4º BIMESTRE – PROGRAMAÇÃO ORIENTADA A OBJETOS - PYTHON

Sumário

38 – MANIPULAÇÃO DE ARQUIVOS TEXTO

Um poderoso e importante recurso utilizado em linguagens de programações é manipulação de arquivos no formato de texto.

Os arquivos do tipo texto podem ser criados e manipulados, por exemplo, no bloco de notas do Windows.

Vale dizer que no word podemos criar também arquivos no formato texto (.txt). Entretanto, no word, por padronização geramos inicialmente arquivos .doc ou .docx, que são textos formatados.

Os arquivos texto podem gerar por exemplo:

- Sequência de dados não formatados
- Arquivos de programas como HTML, C, ALGORITMOS, etc...
- Arquivos de configurações
- Entre outros

Saibam que, manipular arquivo texto, nada mais é do que ABRIR, ESCREVER, ALTERAR e LER dados nestes tipos de arquivos.

38.1 – CRIAR/ABRIR ARQUIVOS TEXTOS

Para criar e abrir um arquivo texto usaremos um objeto FILE, para o qual usaremos o método ***open()*** com parâmetros que identificarão os tipos de abertura de arquivos possíveis.

Primeiramente veja abaixo a sintaxe do método open, e suas diferentes formas de abertura de arquivos:

Sintaxe:

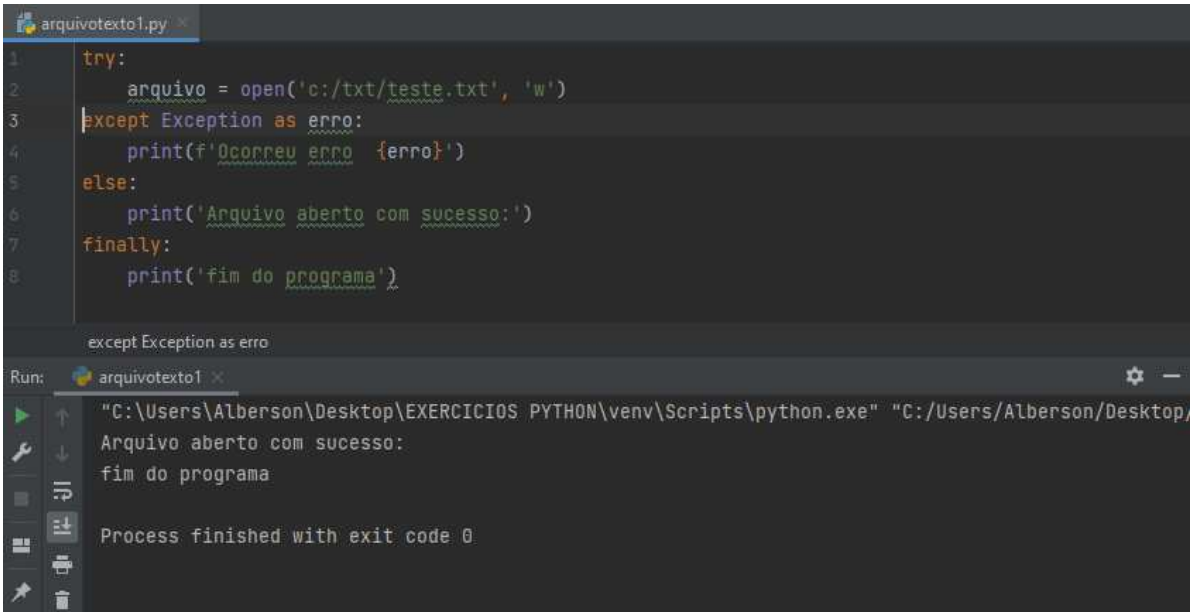
<nome_do_objetofile> = open(“caminho e nome do arquivo texto”, “modo_abertura”)

Os modos de abertura são:

MODOS DE ABERTURA DE ARQUIVOS TEXTOS MAIS USADOS	PARA QUE SERVE?
“r”	somente leitura do arquivo texto existente. Ocorrerá erro se não existir arquivo.
“r+”	Cria e lê arquivo texto. Não perde conteúdo se já existir.
“w”	Cria novo arquivo texto, perdendo conteúdo antigo se já existisse.
“a”	append (inclusão de novas linhas ao final do arquivo texto)
“x”	Abre arquivo para criação exclusiva. Ocorre erro se já existir

Vamos exemplos:

Exemplo 1: Criando arquivo texto pela primeira vez ou recriando arquivo já existente (substituindo arquivo existente):

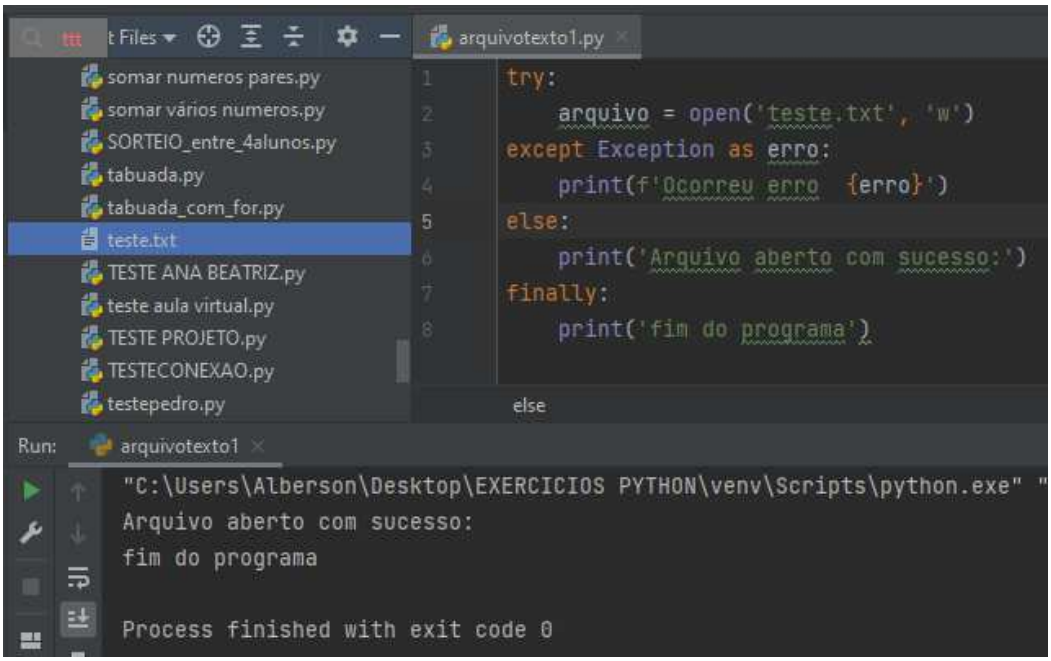


Repare acima:

a) na pasta txt na unidade c:\ do meu computador.
Observação importante: Caso a pasta ou a unidade de gravação não exista vai ocorrer erro!!

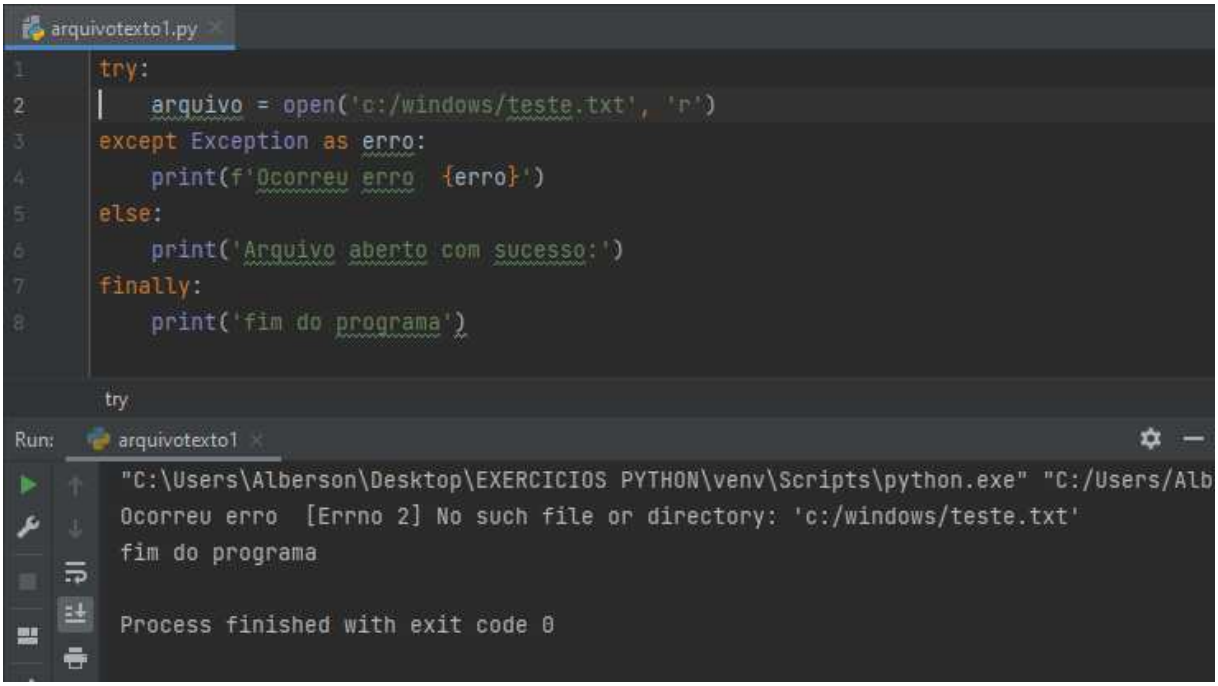
b) No exemplo acima, perceba que criei o arquivo teste.txt e o objeto ***arquivo*** foi criado para representá-lo no meu programa

Exemplo 2: Criando arquivo sem especificar caminho (pasta) para guardá-lo:



Conforme pode-se notar o arquivo teste.txt foi criado na mesma pasta onde meu programa está gravado, ou seja, na pasta do projeto que está atualmente aberta.

Exemplo 3: Tentando abrir um arquivo inexistente dentro de uma pasta qualquer:



Neste exemplo, veja que estou tentando abrir um arquivo para leitura e indiquei que este estaria gravado na pasta Windows. Como não existia nesta pasta, ocorreu um erro que pode ser observado na área de execução acima.

38.2 – ESCREVENDO CONTEÚDO NO ARQUIVO TEXTO E FECHANDO ARQUIVO TEXTO

Para escrevermos num arquivo aberto com modo “w”, “r+”, ou “a” usamos a seguinte sintaxe:

<nome_do_objetofile>.write(“conteúdo do arquivo”)

Podemos escrever todo conteúdo desejado através do uso de variáveis, ou então, determinar valor desejado como string na sintaxe acima.

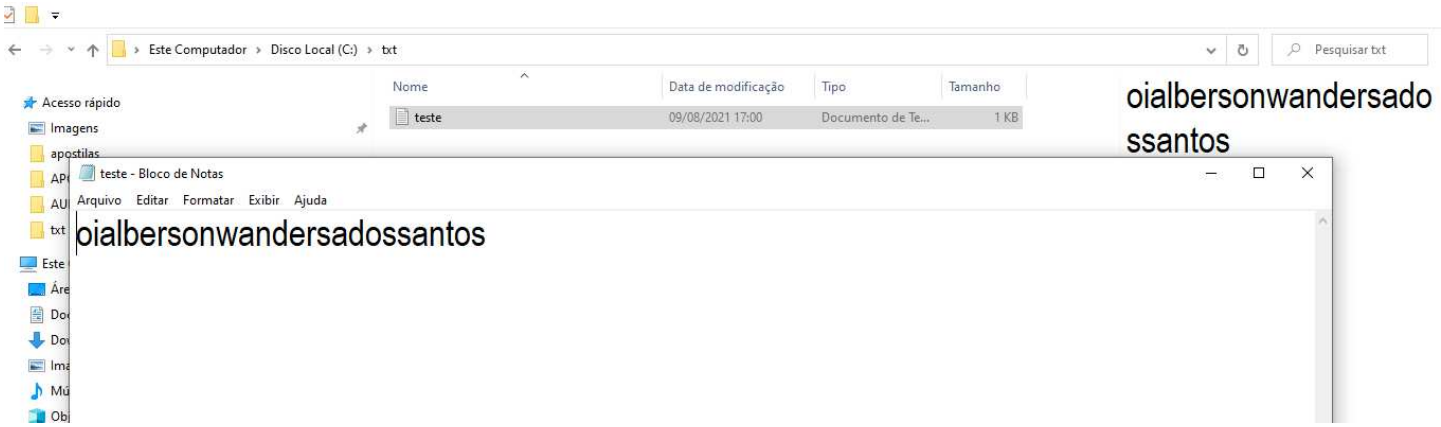
<nome_do_objetofile>.close() – método usado para fechar o arquivo texto aberto. Este método deve ser usado sempre após o uso do arquivo de texto. Nunca deixe um arquivo aberto.

Exemplo 1:

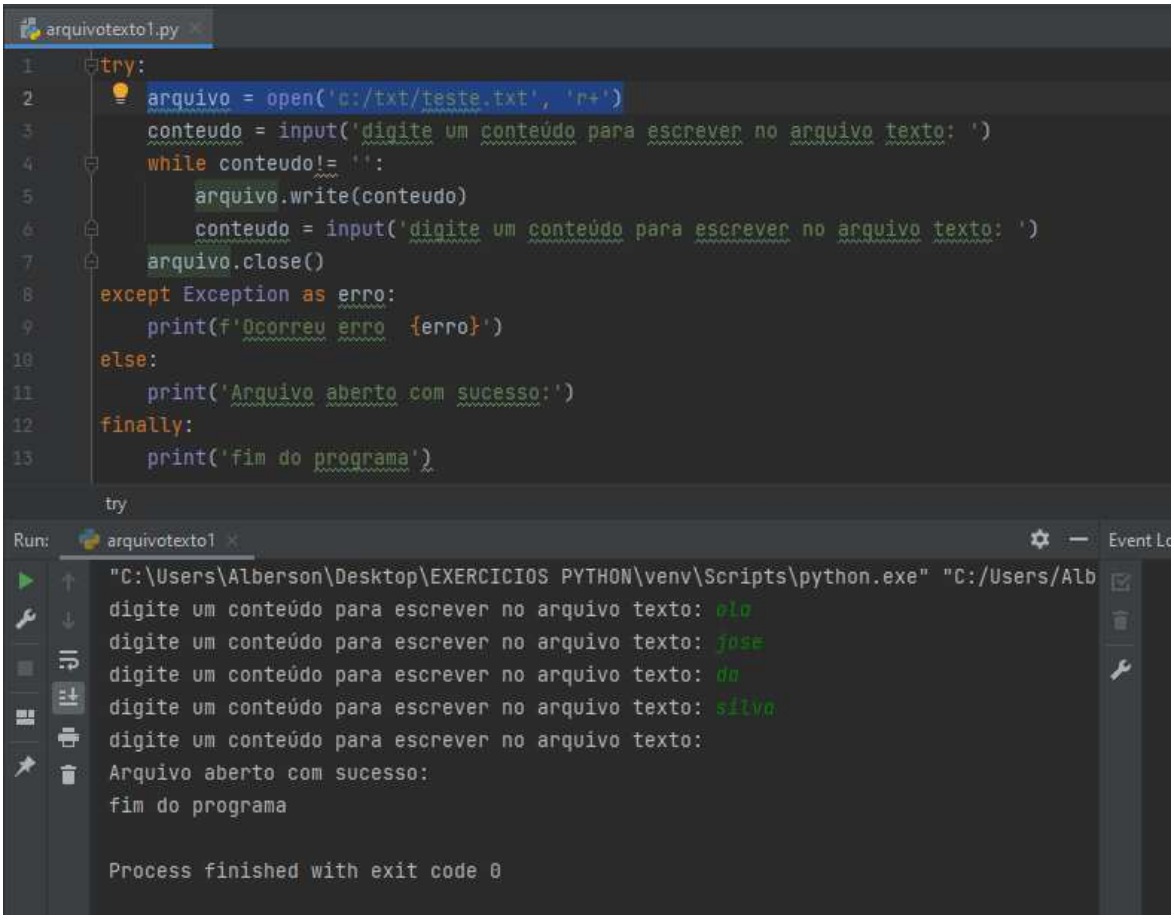
```
arquivotexto1.py
try:
    arquivo = open('c:/txt/teste.txt', 'w')
    conteudo = input('digite um conteúdo para escrever no arquivo texto: ')
    while conteudo!= '':
        arquivo.write(conteudo)
        conteudo = input('digite um conteúdo para escrever no arquivo texto: ')
    arquivo.close()
except Exception as erro:
    print(f'Ocorreu erro  {erro}')
else:
    print('Arquivo aberto com sucesso:')
finally:
    print('fim do programa')

except Exception as erro

Run:  arquivotexto1
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alb
digite um conteúdo para escrever no arquivo texto: oi
digite um conteúdo para escrever no arquivo texto: alberson
digite um conteúdo para escrever no arquivo texto: wander
digite um conteúdo para escrever no arquivo texto: sa
digite um conteúdo para escrever no arquivo texto: dos
digite um conteúdo para escrever no arquivo texto: santos
digite um conteúdo para escrever no arquivo texto:
Arquivo aberto com sucesso:
fim do programa

Process finished with exit code 0
```

Repare:

a) Neste arquivo o usuário digita os valores que serão inseridos um a um no arquivo que foi aberto pelo modo “w”.
b) Repare que no arquivo gerado, as palavras foram gravadas juntas uma das outras, veja abaixo quando abri o arquivo no bloco de notas do windows:

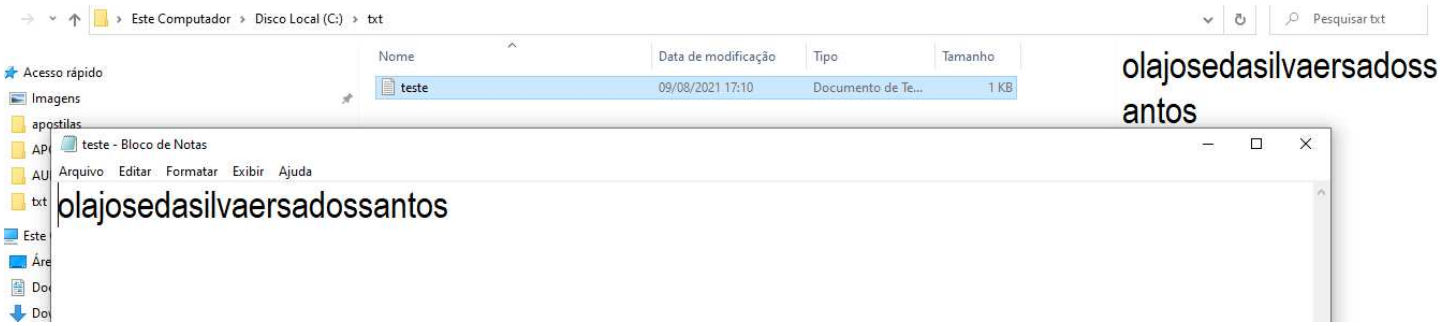


Exemplo 2: Reescrevendo conteúdo do arquivo aberto, usando método “r+”

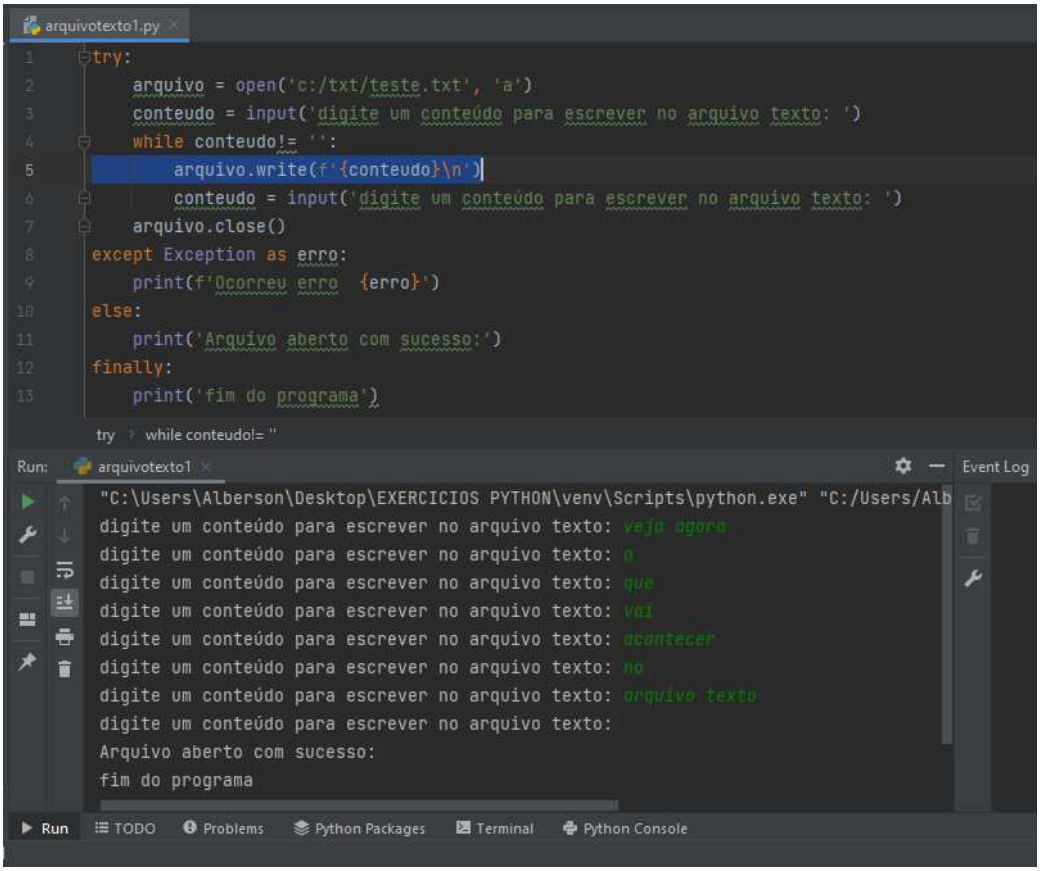


No caso acima repare:

a) A abertura do arquivo foi para leitura e gravação.
b) Como o arquivo já existia, pois o criei no exemplo anterior, ao informar novos valores para o arquivo texto, os dados anteriormente gravados foram perdidos, dando lugar aos novos valores digitados. Veja o arquivo aberto:

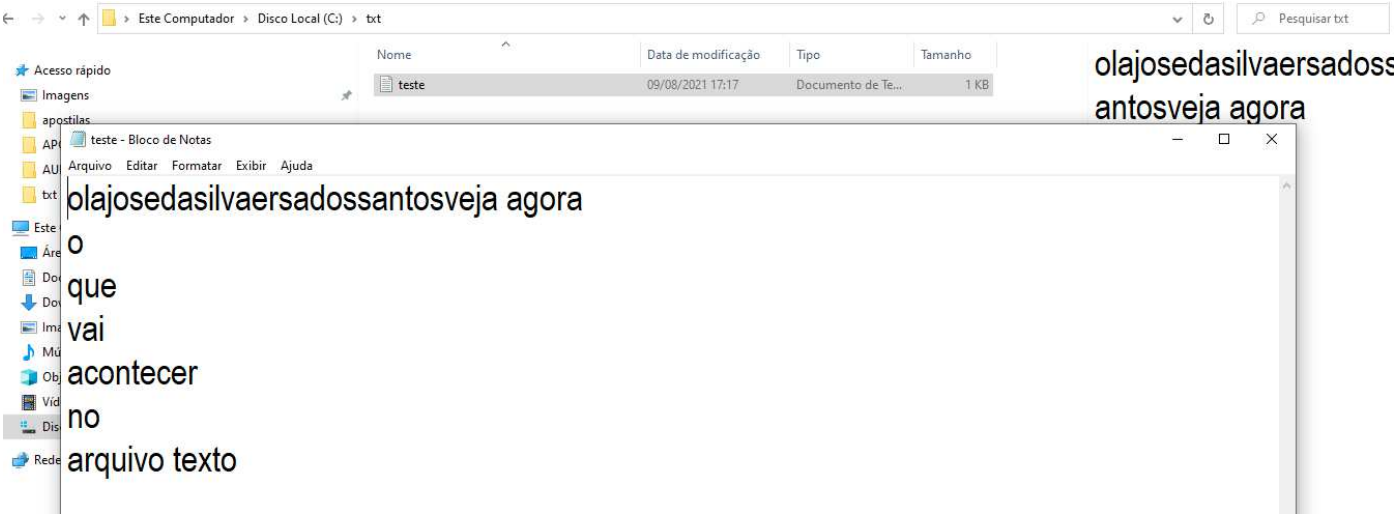


Exemplo 3: Abrindo arquivo no modo “a”:



Veja acima:

a) Abri o arquivo com modo “a” append, o que significa que o arquivo sofrerá inclusão de novas linhas
b) Veja que no método ***write()*** foi usado o ***“\n”*** para provocar salto de linha dentro do arquivo existente.
c) Observe o que os novos valores informados foram gravados em linhas diferentes dentro do arquivo já existente:



Observe que a inclusão se fez a partir da última inclusão anteriormente realizada.

Método close() – Nos exemplos anteriores o método close foi usado para fechar o arquivo texto.

38.3 – Lendo uma linha de um arquivo texto – readline()

Para ler uma linha de um arquivo texto basta usar o método ***readline()***, conforme sintaxe abaixo:

<variável> = <nome_do_objetofile>.readline()

Ou então:

print(f'{<nome_do_objetofile>.readline()}')

Exemplo: Lendo uma linha do arquivo texto criado nos exemplos anteriores:

arquivotexto1.py

```
try:
    arquivo = open('c:/txt/teste.txt', 'r')
    linha1 = arquivo.readline()
    linha2 = arquivo.readline()
    linha3 = arquivo.readline()
    print(f'{linha1}')
    print(f'{linha2}')
    print(f'{linha3}')
    print(f'{arquivo.readline()}')
    arquivo.close()
except Exception as erro:
    print(f'Ocorreu erro  {erro}')
else:
    print('Arquivo aberto com sucesso:')
finally:
    print('fim do programa')
```

O resultado será o seguinte:

Run: arquivotexto1

```
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alb
olajosedasilvaersadossantosveja agora

o

que

vai

Arquivo aberto com sucesso:
fim do programa

Process finished with exit code 0
```

Repare que cada readline() imprime somente uma linha do arquivo que foi aberto.

38.4 – Lendo todo o arquivo texto – readlines()

Para leitura total de um arquivo texto podemos usar o método ***readlines().*** Com esta forma de leitura cria-se uma lista de conteúdos e linhas a qual pode ser acessada e impressa de várias formas.

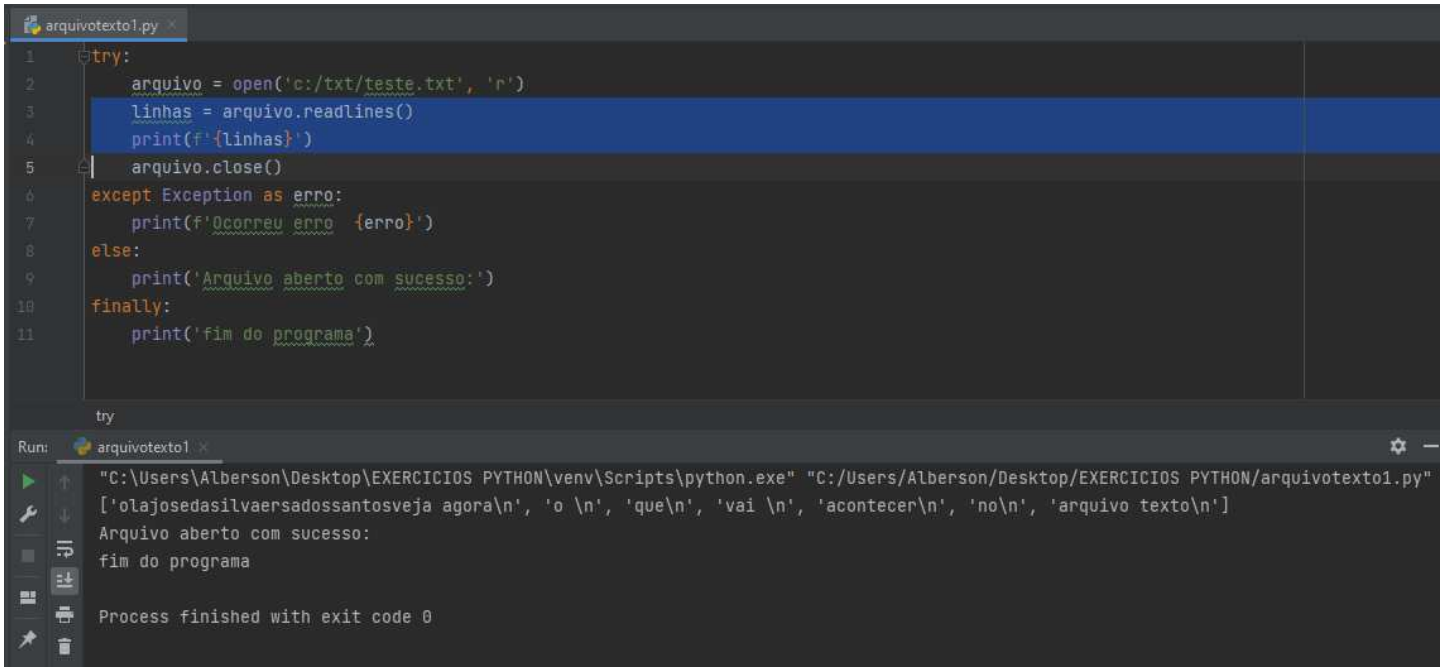
Antes porém vejamos a sintaxe mais comum do uso do método ***readlines():***

<variável> = <nome_do_objetofile>.readlines()

Ou então:

print(f'{<nome_do_objetofile>.readlines()}')

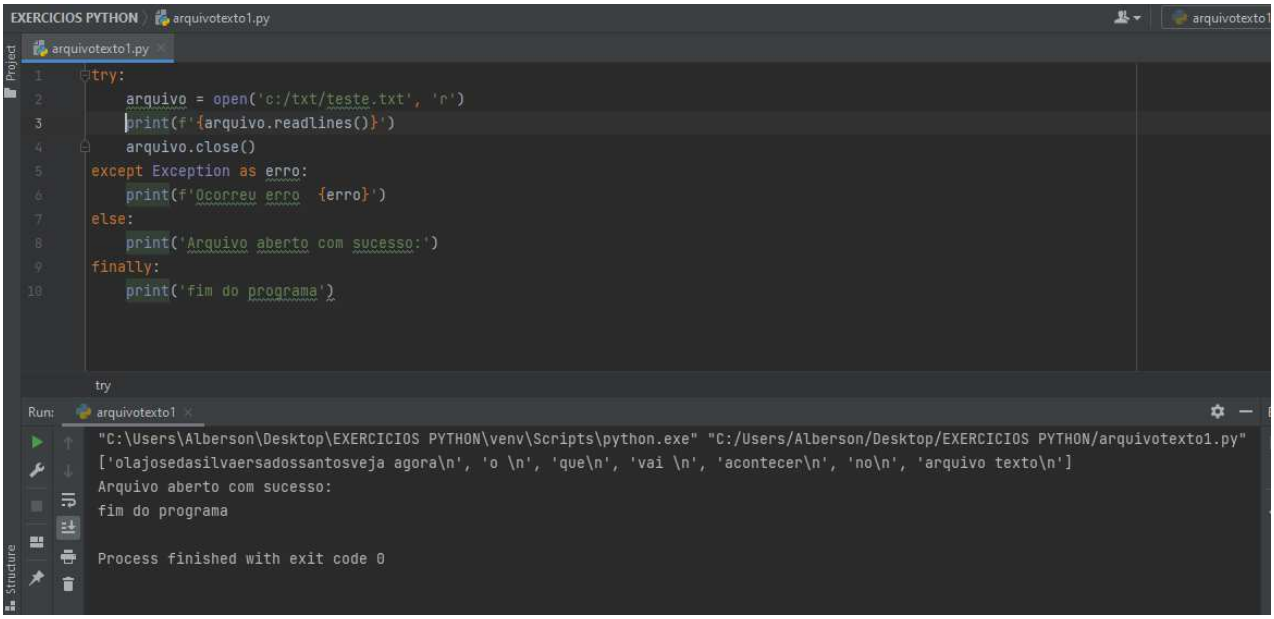
Exemplo 1: Criando uma lista das linhas lidas com readlines() e imprimindo-a na tela:



Repare

a) A lista ***linhas*** foi criada e impressa.
b) O conteúdo do arquivo foi posto em uma lista.
c) Esta impressão para o usuário isto não é a ideal.

Exemplo 2: Veja que imprimindo somente o resultado do readlines() apresentará o mesmo resultado:

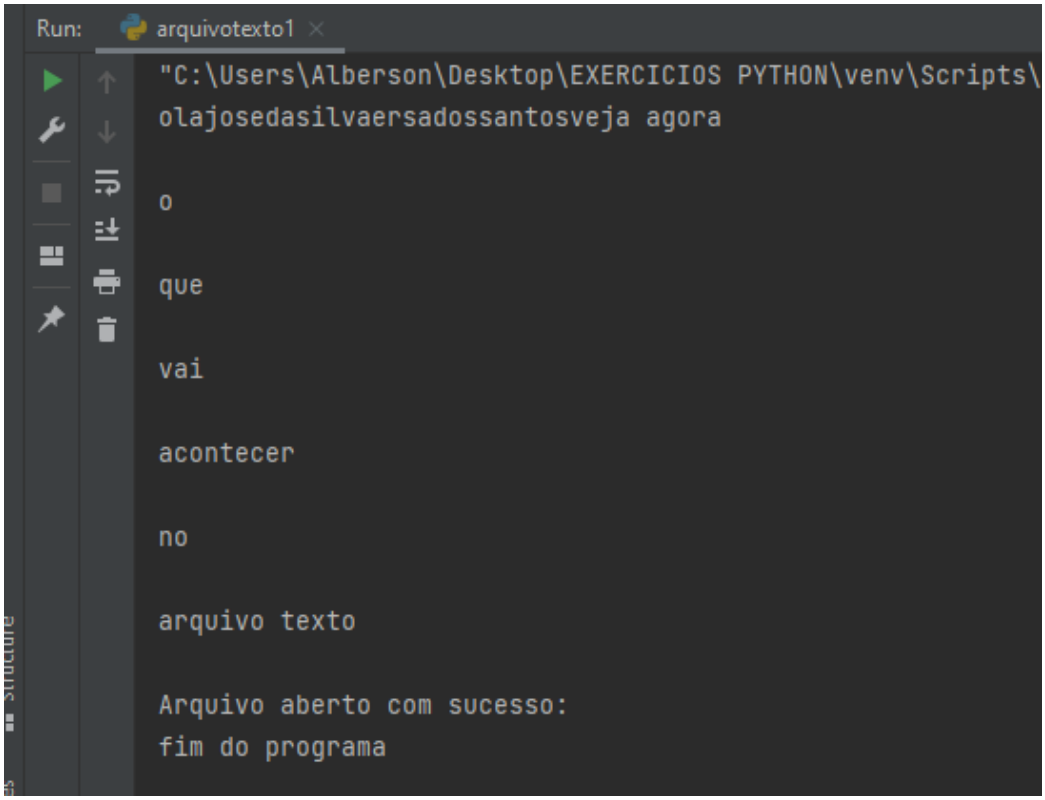


Este também não é o resultado ideal para se apresentar para o usuário

Exemplo 3: Percorrendo a lista das linhas lidas do arquivo texto e imprimindo-as na tela:

Repare:

A) Agora temos todo o arquivo lido exibido corretamente na tela, da forma que foi gravado, veja:



38.4 – Lendo todo o arquivo texto – read()

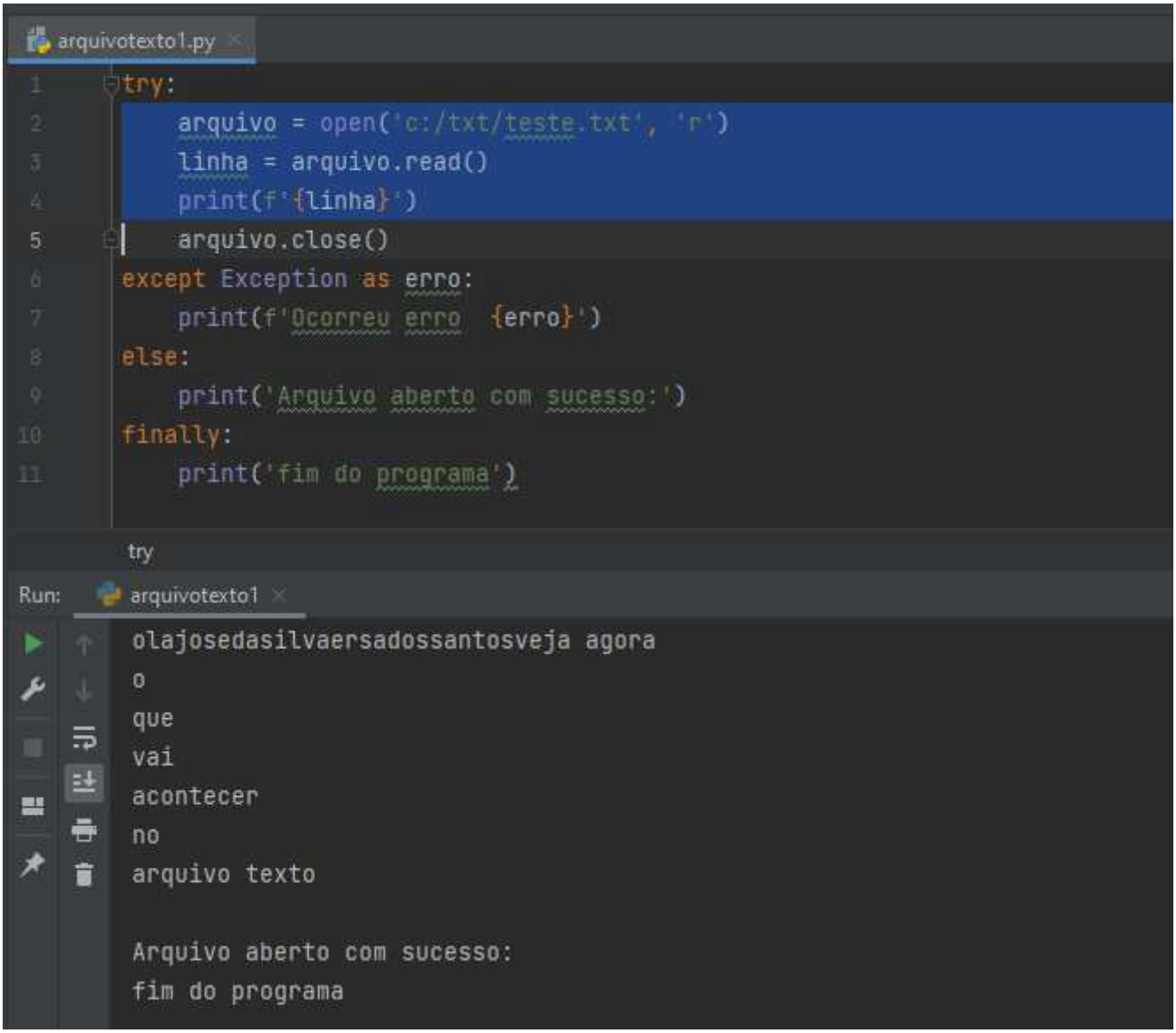
Se um arquivo for lido com o método read(), a posição retornada será sempre a do final do arquivo, pois esse método retorna o arquivo inteiro de uma vez.

Sintaxe:

<variável> = <nome_do_objetofile>.read()

A variável na sintaxe acima armazenará o texto com suas devidas quebras de linhas. Não é criado lista, ou qualquer outra estrutura de dados.

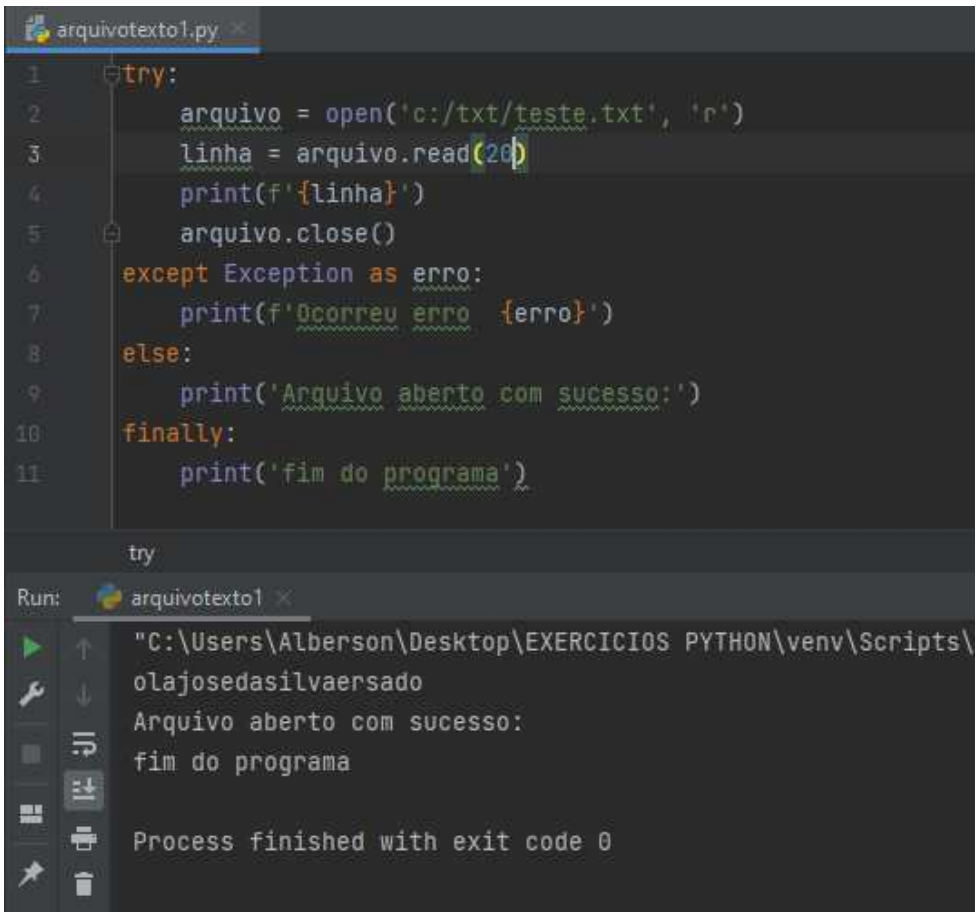
Exemplo 1: Vamos ler e exibir o arquivo teste.txt, criado e usado nos exemplos anteriores:



Observe que:

a) O conteúdo do arquivo texto fica mais bem visualizado com uso desse método
b) Não há necessidade de usar estrutura de repetições para visualizar cada linha do arquivo texto, pois a variável linha guarda todo o conteúdo do mesmo.

Exemplo 2: Podemos ler os N primeiros caracteres do arquivo texto com o método ***read().*** Para isso basta passarmos como parâmetro para o método a quantidade de caracteres desejado. Veja abaixo:



Repare:

a) Na linha 3, o método read() passa o número 20 como parâmetro, o que realizará a leitura de somente os 20 primeiros caracteres do arquivo lido.

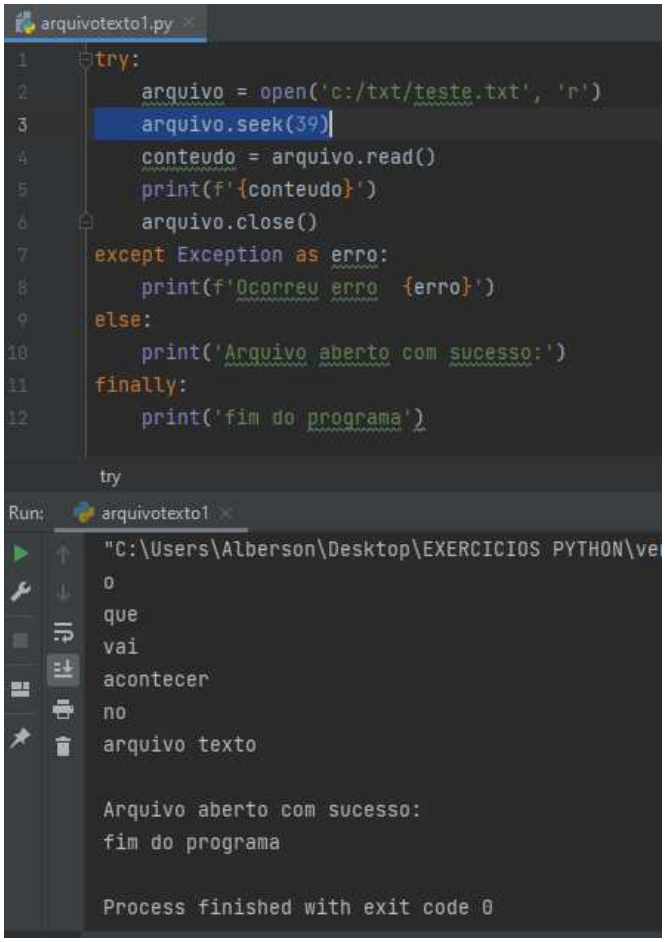
38.5 – POSICIONANDO INICIO DE LEITURA NO ARQUIVO TEXTO – MÉTODO seek()

Podemos iniciar a leitura a partir de uma quantidade de caracteres existentes no arquivo texto.

Sintaxe:

<nome_do_objetofile>.seek (<numerodaposicaoinicialdecaracteres>)

Exemplo 1: Vamos ler o arquivo dos nossos exemplos anteriores a partir do caractere 39. Veja o resultado:



Repare:

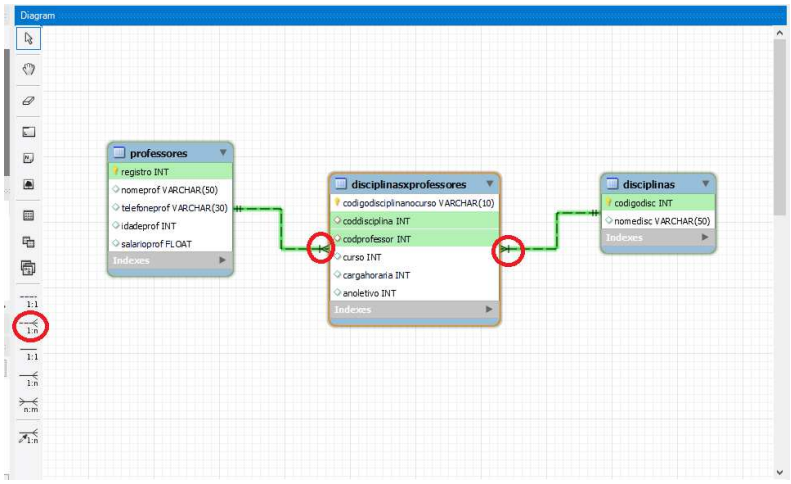
a) Veja que posicionamos o início da impressão a partir do caractere 39, usando o método ***seek()*** com parâmetro 39.
b) Quando usamos o método ***read()***, a leitura se concretizará a partir da posição de caractere 39.

Caso queira iniciar a leitura a partir do primeiro caractere do arquivo, basta usar o método seek com a indicação 0, ou seja, arquivo.seek(0)

38.6 – DESAFIOS ARQUIVOS TEXTO

a) Desenvolver um programa python para criar um editor de código HTML. O usuário deve definir o nome do arquivo HTML desejado. Ao iniciar o programa, deve ser pedido ao usuário um título da página web que será criada. Em seguida, deve ser solicitado o texto da página. Quando estiver criando o texto da página, o usuário poderá inserir qualquer tag HTML, para formatação geral do texto.

b) O banco de dados relacional abaixo mostra o relacionamento entre 3 tabelas, veja:



Fazer um programa python para criar arquivos HTML´s para expor as disciplinas de um professor nos diversos cursos que o mesmo da aula. Perceba que um mesmo código de disciplina pode aparecer em diversos cursos e que também o professor pode dar aulas de diversas disciplinas em diversos cursos. Vejamos exemplos de cadastros abaixo:

disciplinasxprofessores					
coddsciplinanocurso	coddisciplina	codprofessor	curso	cargahoraria	anoletivo
100	10	1	1	20	2021
300	20	1	1	20	2021
400	30	1	2	30	2021
200	10	2	2	40	2021

disciplinas	
codigodisc	nomedisc
10	pooi
20	pvb
30	paw

professores	
registro	nomeprof
1	alberson
2	wagner

Observações:

1. Cada arquivo HTML deve ser nomeado com o código do professor.
2. Ao abrir um arquivo HTML para visualização, como exemplo, veremos uma página com o seguinte layout:

Nome do arquivo: 1.html

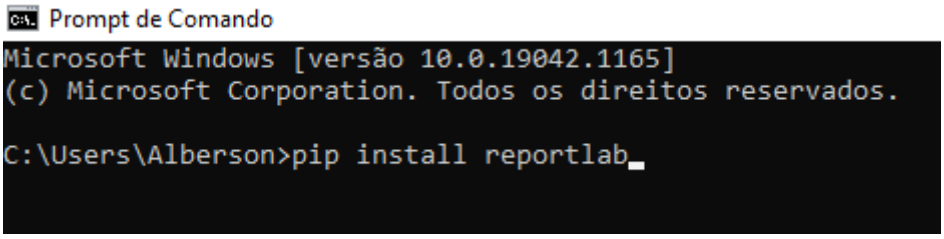
```
Disciplinas do professor: alberson
Curso: 1
CÓDIGO DA DISCIPLINA | NOME DISCIPLINA
10 |pooi
20 |pvb

Curso: 2
30 |paw
```

39 – Arquivos PDF – USO DA BIBLIOTECA reportLab

Um tipo de manipulação de arquivos muito utilizado é a geração de arquivos PDF´s. Logicamente este tipo de arquivo tem várias utilidades, mas para nosso curso usaremos especificamente para impressão de relatórios em PDF.

É bom dizer que relatórios nada mais são do que resultados de consultas diversas, realizadas em banco de dados, ou até mesmo de dados guardados em arquivos textos, entre outros.

Para exemplificarmos a manipulação de arquivos PDF, vamos usar a biblioteca **reportlab**. Para instalar, basta usarmos o pip do python. Digite no seu prompt de comandos do Windows o seguinte comando:



Ao pressionar <enter> perceba que deverá aparecer os passos da instalação da referida biblioteca, conforme figura a seguir:

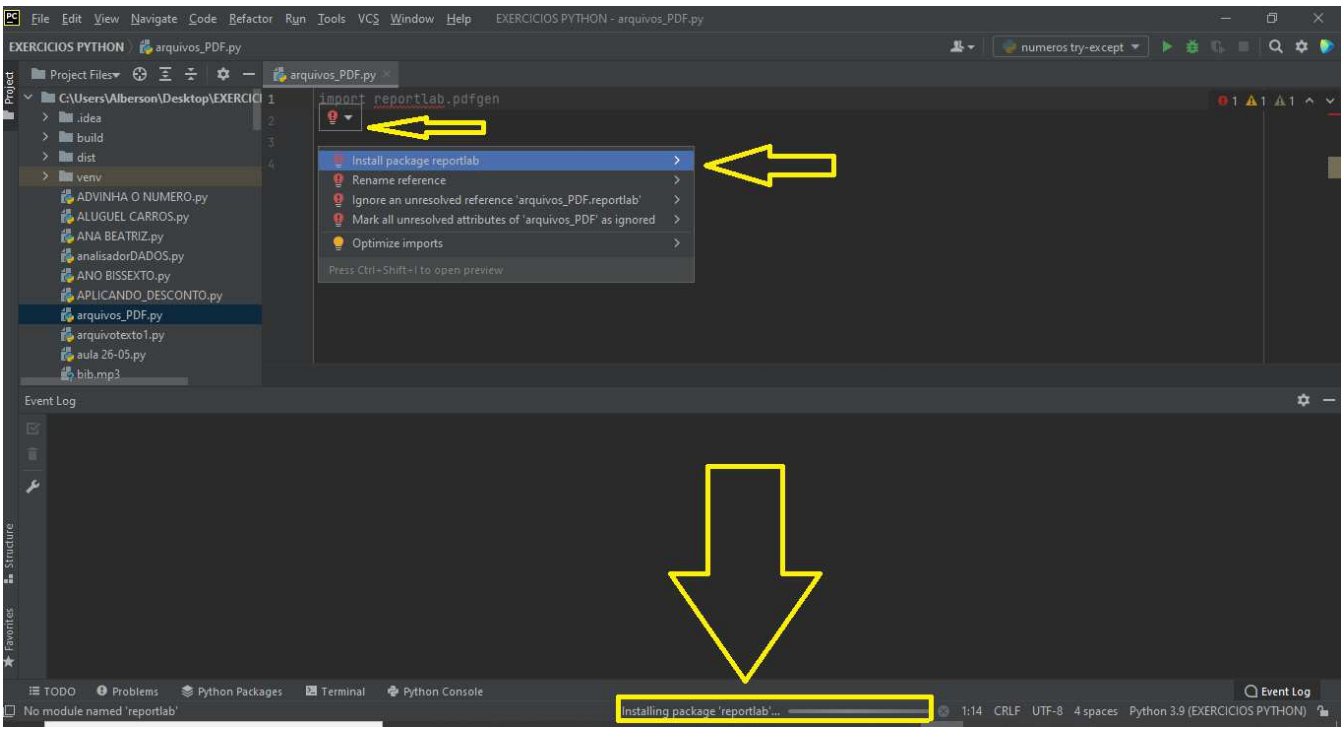
ATENÇÃO:

O INÍCIO DA INSTALAÇÃO PODE DEMORAR ALGUNS MINUTOS.

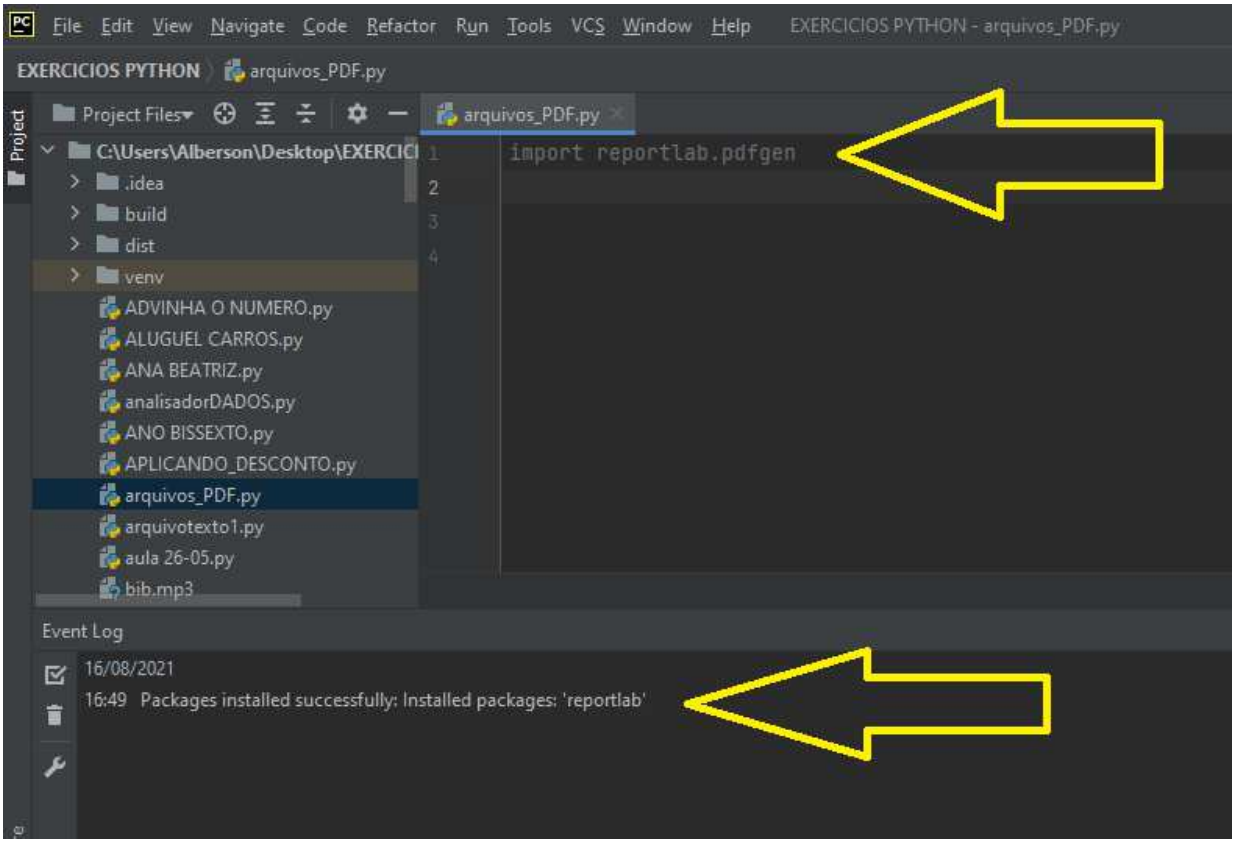
A MENSAGEM EM AMARELO INFORMA APENAS QUE O INSTALADOR pip DO MEU COMPUTADOR ESTÁ DESATUALIZADA

Agora basta abrirmos o PyCharm e importamos a biblioteca reportlab no programa desejado.

Caso a sua biblioteca não tenha sido reconhecida dentro do pycharm, você pode usar do recurso de instalação de bibliotecas, como demonstrado abaixo:



Quando o processor for realizado, perceba que a linha vermelha que existia abaixo da biblioteca **reportlab.pdfgen** deixou de existir e surgirá também uma mensagem informando o final do processo de instalação. Veja abaixo:



Diante do resultado, posso então começar meu projeto para gerar arquivos pdf.

Vamos escrever o seguinte programa, o qual explicarei linha a linha:

arquivos_PDF.py

```
#vamos usar o método canvas da biblioteca reportlab.pdfgen
from reportlab.pdfgen import canvas

def regua(pdf):
    '''
        :param pdf: este parâmetro receberá o objeto pdf criado no rotina gerarpdf
         Esta rotina só foi criada para mostrar os numeros de linhas e colunas da pagina PDF
    '''
    #DEFININDO A COR DO TEXTO (COR DA FONTE) A SER IMPRESSO NO ARQUIVO PDF
    pdf.setFillColor('red')
    for coluna in range(0, 595, 5):
        pdf.setFont("Helvetica-Oblique", 2)
        # imprimindo os numeros das coluna na última linha do arquivo PDF, linha 0
        pdf.drawString(coluna, 0, f'{coluna} ')

    for linha in range(0, 841, 5):
        pdf.setFont("Helvetica-Oblique", 2)
        # imprimindo os numeros de linhas na primeira coluna do arquivo PDF, coluna 0
        pdf.drawString(0, linha, f'{linha} ')
```

```
def gerarpdf(dicionario):
    try:
        #solicitando ao usuário o nome do arquivo PDF
        nomearquivo = input('Informe o nome do arquivo PDF que deseja gerar: ')

        #criando objeto pdf com nome do arquivo definido pelo usuário
        pdf = canvas.Canvas(f'{nomearquivo}.pdf')

        #CHAMANDO A ROTINA regua PARA IMPRIMIR UMA REGUA NA PÁGINA PARA MOSTRAR O POSICIONAMENTO DO TEXTO
        #NO ARQUIVO PDF. ESTA ROTINA NÃO PRECISA SER CHAMADA SEMPRE E NEM PRECISA SER CRIADA NOS SEUS PROGRAMAS
        regua(pdf)

        #DEFININDO A COR DO TEXTO DOS PRÓXIMOS TEXTOS A SEREM IMPRESSOS
        pdf.setFillColor('black')

        #Definindo o texto do titulo do documento a ser impresso
        pdf.setTitle('Relatório de Professores do Curso Técnico')

        # definindo nome da fonte e tamanho da mesma para o próximo texto a ser escrito no PDF
        pdf.setFont("Helvetica-Oblique", 16)

        # ATENÇÃO: AS REFERENCIAS DE LINHAS E COLUNAS NO RELATÓRIO SÃO TROCADAS NO MÉTODO drawString
        # Assim estou ESCREVENDO UM TEXTO NA PÁGINA PDF, NA LINHA 750 A PARTIR DA COLUNA 10
```

```
        pdf.drawString(10, 750, 'Relação de Professores do 2° ano:')

        # definindo nome da fonte e tamanho da mesma para o próximo texto a ser escrito,
        pdf.setFont("Helvetica-Oblique", 14)

        # ATENÇÃO: AS REFERENCIAS DE LINHAS E COLUNAS NO RELATÓRIO SÃO TROCADAS NO MÉTODO drawString
        # Assim estou ESCREVENDO UM TEXTO NA PÁGINA PDF, NA LINHA 750 A PARTIR DA COLUNA 10
        pdf.drawString(10, 720, 'Nome professor | Disciplina ')

        x=700 #MEDIDA EM MILIMETROS
        for nome, disciplina in dicionario.items():
            print(f'{nome} | {disciplina}')
            x-=20 #-20mm
            # o método drawString é usado para escrever na página PDF (FOLHA TOTAL POSSUI
            pdf.drawString(10, x, f'{nome} | {disciplina}')

        #criando arquivo PDF
        pdf.save()
        print(f'{nomearquivo} foi gerado com sucesso: ')
    except Exception as erro:
        print(f'Erro ao gerar o arquivo pdf: {erro}')

dicionario = {'Alberson': 'PooI', 'Bruno': 'Banco de Dados', 'Helio': 'PAWeb', 'Wagner': 'PVBásica'}
gerarpdf(dicionario)
```

SOBRE O CÓDIGO DESTE PROGRAMA:

a) Linha 68 - a rotina gerarpdf() foi chamada e será passado como parâmetro um dicionário de nomes e disciplinas de professores.
b) Linha 25 – O usuário está definindo um nome de arquivo PDF que será gerado. **Não é necessário indicar a extensão .pdf .**
c) Linha 28 – Estou criando um objeto canva, o qual será representado pelo nome “pdf”. Lembro que “pdf” (variável) representa o caminho e nome do arquivo que será criado. Neste caso, omiti o caminho (local) onde o mesmo será gravado. Desta forma será gravado na pasta do meu projeto no pycharm.
d) Linha 32 – Estou chamando a rotina para imprimir a régua na página. Esta rotina pode ser retirada dos seus projetos futuros.
e) Linha 35 – Estou definindo a cor do texto “preto” a ser impresso, usando o método setFillColor('black') do objeto canva chamado ***pdf***
f) Linha 38 –Definindo um título do relatório que será impresso com o método setTitle('Relatório de Professores do Curso Técnico'). Neste caso, o título será “Relatório de Professores do Curso Técnico”)
g) Linha 41 – Definindo tipo da fonte e tamanho da fonte que será usado para impressão do próximo texto. Neste caso estou passando como parâmetro a fonte “Helvetica-Oblique”, tamanho 16 para o método setFont("Helvetica-Oblique", 16)

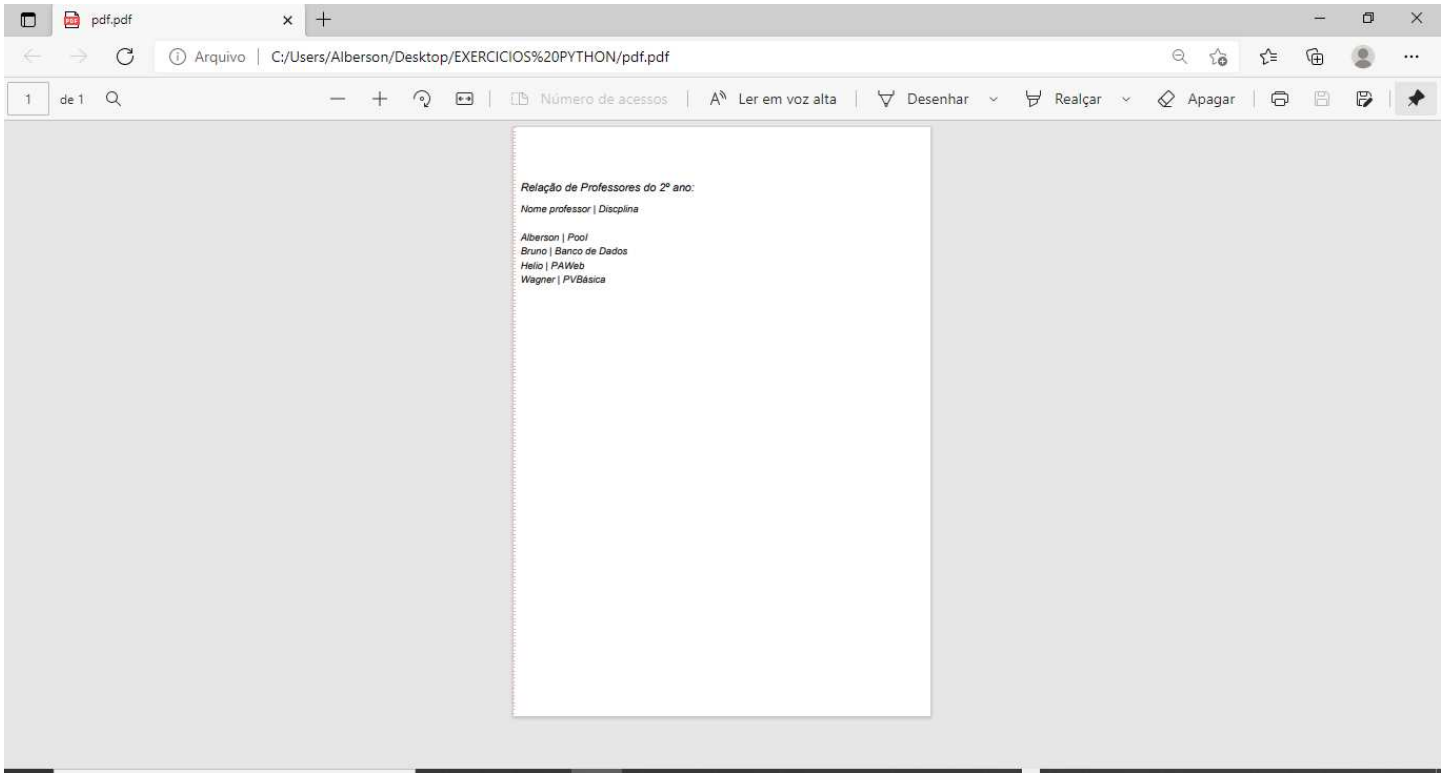
ATENÇÃO:

Vale ressaltar que fontes true types precisam ser registradas para uso. De acordo com o guia do reportlabs vocês precisarão escrever o seguinte código no seu programa no pycharm:

```
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont

Escreva a linha abaixo, por exemplo, no módulo principal do programa. Neste exemplo estou registrand#o
a fonte ‘Arial’, cujo o arquivo existente em c:\windows\fonts é “Arial.ttf”.

pdfmetrics.registerFont(TTFont('Arial', 'Arial.ttf'))
Pronto !! agora já posso usar esta fonte no meu programa
```

h) Linha 45 – O método drawString() é utilizado para escrever um texto numa colunaY de uma linhaX. Neste caso foi impresso “Relatório de Professores do 2º Ano”, na coluna 10 da linha 750, portanto escrevi pdf.drawString(10, 750, 'Relação de Professores do 2º ano:'). Lembro que esta medida está expressa em milímetro na folha A4.
i) Linha 48 – Redefini a fonte para impressão das próximas linhas, alterando o tamanho anterior para 14. Assim sendo o próximo comando uso do drawString já usará esta fonte e tamanho.
j) Linha 52 – Imprimindo uma legenda para as colunas de dados que serão impressas no meu relatório.
k) Linhas 54 até 59 – Estou percorrendo o dicionário, imprimindo os dados de nome e disciplina de cada professor. Repare que estou variando as linhas no método drawString, subtraindo sempre 10mm da variável ***x.***
l) Linha 62 – Esta linha, quando for executada, salva (cria) efetivamente o arquivo de relatório definido quando criamos o objeto canva, no código representado por ***pdf***. Portanto, o método save() é utilizado para gerar fisicamente o relatório definido pelo usuário no início da execução do programa. Lembro que o relatório será criado, neste exemplo, na pasta do projeto onde meu programa python está gravado.

ABRINDO O ARQUIVO PDF GERADO TEREMOS O SEGUINTE LAYOUT GERADO:



ATENÇÃO: IMPORTANTE SABER SOBRE MEDIDAS DE LINHAS E COLUNAS DE UMA PÁGINA A4 EM FORMATO PDF

A função drawString(y,x,texto) utiliza a folha do pdf como um plano cartesiano com eixos X e Y (a página possui 595.27 de largura e 841.89 de altura no padrão A4).

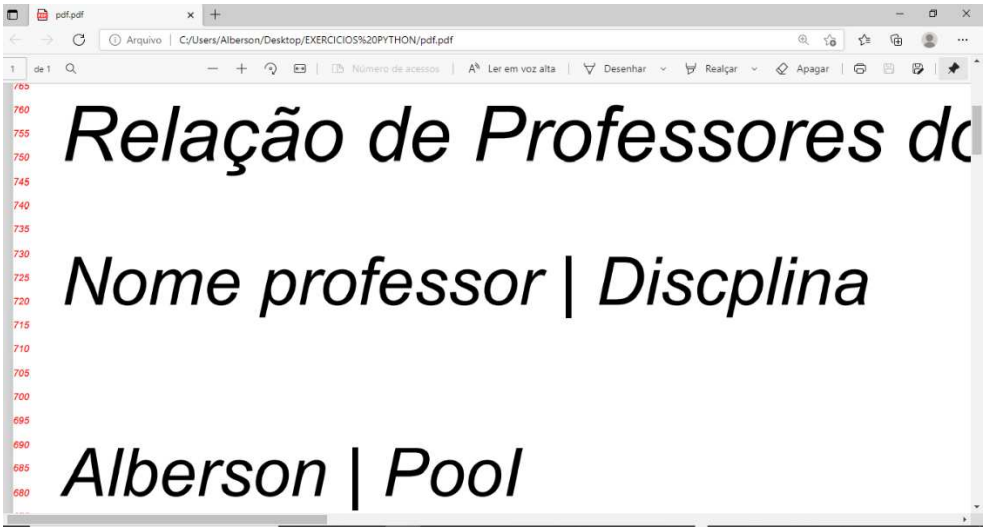
Então, basicamente a posição y = 247 e x = 700 para centralizar na tela.

REPARE QUE:

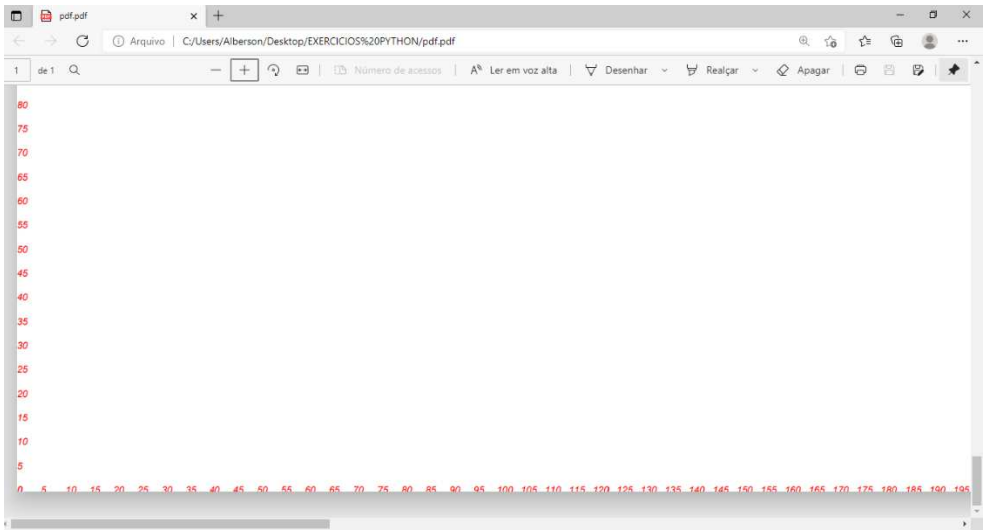
y CORRESPONDE A POSIÇÃO DE COLUNA ONDE O TEXTO SERÁ IMPRESSO

x CORRESPONDE A POSIÇÃO DA LINHA QUE O TEXTO SERÁ IMPRESSA

Dando um zoom na página gerada você perceberá que foi impressa uma "régua" do lado esquerdo e no rodapé da página PDF. Observe:



ATENÇÃO, REPARE QUE A ÚLTIMA LINHA DO ARQUIVO PDF É A ZERO E A PRIMEIRA COLUNA É TAMBÉM ZERO



Veja na tabela abaixo os métodos usados no programa acima do objeto canva, para criação de arquivos PDF´s:

Método	Para que serve
setFillCollor('<nomecor')	Define a cor da fonte do texto que será impresso na próxima impressão do método drawString()
setFont("nomefonte", tamanhofonte)	Define o nome da fonte e tamanho, para próximo texto impresso pelo método drawString()
drawString(coluna, linha, "texto")	Imprime um texto numa coluna e linha do arquivo PDF.
Canvas("caminho\nomearquivo.pdf")	Cria o objeto canva que representará o no caminho e nome definidos para o arquivo PDF
setTitle("titulo da Pagina")	Definindo titulo da página PDF que será criada pelo método save()
save()	Cria fisicamente o arquivo PDF definido no objeto canva instanciado.

39.1 – REGISTRO DE FONTE TRUE-TYPE PARA USO NUM ARQUIVO PDF

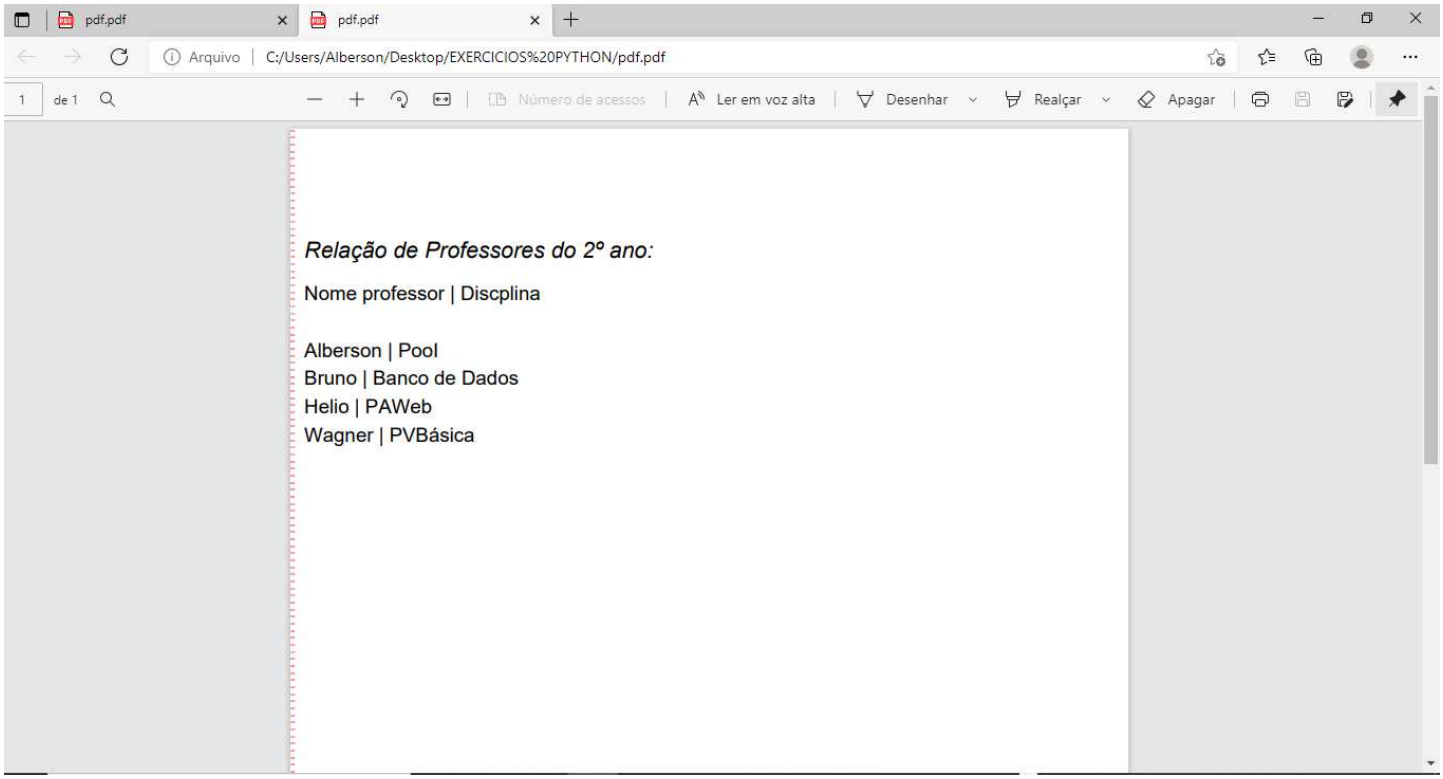
Neste capítulo vou somente reescrever o programa acima destacando as linhas necessárias para registro de uma fonte true-type, possibilitando com isso usá-las no seu arquivo PDF.

No programa anterior, exposto para gerar arquivo PDF, só usamos fonte "Helvetica-Oblique". Para usar a fonte "Arial", por exemplo, teremos que registrá-la. Veja nas linhas destacadas em amarelo o que foi necessário para isso:

```
#vamos usar o método canvas da biblioteca reportlab.pdfgen
from reportlab.pdfgen import canvas

#importando a biblioteca pdfmetrics que permite o uso do método registerFont(), escrito no módulo principal do
#programa
from reportlab.pdfbase import pdfmetrics

#importando o método TTFonts que permite reconhecer uma fonte true-type existente na pasta fonts do #windows
from reportlab.pdfbase.ttfonts import TTFont

def regua(pdf):
    '''
        :param pdf: este parâmetro receberá o objeto pdf criado no rotina gerarpdf
         Esta rotina só foi criada para mostrar os numeros de linhas e colunas da pagina PDF
    '''
    #DEFININDO A COR DO TEXTO (COR DA FONTE) A SER IMPRESSO NO ARQUIVO PDF
    pdf.setFillColor('red')
    for coluna in range(0, 595, 5):
        pdf.setFont("Arial", 2 )
        # imprimindo os numeros das coluna na última linha do arquivo PDF, linha 0
        pdf.drawString(coluna, 0, f'{coluna} ')

    for linha in range(0, 841, 5):
        pdf.setFont("Helvetica-Oblique", 2)
        # imprimindo os numeros de linhas na primeira coluna do arquivo PDF, coluna 0
        pdf.drawString(0, linha, f'{linha} ')


def gerarpdf(dicionario):
    try:
        #solicitando ao usuário o nome do arquivo PDF
        nomearquivo = input('Informe o nome do arquivo PDF que deseja gerar: ')
```

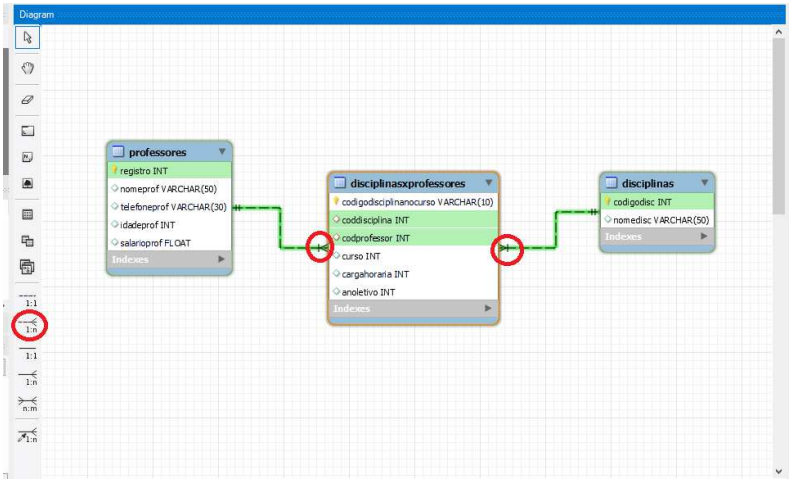
```
    #criando objeto pdf com nome do arquivo definido pelo usuário
    pdf = canvas.Canvas(f'{nomearquivo}.pdf')

    #CHAMANDO A ROTINA regua PARA IMPRIMIR UMA REGUA NA PÁGINA PARA MOSTRAR O POSICIONAMENTO DO TEXTO
    #NO ARQUIVO PDF. ESTA ROTINA NÃO PRECISA SER CHAMADA SEMPRE E NEM PRECISA SER CRIADA NOS SEUS PROGRAMAS
    regua(pdf)

    #DEFININDO A COR DO TEXTO DOS PRÓXIMOS TEXTOS A SEREM IMPRESSOS
    pdf.setFillColor('black')

    #Definindo o texto do título do documento a ser impresso
    pdf.setTitle('Relatório de Professores do Curso Técnico')

    # definindo nome da fonte e tamanho da mesma para o próximo texto a ser escrito no PDF
    pdf.setFont("Helvetica-Oblique", 16)

    # ATENÇÃO: AS REFERENCIAS DE LINHAS E COLUNAS NO RELATÓRIO SÃO TROCADAS NO MÉTODO drawString
    # Assim estou ESCREVENDO UM TEXTO NA PÁGINA PDF, NA LINHA 750 A PARTIR DA COLUNA 10
    pdf.drawString(10, 750, 'Relação de Professores do 2º ano:')

    # definindo nome da fonte e tamanho da mesma para o próximo texto a ser escrito,
    pdf.setFont("Arial", 14)

    # ATENÇÃO: AS REFERENCIAS DE LINHAS E COLUNAS NO RELATÓRIO SÃO TROCADAS NO MÉTODO drawString
    # Assim estou ESCREVENDO UM TEXTO NA PÁGINA PDF, NA LINHA 750 A PARTIR DA COLUNA 10
    pdf.drawString(10, 720, 'Nome professor | Discplina ')

    x=700 #MEDIDA EM MILIMETROS
    for nome, disciplina in dicionario.items():
        print(f'{nome} | {disciplina}')
        x-=20 #-20mm
        # o método drawString é usado para escrever na página PDF (FOLHA TOTAL POSSUI
        pdf.drawString(10, x, f'{nome} | {disciplina}')

    #criando arquivo PDF
    pdf.save()
    print(f'{nomearquivo} foi gerado com sucesso: ')
  except Exception as erro:
    print(f'Erro ao gerar o arquivo pdf: {erro}')

##################módulo principal do programa
#A linha abaixo registra a fonte Arial para uso
pdfmetrics.registerFont(TTFont('Arial', 'Arial.ttf'))

dicionario = {'Alberson': 'Pool', 'Bruno': 'Banco de Dados', 'Helio': 'PAWeb', 'Wagner': 'PVBásica'}
gerarpdf(dicionario)
```

Com a execução deste programa, perceba que os números de linhas e colunas, bem como a parte do texto que imprime o nome e as disciplinas de cada professor, estão sendo impressos com a fonte “Arial”.



39.2 - DESAFIOS

Observe as seguintes tabelas de um banco de dados, usadas em outras partes desta apostila:



Criar um programa python para permitir ao usuário gerar arquivos pdf´s das seguintes consultas, as quais podem ser disponibilizadas em menus para escolha:

- Dados completos de um professor diante de um número de registro informado pelo usuário
- Dados completos de professores cujo nomes iniciem com caracteres informados pelo usuário
- Nomes de todas as disciplinas de um curso informado pelo usuário
- Nomes de professores que dão aulas em um curso informado pelo usuário
- Carga horária TOTAL de um curso, cujo código tenha sido informado pelo usuário PARA UM DETERMINADO ANO LETIVO.

Atenção:

a) Você deve gerar os arquivos pdf´s acima mediante desejo expresso do usuário pela geração de cada consulta desejada
b) Crie menu com opções de geração do arquivo PDF
c) Não usem dados diferentes dos que estão disponíveis nas tabelas existentes neste banco de dados.

40 – Pandas x Openpyxl – Análise de dados em Excel

Trataremos agora sobre duas bibliotecas muito importantes para análise de dados em python. Vamos manipular arquivos em excel usando as duas bibliotecas mais utilizadas no mercado, a saber:

- Pandas
- Openpyxl

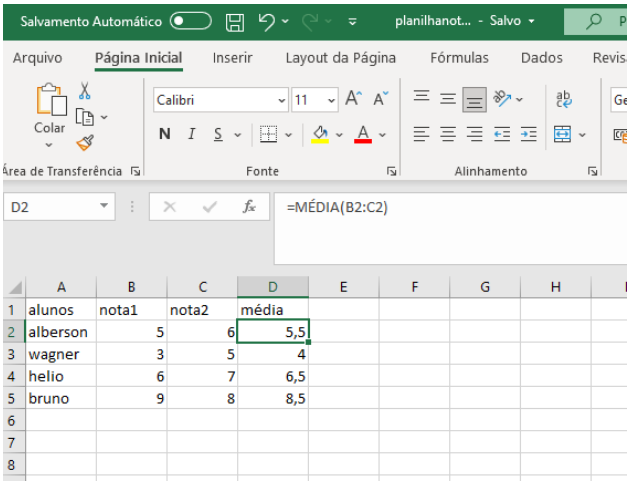
Características no uso do Pandas com Python:

- Mais utilizada no mercado
- Trata o Excel como **uma simples base de dados e não como planilha** que pode ter fórmulas, gráficos e/ou figuras
- Podemos fazer o que quiser com o arquivo manipulado.
- Poderá destruir a estrutura original do arquivo excel, caso regrave ou edite o arquivo.

Características no uso da biblioteca Openpyxl com Python:

- Usamos o arquivo excel como uma planilha, contendo fórmulas e gráficos principalmente
- Esta biblioteca trata o arquivo python como um CÓDIGO VBA
- É MENOS EFICIENTE QUE A BIBLIOTECA PANDAS
- Tenha cuidado ao usar esta biblioteca, apesar desta manter a estrutura original do arquivo excel. Você poderá ter surpresas. Faça testes antes de usá-la.

Para exemplificarmos o manuseio de dados em excel, usarei o arquivo que criei no meu computador, na unidade "c:\notas\planilhanotas.xlsx". Veja abaixo a estrutura deste arquivo:

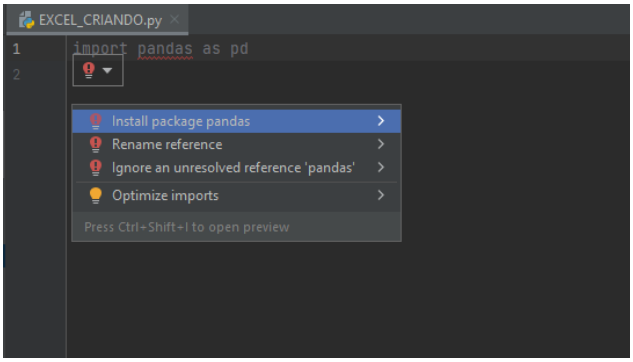


	A	B	C	D
1	alunos	nota1	nota2	média
2	alberson	5	6	5,5
3	wagner	3	5	4
4	helio	6	7	6,5
5	bruno	9	8	8,5
6				
7				
8				

Repare que a última coluna "média" possui uma fórmula.

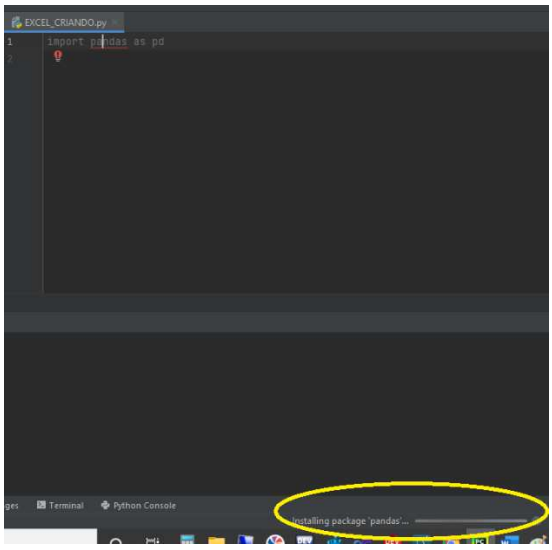
40.1 – ABRINDO E LENDO ARQUIVO EXCEL COM pandas E openpyxl

Primeiramente para manipularmos arquivos excel com pandas temos que importar a biblioteca pandas para dentro do seu código. Veja abaixo:

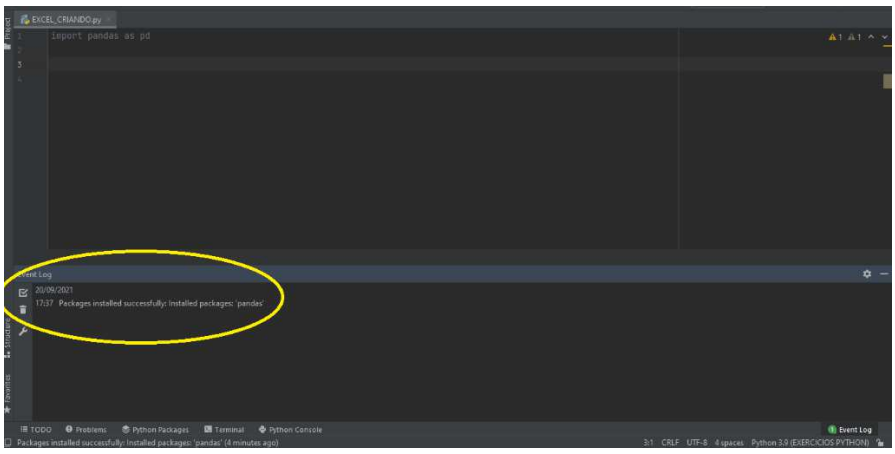


Como já tratado em capítulos anteriores, teremos que instalar o pacote pandas. Faça isso usando o pip, ou então, como ilustrado acima, através da opção ***"install package pandas"***.

Quando iniciamos o processo de instalação da biblioteca observe a área no canto inferior direito do seu pycharm, caso esteja usando esta ferramenta



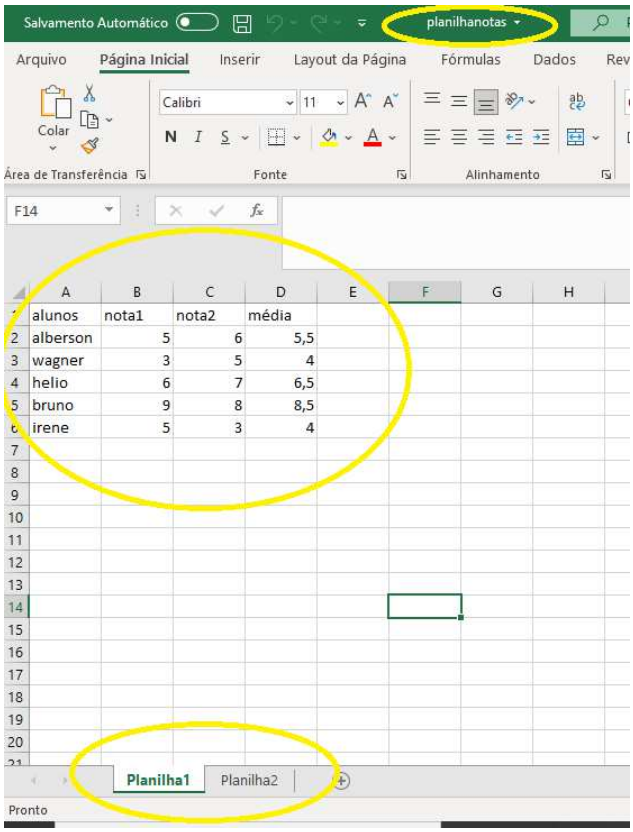
A instalação será iniciada e você poderá acompanhar o progresso na área circulada na figura acima. Aguarde até o processo ser finalizado. Veja abaixo:



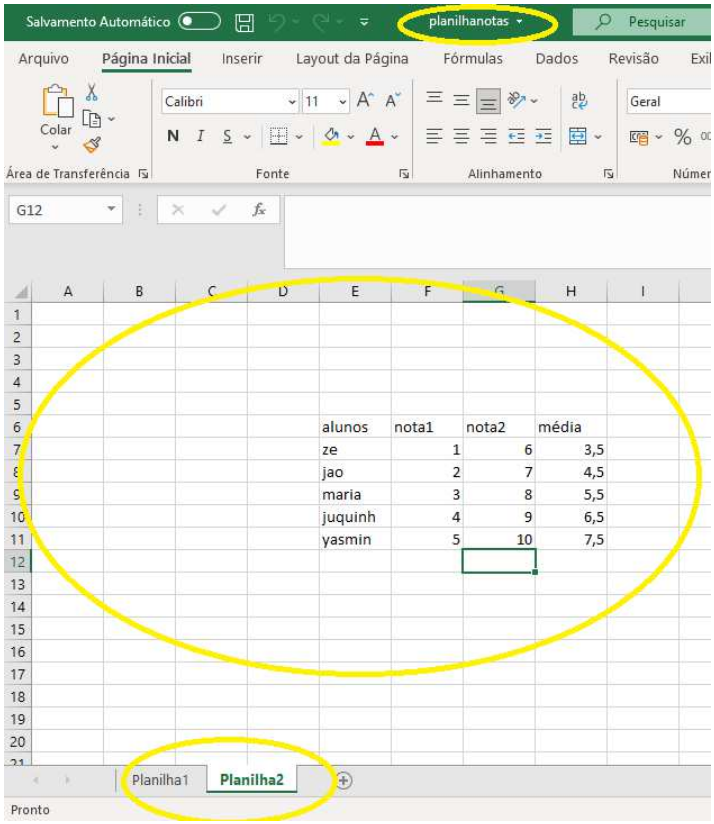
Para instalar a biblioteca Openpyxl, usei o pip, veja abaixo:

```
Prompt de Comando
Microsoft Windows [versão 10.0.19042.1237]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Alberson>pip install openpyxl
Collecting openpyxl
  Downloading openpyxl-3.0.8-py2.py3-none-any.whl (244 kB)
     |████████████████████████████████| 244 kB 2.2 MB/s
Collecting et-xmlfile
  Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.0.8

C:\Users\Alberson>
```

Agora vou digitar o código para abrir e imprimir a planilha lida na área do pycharm. Veja os exemplos de acesso aos dados de planilhas de um arquivo excel. Usarei as duas planilhas abaixo, criadas no arquivo “planilhanotas.xlsx”:



	A	B	C	D
1	alunos	nota1	nota2	média
2	alberson	5	6	5,5
3	wagner	3	5	4
4	helio	6	7	6,5
5	bruno	9	8	8,5
6	irene	5	3	4



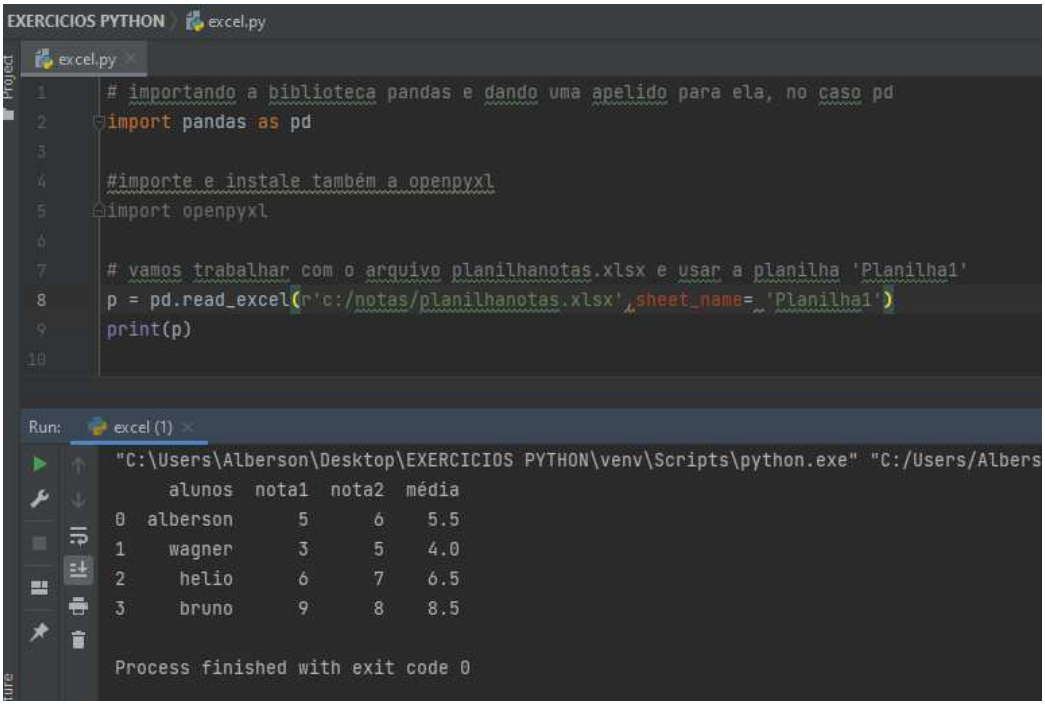
	E	F	G	H
6	alunos	nota1	nota2	média
7	ze	1	6	3,5
8	jao	2	7	4,5
9	maria	3	8	5,5
10	juquinh	4	9	6,5
11	yasmin	5	10	7,5

Repare:

a) Temos então o arquivo “planilhanotas.xlsx” e neste temos duas planilhas chamadas respectivamente “planilha1” e “planilha2”.
b) Veja que na “planilha2” os dados estão localizados em células mais afastadas da borda esquerda e do topo, a partir da célula E6, isso trará algumas consequências no momento da leitura dos dados.

Vou acessar a ***planilha1*** usando o método ***read_excel*** do objeto pandas, veja:

Exemplo 1:



Repare:

a) A planilha foi criada no excel e foi gravada na pasta "C:\notas\" com nome de "planilhanotas.xlsx". Quando temos um arquivo excel gravado numa pasta diferente da que está o programa python, temos que indicar o

r antes do caminho indicado como primeiro parâmetro do método *read_excel*, para informar que faremos leitura no python.

b) O parâmetro ***sheet_name*** é usado para indicar o nome da planilha existente que vou acessar, no arquivo que será aberto para leitura.

c) Por fim, o objeto ***p*** criado representará a nossa planilha dentro do python.

d) Quando usei o ***print(p)***, os dados da ***"planilha1"*** foram impressos na tela para o usuário. Simples assim!!

Exemplo 2: Podemos usar também o método ***ExcelFile*** para abrir e ler um arquivo excel com o pandas. Quando usamos este método, podemos abrir diversas planilhas conforme exemplo a seguir, veja:

```
excel.py
1  # importando a biblioteca pandas e dando uma apelido para ela, no caso pd
2  import pandas as pd
3
4  #criando um dataframe para representar o arquivo planilhanotas.xlsx
5  df = pd.ExcelFile ('c:/notas/planilhanotas.xlsx')
6
7  #Impriminfo od nomrd fr planilhas existentes no dataframe criado
8  print(f'Planilhas do arquivo planilhasnotas.xls ==> {df.sheet_names}')
9
10 #método parse serve para trabalharmos com planilhas especificas de uma pasta de trabalho do excel
11 aba1 = df.parse('Planilha1')
12 aba2 = df.parse('Planilha2')
13
14 #imprimindo a planilha 1, representada por aba1
15 print(aba1)
16
17 #imprimindo a planilha 2, representada por aba2
18 print(aba2)
```

Repare:

a) Na linha 2: importei a biblioteca pandas e a referenciarei no código como sendo ***pd***
b) Na linha 5: Criando um ***DataFrame*** chamado ***df*** para representar o arquivo "***planilhanotas.xlsx"*** que está gravado em ***"c:\notas\".*** (DataFrame = Planilha lida).
c) Na linha 8: estou apenas imprimindo os nomes de planilhas existentes no arquivo o método ***sheet_names*** DO OBJETO ***DATAFRAME***.
d) Linhas 11 e 12: com uso do método ***parse*** do objeto ***DataFrame***, atribuo para aba1 e aba2 as planilhas existentes no arquivo até então aberto.
e) Nas linhas 15 e 18: imprimindo as planilhas existentes conforme figura a seguir:

```
Run: excel (1)
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/EXERCICIOS PYTHON/excel.py"
Planilhas do arquivo planilhasnotas.xls ==> ['Planilha1', 'Planilha2']
     alunos  nota1  nota2  média
0  alberson      5      6    5.5
1    wagner      3      5    4.0
2     helio      6      7    6.5
3     bruno      9      8    8.5
4     irene      5      3    4.0
   Unnamed: 0  Unnamed: 1  Unnamed: 2  ...  Unnamed: 5 Unnamed: 6 Unnamed: 7
0         NaN         NaN         NaN  ...         NaN        NaN        NaN
1         NaN         NaN         NaN  ...         NaN        NaN        NaN
2         NaN         NaN         NaN  ...         NaN        NaN        NaN
3         NaN         NaN         NaN  ...         NaN        NaN        NaN
4         NaN         NaN         NaN  ...       nota1      nota2      média
5         NaN         NaN         NaN  ...           1          6        3.5
6         NaN         NaN         NaN  ...           2          7        4.5
7         NaN         NaN         NaN  ...           3          8        5.5
8         NaN         NaN         NaN  ...           4          9        6.5
9         NaN         NaN         NaN  ...           5         10        7.5
```

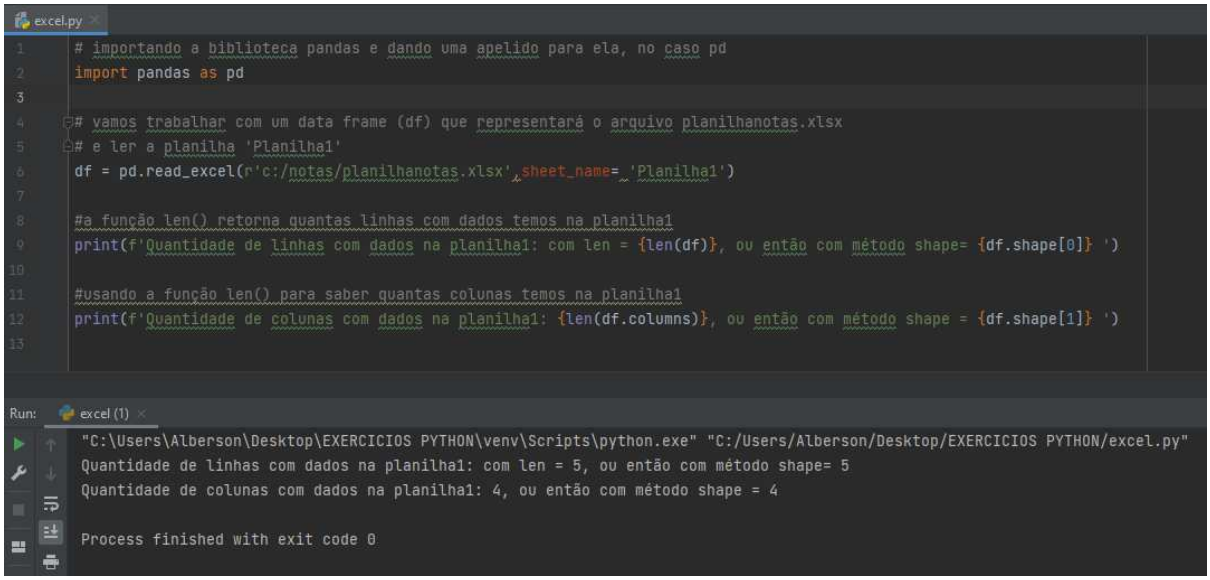
IMPORTANTE!

NOTE QUE A PLANILHA2, FOI IMPRESSA COM CÉLULAS CONTENDO "NaN".

ISSO ACONTECEU PORQUE TAIS CÉLULAS ESTÃO VAZIAS E OS DADOS ESTÃO CONTIDOS A PARTIR DA COLUNA 4

DA PLANILHA2

Exemplo 3:



Neste exemplo:

a) Na linha 2: importei a biblioteca pandas e a "apelidei" como ***pd***, para uso no programa.
b) Na linha 6: Abri a "***planilha1"*** do arquivo ***"c:\notas\planilhanotas.xlsx".***
c) Na linha 9: A função ***len(df)*** e o método ***shape(0)*** retornam a quantidade de **linhas DE DADOS** da planilha, **a PARTIR DA LINHA DE ÍNDICE 0 (NO CASO LINHA 2 DA PLANILHA1)**, ou seja, neste caso 5 dados gravados.

ATENÇÃO:

A LINHA 1 DA PLANILHA É CONSIDERADA APENAS DE COLUNAS. DESTA FORMA, NÃO É CONTADA COMO LINHA DE DADOS DA PLANILHA, POR ISSO A FUNÇÃO len() RETORNOU 5

d) Na linha 12: a função ***len(df.columns)*** e o método ***shape[1]*** retornam a quantidade exata de colunas da ***"planilha1"***.

Só para lembrar, observe a "planilha1":

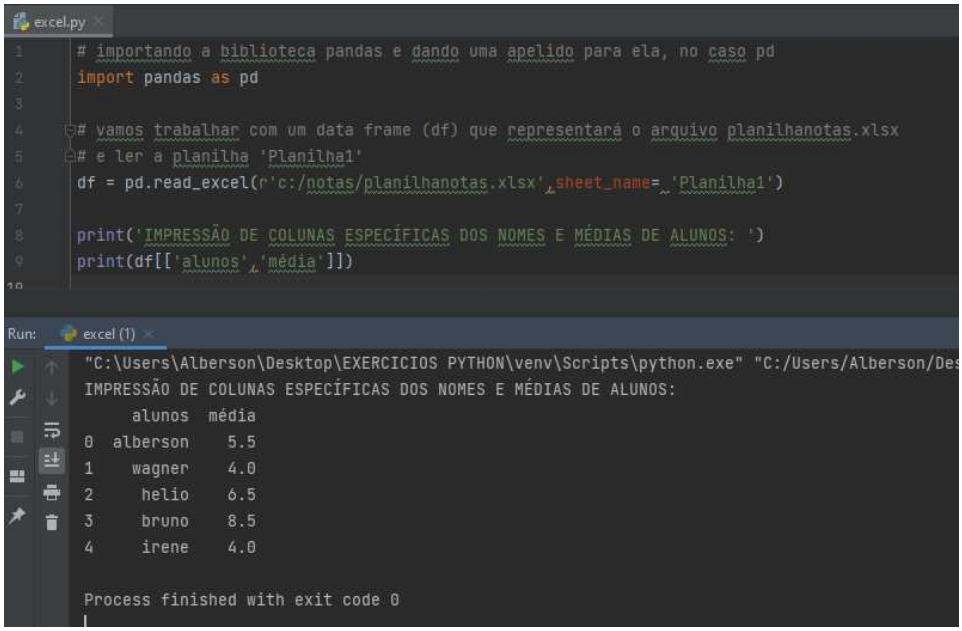
	A	B	C	D
1	alunos	nota1	nota2	média
2	alberson	5	6	5,5
3	wagner	3	5	4
4	helio	6	7	6,5
5	bruno	9	8	8,5
6	irene	5	3	4
7				

IMPORTANTE:

CASO DESEJÁSSEMOS LER DADOS DA *"PLANILHA2"*:

AS CÉLULAS VAZIAS MAIS A ESQUERDA E NO TOPO DA MESMA SERIAM CONSIDERADAS NOS TOTAIS RETORNADOS:

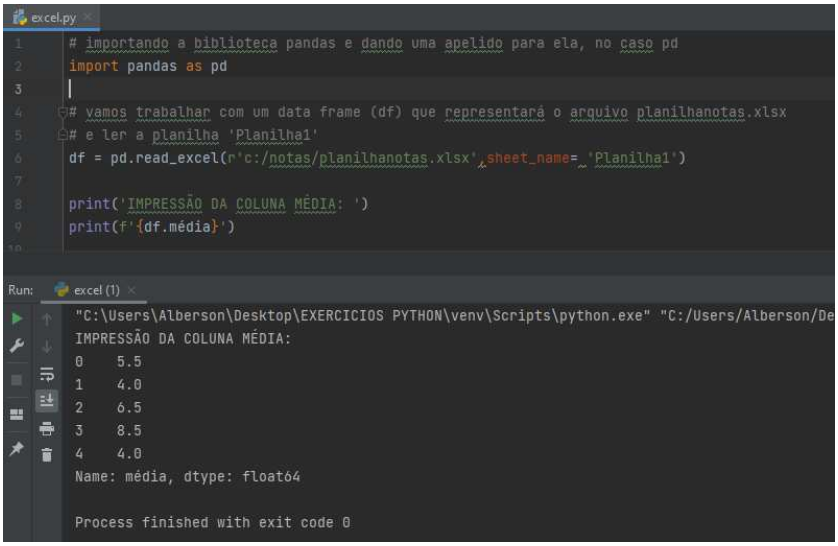
Exemplo 4: Impressão de somente algumas colunas de uma planilha:



Repare:

a) Linha 9: Neste exemplo são mostradas somente as colunas ***"alunos"*** e ***"média"*** da "***planilha1"***.

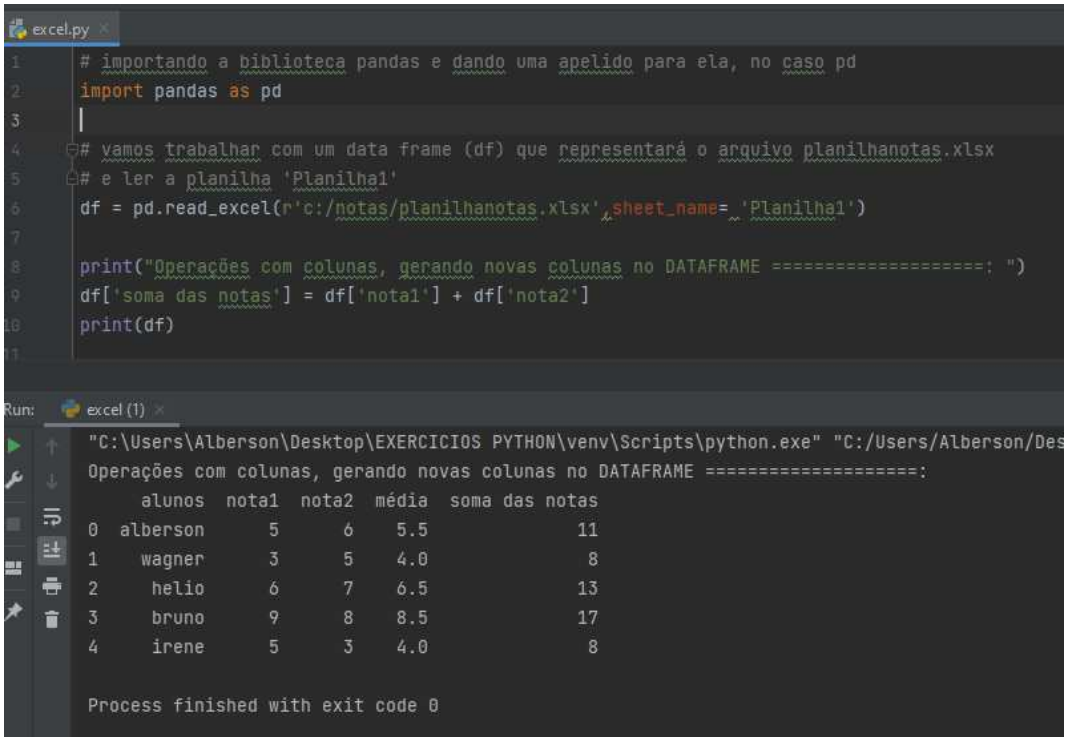
Exemplo 5: Impressão de dados de uma só coluna, podemos indicar desta forma também:



Repare:

a) Neste caso, no final da impressão dos dados da coluna média, são exibidos informações do objeto "***df.media"***. Podem ser desconsiderados.

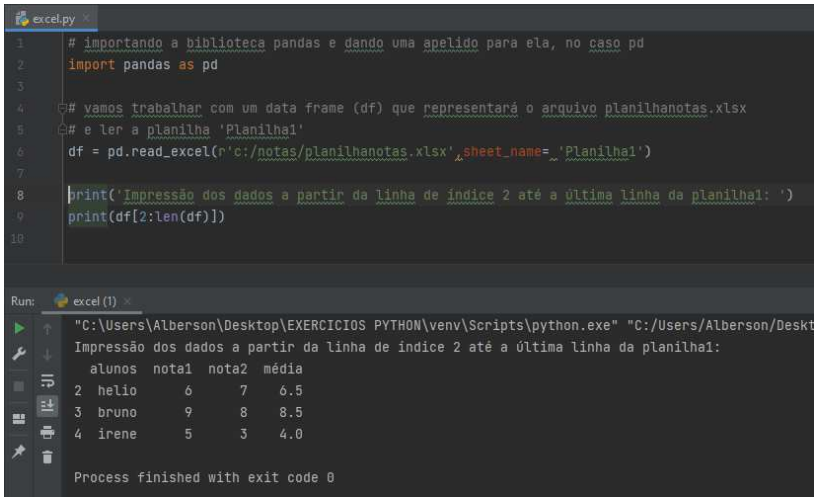
Exemplo 6: Realizando cálculos com colunas do DataFrame:



Vejamos:

a) Linha 9: Está sendo criada uma coluna no ***DataFrame*** que será chamada de ***"soma das notas"***. Esta nova coluna será resultante da soma realizada entre as colunas ***"nota1"*** e ***"nota2".***
b) Linha 10: Quando mando imprimir o ***df (DataFrame)***, observe que a nova coluna já é parte constante da impressão. **OBSERVAÇÃO: ESTA NOVA COLUNA NÃO É CRIADA NA PLANILHA FÍSICA DO EXCEL.**

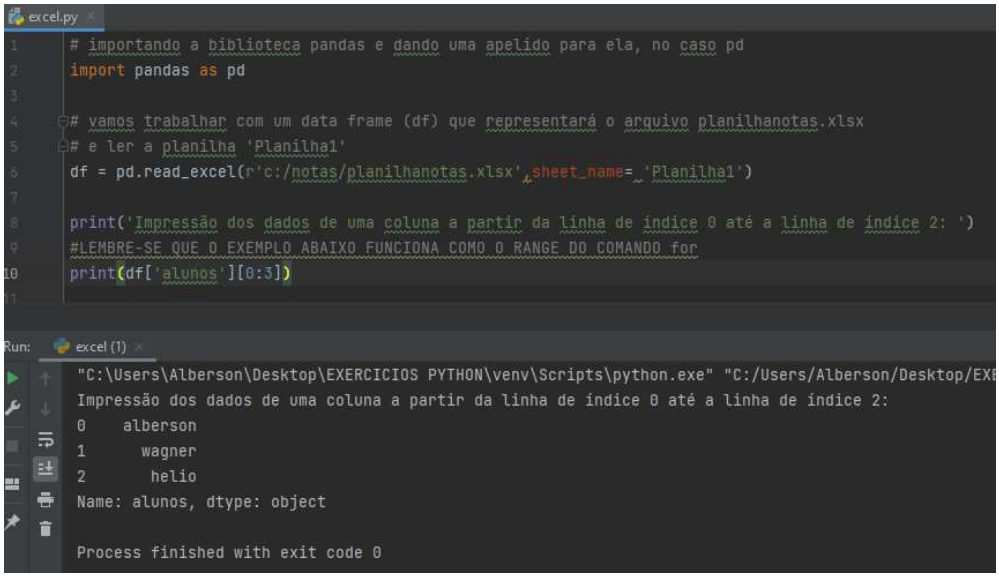
Exemplo 7: Impressão de dados a partir de uma linha específica da planilha:



Observem:

a) Linha 9: No comando print desta linha, limitei o intervalo de impressão a partir da linha de **ÍNDICE 2** até o final da planilha. **OBSERVAÇÃO: LEMBREM-SE QUE OS ÍNDICES DE LINHAS INICIAM A PARTIR DA LINHA 0 (ZERO).**
b) Veja que na impressão, a linha dos cabeçalhos também são impressas, pois como mencionado anteriormente elas não pertencem ao range de DADOS DA PLANILHA.

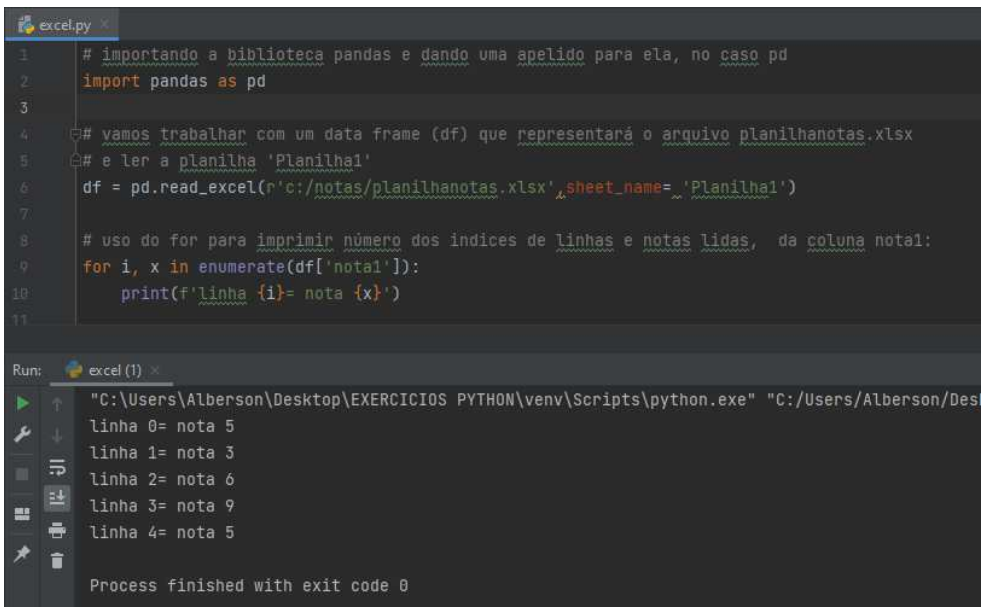
Exemplo 8: Imprimindo intervalos de dados de uma coluna específica:



Repare:

a) Linha 10: Defini o nome da coluna a ser impressa ***('alunos')*** e indiquei o limite de linhas de dados que deveriam impressas, ou seja, linhas 0,1 e 2. **IMPORTANTE: LEMBRE-SE QUE [0:3] INDICA DA LINHA DE ÍNDICE 0 ATÉ A LINHA DE INDICE 2, pois o número 3 indicado informa limite da impressão (lembre-se: 3-1= 2).**
b) Neste exemplo a linha dos cabeçalhos foi omitida, pois o comando de impressão referenciou LINHAS DE DADOS DE ÍNDICE 0 ATÉ ÍNDICE 2 (**[0:3]**).

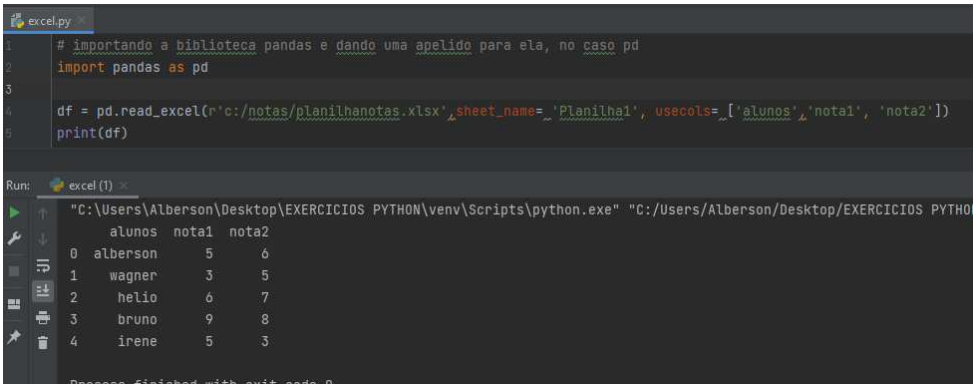
Exemplo 9: Percorrendo com laço de repetição dados de uma coluna específica:



Observem:

a) Linha 9: Só para lembrar, observem que a variável ***i*** receberá o índice de cada nota (cada linha) lida na coluna ***"nota1"***. As notas lidas serão armazenadas em ***x***.
b) Quando realizamos impressões dentro de laços de repetições como o exemplificado acima, somente as linhas de dados são exibidas

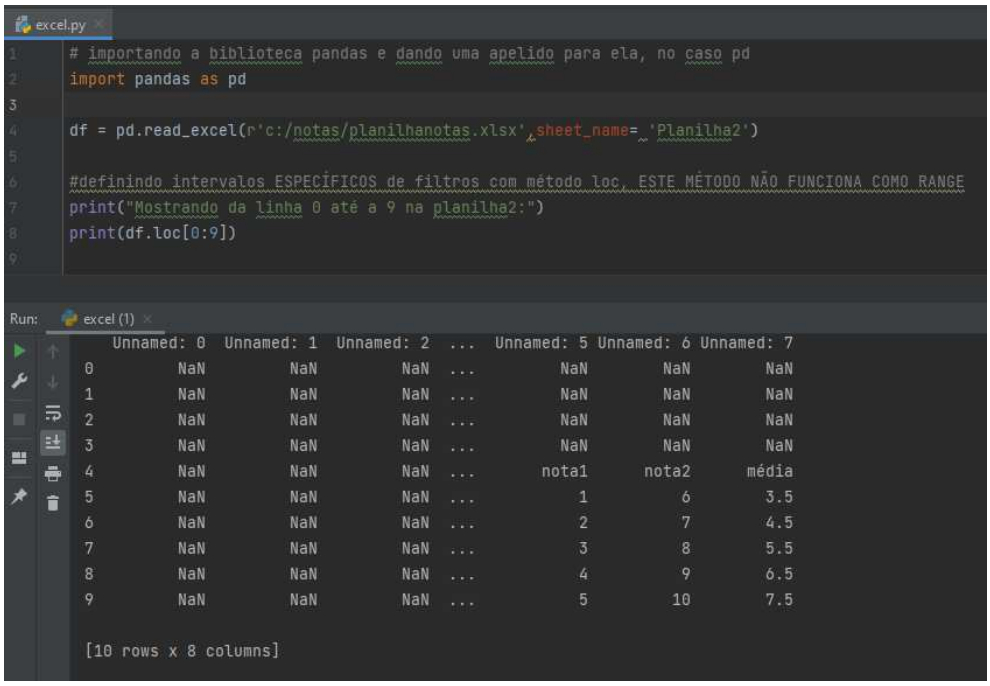
Exemplo 10: Abrindo planilha e determinando quais colunas serão usadas:



Repare:

a) Linha 4= no método "***read_excel"***, foi informado parâmetro "***usecols"***, na qual determinei as colunas que serão lidas da planilha.

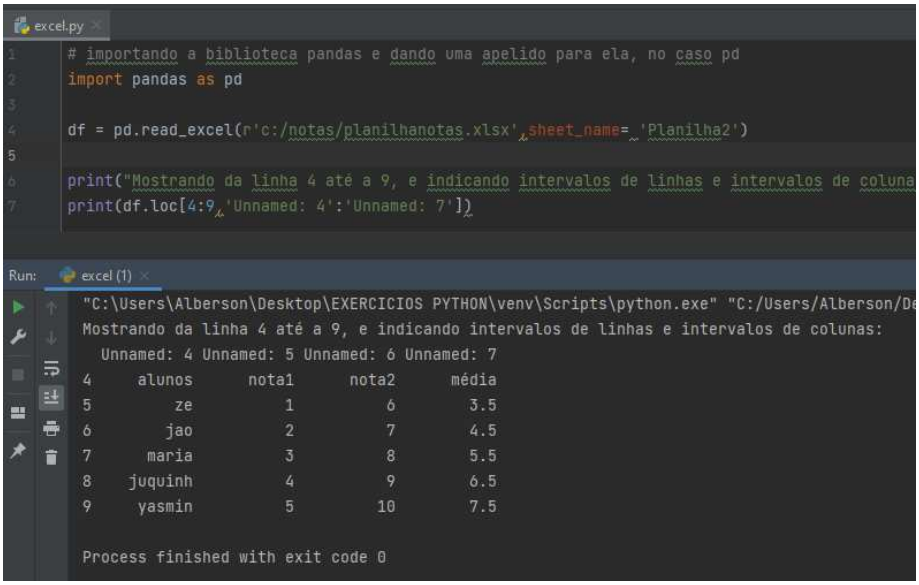
Exemplo 11: **<u>MÉTODO loc[]</u>**, para buscar dados específicos da planilha:



IMPORTANTE OBSERVAR:

a) Linha 8: estou imprimindo dados das linhas 0 até a 9 da "***planilha2"***. Neste caso os limites informados com loc[] ***<u>SÃO EXATOS.</u>***
b) Repare também que algumas colunas da planilha não são impressas, porém estão contidas no ***DataFrame (df)***

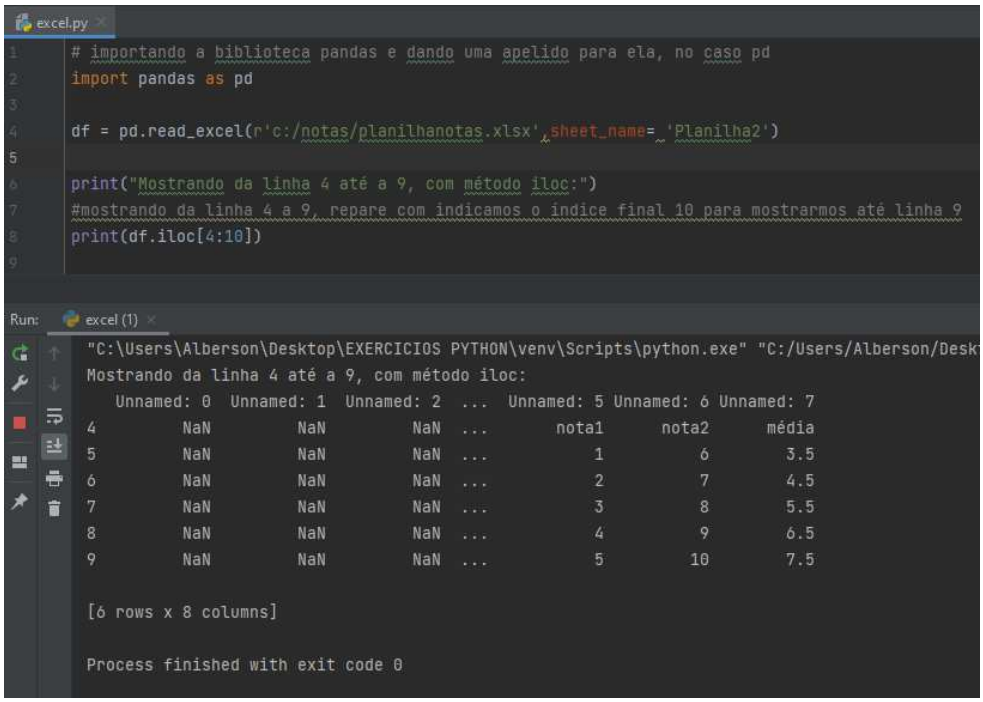
Exemplo 12: Indicando com método loc[], intervalos de linhas e colunas a serem impressos:



Repare:

a) Linha 7: Determinei que os dados impressos estão contidos no intervalo a ***partir da linha 4 até a 9*** e nas colunas do intervalo entre ***Unnamed 4 até Unnamed 7***. Lembre-se que os limites SÃO EXATOS COM MÉTODO loc[]
b) Repare que quando temos a planilha com dados afastados das bordas superior e esquerda, os nomes das colunas são definidos com ***Unnamed.***

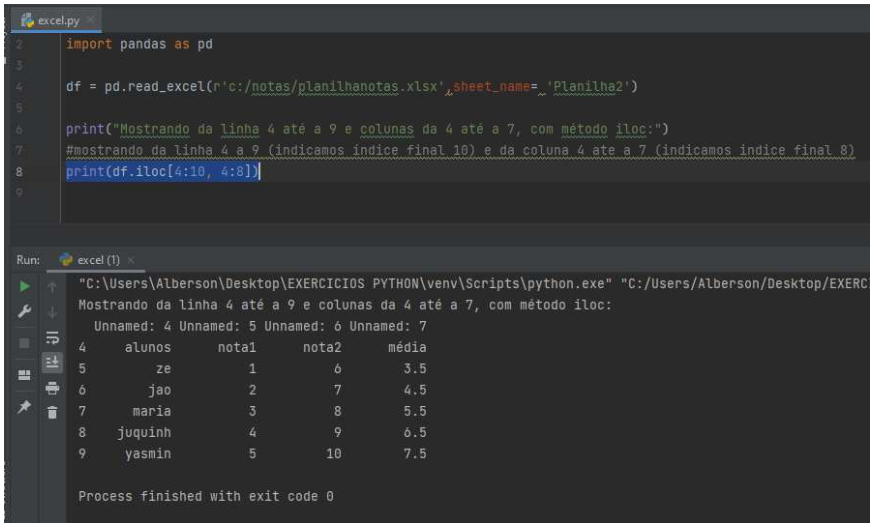
Exemplo 13: USO DO MÉTODO iloc[], VEJA:



Vejamos as diferenças da impressão com ***iloc[]***:

a) Neste caso, os limites informados ***NÃO SÃO EXATOS***, ou seja, funciona como range [4:10] definindo desta forma limites de impressão entre linhas de índice 4 até índice 9.

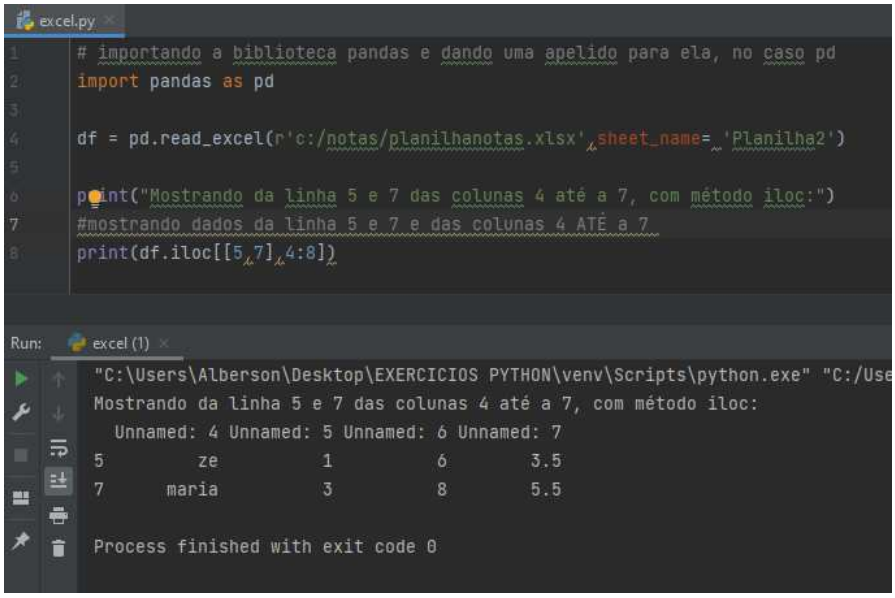
Exemplo 14: iloc[] PARA LIMITAÇÃO DE FAIXAS DE LINHAS E COLUNAS



Repare:

a) Linha 8: os limites de impressão informados mostram dados das LINHAS 4 até 9 e COLUNAS 4 até 7. Veja que neste caso não precisei indicar os nomes ***Unnamed*** de colunas.
b) Neste exemplo os cabeçalhos de colunas são impressos.

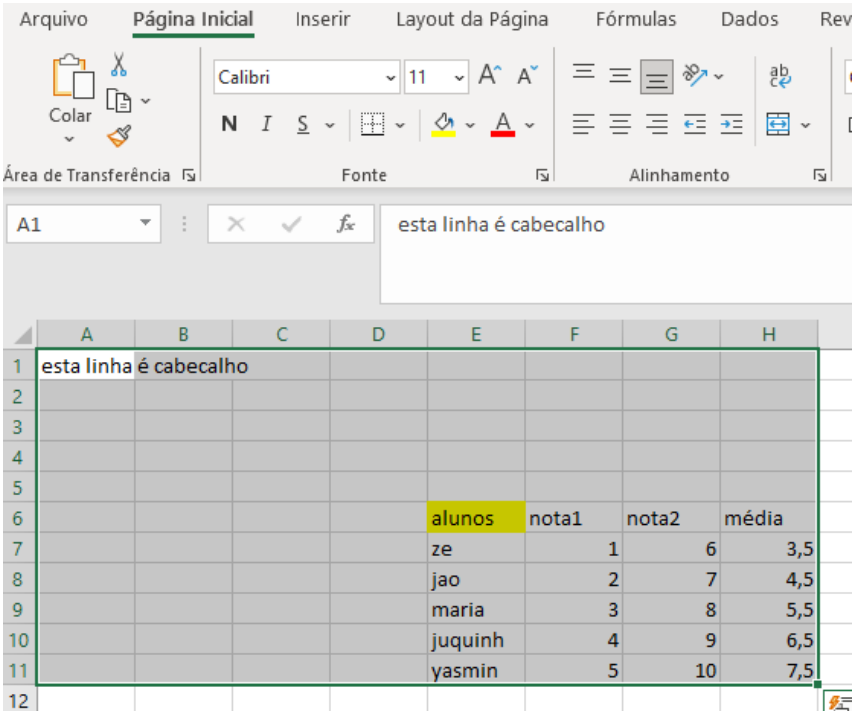
Exemplo 15: Filtrando dados de linhas específicas e range de colunas.



Observem:

a) Linha 8: Nesta linha indiquei a lista de linhas que devem ser apresentadas, e a faixa de colunas (entre 4 e 7) destas linhas.
b) Neste caso os cabeçalhos de colunas são impressos.

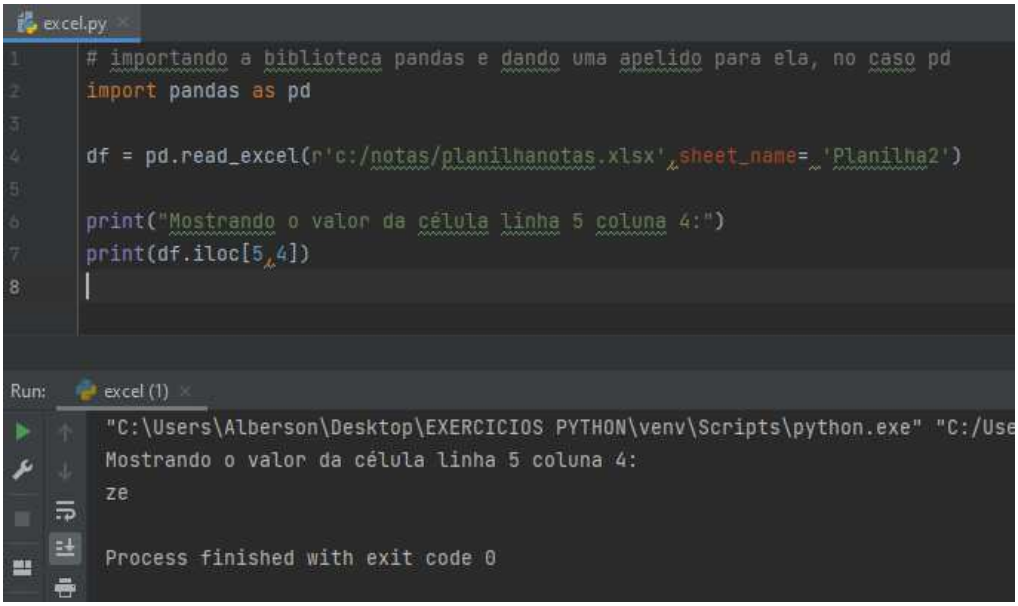
Exemplo 16: Filtrando dado de uma célula específica da planilha. Para este exemplo vamos considerar o seguinte conteúdo da ***"planilha2":***



	A	B	C	D	E	F	G	H
1	esta linha	é cabecalho						
2								
3								
4								
5								
6					alunos	nota1	nota2	média
7					ze	1	6	3,5
8					jao	2	7	4,5
9					maria	3	8	5,5
10					juquinh	4	9	6,5
11					yasmin	5	10	7,5
12								

IMPORTANTE:

A LINHA 1 DESTA PLANILHA ACIMA, NÃO É CONSIDERADA A LINHA DE ÍNDICE 0 (ZERO), POIS ELA É A LINHA DE CABEÇAHOS DE COLUNAS



Veja:

a) Linha 7: Nesta linha estou imprimindo o conteúdo da célula de linha 5 coluna 4, que é ***"zé"***.
b) Conforme destacado no início deste item da apostilia, a linha 1 da planilha excel física é encarada como sendo de cabeçalhos das colunas. **A linha de dados que inicia com índice 0 é a linha 2 da planilha**. Assim sendo, **linha de índice 5 é a linha 7 da planilha**. Lembre-se que a **coluna A** da planilha **é a de índice 0** e que a **coluna E é a de índice 4**, observe:

	A	B	C	D	E	F	G	H
1	esta linha é cabecalho							
2								
3								
4								
5								
6					alunos	nota1	nota2	média
7					ze	1	6	3,5
8					jao	2	7	4,5
9					maria	3	8	5,5
10					juquinh	4	9	6,5
11					yasmin	5	10	7,5
12								

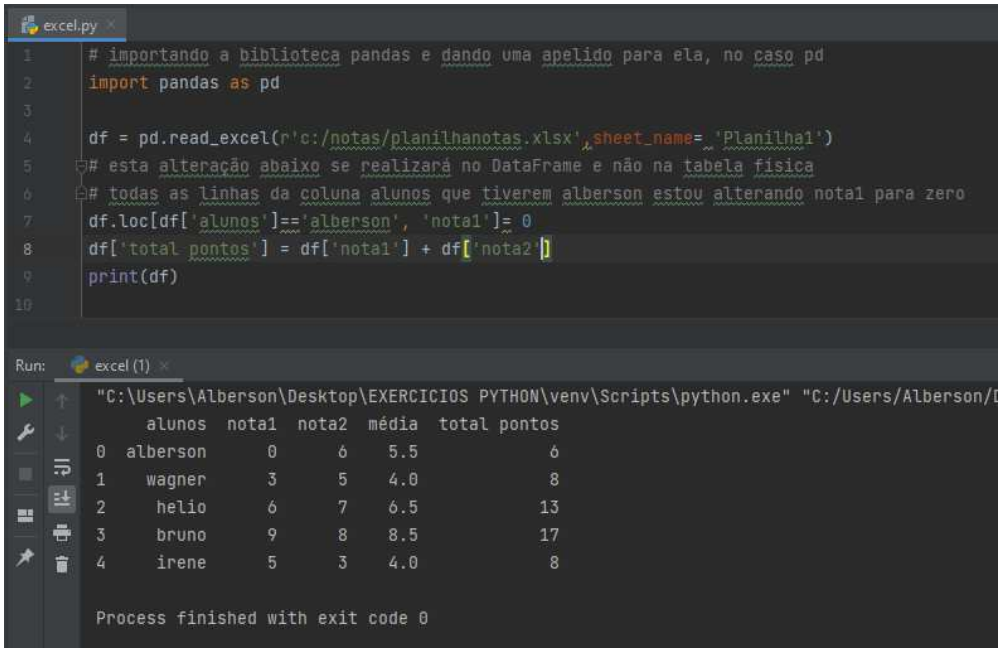
Exemplo 17: Alterando dados do ***DataFrame***:

```
excel.py
1  # importando a biblioteca pandas e dando uma apelido para ela, no caso pd
2  import pandas as pd
3
4  df = pd.read_excel(r'c:/notas/planilhanotas.xlsx', sheet_name= 'Planilha1')
5  # esta alteração abaixo se realizará no DataFrame e não na tabela física
6  # todas as linhas da coluna alunos que tiverem alberson estou alterando nota1 para zero
7  df.loc[df['alunos']=='alberson', 'nota1']= 0
8  print(df)
9

Run:  excel (1)
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Des
     alunos  nota1  nota2  média
0  alberson      0      6    5.5
1    wagner      3      5    4.0
2     helio      6      7    6.5
3     bruno      9      8    8.5
4     irene      5      3    4.0

Process finished with exit code 0
```

Veja no exemplo acima:

a) Linha 7: Estou localizando todos os nomes ***'alberson'***, na coluna ***'alunos'*** e alterando suas ***'nota1'*** para ***0 (zero)***
b) A planilha mostrada é o ***DataFrame*** e não a planilha física.

Exemplo 18: Somando dados de colunas e gerando novas colunas no ***DataFrame***:



Repare:

a) Na linha 8: Estou somando cada ‘nota1’ com ‘nota2’ e criando nova coluna no DataFrame chamada ‘total pontos’

40.2 –CRIANDO ARQUIVO NO EXCEL COM PANDAS

Para alterarmos uma planilha excel já existente, podemos gravar os dados processados e armazenados num DataFrame na referida tabela, fazendo com que a tabela original seja alterada.

Para isso usaremos o método to_excel. Veja sintaxe básica abaixo:

df.**to_excel**('**caminhonomearquivo**', sheet_name = '**nomeplanilha**', na_rep = '**#N/A**', header = **True**, index = **False**)

Onde:

to_excel = método do objeto DataFrame do pandas que gera uma tabela excel.

sheet_name = para indicarmos o nome da planilha que será criada dentro do arquivo excel

na_rep = indicamos ‘#N/A’ para o caso de existirem células vazias. Caso omita irá inserir ‘NaN’ nas células vazias

header = indicamos **True** para informar que os títulos das colunas serão os mesmos do DataFrame já criado

index = indicamos **False** para que a tabela não seja gerada com os índices de linhas do DataFrame

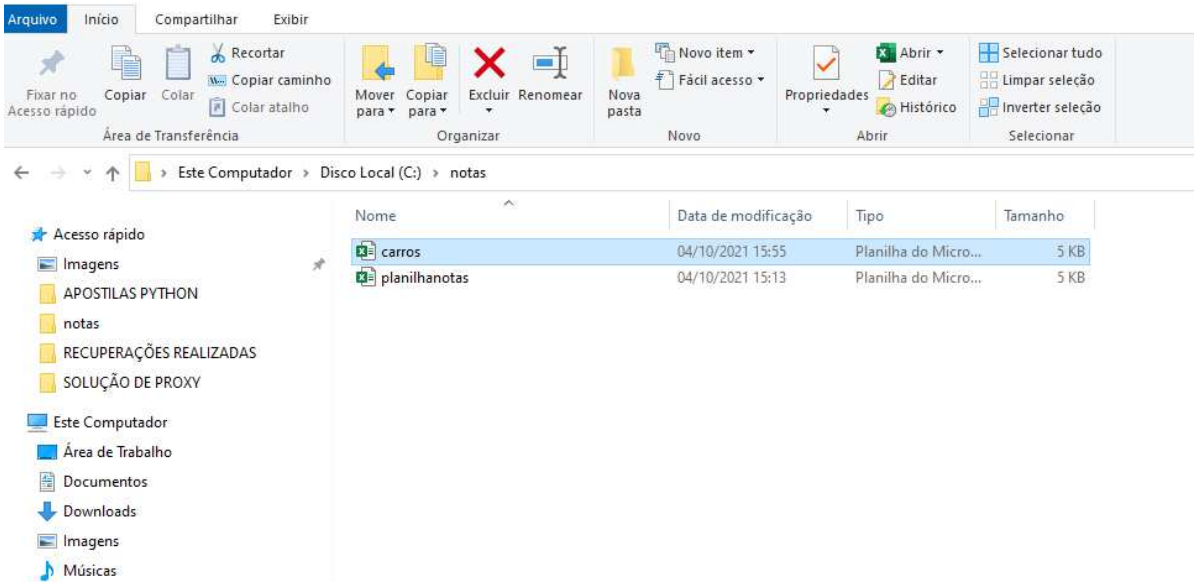
Exemplo 1:

```
excel.py
1  # importando a biblioteca pandas e dando uma apelido para ela, no caso pd
2  import pandas as pd
3  #Criando um dicionário com dados a serem gerados numa tabela excel
4  dicionario = {'Carros': ['Onix','Gol','Civic'],
5                'Ano': [2015,2019, 2021],
6                'Valores': [50000.00, 60000.0, 100000.0]
7               }
8  #imprimindo o dicionário para breve conferência
9  print(dicionario)
10 #Criando um DataFrame com o dicionário de dados
11 df = pd.DataFrame(dicionario)
12 #imprimindo o DataFrame que foi criado em formato de tabela
13 print(df)
14
15 #Gerando o arquivo excel carros.xlsx, contendo uma planilha de dados chamada 'Planilha1', que tenha células vazias com
16 #conteúdo '#N/A', que contenha cabeçalhos informados e não grave os índices de linhas na nova planilha excel
17 df.to_excel('c:/notas/carros.xlsx', sheet_name='Planilha1', na_rep='#N/A', header=True, index=False)
18

Run: excel (1)
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/EXERCICIOS PYTHON/excel.py"
{'Carros': ['Onix', 'Gol', 'Civic'], 'Ano': [2015, 2019, 2021], 'Valores': [50000.0, 60000.0, 100000.0]}
  Carros   Ano   Valores
0   Onix  2015   50000.0
1    Gol  2019   60000.0
2  Civic  2021  100000.0
```

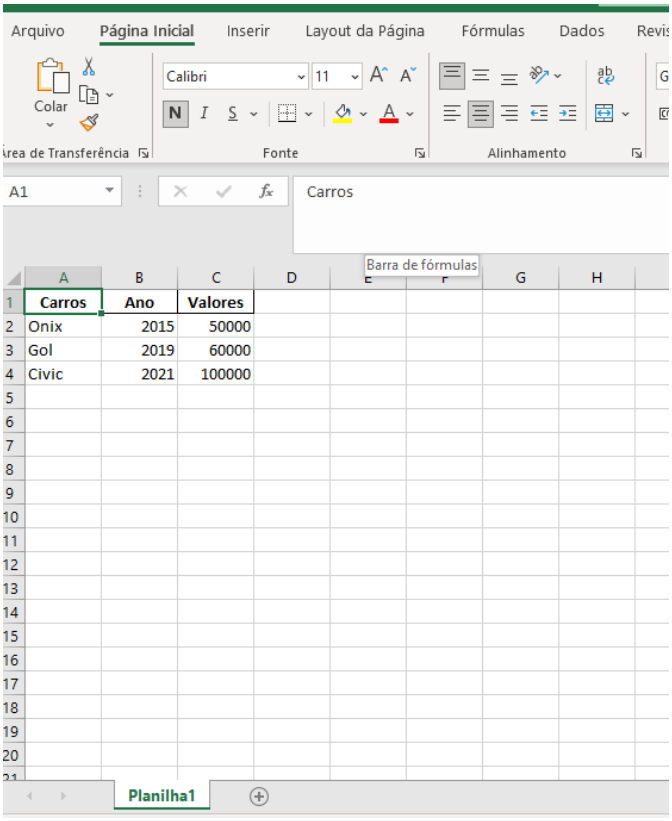
Observe:

a) Linha 4: Primeiramente criei um dicionário com os dados desejados, para transformar em DataFrame
b) Linha 11: Transformei o dicionário em DataFrame, gerando com isso uma tabela na memória.
c) Linha 17: Transformei o DataFrame em arquivo excel com parâmetros definidos.

Com a execução do programa acima, geramos então o seguinte arquivo excel, na pastas notas, chamado carros.xlsx:



Abrindo o arquivo criado no excel, temos a seguinte estrutura:



	A	B	C
1	Carros	Ano	Valores
2	Onix	2015	50000
3	Gol	2019	60000
4	Civic	2021	100000

DICA MUITO USADA!!!!!!

DE AGORA PARA FRENTE, PODEMOS POR EXEMPLO, LER DADOS DE UM BANCO DE DADOS E GERAR UM DICIONÁRIO CONTENDO TODOS OS REGISTROS GRAVADOS. EM SEGUIDA, PODEMOS GERAR UMA PLANILHA EXCEL COM ESTES DADOS.

Exemplo 2:

PROJETO 4ºBIM.py

```
1   import pandas as pd
2   import openpyxl
3
4   # Criando os dataframes Pandas para armazenar os dados DE DICIONÁRIOS
5   df1 = pd.DataFrame({'Data': [2, 4, 6, 8]})
6   df2 = pd.DataFrame({'Data': [100, 150, 200, 250]})
7   df3 = pd.DataFrame({'Data': [3, 6, 9, 12]})
8
9   # Usando o MÉTODO ExcelWriter, cria um ARQUIVO .xlsx, usando engine='openpyxl'
10  arquivo = pd.ExcelWriter('tabelas_exemplo.xlsx', engine='openpyxl')
11
12  # Armazena cada df em uma planilha diferente do mesmo arquivo, OMITINDO INDICES DE LINHAS DOS DADOS DA PLANILHA
13  df1.to_excel(arquivo, sheet_name='Tabela 1', index=False)
14  df2.to_excel(arquivo, sheet_name='Tabela 2', index=False)
15  df3.to_excel(arquivo, sheet_name='Tabela 3', index=False)
16
17  # Fecha o ExcelWriter e gera o arquivo .xlsx
18  arquivo.save()
```

Repare neste exemplo:

a) Linhas 1 e 2 – Importei as bibliotecas pandas e openpyxl
b) Linhas 5,6 e 7 – Criando dataframes com dados de 3 dicionários, os quais serão armazenados posteriormente um em cada planilha do arquivo excel que será gerado
c) Linha 10 – Preparando para criação do arquivo tabelas-exemplo.xlsx, usando o mecanismo openpyxl
d) Linhas 13, 14, e 15, gerando as tabelas dentro de 3 planilhas com nomes, “Tabela 1”, “Tabela 2” e “Tabela 3”, no arquivo indicado na linha 10. Cada planilha foi gerada com dados dos dicionários vinculados a “df1”, “df2” e “df3”, respectivamente.
e) Linha 18 – Esta linha grava fisicamente o arquivo na pasta do projeto em questão.

40.3 – ALTERANDO UM ARQUIVO EXCEL COM OPENPYXL

Com uso da biblioteca openpyxl poderemos alterar dados de uma planilha excel, sem perder demais conteúdos que não foram alterados tais como fórmulas, outras planilhas e gráficos.

De início, é bom saber sobre alguns métodos do openpyxl que usaremos para criar e acessar uma planilha de dados excel, vejamos:

Workbook = cria planilha excel

load_workbook = carregar planilha já existente

IMPORTANTE:

A biblioteca openpyxl trata a planilha aberta como planilha do excel, ou seja, as referências de linhas e colunas são exatamente iguais as que usamos no Excel

Vamos ao exemplo:

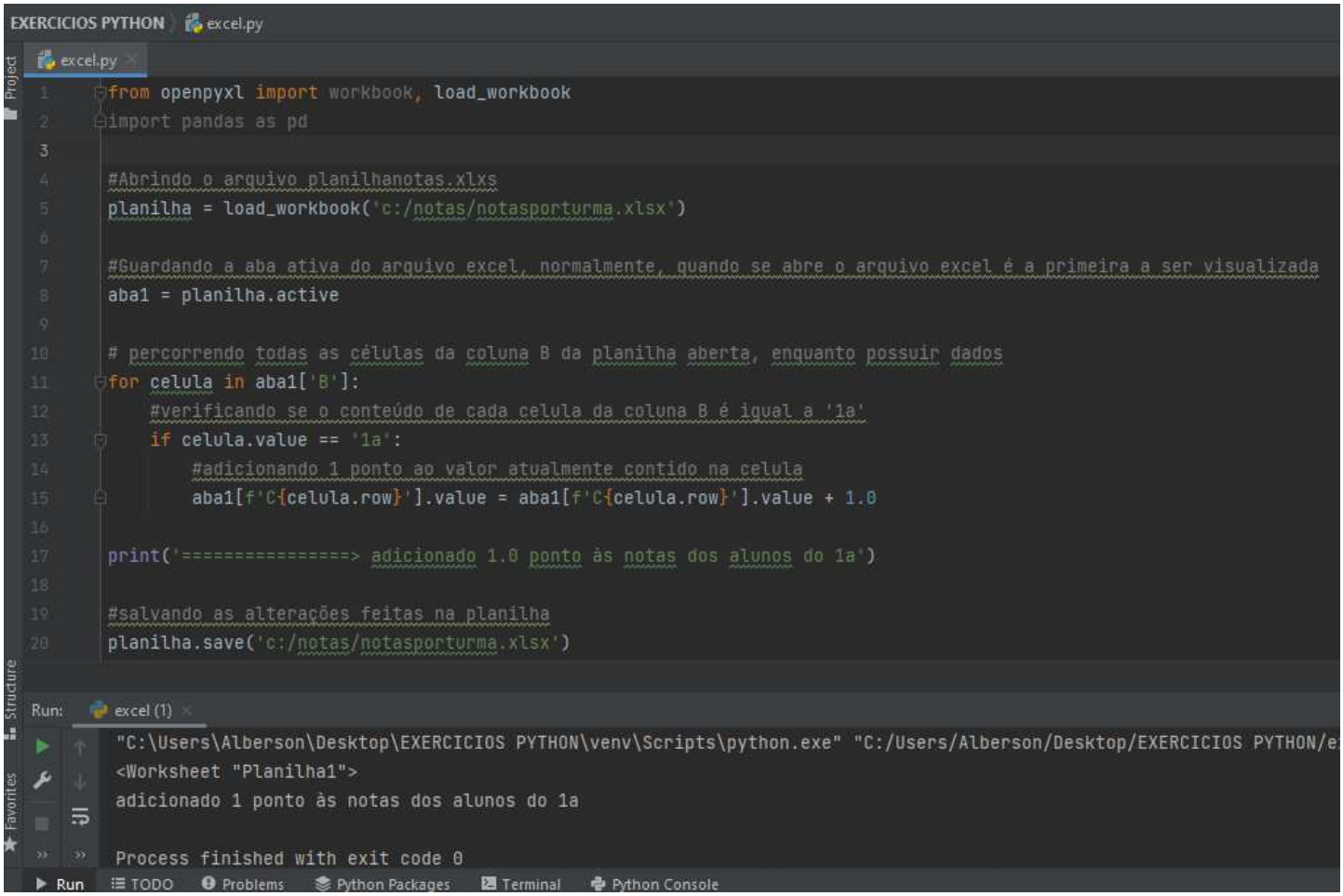
Vou considerar a seguinte planilha excel:

	A	B	C	D	E	F
1	aluno	turma	nota1	nota2	media	total pontos
2	joao	1a	5	4	4,5	9
3	lucas	1a	5	2	3,5	7
4	josé	1f	3	4	3,5	7
5	maria	1g	1	1	1	2
6	lucia	1a	7	7	7	14
7						

Observações importantes:

a) Desejo um programa para somar 1 ponto na nota1 dos alunos do 1ª.
b) Os dados coloridos em amarelo são gerados através de fórmulas do excel, capazes de calcular média e soma de pontos dos alunos.

Vamos ao programa desejado:



Analisemos o programa acima:

a) Linha 1: importei os métodos '***Workbook'*** e '***load_workbook'*** da biblioteca '***openpyxl'***
b) Linha 5: carreguei no objeto planilha uma cópia da planilha '***notasporturma.xlsx'***
c) Linha 11: percorrendo todas as células da coluna B da planilha excel aberta e atribuindo a '***celula'*** o conteúdo lido da planilha.
d) Linha 13: verificando se o conteúdo lido é igual a ***'1a'***
e) Linha 15: caso a turma lida (ou a célula lida) seja igual a ***'1a'***, adicionei 1.0 ponto na '***nota1'*** correspondente a linha atual (identificada pelo método ***row***) deste aluno da turma ***'1a'.***
f) Linha 20: salvando as alterações realizadas na planilha física '***notasporturma.xlsx'.*** **ATENÇÃO: NESTE CASO AS DEMAIS PLANILHAS, FÓRMULAS E GRÁFICOS NÃO SERÃO PERDIDOS E SERÃO ATUALIZADOS DE ACORDO COM A NECESSIDADE.**

NESTE CASO TIVEMOS O SEGUINTE RESULTADO:

	A	B	C	D	E	F
1	aluno	turma	nota1	nota2	media	total pontos
2	joao	1a	6	4	5	10
3	lucas	1a	6	2	4	8
4	josé	1f	3	4	3,5	7
5	maria	1g	1	1	1	2
6	lucia	1a	8	7	7,5	15
7						
8						
9						

OBSERVEM:

a) As linhas da coluna C nota1 dos alunos do 1a tiveram acréscimo de 1 ponto.
b) As médias e total pontos dos alunos que tiveram as notas alteradas, automaticamente foram atualizadas.
c) Caso tivéssemos gráficos vinculados a esta planilha estes seriam atualizados
d) Se existissem outras abas de planilhas neste arquivo excel, após a gravação estas NÃO SERIAM PERDIDAS.