # João da Silva,55,"Rua 22, 44, Nova Era"
    # A primeira linha do arquivo define os nomes das colunas da, enquanto as
    # linhas seguintes contem os valores das linhas, separados por virgulas.
    # Regras simples do CSV
    # 1 - Separe os valores das colunas com um delimitador unico (,)
    # 2 - Cada registro deve estar em uma linha
    # 3 - Não deixar linhas ou espaços sobrando
    # 4 - Use o caractere de escape (") quando o delimitador aparecer no valor.
```

6.1 csv.reader e csv.DictReader

```
    Nome,Idade,Endereço
Luiz Otávio,32,"Av Brasil, 21, ""Centro"""
João da Silva,55,"Rua 22, 44, Nova Era"

    # csv.reader e csv.DictReader
# csv.reader le o CSV em formato de lista
# csv.DictReader le o CSV em formato de dicionario
import csv
from pathlib import Path

CAMINHO_CSV = Path(__file__).parent / 'aula179.csv'

with open(CAMINHO_CSV, 'r') as arquivo:
    leitor = csv.DictReader(arquivo)

    for linha in leitor:
        print(linha['Nome'], linha['Idade'], linha['Endereço'])

# with open(CAMINHO_CSV, 'r') as arquivo:
#     leitor = csv.reader(arquivo)
```

```
#     for linha in leitor:
#         print(linha)
```

6.2 csv.writer e csv.DictWriter para escrever em CSV

```
    Luiz Otávio,"Av 1, 22"
João Silva,"R. 2, ""1"""
Maria Sol,"Av B, 3A"

    # csv.writer e csv.DictWriter para escrever em CSV
# csv.reader le o CSV em formato de lista
# csv.DictReader le o CSV em formato de dicionario
import csv
from pathlib import Path

CAMINHO_CSV = Path(__file__).parent / 'aula180.csv'

lista_clientes = [
    {'Nome': 'Luiz Otávio', 'Endereço': 'Av 1, 22'},
    {'Nome': 'João Silva', 'Endereço': 'R. 2, "1"'},
    {'Nome': 'Maria Sol', 'Endereço': 'Av B, 3A'},
]

with open(CAMINHO_CSV, 'w') as arquivo:
    nome_colunas = lista_clientes[0].keys()
    escritor = csv.DictWriter(
        arquivo,
        fieldnames=nome_colunas
    )
    escritor.writeheader()

    for cliente in lista_clientes:
        print(cliente)
        escritor.writerow(cliente)


# lista_clientes = [
#     ['Luiz Otávio', 'Av 1, 22'],
#     ['João Silva', 'R. 2, "1"'],
#     ['Maria Sol', 'Av B, 3A'],
# ]
# with open(CAMINHO_CSV, 'w') as arquivo:
#     # nome_colunas = lista_clientes[0].keys()
#     nome_colunas = ['Nome', 'Endereço']
#     escritor = csv.writer(arquivo)

#     escritor.writerow(nome_colunas)

#     for cliente in lista_clientes:
#         escritor.writerow(cliente)
```

7 random tem geradores de números pseudoaleatórios

```
    # random tem geradores de numeros pseudoaleatorios
# Obs.: numeros pseudoaleatorios significa que os numeros
# parecem ser aleatorios, mas na verdade não são. Portanto,
# este modulo nao deve ser usado para seguranca ou uso criptografico.
# O motivo disso e que quando temos uma mesma entrada e um mesmo algoritimo,
# a saida pode ser previsivel.
# doc: https://docs.python.org/pt-br/3/library/random.html
import random

# Funcoes:
# seed
#   -> Inicializa o gerador de random (por isso "numeros pseudoaleatorios")
# random.seed(0)

# random.randrange(inicio, fim, passo)
#   -> Gera um numero inteiro aleatorio dentro de um intervalo especifico
r_range = random.randrange(10, 20, 2)
# print(r_range)

# random.randint(inicio, fim)
#   -> Gera um numero inteiro aleatorio dentro de um intervalo "sem passo"
r_int = random.randint(10, 20)
# print(r_int)

# random.uniform(inicio, fim)
#   -> Gera um numero flutuante aleatorio dentro de um intervalo "sem passo"
r_uniform = random.uniform(10, 20)
```

```
# print(r_uniform)

# random.shuffle(SequenciaMutável) -> Embaralha a lista original
nomes = ['Luiz', 'Maria', 'Helena', 'Joana']
# random.shuffle(nomes)
# print(nomes)

# random.sample(Iterável, k=N)
#   -> Escolhe elementos do iterável e retorna outro iterável (não repete)
novos_nomes = random.sample(nomes, k=3)
# print(nomes)
# print(novos_nomes)

# random.choices(Iterável, k=N)
#   -> Escolhe elementos do iterável e retorna outro iterável (repete valores)
novos_nomes = random.choices(nomes, k=3)
print(nomes)
print(novos_nomes)

# random.choice(Iterável) -> Escolhe um elemento do iterável
print(random.choice(nomes))
```

7.1 part II

```
    # random tem geradores de numeros pseudoaleatorios
# Obs.: numeros pseudoaleatorios significa que os numeros
# parecem ser aleatorios, mas na verdade nao sao. Portanto,
# este modulo nao deve ser usado para seguranca ou uso criptografico.
# O motivo disso e que quando temos uma mesma entrada e um mesmo algoritimo,
# a saida pode ser previsivel.
# doc: https://docs.python.org/pt-br/3/library/random.html
import random

# Funcoes:
# seed
#   -> Inicializa o gerador de random (por isso "numeros pseudoaleatorios")
# random.seed(0)

# random.randrange(inicio, fim, passo)
#   -> Gera um numero inteiro aleatorio dentro de um intervalo especifico
r_range = random.randrange(10, 20, 2)
# print(r_range)

# random.randint(inicio, fim)
#   -> Gera um numero inteiro aleatorio dentro de um intervalo "sem passo"
r_int = random.randint(10, 20)
# print(r_int)

# random.uniform(inicio, fim)
#   -> Gera um numero flutuante aleatorio dentro de um intervalo "sem passo"
r_uniform = random.uniform(10, 20)
# print(r_uniform)

# random.shuffle(SequenciaMutável) -> Embaralha a lista original
nomes = ['Luiz', 'Maria', 'Helena', 'Joana']
# random.shuffle(nomes)
# print(nomes)

# random.sample(Iterável, k=N)
#   -> Escolhe elementos do iterável e retorna outro iterável (não repete)
novos_nomes = random.sample(nomes, k=3)
# print(nomes)
# print(novos_nomes)

# random.choices(Iterável, k=N)
#   -> Escolhe elementos do iterável e retorna outro iterável (repete valores)
novos_nomes = random.choices(nomes, k=3)
print(nomes)
print(novos_nomes)

# random.choice(Iterável) -> Escolhe um elemento do iterável
print(random.choice(nomes))
```

7.2 secrets gera números aleatórios seguros

```
import secrets

# import string as s
# from secrets import SystemRandom as Sr
```

```
# print(''.join(Sr().choices(s.ascii_letters + s.digits + s.punctuation, k=64)))
# python -c "import string as s;from secrets import SystemRandom as Sr; print(''.join(Sr
    ().choices(s.ascii_letters + s.punctuation + s.digits,k=12)))"

random = secrets.SystemRandom()

# print(secrets.randbelow(100))
# print(secrets.choice([10, 11, 12]))

# Funcoes:
# seed
#   -> NAO FAZ NADA
random.seed(10)

# random.randrange(inicio, fim, passo)
#   -> Gera um numero inteiro aleatorio dentro de um intervalo especifico
r_range = random.randrange(10, 20, 2)
print(r_range)

# random.randint(inicio, fim)
#   -> Gera um numero inteiro aleatorio dentro de um intervalo "sem passo"
r_int = random.randint(10, 20)
print(r_int)

# random.uniform(inicio, fim)
#   -> Gera um numero flutuante aleatorio dentro de um intervalo "sem passo"
r_uniform = random.uniform(10, 20)
print(r_uniform)

# random.shuffle(SequenciaMutável) -> Embaralha a lista original
nomes = ['Luiz', 'Maria', 'Helena', 'Joana']
# random.shuffle(nomes)
print(nomes)

# random.sample(Iterável, k=N)
#   -> Escolhe elementos do iterável e retorna outro iterável (não repete)
novos_nomes = random.sample(nomes, k=3)
print(nomes)
print(novos_nomes)

# random.choices(Iterável, k=N)
#   -> Escolhe elementos do iterável e retorna outro iterável (repete valores)
novos_nomes = random.choices(nomes, k=3)
print(nomes)
print(novos_nomes)

# random.choice(Iterável) -> Escolhe um elemento do iterável
print(random.choice(nomes))
```

8 string.Template para substituir variáveis em textos

```
    # string.Template para substituir variáveis em textos
# doc: https://docs.python.org/3/library/string.html#template-strings
# Metodos:
# substitute: substitui mas gera erros se faltar chaves
# safe_substitute: substitui sem gerar erros
# Voce tambem pode trocar o delimitador e outras coisas criando uma subclasse
# de template.
import locale
import string
from datetime import datetime
from pathlib import Path

CAMINHO_ARQUIVO = Path(__file__).parent / 'aula183.txt'

locale.setlocale(locale.LC_ALL, '')


def converte_para_brl(numero: float) -> str:
    brl = 'R$ ' + locale.currency(numero, symbol=False, grouping=True)
    return brl


data = datetime(2022, 12, 28)
dados = dict(
    nome='João',
    valor=converte_para_brl(1_234_456),
    data=data.strftime('%d/%m/%Y'),
    empresa='O. M.',
```

```
    telefone='+55 (11) 7890-5432'
)


class MyTemplate(string.Template):
    delimiter = '%'


with open(CAMINHO_ARQUIVO, 'r') as arquivo:
    texto = arquivo.read()
    template = MyTemplate(texto)
    print(template.substitute(dados))

    Prezado(a) %nome,

Informamos que sua mensalidade será cobrada no valor de %{valor} no dia %data. Caso
   deseje cancelar o serviço, entre em contato com a %empresa pelo telefone %telefone.

Atenciosamente,

%{empresa},
```

9 Variáveis de ambiente com Python

```
BD_USER="CHANGE-ME"
BD_PASSWORD="CHANGE-ME"
BD_PORT=CHANGE-ME
BD_HOST="CHANGE-ME"


# Para variáveis de ambiente
# Windows PS: $env:VARIAVEL="VALOR" | dir env:
# Linux e Mac: export NOME_VARIAVEL="VALOR" | echo $VARIAVEL
# Para obter o valor das variáveis de ambiente
# os.getenv ou os.environ['VARIAVEL']
# Para configurar variáveis de ambiente
# os.environ['VARIAVEL'] = 'valor'
# Ou usando python-dotenv e o arquivo .env
# pip install python-dotenv
# from dotenv import load_dotenv
# load_dotenv()
# https://pypi.org/project/python-dotenv/
# OBS.: sempre lembre-se de criar um .env-example
import os

from dotenv import load_dotenv  # type: ignore

load_dotenv()

# print(os.environ)
print(os.getenv('BD_PASSWORD'))
```

requirements.txt

```
pycodestyle==2.10.0
pyflakes==3.0.1
python-dateutil==2.8.2
python-dotenv==0.21.0
pytz==2022.6
six==1.16.0
tomli==2.0.1
```

9.1 part II

```
    BD_USER="CHANGE-ME"
BD_PASSWORD="CHANGE-ME"
BD_PORT=CHANGE-ME
BD_HOST="CHANGE-ME"

    # Variáveis de ambiente com Python
# Para variáveis de ambiente
# Windows PS: $env:VARIAVEL="VALOR" | dir env:
# Linux e Mac: export NOME_VARIAVEL="VALOR" | echo $VARIAVEL
# Para obter o valor das variáveis de ambiente
# os.getenv ou os.environ['VARIAVEL']
# Para configurar variáveis de ambiente
# os.environ['VARIAVEL'] = 'valor'
# Ou usando python-dotenv e o arquivo .env
# pip install python-dotenv
# from dotenv import load_dotenv
# load_dotenv()
```

```
# https://pypi.org/project/python-dotenv/
# OBS.: sempre lembre-se de criar um .env-example
import os

from dotenv import load_dotenv  # type: ignore

load_dotenv()

# print(os.environ)
print(os.getenv('BD_PASSWORD'))

pycodestyle==2.10.0
pyflakes==3.0.1
python-dateutil==2.8.2
python-dotenv==0.21.0
pytz==2022.6
six==1.16.0
tomli==2.0.1
```

9.2 Variáveis de ambiente com Python

```
    BD_USER="CHANGE-ME"
BD_PASSWORD="CHANGE-ME"
BD_PORT=CHANGE-ME
BD_HOST="CHANGE-ME"

    # Variáveis de ambiente com Python
# Para variáveis de ambiente
# Windows PS: $env:VARIAVEL="VALOR" | dir env:
# Linux e Mac: export NOME_VARIAVEL="VALOR" | echo $VARIAVEL
# Para obter o valor das variáveis de ambiente
# os.getenv ou os.environ['VARIAVEL']
# Para configurar variáveis de ambiente
# os.environ['VARIAVEL'] = 'valor'
# Ou usando python-dotenv e o arquivo .env
# pip install python-dotenv
# from dotenv import load_dotenv
# load_dotenv()
# https://pypi.org/project/python-dotenv/
# OBS.: sempre lembre-se de criar um .env-example
import os

from dotenv import load_dotenv  # type: ignore

load_dotenv()

# print(os.environ)
print(os.getenv('BD_PASSWORD'))

    pycodestyle==2.10.0
pyflakes==3.0.1
python-dateutil==2.8.2
python-dotenv==0.21.0
pytz==2022.6
six==1.16.0
tomli==2.0.1
```

10 Enviando E-mails SMTP com Python

.env-exeample Esse codigo depende de configurações advidas do gmail password

```
    BD_PASSWORD="CHANGE-ME"
BD_PORT=CHANGE-ME
BD_HOST="CHANGE-ME"

FROM_EMAIL="CHANGE-ME"
EMAIL_PASSWORD="CHANGE-ME"
```

aula185.html

```
    !DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Arquivo para o e-mail</title>
  </head>
  <body>
    Olá ${nome},
    <br />
    Estou testando
```

```
    <span style="color: red; font-weight: bold;">este e-mail</span>
    em HTML.
    <br />
    <br />

    <em>
      Atenciosamente,
      <br />
      Luiz Otávio.
    </em>
  </body>
</html>
```

```
    # Enviando E-mails SMTP com Python
import os
import pathlib
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from string import Template

from dotenv import load_dotenv  # type: ignore

load_dotenv()

# Caminho arquivo HTML
CAMINHO_HTML = pathlib.Path(__file__).parent / 'aula185.html'

# Dados do remetente e destinatário
remetente = os.getenv('FROM_EMAIL', '')
destinatario = remetente

# Configuracoes SMTP
smtp_server = 'smtp.gmail.com'
smtp_port = 587
smtp_username = os.getenv('FROM_EMAIL', '')
smtp_password = os.getenv('EMAIL_PASSWORD', '')

# Mensagem de texto
with open(CAMINHO_HTML, 'r') as arquivo:
    texto_arquivo = arquivo.read()
    template = Template(texto_arquivo)
    texto_email = template.substitute(nome='Helena')

# Transformar nossa mensagem em MIMEMultipart
mime_multipart = MIMEMultipart()
mime_multipart['from'] = remetente
mime_multipart['to'] = destinatario
mime_multipart['subject'] = 'Este e o assunto do e-mail'

corpo_email = MIMEText(texto_email, 'html', 'utf-8')
mime_multipart.attach(corpo_email)

# Envia o e-mail
with smtplib.SMTP(smtp_server, smtp_port) as server:
    server.ehlo()
    server.starttls()
    server.login(smtp_username, smtp_password)
    server.send_message(mime_multipart)
    print('E-mail enviado com  sucesso!')
```

11 ZIP - Compactando / Descompactando arquivos com zipfile.ZipFile

```
import os
import shutil
from pathlib import Path
from zipfile import ZipFile

# Caminhos
CAMINHO_RAIZ = Path(__file__).parent
CAMINHO_ZIP_DIR = CAMINHO_RAIZ / 'aula_186_diretorio_zip'
CAMINHO_COMPACTADO = CAMINHO_RAIZ / 'aula186_compactado.zip'
CAMINHO_DESCOMPACTADO = CAMINHO_RAIZ / 'aula186_descompactado'

shutil.rmtree(CAMINHO_ZIP_DIR, ignore_errors=True)
Path.unlink(CAMINHO_COMPACTADO, missing_ok=True)
shutil.rmtree(str(CAMINHO_COMPACTADO).replace('.zip', ''), ignore_errors=True)
shutil.rmtree(CAMINHO_DESCOMPACTADO, ignore_errors=True)
```

```
# raise Exception()

# Cria o diretorio para a aula
CAMINHO_ZIP_DIR.mkdir(exist_ok=True)


def criar_arquivos(qtd: int, zip_dir: Path):
    for i in range(qtd):
        texto = 'arquivo_%s' % i
        with open(zip_dir / f'{texto}.txt', 'w') as arquivo:
            arquivo.write(texto)


criar_arquivos(10, CAMINHO_ZIP_DIR)
```

11.1 ZIP - Compactando / Descompactando arquivos com zipfile.ZipFile

```
    criar_arquivos(10, CAMINHO_ZIP_DIR)

    # Criando um zip e adicionando arquivos
    with ZipFile(CAMINHO_COMPACTADO, 'w') as zip:
        for root, dirs, files in os.walk(CAMINHO_ZIP_DIR):
            for file in files:
                # print(file)
                zip.write(os.path.join(root, file), file)

    # Lendo arquivos de um zip
    with ZipFile(CAMINHO_COMPACTADO, 'r') as zip:
        for arquivo in zip.namelist():
            print(arquivo)

    # Extraindo arquivos de um zip
    with ZipFile(CAMINHO_COMPACTADO, 'r') as zip:
        zip.extractall(CAMINHO_DESCOMPACTADO)
```

12 sys.argv - Executando arquivos com argumentos no sistema

```
    "code-runner.ignoreSelection": true,
  "editor.fontFamily": "'Fira Code', Consolas, 'Dank Mono', 'Source Code Pro', 'Fira
     Code', Menlo, 'Inconsolata', 'Droid Sans Mono', 'DejaVu Sans Mono', 'Ubuntu Mono',
     'Courier New', Courier, Monaco, monospace",
  "terminal.integrated.fontFamily": "",
  "editor.fontLigatures": false,
  "[python]": {
    "editor.defaultFormatter": "ms-python.python",
    "editor.tabSize": 4,

    # sys.argv - Executando arquivos com argumentos no sistema
# Fonte = Fira Code
import sys

argumentos = sys.argv
qtd_argumentos = len(argumentos)

if qtd_argumentos <= 1:
    print('Voce nao passou argumentos')
else:
    try:
        print(f'Voce passou os argumentos {argumentos[1:]}')
        print(f'Faça alguma coisa com {argumentos[1]}')
        print(f'Faça outra coisa com {argumentos[2]}')
    except IndexError:
        print('Faltam argumentos')
```

13 argparse.ArgumentParser para argumentos mais complexos

```
  "window.zoomLevel": 2,
  "window.zoomLevel": 3,
  "editor.fontSize": 24,
  "editor.hover.enabled": true,
  "workbench.startupEditor": "none",

    # argparse.ArgumentParser para argumentos mais complexos
# Tutorial Oficial:
# https://docs.python.org/pt-br/3/howto/argparse.html
from argparse import ArgumentParser
```

```
parser = ArgumentParser()

parser.add_argument(
    '-b', '--basic',
    help='Mostra "Olá mundo" na tela',
    # type=str, # Tipo do argumento
    metavar='STRING',
    # default='Olá mundo', # Valor padrão
    required=False,
    action='append',  # Recebe o argumento mais de uma vez
    # nargs='+', # Recebe mais de um valor
)
parser.add_argument(
    '-v', '--verbose',
    help='Mostra logs',
    action='store_true'
)
args = parser.parse_args()

if args.basic is None:
    print('Voce nao passou o valor de b.')
    print(args.basic)
else:
    print('O valor de basic:', args.basic)

print(args.verbose)
```

14 Básico do protocolo HTTP (HyperText Transfer Protocol)

```
    # (Parte 1) Básico do protocolo HTTP (HyperText Transfer Protocol)
# HTTP (HyperText Transfer Protocol) e um protocolo usado enviar e receber
# dados na Internet. Ele funciona no modo cliente/servidor, onde o cliente
# (seu navegador, por exemplo) faz uma requisição ao servidor
# (site, por exemplo), que responde com os dados adequados.
#
# A mensagem de requisição do cliente deve incluir dados como:
# - O metodo HTTP
#     - leitura (safe) - GET, HEAD (cabeçalhos), OPTIONS (metodos suportados)
#     - escrita - POST, PUT (substitui), PATCH (atualiza), DELETE
# - O endereço do recurso a ser acessado (/users/)
# - Os cabeçalhos HTTP (Content-Type, Authorization)
# - O Corpo da mensagem (caso necessario, de acordo com o metodo HTTP)
#
# A mensagem de resposta do servidor deve incluir dados como:
# - codigo de status HTTP (200 success, 404 Not found, 301 Moved Permanently)
# https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
# - Os cabeçalhos HTTP (Content-Type, Accept)
# - O corpo da mensagem (Pode estar em vazio em alguns casos)
```

15 requests para requisições HTTP com Python (entenda request e response)

```
    # requests para requisicoes HTTP
# Tutorial -> https://youtu.be/Qd8JT0bnJGs
import requests

# http:// -> 80
# https:// -> 443
url = 'http://localhost:3333/'
response = requests.get(url)

print(response.status_code)
# print(response.headers)
# print(response.content)
# print(response.json())
print(response.text)
```

16 Web Scraping com Python usando requests e bs4 BeautifulSoup

```
    +# Web Scraping com Python usando requests e bs4 BeautifulSoup
    # - Web Scraping e o ato de "raspar a web" buscando informacoes de forma
    # automatizada, com determinada linguagem de programação, para uso posterior.
    # - O modulo requests consegue carregar dados da Internet para dentro do seu
    # codigo. Ja o bs4.BeautifulSoup e responsavel por interpretar os dados HTML
    # em formato de objetos Python para facilitar a vida do desenvolvedor.
    # - Doc: https://www.crummy.com/software/BeautifulSoup/bs4/doc.ptbr/
    # + Instalação
    # - pip install requests types-requests bs4
```

```
    import re

    import requests
    from bs4 import BeautifulSoup

    url = 'http://127.0.0.1:3333/'
    response = requests.get(url)
    raw_html = response.text
    parsed_html = BeautifulSoup(raw_html, 'html.parser')

    # if parsed_html.title is not None:
    #     print(parsed_html.title.text)

    top_jobs_heading = parsed_html.select_one('#intro > div > div > article > h2')

    if top_jobs_heading is not None:
        article = top_jobs_heading.parent

        if article is not None:
            for p in article.select('p'):
                print(re.sub(r'\s{1,}', ' ', p.text).strip())
```

16.1 Web Scraping com Python usando requests e bs4 BeautifulSoup

```
    + Web Scraping com Python usando requests e bs4 BeautifulSoup
    # - Web Scraping e o ato de "raspar a web" buscando informacoes de forma
    # automatizada, com determinada linguagem de programacao, para uso posterior.
    # - O modulo requests consegue carregar dados da Internet para dentro do seu
    # codigo. Ja o bs4.BeautifulSoup e responsavel por interpretar os dados HTML
    # em formato de objetos Python para facilitar a vida do desenvolvedor.
    # - Doc: https://www.crummy.com/software/BeautifulSoup/bs4/doc.ptbr/
    # + Instalação
    # - pip install requests types-requests bs4
    import re

    import requests
    from bs4 import BeautifulSoup

    url = 'http://127.0.0.1:3333/'
    response = requests.get(url)
    raw_html = response.text
    parsed_html = BeautifulSoup(raw_html, 'html.parser')

    # if parsed_html.title is not None:
    #     print(parsed_html.title.text)

    top_jobs_heading = parsed_html.select_one('#intro > div > div > article > h2')

    if top_jobs_heading is not None:
        article = top_jobs_heading.parent

        if article is not None:
            for p in article.select('p'):
                print(re.sub(r'\s{1,}', ' ', p.text).strip())
```

17 Adicionando "encoding" no BeautifulSoup 4 para evitar problemas

```
    url = 'http://127.0.0.1:3333/'
response = requests.get(url)
raw_html = response.text
parsed_html = BeautifulSoup(raw_html, 'html.parser')
bytes_html = response.content
parsed_html = BeautifulSoup(bytes_html, 'html.parser', from_encoding='utf-8')

# if parsed_html.title is not None:
#     print(parsed_html.title.text)
```

18 Selenium Automatizando tarefas no navegador

```
print('Hello world!')
# Selenium - Automatizando tarefas no navegador
from pathlib import Path
from time import sleep

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
```

```
ROOT_FOLDER = Path(__file__).parent
# Caminho para a pasta onde o chromedriver está
CHROME_DRIVER_PATH = ROOT_FOLDER / 'drivers' / 'chromedriver'


def make_chrome_browser(*options: str) -> webdriver.Chrome:
    chrome_options = webdriver.ChromeOptions()

    # chrome_options.add_argument('--headless')
    if options is not None:
        for option in options:
            chrome_options.add_argument(option)  # type: ignore

    chrome_service = Service(
        executable_path=str(CHROME_DRIVER_PATH),
    )

    browser = webdriver.Chrome(
        service=chrome_service,
        options=chrome_options
    )

    return browser


if __name__ == '__main__':
    # Example
    # options = '--headless', '--disable-gpu',
    options = ()
    browser = make_chrome_browser(*options)

    # Como antes
    browser.get('https://www.google.com')

    # Dorme por 10 segundos
    sleep(10)
```

18.1 Selenium Selecionando elementos com By, expected conditions e WebDriver

```
    # type: ignore
# Selenium - Automatizando tarefas no navegador
from pathlib import Path
from time import sleep

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait

# Chrome Options
# https://peter.sh/experiments/chromium-command-line-switches/
# Doc Selenium
# https://selenium-python.readthedocs.io/locating-elements.html


# Caminho para a raiz do projeto
ROOT_FOLDER = Path(__file__).parent
# Caminho para a pasta onde o chromedriver está
CHROME_DRIVER_PATH = ROOT_FOLDER / 'drivers' / 'chromedriver'


def make_chrome_browser(*options: str) -> webdriver.Chrome:
    chrome_options = webdriver.ChromeOptions()

    # chrome_options.add_argument('--headless')
    if options is not None:
        for option in options:
            chrome_options.add_argument(option)

    chrome_service = Service(
        executable_path=str(CHROME_DRIVER_PATH),
    )

    browser = webdriver.Chrome(
        service=chrome_service,
        options=chrome_options
    )

    return browser
```

```python
if __name__ == '__main__':
    TIME_TO_WAIT = 10

    # Example
    # options = '--headless', '--disable-gpu',
    options = ()
    browser = make_chrome_browser(*options)

    # Como antes
    browser.get('https://www.google.com')

    # Espere para encontrar o input
    search_input = WebDriverWait(browser, TIME_TO_WAIT).until(
        EC.presence_of_element_located(
            (By.NAME, 'q')
        )
    )
    search_input.send_keys('Hello World!')

    # Dorme por 10 segundos
    sleep(TIME_TO_WAIT)
```

18.2 Selenium - Enviando teclas com a classe Keys

```python
    # type: ignore
# Selenium - Automatizando tarefas no navegador
from pathlib import Path
from time import sleep

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait

# Chrome Options
# https://peter.sh/experiments/chromium-command-line-switches/
# Doc Selenium
# https://selenium-python.readthedocs.io/locating-elements.html


# Caminho para a raiz do projeto
ROOT_FOLDER = Path(__file__).parent
# Caminho para a pasta onde o chromedriver está
CHROME_DRIVER_PATH = ROOT_FOLDER / 'drivers' / 'chromedriver'


def make_chrome_browser(*options: str) -> webdriver.Chrome:
    chrome_options = webdriver.ChromeOptions()

    # chrome_options.add_argument('--headless')
    if options is not None:
        for option in options:
            chrome_options.add_argument(option)

    chrome_service = Service(
        executable_path=str(CHROME_DRIVER_PATH),
    )

    browser = webdriver.Chrome(
        service=chrome_service,
        options=chrome_options
    )

    return browser


if __name__ == '__main__':
    TIME_TO_WAIT = 10

    # Example
    # options = '--headless', '--disable-gpu',
    options = ()
    browser = make_chrome_browser(*options)

    # Como antes
    browser.get('https://www.google.com')

    # Espere para encontrar o input
```

```
    search_input = WebDriverWait(browser, TIME_TO_WAIT).until(
        EC.presence_of_element_located(
            (By.NAME, 'q')
        )
    )
    search_input.send_keys('Hello World!')
    search_input.send_keys(Keys.ENTER)

    # Dorme por 10 segundos
    sleep(TIME_TO_WAIT)
```

18.3 Selenium find element e find elements By

```
    # type: ignore
# Selenium - Automatizando tarefas no navegador
from pathlib import Path
from time import sleep

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait

# Chrome Options
# https://peter.sh/experiments/chromium-command-line-switches/
# Doc Selenium
# https://selenium-python.readthedocs.io/locating-elements.html


# Caminho para a raiz do projeto
ROOT_FOLDER = Path(__file__).parent
# Caminho para a pasta onde o chromedriver está
CHROME_DRIVER_PATH = ROOT_FOLDER / 'drivers' / 'chromedriver'


def make_chrome_browser(*options: str) -> webdriver.Chrome:
    chrome_options = webdriver.ChromeOptions()

    # chrome_options.add_argument('--headless')
    if options is not None:
        for option in options:
            chrome_options.add_argument(option)

    chrome_service = Service(
        executable_path=str(CHROME_DRIVER_PATH),
    )

    browser = webdriver.Chrome(
        service=chrome_service,
        options=chrome_options
    )

    return browser


if __name__ == '__main__':
    TIME_TO_WAIT = 10

    # Example
    # options = '--headless', '--disable-gpu',
    options = ()
    browser = make_chrome_browser(*options)

    # Como antes
    browser.get('https://www.google.com')

    # Espere para encontrar o input
    search_input = WebDriverWait(browser, TIME_TO_WAIT).until(
        EC.presence_of_element_located(
            (By.NAME, 'q')
        )
    )
    search_input.send_keys('Hello World!')
    search_input.send_keys(Keys.ENTER)

    results = browser.find_element(By.ID, 'search')
    links = results.find_elements(By.TAG_NAME, 'a')
    links[0].click()
```

```
    # Dorme por 10 segundos
    sleep(TIME_TO_WAIT)
```

19 Usando subprocess para executar e comandos externos

```
    # Usando subprocess para executar e comandos externos
# subprocess e um modulo do Python para executar
# processos e comandos externos no seu programa.
# O metodo mais simples para atingir o objetivo e usando subprocess.run().
# Argumentos principais de subprocess.run():
# - stdout, stdin e stderr -> Redirecionam saida, entrada e erros
# - capture_output -> captura a saida e erro para uso posterior
# - text -> Se True, entradas e saidas serao tratadas como texto
# e automaticamente codificadas ou decodificadas com o conjunto
# de caracteres padrão da plataforma (geralmente UTF-8).
# - shell -> Se True, terá acesso ao shell do sistema. Ao usar
# shell (True), recomendo enviar o comando e os argumentos juntos.
# - executable -> pode ser usado para especificar o caminho
# do executável que iniciará o subprocesso.
# Retorno:
# stdout, stderr, returncode e args
# Importante: a codificação de caracteres do Windows pode ser
# diferente. Tente usar cp1252, cp852, cp850 (ou outros). Linux e
# mac, use utf_8.
# Comando de exemplo:
# Windows: ping 127.0.0.1
# Linux/Mac: ping 127.0.0.1 -c 4
```

19.1 Implementação

```
import subprocess
import sys

# sys.platform = linux, linux2, darwin, win32

cmd = ['ls -lah /']
encoding = 'utf_8'
system = sys.platform # checa sistema operacional

if system == "win32":
    cmd = ['ping', '127.0.0.1']
    encoding = 'cp850'


proc = subprocess.run(
    cmd, capture_output=True,
    text=True, encoding=encoding,
    shell=True,
)

print()

# print(proc.args)
# print(proc.stderr)
print(proc.stdout)
# print(proc.returncode)
```

20 Jupyter Notebook

20.1 Instalação e teste

```
    {
 "cells": [
  {
   "cell_type": "markdown",
   "id": "9dcb46f7",
   "metadata": {},
   "source": [
    "# Esse e o meu titulo\n",
    "\n",
    "Essa linha cria uma variável `a`."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "be0d3296",
   "metadata": {},
   "source": [
```

```
   "Essa linha tem o valor da variável ‘a‘ em dobro"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 7,
  "id": "f279e39a",
  "metadata": {
   "scrolled": true
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "2"
     ]
    },
    "execution_count": 7,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "a = 2\n",
   "a"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 8,
  "id": "1f5d7348",
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "4"
     ]
    },
    "execution_count": 8,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "b = a * 2\n",
   "b"
  ]
 },
 {
  "cell_type": "markdown",
  "id": "9c7f93b9",
  "metadata": {},
  "source": [
   "Essa linha mostra o valor de ‘a‘ + ‘b‘"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 9,
  "id": "6b735c9e",
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "6"
     ]
    },
    "execution_count": 9,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "a + b"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "id": "20fba249",
```

```
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.11.1"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```

```
    # Jupyter Notebook - Instalação e teste

    # Essa linha cria uma variável a
    a = 2

    # Essa linha tem o valor da variável ‘a‘ em dobro
    b = a * 2  # 4

    # Essa linha mostra o valor de ‘a‘ + ‘b‘
    print(a + b)  # 6
```

21 (Parte 1) Threads - Executando processamentos em paralelo

```
    from threading import Thread
from time import sleep


class MeuThread(Thread):
    def __init__(self, texto: str, tempo: int):
        self.texto = texto
        self.tempo = tempo

        super().__init__()

    def run(self):
        sleep(self.tempo)
        print(self.texto)


t1 = MeuThread('Thread 1', 5)
t1.start()

t2 = MeuThread('Thread 2', 3)
t2.start()

t3 = MeuThread('Thread 3', 2)
t3.start()

for i in range(20):
    print(i)
    sleep(1)
```

22 Part II

```
from threading import Thread
from time import sleep

"""
class MeuThread(Thread):
    def __init__(self, texto, tempo):
```

```
        self.texto = texto
        self.tempo = tempo

        super().__init__()

    def run(self):
        sleep(self.tempo)
        print(self.texto)


t1 = MeuThread('Thread 1', 5)
t1.start()

t2 = MeuThread('Thread 2', 3)
t2.start()

t3 = MeuThread('Thread 3', 2)
t3.start()

for i in range(20):
    print(i)
    sleep(1)
"""


def vai_demorar(texto: str, tempo: int):
    sleep(tempo)
    print(texto)


t1 = Thread(target=vai_demorar, args=('Olá mundo 1!', 5))
t1.start()

t2 = Thread(target=vai_demorar, args=('Olá mundo 2!', 1))
t2.start()

t3 = Thread(target=vai_demorar, args=('Olá mundo 3!', 2))
t3.start()

for i in range(20):
    print(i)
    sleep(.5)
```

23 part III

```
from threading import Lock, Thread
from time import sleep

"""
class MeuThread(Thread):
    def __init__(self, texto, tempo):
        self.texto = texto
        self.tempo = tempo

        super().__init__()

    def run(self):
        sleep(self.tempo)
        print(self.texto)


t1 = MeuThread('Thread 1', 5)
t1.start()

t2 = MeuThread('Thread 2', 3)
t2.start()

t3 = MeuThread('Thread 3', 2)
t3.start()

for i in range(20):
    print(i)
    sleep(1)
"""

"""
def vai_demorar(texto, tempo):
    sleep(tempo)
```

```
    print(texto)


t1 = Thread(target=vai_demorar, args=('Olá mundo 1!', 5))
t1.start()

t2 = Thread(target=vai_demorar, args=('Olá mundo 2!', 1))
t2.start()

t3 = Thread(target=vai_demorar, args=('Olá mundo 3!', 2))
t3.start()

for i in range(20):
    print(i)
    sleep(.5)
"""


"""
def vai_demorar(texto, tempo):
    sleep(tempo)
    print(texto)


t1 = Thread(target=vai_demorar, args=('Olá mundo 1!', 10))
t1.start()
t1.join()

print('Thread acabou!')
"""


class Ingressos:
    """
    Classe que vende ingressos
    """

    def __init__(self, estoque: int):
        """ Inicializando...

        :param estoque: quantidade de ingressos em estoque
        """
        self.estoque = estoque
        # Nosso cadeado
        self.lock = Lock()

    def comprar(self, quantidade: int):
        """
        Compra determinada quantidade de ingressos

        :param quantidade: A quantidade de ingressos que deseja comprar
        :type quantidade: int
        :return: Nada
        :rtype: None
        """
        # Tranca o metodo
        self.lock.acquire()

        if self.estoque < quantidade:
            print('Não temos ingressos suficientes.')
            # Libera o metodo
            self.lock.release()
            return

        sleep(1)

        self.estoque -= quantidade
        print(f'Voce comprou {quantidade} ingresso(s). '
              f'Ainda temos {self.estoque} em estoque.')

        # Libera o metodo
        self.lock.release()


if __name__ == '__main__':
    ingressos = Ingressos(10)

    for i in range(1, 20):
        t = Thread(target=ingressos.comprar, args=(i,))
        t.start()

    print(ingressos.estoque)
```

24 PyPDF2 para manipular arquivos PDF

24.1 instalação

```
    # PyPDF2 para manipular arquivos PDF (Instalacao)
    # PyPDF2 e uma biblioteca de manipulação de arquivos PDF feita em Python puro,
    # gratuita e de codigo aberto. Ela e capaz de ler, manipular, escrever e unir
    # dados de arquivos PDF, assim como adicionar anotacoes, transformar paginas,
    # extrair texto e imagens, manipular metadados, e mais.
    # A documentacao contem todas as informacoes necessarias para usar PyPDF2.
    # Link: https://pypdf2.readthedocs.io/en/3.0.0/
    # Ative seu ambiente virtual
    # pip install pypdf2
```

24.2 PdfReader

```
    # # PyPDF2 para manipular arquivos PDF (Instalação)
# PyPDF2 para manipular arquivos PDF (PdfReader)
# PyPDF2 e uma biblioteca de manipulacao de arquivos PDF feita em Python puro,
# gratuita e de codigo aberto. Ela e capaz de ler, manipular, escrever e unir
# dados de arquivos PDF, assim como adicionar anotacoes, transformar paginas,

    from pathlib import Path

    from PyPDF2 import PdfReader

    PASTA_RAIZ = Path(__file__).parent
    PASTA_ORIGINAIS = PASTA_RAIZ / 'pdfs_originais'
    PASTA_NOVA = PASTA_RAIZ / 'arquivos_novos'

    RELATORIO_BACEN = PASTA_ORIGINAIS / 'R20230210.pdf'

    PASTA_NOVA.mkdir(exist_ok=True)

    reader = PdfReader(RELATORIO_BACEN)

    # print(len(reader.pages))
    # for page in reader.pages:
    #     print(page)
    #     print()

    page0 = reader.pages[0]
    imagem0 = page0.images[0]

    # print(page0.extract_text())
    # with open(PASTA_NOVA / imagem0.name, 'wb') as fp:
    #     fp.write(imagem0.data)
```

24.3 PdfWriter

```
    # PyPDF2 para manipular arquivos PDF (PdfWriter)
# PyPDF2 para manipular arquivos PDF (PdfMerger)
# PyPDF2 e uma biblioteca de manipulacao de arquivos PDF feita em Python puro,
# gratuita e de codigo aberto. Ela e capaz de ler, manipular, escrever e unir
# dados de arquivos PDF, assim como adicionar anotacoes, transformar paginas,
@@ -9,7 +9,7 @@
# pip install pypdf2
    from pathlib import Path

from PyPDF2 import PdfReader
from PyPDF2 import PdfReader, PdfWriter

PASTA_RAIZ = Path(__file__).parent
PASTA_ORIGINAIS = PASTA_RAIZ / 'pdfs_originais'
@@ -32,3 +32,10 @@
# print(page0.extract_text())
# with open(PASTA_NOVA / imagem0.name, 'wb') as fp:
#     fp.write(imagem0.data)


for i, page in enumerate(reader.pages):
    writer = PdfWriter()
    with open(PASTA_NOVA / f'page{i}.pdf', 'wb') as arquivo:
        writer.add_page(page)
        writer.write(arquivo)  # type: ignore
```

24.4 PdfMerger

```
    from pathlib import Path

from PyPDF2 import PdfReader, PdfWriter
from PyPDF2 import PdfMerger, PdfReader, PdfWriter

PASTA_RAIZ = Path(__file__).parent
PASTA_ORIGINAIS = PASTA_RAIZ / 'pdfs_originais'
@@ -39,3 +39,17 @@
    with open(PASTA_NOVA / f'page{i}.pdf', 'wb') as arquivo:
        writer.add_page(page)
        writer.write(arquivo)  # type: ignore


files = [
    PASTA_NOVA / 'page1.pdf',
    PASTA_NOVA / 'page0.pdf',

]

merger = PdfMerger()
for file in files:
    merger.append(file)  # type: ignore

merger.write(PASTA_NOVA / 'MERGED.pdf')  # type: ignore
merger.close()
```

25 Deque - Trabalhando com LIFO e FIFO

```
    # Deque - Trabalhando com LIFO e FIFO
# deque - Double-ended queue
#
# Lifo  e fifo
# pilha e fila

# LIFO (Last In First Out)
# Pilha (stack)
# Significa que o ultimo item a entrar sera o primeiro a sair (list)
# Artigo:
# https://www.otaviomiranda.com.br/2020/pilhas-em-python-com-listas-stack/
# Video:
# https://youtu.be/svWVHEihyNI
# Para tirar itens do final: O(1) Tempo constante
# Para tirar itens do inicio: O(n) Tempo Linear

from collections import deque

lista = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# Legal (LIFO com lista)
#  0  1  2  3  4  5  6  7  8  9
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
lista.append(10)
#  0  1  2  3  4  5  6  7  8  9  10
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
lista.append(11)
#  0  1  2  3  4  5  6  7  8  9  10, 11
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
ultimo_removido = lista.pop()
#  0  1  2  3  4  5  6  7  8  9  10
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print('ultimo: ', ultimo_removido)
print('Lista:', lista)
#  0  1  2  3  4  5  6  7  8  9  10
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print()


lista = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

#  Ruim (FIFO com lista)
#  0  1  2  3  4  5  6  7  8  9
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
lista.insert(0, 10)
#   0  1  2  3  4  5  6  7  8  9, 10
# [10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
lista.insert(0, 11)
#  0   1   2  3  4  5  6  7  8  9, 10 11
# [11, 10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
primeiro_removido = lista.pop(0)  # 11
#  0   1  2  3  4  5  6  7  8  9, 10
```

```
# [10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print('Primeiro: ', primeiro_removido)  # 11
print('Lista:', lista)  # [10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print()

# FIFO (First In First Out)
# Filas (queue)
# Significa que o primeiro item a entrar será o primeiro a sair (deque)
# Artigo:
# https://www.otaviomiranda.com.br/2020/filas-em-python-com-deque-queue/
# Video:
# https://youtu.be/RHb-8hXs3HE
# Para tirar itens do final: O(1) Tempo constante
# Para tirar itens do inicio: O(1) Tempo constante

#  Legal (FIFO com deque)

fila_correta: deque[int] = deque()
fila_correta.append(3)  # Adiciona no final
fila_correta.append(4)  # Adiciona no final
fila_correta.append(5)  # Adiciona no final
fila_correta.appendleft(2)  # Adiciona no começo
fila_correta.appendleft(1)  # Adiciona no começo
fila_correta.appendleft(0)  # Adiciona no começo
print(fila_correta)  # deque([0, 1, 2, 3, 4, 5])
fila_correta.pop()  # 5
fila_correta.popleft()  # 0
print(fila_correta)  # deque([1, 2, 3, 4])
```

26 Remover regras de tipos Unknown do linter do VS Code

```
    "python.linting.mypyEnabled": true,
    "python.testing.unittestEnabled": false,
    "python.testing.pytestEnabled": true,
    "python.analysis.diagnosticSeverityOverrides": {},
    "python.analysis.diagnosticSeverityOverrides": {
      "reportUnknownMemberType": "none",
      "reportUnknownArgumentType": "none",
      "reportUnknownVariableType": "none",
      "reportUnknownLambdaType": "none",
      "reportUnknownParameterType": "none"
    },
    // "python.defaultInterpreterPath": "./venv/bin/python",
    "python.analysis.typeCheckingMode": "strict",
    "python.analysis.typeCheckingMode": "basic",
    "cSpell.enabled": true
  }
```

27 openpyxl para arquivos Excel xlsx, xlsm, xltx e xltm

27.1 instalação

```
    # openpyxl para arquivos Excel xlsx, xlsm, xltx e xltm (instalação)
# Com essa biblioteca sera possivel ler e escrever dados em celulas
# especificas, formatar celulas, inserir graficos,
# criar formulas, adicionar imagens e outros elementos graficos as suas
# planilhas. Ela e util para automatizar tarefas envolvendo planilhas do
# Excel, como a criacao de relatorios e analise de dados e/ou facilitando a
# manipulacao de grandes quantidades de informacoes.
# Instalação necessária: pip install openpyxl
# Documentação: https://openpyxl.readthedocs.io/en/stable/
```

27.2 criando uma planilha do Excel (Workbook e Worksheet)

```
    from pathlib import Path

from openpyxl import Workbook
from openpyxl.worksheet.worksheet import Worksheet

ROOT_FOLDER = Path(__file__).parent
WORKBOOK_PATH = ROOT_FOLDER / 'workbook.xlsx'

workbook = Workbook()
worksheet: Worksheet = workbook.active

# Criando os cabeçalhos
worksheet.cell(1, 1, 'Nome')
worksheet.cell(1, 2, 'Idade')
```

```
worksheet.cell(1, 3, 'Nota')

students = [
    # nome      idade nota
    ['João',    14,   5.5],
    ['Maria',   13,   9.7],
    ['Luiz',    15,   8.8],
    ['Alberto', 16,   10],
]

# for i, student_row in enumerate(students, start=2):
#     for j, student_column in enumerate(student_row, start=1):
#         worksheet.cell(i, j, student_column)

for student in students:
    worksheet.append(student)

workbook.save(WORKBOOK_PATH)
```

27.3 manipulando as planilhas do Workbook

```
    WORKBOOK_PATH = ROOT_FOLDER / 'workbook.xlsx'

workbook = Workbook()
worksheet: Worksheet = workbook.active
# worksheet: Worksheet = workbook.active

# Nome para a planilha
sheet_name = 'Minha planilha'
# Criamos a planilha
workbook.create_sheet(sheet_name, 0)
# Selecionou a planilha
worksheet: Worksheet = workbook[sheet_name]

# Remover uma planilha
workbook.remove(workbook['Sheet'])

# Criando os cabeçalhos
worksheet.cell(1, 1, 'Nome')
```

27.4 ler e alterar dados de uma planilha

```
    from pathlib import Path

from openpyxl import Workbook, load_workbook
from openpyxl.cell import Cell
from openpyxl.worksheet.worksheet import Worksheet

ROOT_FOLDER = Path(__file__).parent
WORKBOOK_PATH = ROOT_FOLDER / 'workbook.xlsx'

# Carregando um arquivo do excel
workbook: Workbook = load_workbook(WORKBOOK_PATH)

# Nome para a planilha
sheet_name = 'Minha planilha'

# Selecionou a planilha
worksheet: Worksheet = workbook[sheet_name]

row: tuple[Cell]
for row in worksheet.iter_rows(min_row=2):
    for cell in row:
        print(cell.value, end='\t')

        if cell.value == 'Maria':
            worksheet.cell(cell.row, 2, 23)
    print()

# worksheet['B3'].value = 14

workbook.save(WORKBOOK_PATH)
```

28 Pillow: redimensionando imagens com Python

```
# Pillow: redimensionando imagens com Python
# Essa biblioteca e o Photoshop do Python
from pathlib import Path
```

```
from PIL import Image

ROOT_FOLDER = Path(__file__).parent
ORIGINAL = ROOT_FOLDER / 'original.JPG'
NEW_IMAGE = ROOT_FOLDER / ' new.JPG'

pil_image = Image.open(ORIGINAL)
width, height = pil_image.size
exif = pil_image.info['exif']

# width     new_width
# height    ??
new_width = 640
new_height = round(height * new_width / width)

new_image = pil_image.resize(size=(new_width, new_height))
new_image.save(
    NEW_IMAGE,
    optimize=True,
    quality=70,
    # exif=exif,
)
```