

# LABORATÓRIOS DE PROGRAMAÇÃO

Arquivos

Aula 6

Prof<sup>a</sup>. Kamilla Dória

# TÓPICOS

- Criação de Arquivos
- Escrita em Arquivos
- Leitura em Arquivos



# CRIAÇÃO DE ARQUIVOS

# INTRODUÇÃO

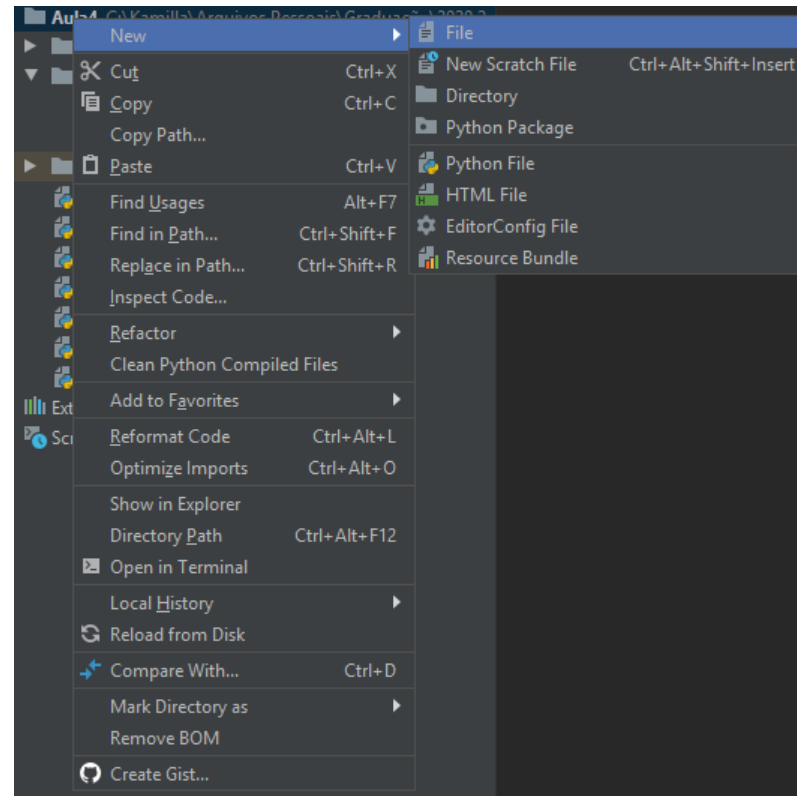
- Computadores normalmente trabalham com duas memórias:
  - Primária: usualmente com base eletrônica e que normalmente se apaga quando o computador desliga;
  - Secundária: usualmente organizada em arquivos gravados em dispositivos de armazenamento persistente, e que não se apaga quando o computador desliga.

# INTRODUÇÃO

- Arquivos normalmente se classificam em dois tipos:
  - Arquivos executáveis: são os programas ou aplicativos;
  - Arquivos de dados: repositório de informações que eventualmente são usados por programas ou aplicativos.
    - Ex: arquivos txt.

# CRIAÇÃO DE ARQUIVO

- Para criar um novo arquivo de forma manual selecione a opção de arquivos no menu de criação:



## criação de arquivo

- Para criar um novo arquivo via código, utilize o comando:

```
arquivo = open ('arquivo.txt', 'w')
```

```
arquivo = open ('arquivo.txt', 'w')
```

- A variável arquivo tem a referência para o arquivo no projeto e será utilizada para escrever e ler dados do arquivo;
- O primeiro argumento indica o nome do arquivo que será criado;
  - Caso o arquivo não exista, será criado um novo arquivo em branco;
  - Caso já exista um arquivo de mesmo nome, o arquivo será sobrescrito por um arquivo em branco;
- O segundo argumento indica que um novo arquivo em branco deverá ser criado.



# EXERCÍCIO 1

Crie um projeto com um programa principal que chama uma função para criar arquivos.

O programa principal deve criar arquivos de forma dinâmica a medida que o usuário deseja.

arquivo1

arquivo2

.

.

.



# ESCRITA EM ARQUIVOS

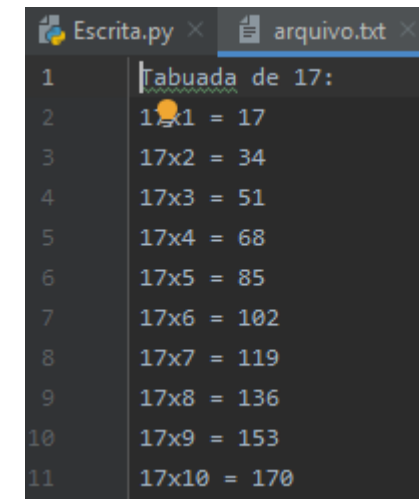
- Os arquivos podem ser usados como uma espécie de banco de dados para um projeto, armazenando os dados que antes ficavam na memória (variáveis);
- Antes de iniciar a escrita em um arquivo é necessário abri-lo;
  - Atenção: ao utilizar o comando open com o parâmetro 'w', o programa irá sobrescrever o arquivo por outro em branco.

- Para escrever em um arquivo, utiliza-se o comando *write* associada à variável que contém a referência do arquivo;
  - O *write* não é tão flexível quanto o *print*; aceita apenas um parâmetro e que seja do tipo string;
    - Deve-se converter o conteúdo para string: *str(conteúdo)* / *f'{conteúdo}'*
  - O *write* também não faz quebra de linha automática como o *print*;
    - Deve-se acrescentar o comando '\n' ao final de cada linha.
- O comando *close()* garante que um arquivo foi gravado e fechado.

# ESCRITA EM ARQUIVO

- Exemplo: escrita de uma tabuada em um arquivo.

```
TABUADA = 17
arq_tab = open('arquivo.txt', 'w')
arq_tab.write(f'Tabuada de {TABUADA}:\n')
for i in range(1, 11):
    arq_tab.write(f'{TABUADA}x{i} = {TABUADA*i}\n')
arq_tab.close()
```



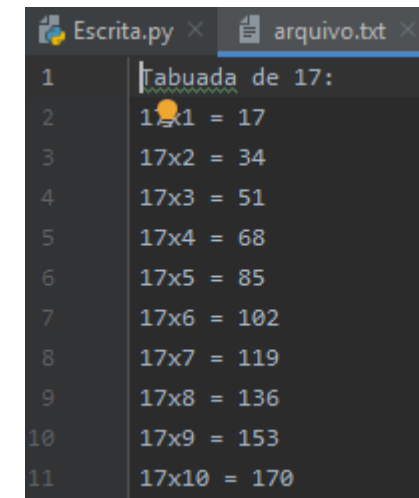
```
Escrita.py x arquivo.txt x
1 Tabuada de 17:
2 17x1 = 17
3 17x2 = 34
4 17x3 = 51
5 17x4 = 68
6 17x5 = 85
7 17x6 = 102
8 17x7 = 119
9 17x8 = 136
10 17x9 = 153
11 17x10 = 170
```

- O comando *with – as* pode ser utilizado para substituir a necessidade de usar o *close* para fechar o arquivo;
- Somente os comandos subordinados ao *with – as* poderão operar sobre o arquivo;
- Após o encerramento do bloco *with – as* o arquivo é encerrado automaticamente.

# ESCRITA EM ARQUIVO

- Exemplo: escrita de uma tabuada em um arquivo.

```
TABUADA = 17
with open('arquivo.txt', 'w') as arq_tab:
    arq_tab.write(f'Tabuada de {TABUADA}:\n')
    for i in range(1, 11):
        arq_tab.write(f'{TABUADA}x{i} = {TABUADA*i}\n')
```

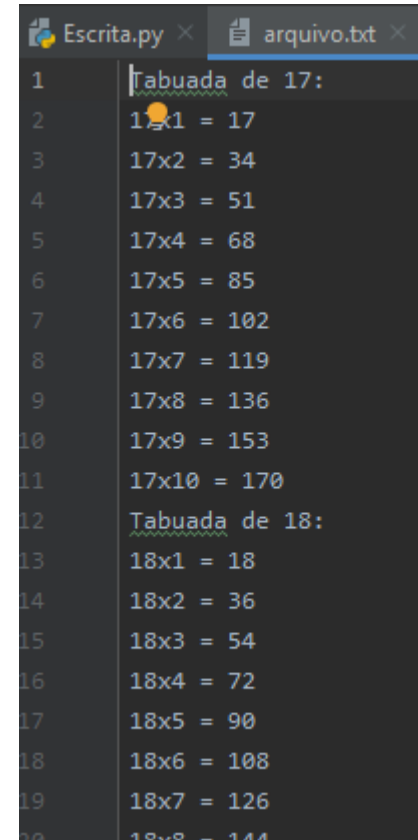


```
Escrita.py x arquivo.txt x
1 Tabuada de 17:
2 17x1 = 17
3 17x2 = 34
4 17x3 = 51
5 17x4 = 68
6 17x5 = 85
7 17x6 = 102
8 17x7 = 119
9 17x8 = 136
10 17x9 = 153
11 17x10 = 170
```

# ESCRITA EM ARQUIVO

- Para adicionar novos conteúdos a um arquivo já populado, utiliza-se o parâmetro 'a' no lugar do 'w'.

```
TABUADA = 18
with open('arquivo.txt', 'a') as arq_tab:
    arq_tab.write(f'Tabuada de {TABUADA}:\n')
    for i in range(1, 11):
        arq_tab.write(f'{TABUADA}x{i} = {TABUADA*i}\n')
```



```
Escrita.py x arquivo.txt x
1 Tabuada de 17:
2 17x1 = 17
3 17x2 = 34
4 17x3 = 51
5 17x4 = 68
6 17x5 = 85
7 17x6 = 102
8 17x7 = 119
9 17x8 = 136
10 17x9 = 153
11 17x10 = 170
12 Tabuada de 18:
13 18x1 = 18
14 18x2 = 36
15 18x3 = 54
16 18x4 = 72
17 18x5 = 90
18 18x6 = 108
19 18x7 = 126
20 18x8 = 144
```



## EXERCÍCIO 2

Crie um programa para chamar uma função que realizará o input de dados de nome do usuário e quanto de espaço em disco o mesmo ocupa em bytes.

Na sequência crie uma outra função para gravar o seguinte relatório em um arquivo chamado relatório.txt (a partir dos dados inseridos pelo usuário):

ACME Inc.		Uso do espaço em disco pelos usuários	
Nr.	Usuário	Espaço utilizado	% do uso
1	alexandre	434,99 MB	16,85%
2	anderson	1187,99 MB	46,02%
3	antonio	117,73 MB	4,56%
4	carlos	87,03 MB	3,37%
5	cesar	0,94 MB	0,04%
6	rosemary	752,88 MB	29,16%
Espaço total ocupado: 2581,57 MB			
Espaço médio ocupado: 430,26 MB			

Também crie funções separadas para a conversão de bytes para MB e para o cálculo do percentual de uso.



# LEITURA EM ARQUIVOS

# LEITURA EM ARQUIVOS

- O conteúdo de um arquivo pode ser lido todo de uma vez e atribuído a uma variável no formato de string;
- É necessário abrir o arquivo utilizando o parâmetro 'r', e na sequência utilizar o comando *read()* associado à referência do arquivo.

```
with open('arquivo.txt', 'r') as arquivo:  
    x = arquivo.read()  
print(x)
```

```
"Leitura.py"  
Tabuada de 17:  
17x1 = 17  
17x2 = 34  
17x3 = 51  
17x4 = 68  
17x5 = 85  
17x6 = 102  
17x7 = 119  
17x8 = 136  
17x9 = 153  
17x10 = 170  
Tabuada de 18:  
18x1 = 18  
18x2 = 36  
18x3 = 54  
18x4 = 72  
18x5 = 90  
18x6 = 108  
18x7 = 126  
18x8 = 144  
18x9 = 162  
18x10 = 180
```

Process finished with exit code 0

# LEITURA EM ARQUIVOS

- Arquivos também podem ser lidos linha a linha utilizando o comando *readline()* associado à referência do arquivo;
- A cada vez que o comando é executado em um programa, ele passa um "cursor" de leitura para a linha seguinte.

```
with open('arquivo.txt', 'r') as arquivo:  
    x = arquivo.readline()  
    print(x)  
    x = arquivo.readline()  
    print(x)
```

"Leitura.py"  
Tabuada de 17:

17x1 = 17

Process finished with exit code 0

- Para ler todas as linhas de um arquivo pode-se utilizar um laço *while* percorrendo todo o cursor até que o resultado do comando *readline()* seja uma string vazia.

```
with open('arquivo.txt', 'r') as arquivo:
    i = 1
    while True:
        x = arquivo.readline()
        if x == '':
            break
        print(f'Linha {i}: {x}')
        i += 1
```

- O mesmo pode ser feito utilizando um laço *for*;
- Essa instrução atribui à variável de interação o conteúdo do retorno do comando *readline()*; a cada interação o cursor passa para a próxima linha do arquivo.

```
with open('arquivo.txt', 'r') as arquivo:  
    i = 1  
    for x in arquivo:  
        print(f'Linha {i}: {x}')  
        i += 1
```

- Dica: para remover a quebra linha de final da linha do arquivo pode-se utilizar a expressão `x[:-1]`, a qual irá retornar a linha menos o último caractere (a quebra de linha).

```
with open('arquivo.txt', 'r') as arquivo:
    i = 1
    for x in arquivo:
        linha = x[:-1]
        print(f'Linha {i}: {linha}')
        i += 1
```

## EXERCÍCIO 3

Utilizando o arquivo criado no exercício anterior, crie um programa para realizar a leitura desse arquivo, e para cada linha do arquivo, verificar se o usuário gostaria de fazer uma limpeza de disco ou não. Ao final, grave um novo arquivo com essa informação adicional.





OBRIGADA!