

4º BIMESTRE – PROGRAMAÇÃO ORIENTADA A OBJETOS - PYTHON

Sumário

38 – MANIPULAÇÃO DE ARQUIVOS TEXTO	165
38.1 – CRIAR/ABRIR ARQUIVOS TEXTOS.....	165
38.2 – ESCRIVENDO CONTEÚDO NO ARQUIVO TEXTO E FECHANDO ARQUIVO TEXTO.....	167
38.3 – Lendo uma linha de um arquivo texto – readline()	171
38.4 – Lendo todo o arquivo texto – readlines()	172
38.4 – Lendo todo o arquivo texto – read().....	175
38.5 – POSICIONANDO INÍCIO DE LEITURA NO ARQUIVO TEXTO – MÉTODO seek().....	177
39 – Arquivos PDF – USO DA BIBLIOTECA reportLab.....	178
39.1 – REGISTRO DE FONTE TRUE-TYPE PARA USO NUM ARQUIVO PDF.....	184

38 – MANIPULAÇÃO DE ARQUIVOS TEXTO

Um poderoso e importante recurso utilizado em linguagens de programação 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:

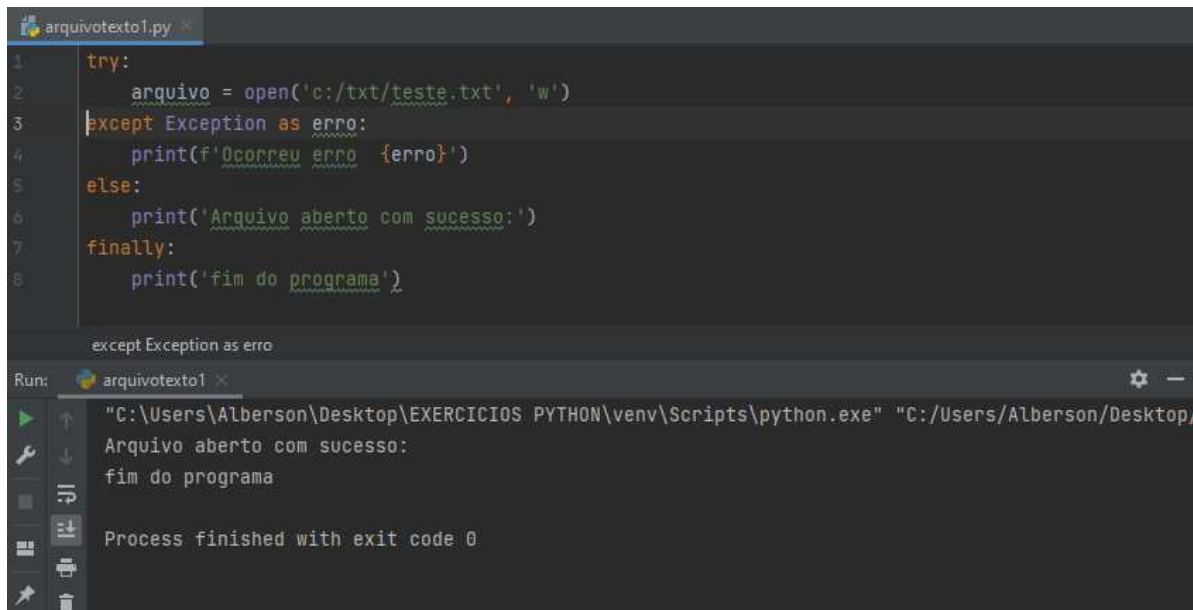
<nome_do_objetofile> = open("caminho e nome do arquivo texto", "modo_abertura")

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):



```
1 try:
2     arquivo = open('c:/txt/teste.txt', 'w')
3 except Exception as erro:
4     print(f'Ocorreu erro {erro}')
5 else:
6     print('Arquivo aberto com sucesso:')
7 finally:
8     print('fim do programa')
```

Run: "C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/Arquivo aberto com sucesso:
fim do programa
Process finished with exit code 0

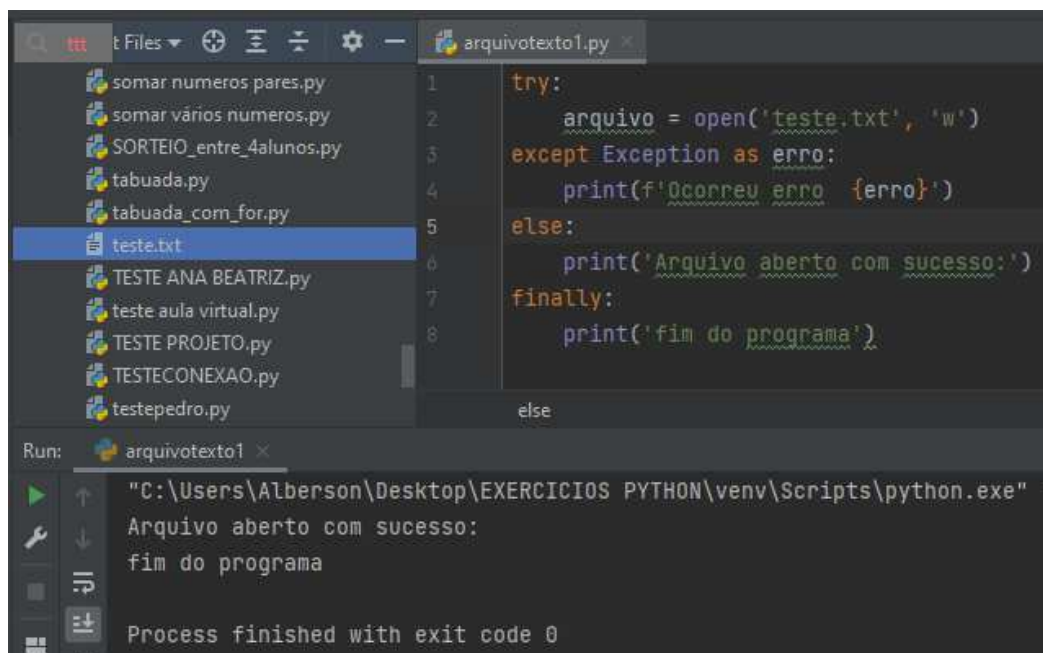
Repare acima:

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:

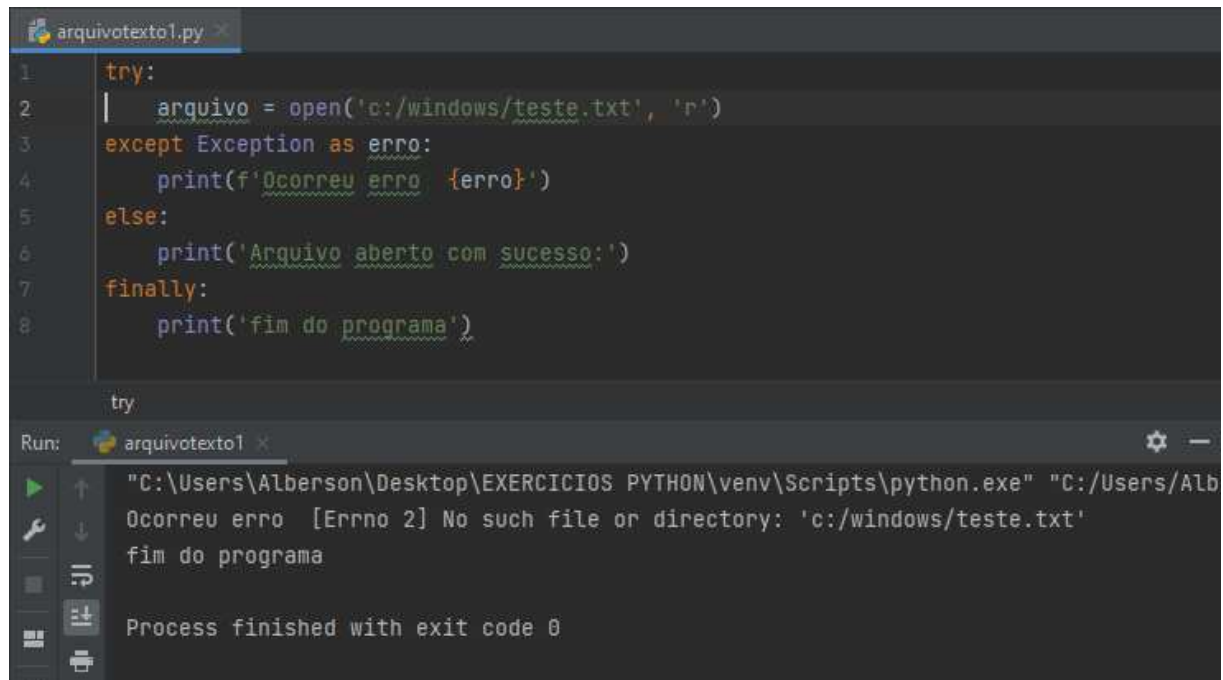


```
1 try:
2     arquivo = open('teste.txt', 'w')
3 except Exception as erro:
4     print(f'Ocorreu erro {erro}')
5 else:
6     print('Arquivo aberto com sucesso:')
7 finally:
8     print('fim do programa')
```

Run: "C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "Arquivo aberto com sucesso:
fim do programa
Process finished with exit code 0

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:



```
1 try:
2     | arquivo = open('c:/windows/teste.txt', 'r')
3 except Exception as erro:
4     print(f'Ocorreu erro {erro}')
5 else:
6     print('Arquivo aberto com sucesso:')
7 finally:
8     print('fim do programa')
```

try

Run: arquivotexto1 x

```
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/EXERCICIOS PYTHON/arquivotexto1.py"
Ocorreu erro [Errno 2] No such file or directory: 'c:/windows/teste.txt'
fim do programa

Process finished with exit code 0
```

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 – ESCRIVENDO 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:

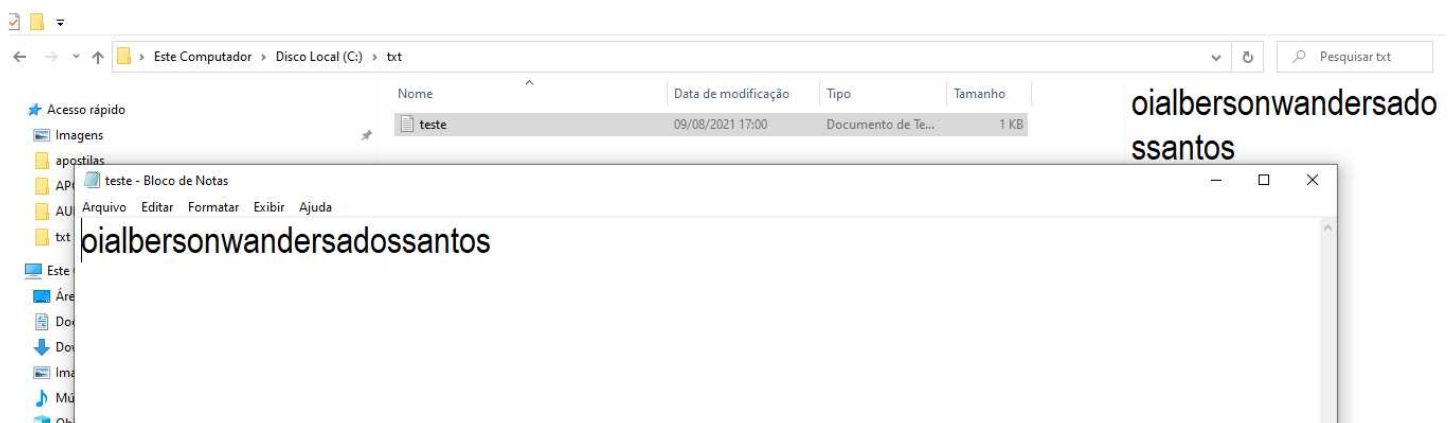
```
arquivotexto1.py
1 try:
2     arquivo = open('c:/txt/teste.txt', 'w')
3     conteudo = input('digite um conteúdo para escrever no arquivo texto: ')
4     while conteudo != '':
5         arquivo.write(conteudo)
6         conteudo = input('digite um conteúdo para escrever no arquivo texto: ')
7     arquivo.close()
8 except Exception as erro:
9     print(f'Ocorreu erro {erro}')
10 else:
11     print('Arquivo aberto com sucesso:')
12 finally:
13     print('fim do programa')

except Exception as erro

Run: arquivotexto1
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/EXERCICIOS PYTHON/arquivotexto1.py"
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
Arquivo aberto com sucesso:
fim do programa
Process finished with exit code 0
```

Repare:

- Neste arquivo o usuário digita os valores que serão inseridos um a um no arquivo que foi aberto pelo modo "w".
- 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+"

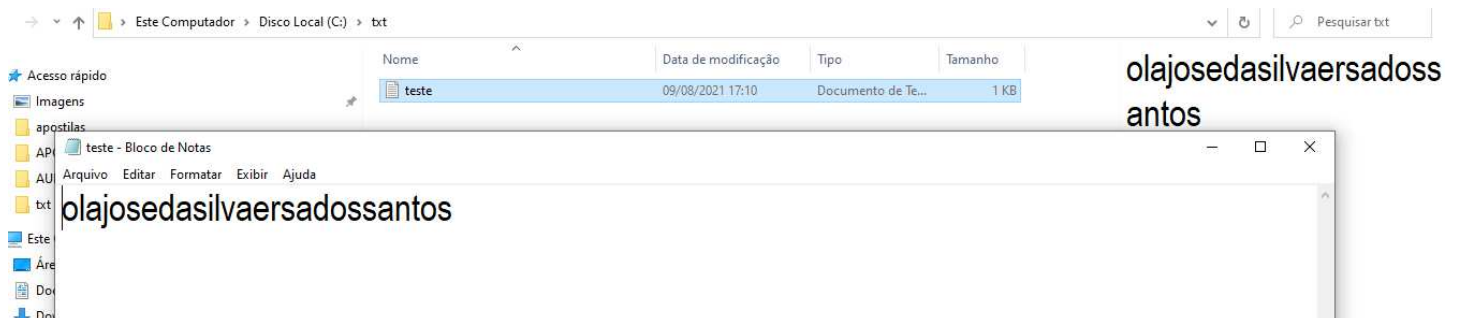
```
arquivotexto1.py
1 try:
2     arquivo = open('c:/txt/teste.txt', 'r+')
3     conteudo = input('digite um conteúdo para escrever no arquivo texto: ')
4     while conteudo != '':
5         arquivo.write(conteudo)
6         conteudo = input('digite um conteúdo para escrever no arquivo texto: ')
7     arquivo.close()
8 except Exception as erro:
9     print(f'Ocorreu erro {erro}')
10 else:
11     print('Arquivo aberto com sucesso:')
12 finally:
13     print('fim do programa')

Run: arquivotexto1
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/EXERCICIOS PYTHON/arquivotexto1.py"
digite um conteúdo para escrever no arquivo texto: ola
digite um conteúdo para escrever no arquivo texto: jose
digite um conteúdo para escrever no arquivo texto: da
digite um conteúdo para escrever no arquivo texto: silva
digite um conteúdo para escrever no arquivo texto:
Arquivo aberto com sucesso:
fim do programa

Process finished with exit code 0
```

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”:

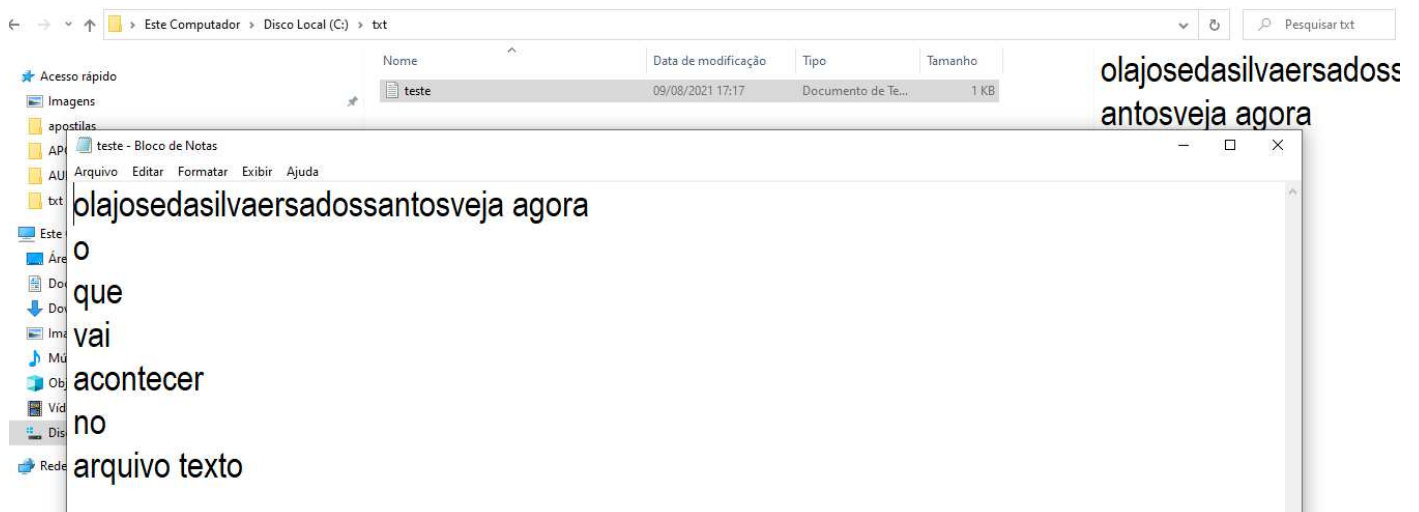
```

arquivotexto1.py
1  try:
2      arquivo = open('c:/txt/teste.txt', 'a')
3      conteudo = input('digite um conteúdo para escrever no arquivo texto: ')
4      while conteudo != '':
5          arquivo.write(f'{conteudo}\n')
6          conteudo = input('digite um conteúdo para escrever no arquivo texto: ')
7      arquivo.close()
8  except Exception as erro:
9      print(f'Ocorreu erro {erro}')
10 else:
11     print('Arquivo aberto com sucesso:')
12 finally:
13     print('fim do programa')

Run: arquivotexto1
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/EXERCICIOS PYTHON/venv/Scripts/python.exe"
digite um conteúdo para escrever no arquivo texto: veja agora
digite um conteúdo para escrever no arquivo texto: a
digite um conteúdo para escrever no arquivo texto: que
digite um conteúdo para escrever no arquivo texto: vai
digite um conteúdo para escrever no arquivo texto: acontecer
digite um conteúdo para escrever no arquivo texto: no
digite um conteúdo para escrever no arquivo texto: arquivo texto
Arquivo aberto com sucesso:
fim do programa
    
```

Veja acima:

- Abri o arquivo com modo “a” append, o que significa que o arquivo sofrerá inclusão de novas linhas
- Veja que no método **write()** foi usado o “\n” para provocar salto de linha dentro do arquivo existente.
- 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()

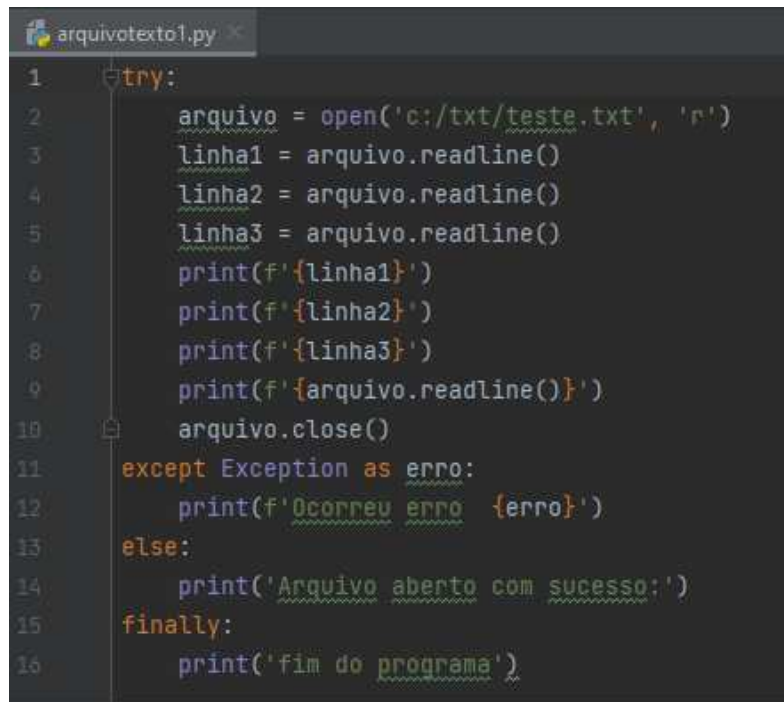
Para ler uma linha de um arquivo texto basta usar o método **readline()**, conforme sintaxe abaixo:

<variável> = <nome_do_objetofile>.readline()

Ou então:

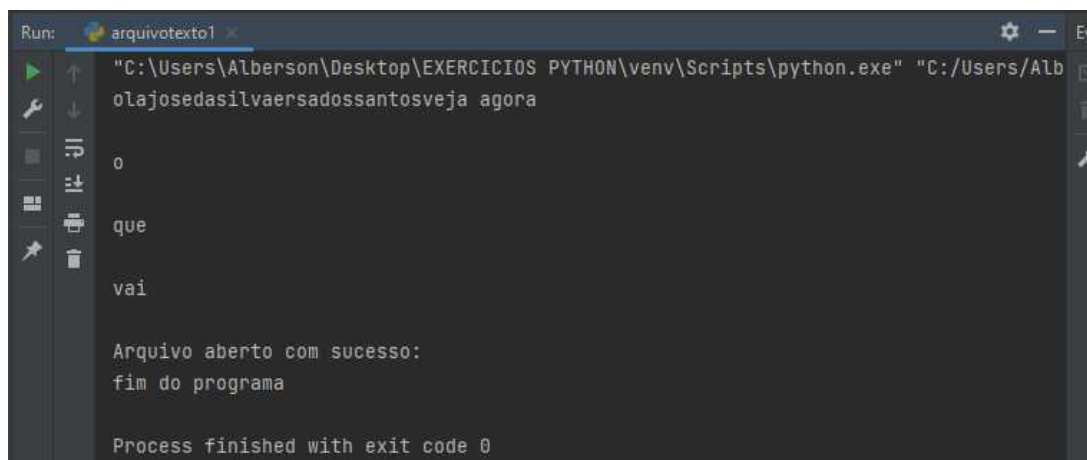
```
print(f'{<nome_do_objetofile>.readline()}')
```

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



```
1  try:
2      arquivo = open('c:/txt/teste.txt', 'r')
3      linha1 = arquivo.readline()
4      linha2 = arquivo.readline()
5      linha3 = arquivo.readline()
6      print(f'{linha1}')
7      print(f'{linha2}')
8      print(f'{linha3}')
9      print(f'{arquivo.readline()}')
10     arquivo.close()
11 except Exception as erro:
12     print(f'Ocorreu erro {erro}')
13 else:
14     print('Arquivo aberto com sucesso:')
15 finally:
16     print('fim do programa')
```

O resultado será o seguinte:



```
Run: arquivotexto1
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\env\Scripts\python.exe" "C:/Users/Alberson/Desktop/EXERCICIOS PYTHON/arquivotexto1.py"
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.

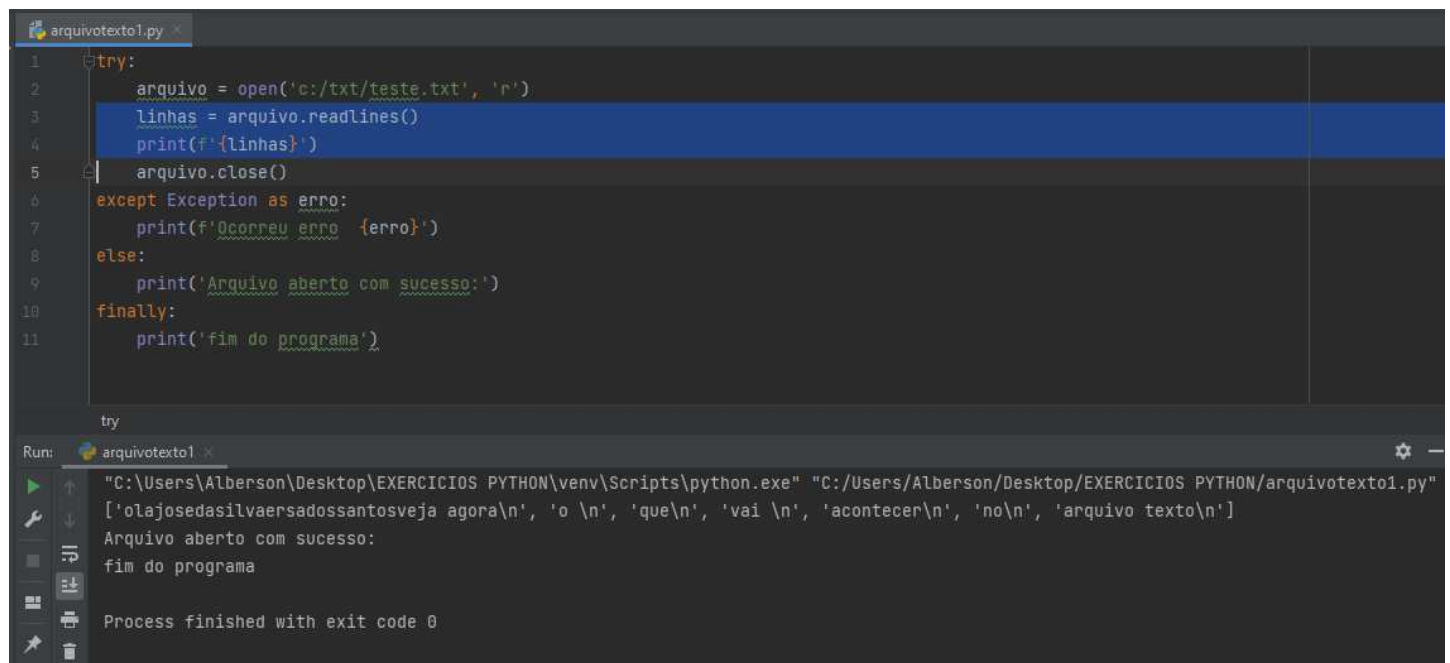
Antes porém vejamos a sintaxe mais comum do uso do método **readlines()**:

```
<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:



```
arquivotexto1.py
1 try:
2     arquivo = open('c:/txt/teste.txt', 'r')
3     linhas = arquivo.readlines()
4     print(f'{linhas}')
5     arquivo.close()
6 except Exception as erro:
7     print(f'Ocorreu erro {erro}')
8 else:
9     print('Arquivo aberto com sucesso:')
10 finally:
11     print('fim do programa')
```

Run: arquivotexto1

```
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/EXERCICIOS PYTHON/arquivotexto1.py"
['olajoseda silva ersa dos santos veja agora\n', 'o \n', 'que\n', 'vai \n', 'acontecer\n', 'no\n', 'arquivo texto\n']
Arquivo aberto com sucesso:
fim do programa
Process finished with exit code 0
```

Repare

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

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



```
EXERCICIOS PYTHON / arquivotexto1.py
1 try:
2     arquivo = open('c:/txt/teste.txt', 'r')
3     print(f'{arquivo.readlines()}')
4     arquivo.close()
5 except Exception as erro:
6     print(f'Ocorreu erro {erro}')
7 else:
8     print('Arquivo aberto com sucesso:')
9 finally:
10    print('fim do programa')
```

Run: arquivotexto1 x

"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\python.exe" "C:/Users/Alberson/Desktop/EXERCICIOS PYTHON/arquivotexto1.py"
['olajosedasilvaersadossantosveja agora\n', 'o \n', 'que\n', 'vai \n', 'acontecer\n', 'no\n', 'arquivo texto\n']
Arquivo aberto com sucesso:
fim do programa

Process finished with exit code 0

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

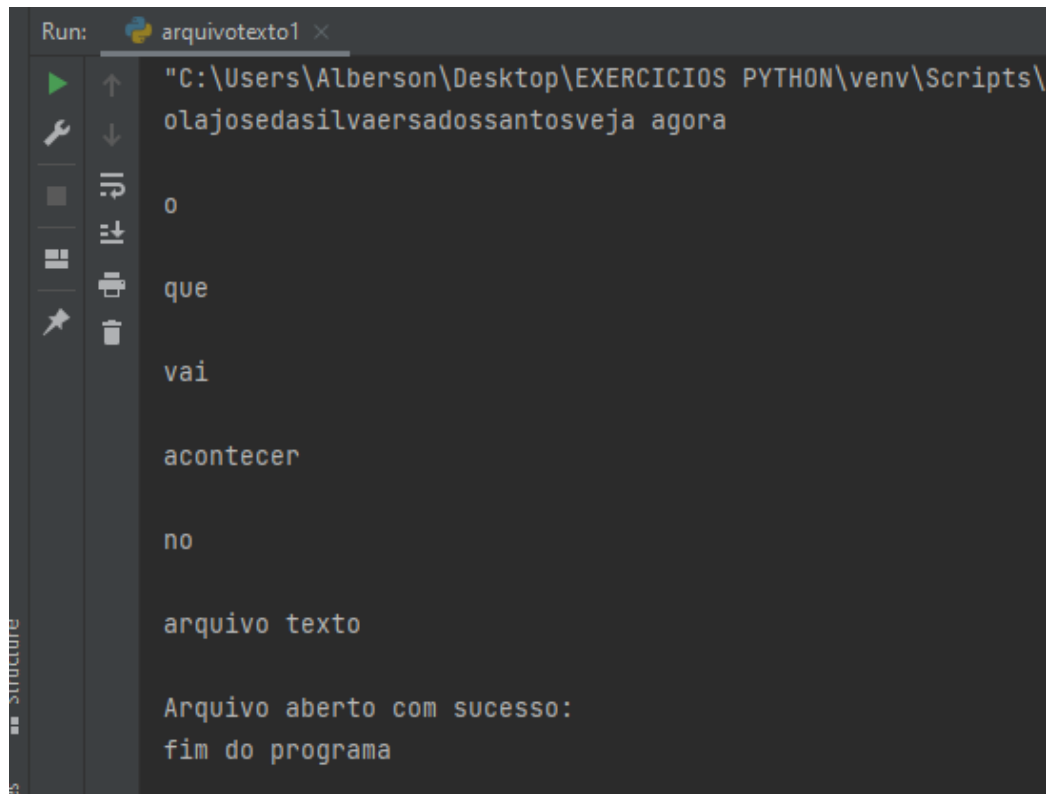
Exemplo 3: Percorrendo a lista das linhas lidas do arquivo texto e imprimindo-as na tela:



```
arquivotexto1.py x
1 try:
2     arquivo = open('c:/txt/teste.txt', 'r')
3     lista = arquivo.readlines()
4     for linha in lista:
5         print(f'{linha}')
6     arquivo.close()
7 except Exception as erro:
8     print(f'Ocorreu erro {erro}')
9 else:
10    print('Arquivo aberto com sucesso:')
11 finally:
12    print('fim do programa')
```

Repare:

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



```
Run: arquivotexto1 x
" C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\
olajosedasilvaersadossantosveja agora
o
que
vai
acontecer
no
arquivo texto
Arquivo aberto com sucesso:
fim do programa
```

38.4 – Lendo todo o arquivo texto – read()

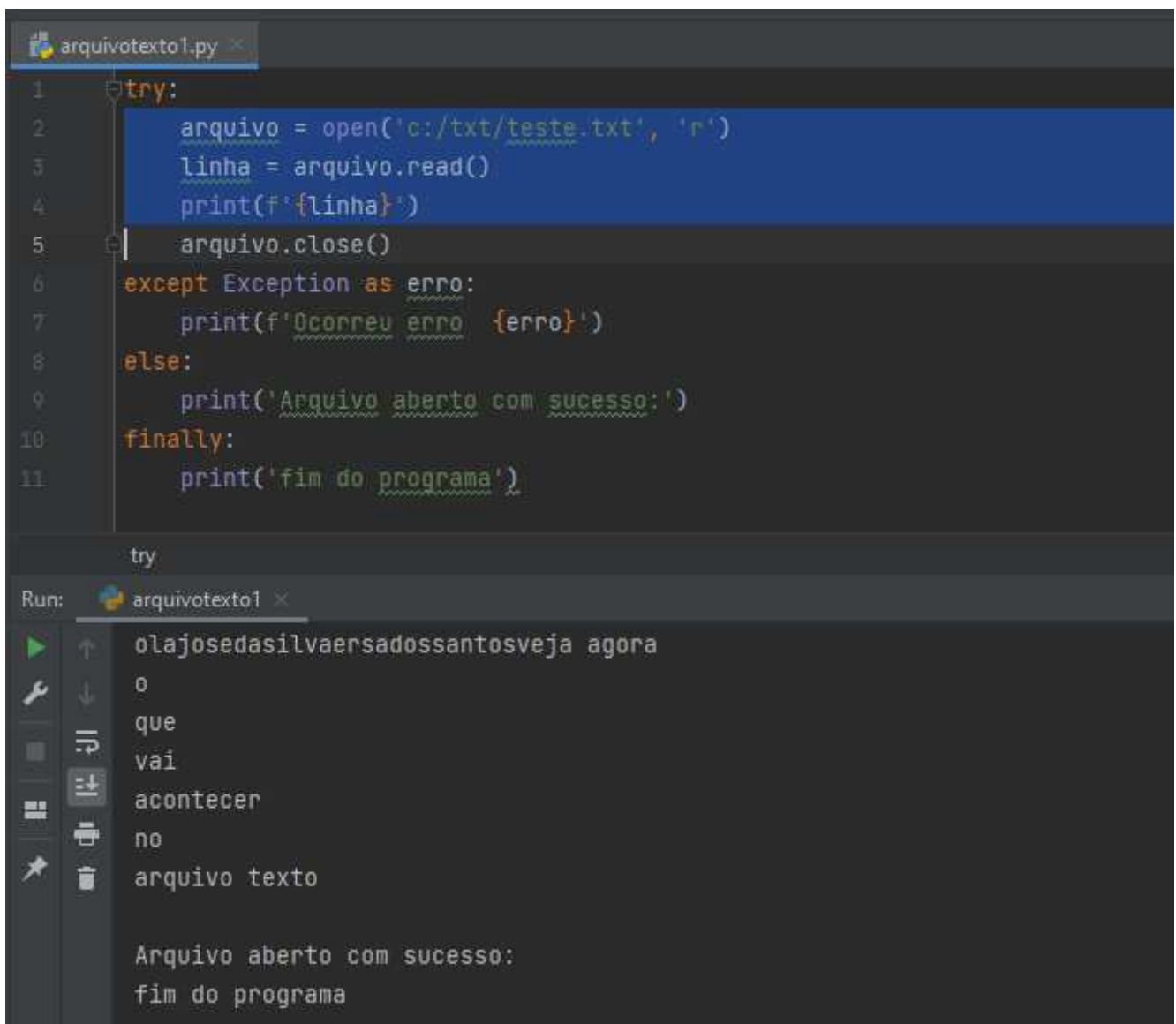
Se um arquivo for lido com o método read(), a posição retornada será sempre a do final do arquivo, pois esse método retorna o arquivo inteiro de uma vez.

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:



```
1 try:
2     arquivo = open('c:/txt/teste.txt', 'r')
3     linha = arquivo.read()
4     print(f'{linha}')
5     arquivo.close()
6 except Exception as erro:
7     print(f'Ocorreu erro {erro}')
8 else:
9     print('Arquivo aberto com sucesso:')
10 finally:
11     print('fim do programa')
```

Run: arquivotexto1

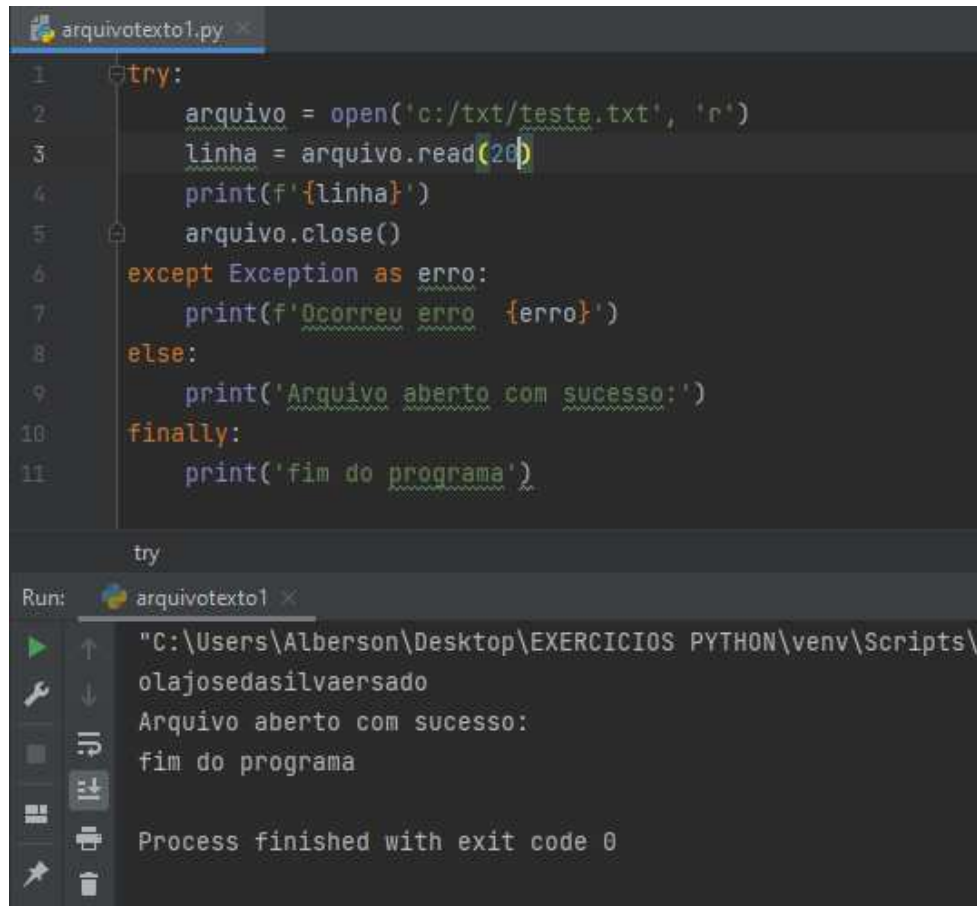
ola josedasilvaersadossantosveja agora
o
que
vai
acontecer
no
arquivo texto

Arquivo aberto com sucesso:
fim do programa

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:



```
1 try:
2     arquivo = open('c:/txt/teste.txt', 'r')
3     linha = arquivo.read(20)
4     print(f'{linha}')
5     arquivo.close()
6 except Exception as erro:
7     print(f'Ocorreu erro {erro}')
8 else:
9     print('Arquivo aberto com sucesso:')
10 finally:
11     print('fim do programa')
```

Run: arquivotexto1

```
"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\venv\Scripts\
olajosedasilvaersado
Arquivo aberto com sucesso:
fim do programa

Process finished with exit code 0
```

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.

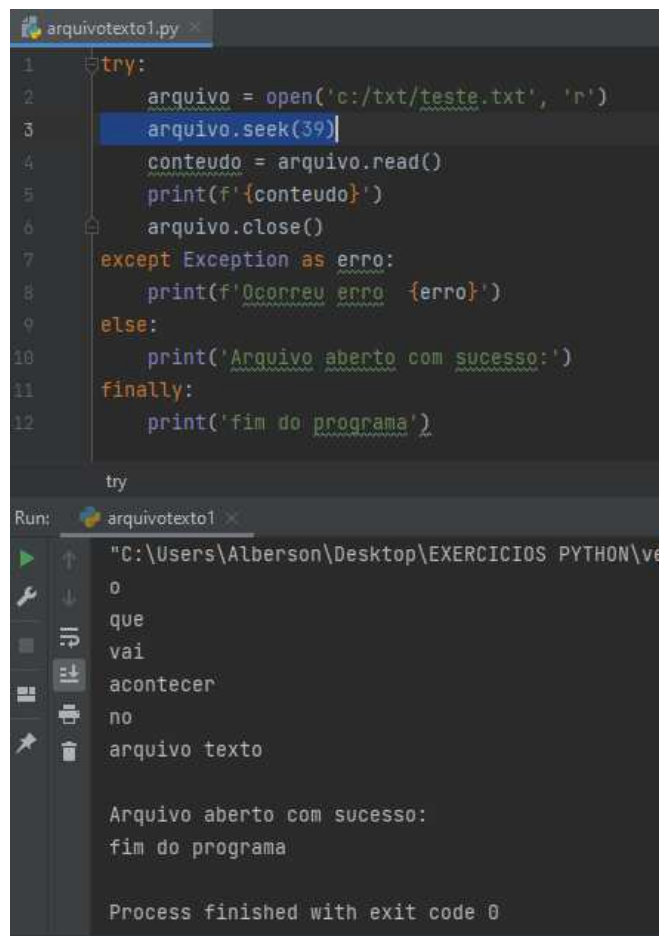
38.5 – POSICIONANDO INICIO DE LEITURA NO ARQUIVO TEXTO – MÉTODO `seek()`

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:



```
1 try:
2     arquivo = open('c:/txt/teste.txt', 'r')
3     arquivo.seek(39)
4     conteudo = arquivo.read()
5     print(f'{conteudo}')
6     arquivo.close()
7 except Exception as erro:
8     print(f'Ocorreu erro {erro}')
9 else:
10    print('Arquivo aberto com sucesso:')
11 finally:
12    print('fim do programa')
```

Run: arquivotexto1

"C:\Users\Alberson\Desktop\EXERCICIOS PYTHON\ve
o
que
vai
acontecer
no
arquivo texto

Arquivo aberto com sucesso:
fim do programa

Process finished with exit code 0

Repare:

- Veja que posicionamos o início da impressão a partir do caractere 39, usando o método **`seek()`** com parâmetro 39.
- 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)`

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:

```
C:\> Prompt de Comando  
Microsoft Windows [versão 10.0.19042.1165]  
(c) Microsoft Corporation. Todos os direitos reservados.  
C:\Users\Alberson>pip install reportlab_
```

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

```
C:\> Prompt de Comando  
Microsoft Windows [versão 10.0.19042.1165]  
(c) Microsoft Corporation. Todos os direitos reservados.  
C:\Users\Alberson>pip install reportlab  
Collecting reportlab  
  Downloading reportlab-3.6.1-cp39-cp39-win_amd64.whl (2.3 MB)  
    | 2.3 MB 2.2 MB/s  
Collecting pillow>=4.0.0  
  Downloading Pillow-8.3.1-1-cp39-cp39-win_amd64.whl (3.2 MB)  
    | 3.2 MB 3.3 MB/s  
Installing collected packages: pillow, reportlab  
Successfully installed pillow-8.3.1 reportlab-3.6.1  
WARNING: You are using pip version 21.1.3; however, version 21.2.4 is available.  
You should consider upgrading via the 'c:\users\alberson\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.  
C:\Users\Alberson>
```

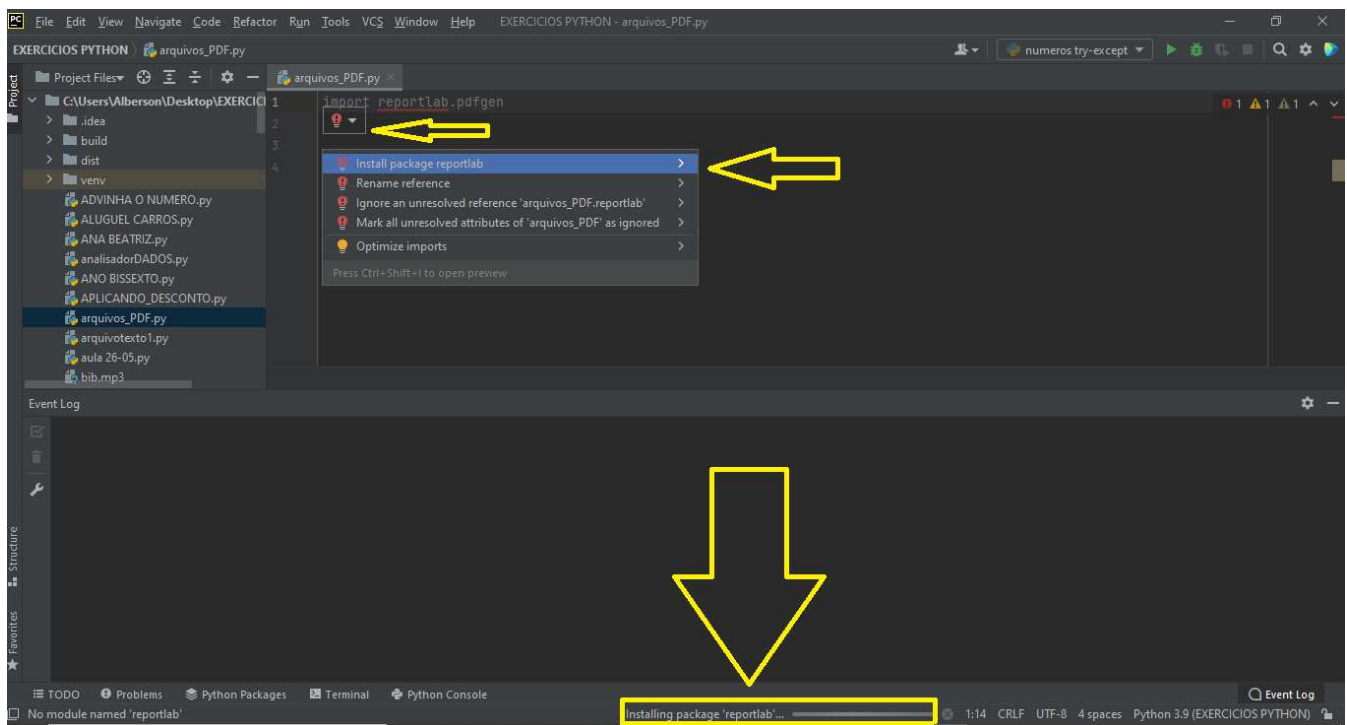
ATENÇÃO:

O INÍCIO DA INSTALAÇÃO PODE DEMORAR ALGUNS MINUTOS.

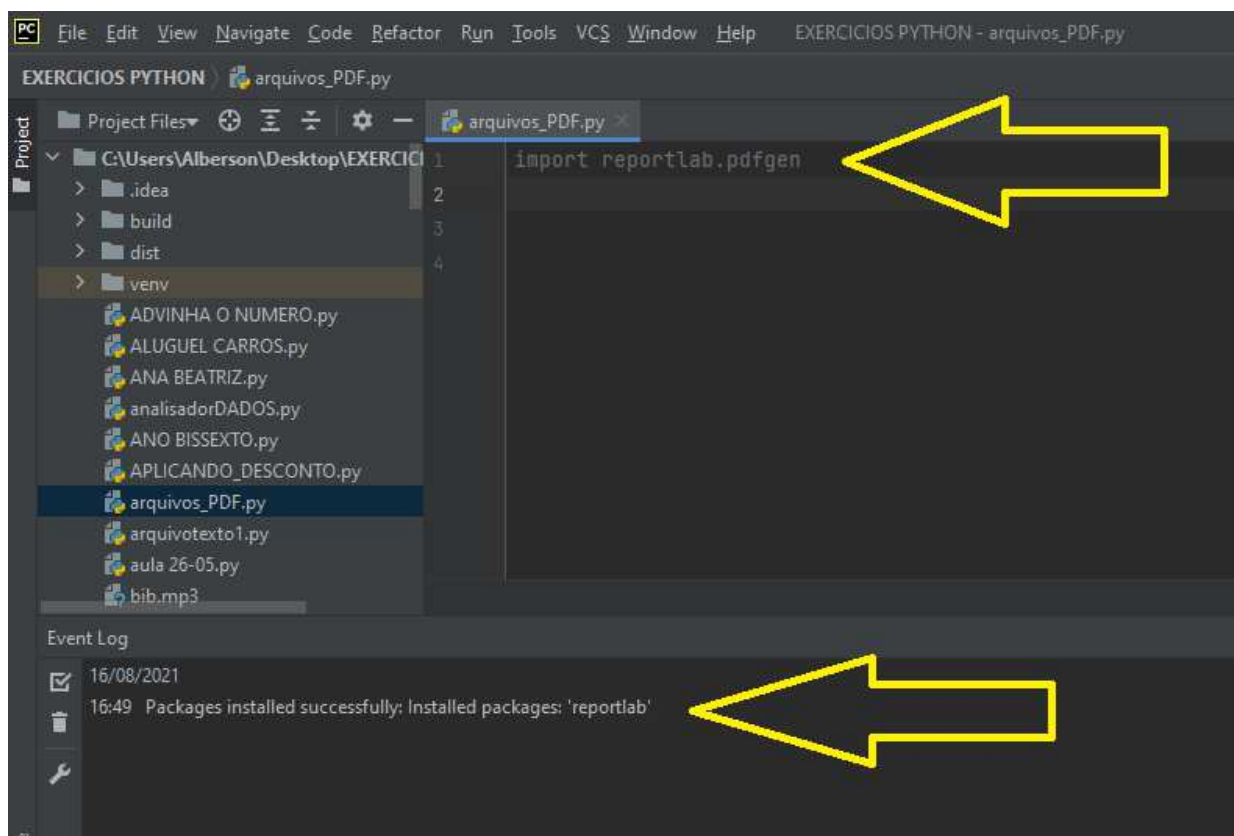
A MENSAGEM EM AMARELO INFORMA APENAS QUE O INSTALADOR pip DO MEU COMPUTADOR ESTÁ DESATUALIZADA, O QUE NÃO IMPEDE A INSTALAÇÃO DA BIBLIOTECA. VEJA QUE NA PRÓPRIA MENSAGEM JÁ ESTÁ SENDO INDICADO O CAMINHO E O COMANDO PARA ATUALIZAÇÃO DO INSTALADOR pip.

Agora basta abriremos 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 processo 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
1  #vamos usar o método canvas da biblioteca reportlab.pdfgen
2  from reportlab.pdfgen import canvas
3
4  def regua(pdf):
5      """
6      :param pdf: este parâmetro receberá o objeto pdf criado no rotina gerarpdf
7      Esta rotina só foi criada para mostrar os números de linhas e colunas da página PDF
8      """
9      #DEFININDO A COR DO TEXTO (COR DA FONTE) A SER IMPRESSO NO ARQUIVO PDF
10     pdf.setFillColor('red')
11     for coluna in range(0, 595, 5):
12         pdf.setFont("Helvetica-Oblique", 2)
13         # imprimindo os numeros das coluna na última linha do arquivo PDF, linha 0
14         pdf.drawString(coluna, 0, f'{coluna} ')
15
16     for linha in range(0, 841, 5):
17         pdf.setFont("Helvetica-Oblique", 2)
18         # imprimindo os numeros de linhas na primeira coluna do arquivo PDF, coluna 0
19         pdf.drawString(0, linha, f'{linha} ')
20
21
22 def gerarpdf(dicionario):
23     try:
24         #solicitando ao usuário o nome do arquivo PDF
25         nomearquivo = input('Informe o nome do arquivo PDF que deseja gerar: ')
26
27         #criando objeto pdf com nome do arquivo definido pelo usuário
28         pdf = canvas.Canvas(f'{nomearquivo}.pdf')
29
30         #CHAMANDO A ROTINA regua PARA IMPRIMIR UMA REGUA NA PÁGINA PARA MOSTRAR O POSICIONAMENTO DO TEXTO
31         #NO ARQUIVO PDF. ESTA ROTINA NÃO PRECISA SER CHAMADA SEMPRE E NEM PRECISA SER CRIADA NOS SEUS PROGRAMAS
32         regua(pdf)
33
34         #DEFININDO A COR DO TEXTO DOS PRÓXIMOS TEXTOS A SEREM IMPRESSOS
35         pdf.setFillColor('black')
36
37         #Definindo o texto do título do documento a ser impresso
38         pdf.setTitle('Relatório de Professores do Curso Técnico')
39
40         # definindo nome da fonte e tamanho da mesma para o próximo texto a ser escrito no PDF
41         pdf.setFont("Helvetica-Oblique", 16)
42
43         # ATENÇÃO: AS REFERENCIAS DE LINHAS E COLUNAS NO RELATÓRIO SÃO TROCADAS NO MÉTODO drawString
44         # Assim estou ESCRREVENDO UM TEXTO NA PÁGINA PDF, NA LINHA 750 A PARTIR DA COLUNA 10
```

```
45 pdf.drawString(10, 750, 'Relação de Professores do 2º ano:')
46
47 # definindo nome da fonte e tamanho da mesma para o próximo texto a ser escrito,
48 pdf.setFont("Helvetica-Oblique", 14)
49
50 # ATENÇÃO: AS REFERENCIAS DE LINHAS E COLUNAS NO RELATÓRIO SÃO TROCADAS NO MÉTODO drawString
51 # Assim estou ESCRREVENDO UM TEXTO NA PÁGINA PDF, NA LINHA 750 A PARTIR DA COLUNA 10
52 pdf.drawString(10, 720, 'Nome professor | Disciplina ')
53
54 x=700 #MEDIDA EM MILIMETROS
55 for nome, disciplina in dicionario.items():
56     print(f'{nome} | {disciplina}')
57     x-=20 # -20mm
58     # o método drawString é usado para escrever na página PDF (FOLHA TOTAL POSSUI
59     pdf.drawString(10, x, f'{nome} | {disciplina}')
60
61 #criando arquivo PDF
62 pdf.save()
63 print(f'{nomearquivo} foi gerado com sucesso: ')
64 except Exception as erro:
65     print(f'Erro ao gerar o arquivo pdf: {erro}')
66
67 dicionario = {'Alberson': 'PooI', 'Bruno': 'Banco de Dados', 'Helio': 'PAWeb', 'Wagner': 'PVBásica'}
68 gerarpdf(dicionario)
```

SOBRE O CÓDIGO DESTE PROGRAMA:

- Linha 68 - a rotina gerarpdf() foi chamada e será passado como parâmetro um dicionário de nomes e disciplinas de professores.
- Linha 25 – O usuário está definindo um nome de arquivo PDF que será gerado. **Não é necessário indicar a extensão .pdf**.
- 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.
- Linha 32 – Estou chamando a rotina para imprimir a régua na página. Esta rotina pode ser retirada dos seus projetos futuros.
- Linha 35 – Estou definindo a cor do texto “preto” a ser impresso, usando o método setFillColor('black') do objeto canva chamado **pdf**
- 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”
- 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:

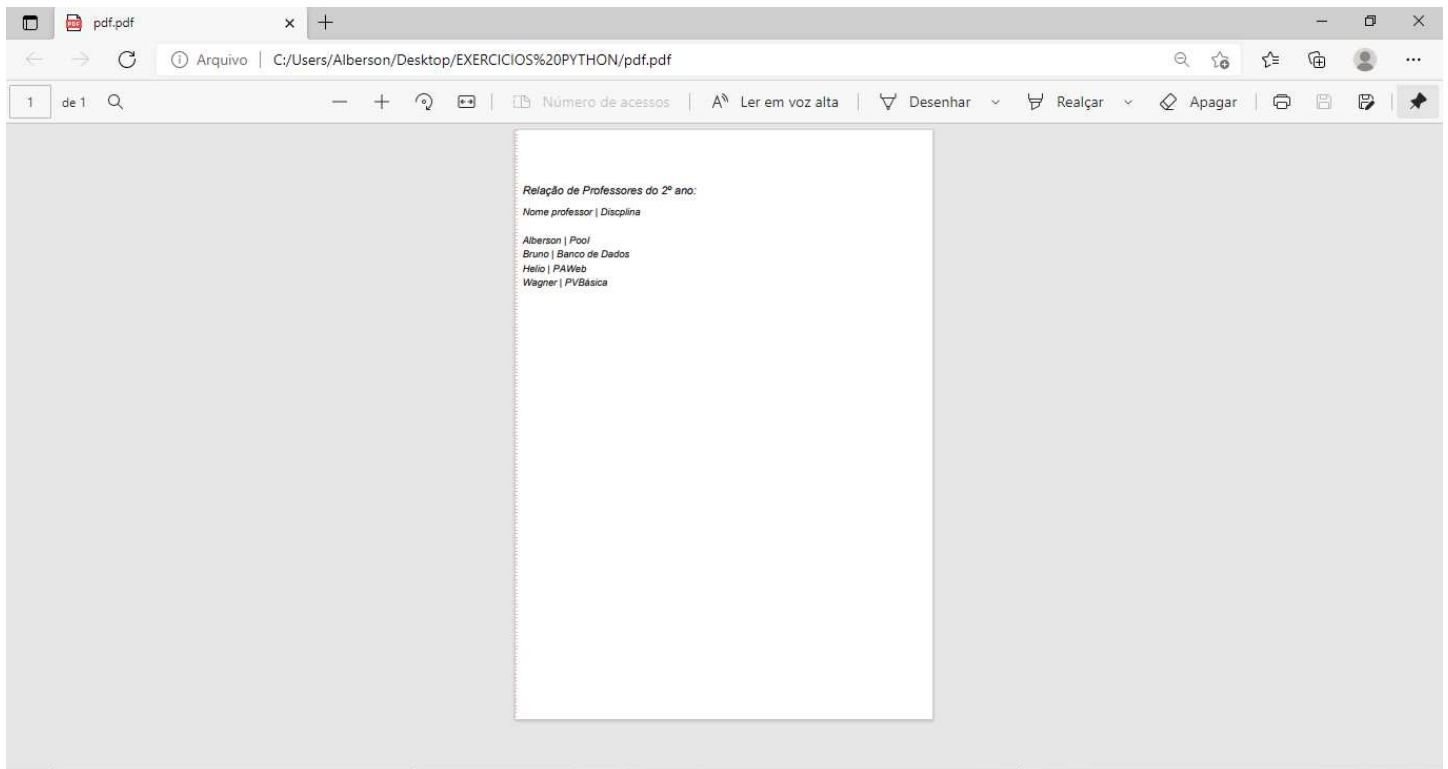
```
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont
```

Escreva a linha abaixo, por exemplo, no módulo principal do programa. Neste exemplo estou registrando 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).

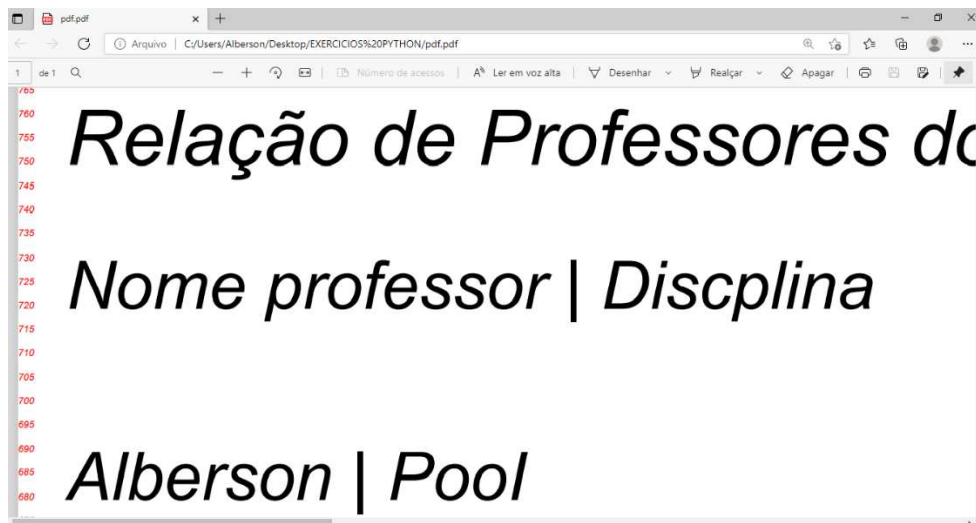
Então, basicamente a posição $y = 247$ e $x = 700$ para centralizar na tela.

REPRE QUE:

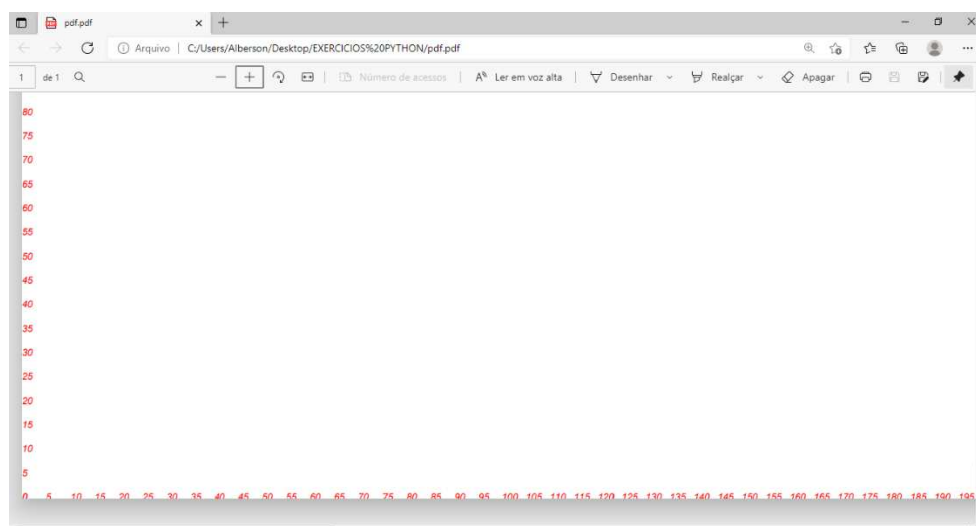
y CORRESPONDE A POSIÇÃO DE COLUNA ONDE O TEXTO SERÁ IMPRESSO

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, REPRE 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
<code>setFillColor('<nomecor')'</code>	Define a cor da fonte do texto que será impresso na próxima impressão do método <code>drawString()</code>
<code>setFont("nomefonte", tamanhofonte)</code>	Define o nome da fonte e tamanho, para próximo texto impresso pelo método <code>drawString()</code>
<code>drawString(coluna, linha, "texto")</code>	Imprime um texto numa coluna e linha do arquivo PDF.
<code>Canvas("caminho\nomearquivo.pdf")</code>	Cria o objeto canva que representará o no caminho e nome definidos para o arquivo PDF
<code>setTitle("titulo da Pagina")</code>	Definindo titulo da página PDF que será criada pelo método <code>save()</code>
<code>save()</code>	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.ttf fonts 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 ESCRIVENDO 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 ESCRIVENDO 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}')

#####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”.

