Exercícios:

1. Classe Bola: Crie uma classe que modele uma bola:
    a. Atributos: Cor, circunferência, material
    b. Métodos: trocaCor e mostraCor

2. Classe Quadrado: Crie uma classe que modele um quadrado:
    a. Atributos: Tamanho do lado
    b. Métodos: Mudar valor do Lado, retornar valor do Lado e calcular Área;

3. Classe Retangulo: Crie uma classe que modele um retângulo:
    a. Atributos: LadoA, LadoB (ou Comprimento e Largura, ou Base e Altura, a escolher)
    b. Métodos: Mudar valor dos lados, retornar valor dos lados, calcular Área e calcular Perímetro;
    c. Crie um programa que utilize esta classe. Ele deve pedir ao usuário que informe as medidas de um local. Depois, deve criar um objeto com as medidas e calcular a quantidade de pisos e de rodapés necessárias para o local.

4. Classe Pessoa: Crie uma classe que modele uma pessoa:
    a. Atributos: nome, idade, peso e altura
    b. Métodos: envelhecer, engordar, emagrecer, crescer. Obs: Por padrão, a cada ano que nossa pessoa envelhece, sendo a idade dela menor que 21 anos, ela deve crescer 0,5 cm.

Manipulando arquivos de texto em Python

É de grande importância para qualquer desenvolvedor saber manipular arquivos, seja para criar backups, consumir uma lista de alguma planilha ou qualquer motivo que seja. Por isso, a maioria das linguagens de programação possuem meios para essa manipulação.

No Python não é diferente e a manipulação de arquivos é mais simples do que você possa imaginar. Por isso, veremos neste artigo como realizar as principais operações com arquivos utilizando o Python.

Criando e abrindo arquivos

Para criar arquivos (e, consequentemente, abri-los), utilizamos o método open(). Este método irá abrir o arquivo que passarmos como parâmetro com um determinado modo de uso (que também será passado como parâmetro).

Há diversos modos de uso, como podemos ver na imagem abaixo:

- "a" - Escreve ao final do arquivo; se este não existir, é criado
- "r" - Abre o arquivo para a leitura, se não existir, lançar um erro de IOError
- "r+" - Abra um arquivo para leitura e escrita. Se não existe, lança um erro IOError
- "w" - Abre um arquivo para escrita. Se existe, ele 'trunca' tudo e escreve por cima. Se não existir o arquivo, ele cria
- "w+" - Abre para leitura e escrita. Se existe, apaga todo conteúdo e escreve por cima. Se não existir o arquivo, ele cria
- "ab", "rb", "r+b", "wb", "w+b" - Abre arquivos para trabalhar com entrada e saída no modo binário, para plataformas Windows e Macintosh

Estes modos são passados como segundo parâmetro do método open. Ou seja, se quisermos abrir um arquivo em modo de escrita, utilizamos a seguinte sintaxe:

```
arquivo = open("aula12.txt", "a")
```

Passamos o nome do arquivo e sua extensão, além do modo que queremos utilizar o arquivo. Este modo pode ser alterado conforme as opções anteriores.

Sendo assim, ao executar o código acima, o arquivo contatos.txt será aberto em modo escrita (caso ele não exista, um novo arquivo será criado).

Escrevendo dados em arquivos

Agora que já sabemos como criar arquivos e seus diferentes modos de uso, podemos realizar as primeiras manipulações. Neste tópico, veremos como escrever dados e salvar em arquivo utilizando o Python.

Para isso, a linguagem fornece dois métodos. O primeiro é o método write() que recebe uma string como parâmetro e a insere no arquivo:

```
arquivo = open("aula12.txt", "a")
arquivo.write("Olá, mundo!")
```

Com a execução do código acima, a string Olá, mundo! será inserida no arquivo aula12.txt.

O segundo método é o writelines() que recebe um objeto iterável (seja uma lista, uma tupla, um dicionário, etc). Com este método, várias linhas poderão ser inseridas no arquivo, diferente do método write() que só recebe uma string por vez:

```
arquivo = open("aula12.txt", "a")

frases = list()
frases.append("Aula12 \n")
frases.append("Python \n")
frases.append("Arquivos \n")
frases.append("Django \n")

arquivo.writelines(frases)
```

Como a lista é um objeto iterável, podemos passá-la como parâmetro do método writelines(). Utilizamos, também, o \n para saltar a linha ao escrevê-la no arquivo.

Lendo dados de arquivos

Além de escrever dados em arquivos, precisamos, também, saber ler os dados que os arquivos possuem. Para isso, existem dois métodos, o primeiro é o readline() que irá ler uma quantidade N de caracteres da primeira linha passadas como parâmetro:

```
arquivo = open("aula12.txt", "r")

print(arquivo.readline(3))
```

A execução acima retornará os três primeiros caracteres da primeira linha do arquivo.

O segundo método disponível é o readlines(). Este método irá retornar todas as linhas do arquivo:

```
arquivo = open("aula12.txt", "r")

print(arquivo.readlines())
```

Tanto na leitura quanto na escrita deve-se fechar o arquivo com o comando:

```
arquivo.close()
```

Aprofundar no estudo de abertura de arquivos:

https://docs.python.org/pt-br/3.8/library/csv.html?highlight=open

Exercícios

Aqui está um arquivo notas_estudantes.txt que contém uma linha para cada aluno de uma turma de estudantes. O nome de cada estudante está no início de cada linha e é seguido pelas suas notas.

notas_estudantes.txt

```
jose 10 15 20 30 40
pedro 23 16 19 22
suzana 8 22 17 14 32 17 24 21 2 9 11 17
gisela 12 28 21 45 26 10
joao 14 32 25 16 89
```

1. Usando o arquivo texto notas_estudantes.txt escreva um programa que imprime o nome dos alunos que têm mais de seis notas.
2. Usando o arquivo texto notas_estudantes.txt escreva um programa que calcula a média das notas de cada estudante e imprime o nome e a média de cada estudante.
3. Usando o arquivo texto notas_estudantes.txt escreva um programa que calcula a nota mínima e máxima de cada estudante e imprima o nome de cada aluno junto com a sua nota máxima e mínima.

Entrega até 11/10/2020 às 23h59

Formato: seu_nome.py ou seu_nome.ipynb

Valor 1 ponto.