Curso de Python Básico 2

revisado 20/6/19

PROGRAMANDO EM PYTHON

TUPLAS

- **Tuplas:** são similares às listas, mas imutáveis. Não podemos reduzir, adicionar ou modificar nenhum de seus ítens.

- **Sintaxe:**

```
<variável> = (info1, info2, info3)
```

ou

```
<variável> = info1, info2, info3
```

- **Exemplo:**

```
dias_uteis = ('Segunda-feira', 'Terça-feira',
'Quarta-feira', 'Quinta-feira', 'Sexta-feira')
```

Dicionário é uma coleção **não ordenada** de pares **chave-valor**.

Diferentemente de **listas** e **strings** em que cada item é acessado por um **índice numérico**, cada item em um dicionário é acessado por uma **chave** (valor de qualquer tipo que seja **imutável** como, por exemplo, **string** ou **numérico** ou **tupla**).

Usa-se dicionário quando se deseja obter a informação por meio de uma **chave**, quando a ordem dos ítens não é importante.

DICIONÁRIOS

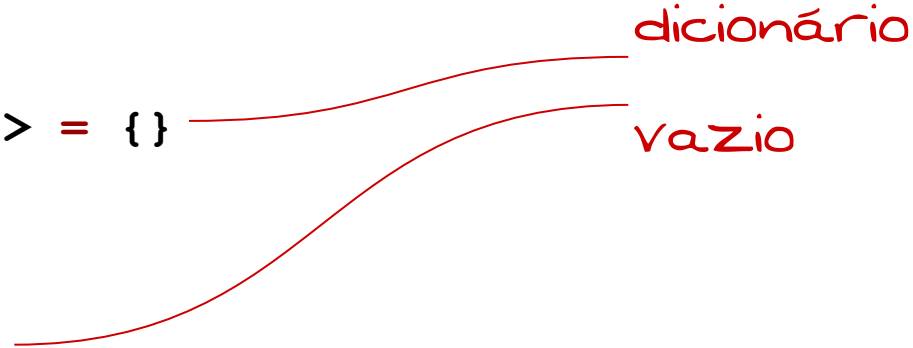
Para criar um dicionário vazio:

- **Sintaxe:**

```
<variável> = {}
```

- **Exemplo:**

```
pessoa = {}
```



DICIONÁRIOS

Para criar um dicionário com ítens:

- **Sintaxe:**

```
<variável> = {<chave1>: <valor1>, <chave2>: <valor2>, }
```

- **Exemplo:**

```
pessoa = {'nome': 'Alice', 'curso': 'Básico 1', 'nota': 8.5}
```

A quantidade de ítens em um dicionário pode variar, pois não é um tipo imutável.
Os valores podem ser de quaisquer tipo, inclusive outro dicionário, lista, etc

DICIONÁRIOS

É possível criar um dicionário a partir do comando **dict()** aplicado a uma **lista** de **tuplas de dois ítens**:

```
>>> nome = ('nome', 'Alice')
>>> curso = ('curso', 'Básico 1')
>>> nota = ('nota', 8.5)

>>> lista = [nome, curso, nota]
>>> print(lista)
[('nome', 'Alice'), ('curso', 'Básico 1'), ('nota', 8.5),]

>>> pessoa = dict(lista)
>>> pessoa
{'nome': 'Alice', 'curso': 'Básico 1', 'nota': 8.5}
```

DICIONÁRIOS

- Para criar um novo valor ou atualizar o valor de um item do dicionário:

```
    <variavel>[<chave>] = <valor>
```

```
>>> pessoa = {'nome': 'Alice', 'curso': 'Básico 1', 'nota': 45.8,}
>>> pessoa['nota'] = 8
>>> pessoa
{'nome': 'Alice', 'curso': 'Básico 1', 'nota': 8, }

>>> pessoa['idade'] = 19
>>> pessoa
{'nome': 'Alice', 'curso': 'Básico 1', 'nota': 8, 'idade': 19}
```

pyladies São Paulo

AGORA É COM VOCÊ

No terminal do Python:

Crie um dicionário que contenha a tradução da palavra GATO para os idiomas inglês (cat), espanhol (gato) e francês (chat). Os idiomas são as chaves no dicionário.

Escreva o comando para imprimir (print) a palavra **gato** em **francês**.

RESPOSTAS POSSÍVEIS

```
>>> gatos = {}
>>> gatos['inglês'] = 'cat'
>>> gatos['espanhol'] = 'gato'
>>> gatos['francês'] = 'chat'
```

ou

```
>>> gatos = {
    'inglês': 'cat',
    'espanhol': 'gato',
    'francês': 'chat', }
```

```
>>> gatos['francês']
chat
```

DICIONÁRIOS

pyladies São Paulo

- Para **apagar** um item, pode ser feito com **del()** ou com **.pop()**

```
# del <variavel>[<chave>]
>>> pessoa = {'nome': 'Alice', 'curso': 'Básico 1', 'país': 'Inglaterra'}
>>> del pessoa['nome']
>>> pessoa
{'curso': 'Básico 1', 'país': 'Inglaterra'}

# <variavel>.pop(<chave>)
>>> pais = pessoa.pop('país')

>>> pessoa
{'curso': 'Básico 1'}
>>> pais
Inglaterra
```

a diferença é que "pop" além de remover do dicionário também retorna o valor

DICIONÁRIOS

- Para acrescentar e/ou atualizar **vários** ítens ao dicionário:
- **Sintaxe:**

```
<variavel>.update(<dicionario>)
```

```
>>> pessoa = {'nome': 'Laura', 'curso': 'Básico 2', 'nota': 78,}

>>> pessoa.update({'nota': 72, 'idade': 30})

>>> pessoa
{'nome': 'Laura', 'curso': 'Básico 2', 'nota': 72, 'idade': 30}
```

DICIONÁRIOS

- Para obter o valor de um item e retornar um valor default caso o item não exista:
- **Sintaxe:**

```
<variável>.get(<chave>, <valor pré-definido>)
```

```
>>> pessoa = {'nome': 'Laura', 'curso': 'Básico 2', 'nota': 72,
'idade': 30}

>>> pessoa.get('cidade', 'Londres')
Londres

>>> pessoa.get('nome', 'Branca de Neve')
Laura
```

AGORA É COM VOCÊ

No terminal do Python:

Crie um dicionário que contenha a tradução da palavra GATO para os idiomas inglês (cat), espanhol (gato) e francês (chat). Os idiomas são as chaves no dicionário.

Execute **comandos (e visualize o resultado)** para:

1. incluir gato em português
2. incluir com apenas 1 comando dois novos ítens: gato em alemão (Katze) e gato em italiano (gatto)
3. apagar gato em inglês
4. obter o valor da chave **russo** e retornar "desconhecido", caso não exista o valor

DICIONÁRIOS

RESPOSTAS

```
>>> gatos = {'inglês': 'cat', 'francês': 'chat', 'espanhol': 'gato', }

>>> gatos['português'] = 'gato'
{'inglês': 'cat', 'francês': 'chat', 'espanhol': 'gato', 'português':
'gato', }

>>> gatos.update({'alemão': 'Katze', 'italiano': 'gatto'})
{'inglês': 'cat', 'francês': 'chat', 'espanhol': 'gato', 'português':
'gato', 'alemão': 'Katze', 'italiano': 'gatto'}

>>> gatos.pop('inglês')
>>> # del gatos['inglês']
{'francês': 'chat', 'espanhol': 'gato', 'português': 'gato', 'alemão':
'Katze', 'italiano': 'gatto'}

>>> gatos.get('russo', 'desconhecido')
desconhecido
```

pyladies São Paulo

- Para obter, ***respectivamente***, todos os **ítens**, os **valores**, as **chaves**, usamos as seguintes funções de dicionários: `.items()`, `.values()`, `.keys()`

```
>>> gatos = {'inglês': 'cat', 'francês': 'chat', 'espanhol': 'gato', }

>>> gatos.items()
dict_items([('inglês','cat'),('francês','chat'),('espanhol','gato')])

>>> gatos.values()
dict_values(['cat', 'chat', 'gato'])

>>> gatos.keys()
dict_keys(['inglês', 'francês', 'espanhol'])
```

dict_items,
dict_values,
dict_keys
são tipos iteráveis

pyladies São Paulo

dict_items são tipos iteráveis como as listas. Então é possível aplicar o comando **for**

```
>>> gatos = {'inglês': 'cat', 'francês': 'chat', 'espanhol': 'gato', }

>>> gatos.items()
dict_items([('inglês','cat'),('francês','chat'),('espanhol','gato')])


>>> for item in gatos.items():
>>>     print(item)
('inglês', 'cat')
('francês', 'chat')
('espanhol', 'gato')
```

PROGRAMANDO EM PYTHON

DICIONÁRIOS

dict_values são tipos iteráveis como as listas. Então é possível aplicar o comando **for**

```
>>> gatos = {'inglês': 'cat', 'francês': 'chat', 'espanhol': 'gato', }

>>> gatos.values()
dict_values(['cat', 'chat', 'gato'])


>>> for valor in gatos.values():
>>>     print(valor)
cat
chat
gato
```

pyladies São Paulo

dict_keys são tipos iteráveis como as listas. Então é possível aplicar o comando **for**

```
>>> gatos = {'inglês': 'cat', 'francês': 'chat', 'espanhol': 'gato', }

>>> gatos.keys()
dict_keys(['inglês', 'francês', 'espanhol'])


>>> for key in gatos.keys():
>>>     print(key)
inglês
francês
espanhol
```

PROGRAMANDO EM PYTHON

DICIONÁRIOS

pyladies São Paulo

```
>>> gatos = {'inglês': 'cat', 'francês': 'chat', 'espanhol': 'gato'}
```

chave valor

```
>>> for idioma, palavra in gatos.items():
>>>     if palavra.startswith('c'):
>>>         print(palavra)
cat
chat

>>> for idioma, palavra in gatos.items():
>>>     if palavra.endswith('o'):
>>>         print(palavra)
gato
```

pyladies São Paulo

AGORA É COM VOCÊ

Em um arquivo (**b2_exercicio_1.py**):

- Crie um dicionário que contenha a tradução da palavra GATO para os idiomas inglês (cat), espanhol (gato) , francês (chat), alemão (Katze), italiano (gatto).

- Escreva os comandos que imprimirão a palavra gato em cada um dos idiomas no seguinte formato:

Exemplo:

- `gato (português)`

RESPOSTAS POSSÍVEIS

```
for idioma, palavra in gatos.items():
    print(palavra + '(' + idioma + ')')
```

ou

```
for idioma, palavra in gatos.items():
    print('{valor1} ({valor2})'.format(valor1=palavra, valor2=idioma))
```

ou

```
for idioma, palavra in gatos.items():
    print('{} ({})'.format(palavra, idioma))
```

ou

```
for idioma, palavra in gatos.items():
    print('%s (%s)' % (palavra, idioma))
```

pyladies São Paulo

- Para ordenar **iteráveis (listas, itens de dicionários, valores de dicionários, chaves de dicionários entre outros)**, usamos a função **sorted()**:
- **Sintaxe:**

```
sorted(<variável>)
sorted(<variável>, reverse=True)
```

crescente

decrescente

- **Exemplo:**

```
>>> gatos = {'inglês': 'cat', 'francês': 'chat', 'espanhol': 'gato', }

>>> sorted(gatos.items())
[('espanhol', 'gato'), ('francês', 'chat'), ('inglês', 'cat')]

>>> sorted(gatos.items(), reverse=True)
[('inglês', 'cat'), ('francês', 'chat'), ('espanhol', 'gato')]
```

Em um arquivo **b2_exercicio_2.py**

AGORA É COM VOCÊ

Use o dicionário:

```
gatos =  {
    'inglês': 'cat',
    'francês': 'chat',
    'espanhol': 'gato',
    'alemão': 'Katze',
    'italiano': 'gatto',}
```

Faça um programa que imprima:

1) Os **ítens** do dicionário em ordem crescente (Dica: use a função **sorted()**)
2) Os **valores** do dicionário em ordem decrescente
3) Em ordem crescente, as **chaves** que terminem em 'ês'
4) Os **valores** dos ítens cujas **chaves** que não terminem em 'ês'

RESPOSTAS POSSÍVEIS

1) Os **ítens** do dicionário em ordem crescente

```
for item in sorted(gatos.items()):
    print(item)
```

ou

```
print(sorted(gatos.items()))
```

RESPOSTAS POSSÍVEIS

2) Os **valores** do dicionário em ordem decrescente

```
valores_em_ordem_decrescente = sorted(gatos.values(), reverse=True)
for valor in valores_em_ordem_decrescente:
    print(valor)
```

ou

```
print(sorted(gatos.values(), reverse=True))
```

reverse é um parâmetro opcional da função sorted

RESPOSTA POSSÍVEL

3) Em ordem crescente, imprime as **chaves** que terminem em 'ês'

```
for idioma in sorted(gatos.keys()):
    if idioma.endswith('ês'):
        print(idioma)
```

4) Imprime os **valores** dos ítens cujas chaves que não terminam em 'ês'

```
for idioma in sorted(gatos.keys()):
    if not idioma.endswith('ês'):
        print(gatos[idioma])
```

LISTAS

- **Listas: uma sequência ou coleção ordenada de valores.**

Seus ítens podem ser do mesmo tipo ou de tipos diferentes entre eles.

- **Sintaxe:**

```
<variável> = [valor1, valor2, valor3]
```

- **Exemplos:**

Lista de strings

```
>>> cursos = ['Básico 1', 'Básico 2', 'NPL', 'Pandas', ]
```

Lista de ítens de tipos diferentes

```
>>> meubicho = ['Gato', 9, True]
```

A quantidade de ítens em um dicionário pode variar, pois não é um tipo imutável.

Os valores podem ser de quaisquer tipo, inclusive outra lista, dicionário etc

- **Exemplos:**

```
>>> atleta1 = {'nome': 'Alice', 'solo': 8.5, 'trave': 7.5}
>>> atleta2 = {'nome': 'Helena', 'solo': 8.5, 'trave': 4.5}
>>> atleta3 = {'nome': 'Julia', 'solo': 9.5, 'trave': 8.5}
>>> atleta4 = {'nome': 'Juana', 'solo': 3.5, 'trave': 7.5}
```

```
>>> atletas = [atleta1, atleta2, atleta3, atleta4]
```

Em um arquivo **b2_exercicio_3.py**

AGORA É COM VOCÊ

Represente os dados de alunas e suas notas das seguintes disciplinas: Matemática e Ciências, usando dicionários.
Um dicionário por aluna.

Depois crie uma lista de alunas.

Faça um programa:

1) Imprima os dados de cada aluna (use for)
2) Imprima o nome e nota de Matemática de cada uma das as alunas (use for)

pyladies São Paulo

RESPOSTA POSSÍVEL

```
aluna1 = {'nome': 'Alice', 'Matemática': 8.5, 'Ciências': 7.5}
aluna2 = {'nome': 'Helena', 'Matemática': 8.5, 'Ciências': 4.5}
aluna3 = {'nome': 'Julia', 'Matemática': 9.5, 'Ciências': 8.5}
aluna4 = {'nome': 'Juana', 'Matemática': 3.5, 'Ciências': 7.5}

alunas = [aluna1, aluna2, aluna3, aluna4]

for aluna in alunas:
    print(aluna)
```

pyladies São Paulo

RESPOSTA POSSÍVEL

```
aluna1 = {'nome': 'Alice', 'Matemática': 8.5, 'Ciências': 7.5}
aluna2 = {'nome': 'Helena', 'Matemática': 8.5, 'Ciências': 4.5}
aluna3 = {'nome': 'Julia', 'Matemática': 9.5, 'Ciências': 8.5}
aluna4 = {'nome': 'Juana', 'Matemática': 3.5, 'Ciências': 7.5}

alunas = [aluna1, aluna2, aluna3, aluna4]

for aluna in alunas:
    print(aluna['nome'], aluna['Matemática'])
```

PROGRAMANDO EM PYTHON

LIST COMPREHENSION

Uma expressão entre colchetes que usa as palavras reservadas **for** e **in** para criar uma lista **processando** e **filtrando** elementos de um ou mais iteráveis.

- **Sintaxe:**

```
<resultado> = [<expressao> for <item> in <iteravel>]

<resultado> = [<expressao> for <item> in <iteravel> <filtragem>]


<filtragem> é opcional
```

LIST COMPREHENSION

- **Sintaxe:**

```
<resultado> = [<expressao> for <item> in <iteravel>]
```

é equivalente a

```
lista_resultante = []
for <item> in <iteravel>:
    lista_resultante.append(<expressao>)
```

PROGRAMANDO EM PYTHON

LIST COMPREHENSION

- **Sintaxe:**

```
<resultado> = [<expressao> for <item> in <iteravel> <filtragem>]
```

é equivalente a

```
lista_resultante = []
for <item> in <iteravel>:
    <filtragem>
        lista_resultante.append(<expressao>)
```

Exemplo:

```
lista_resultante = []
for num in range(1, 10):
    if num % 2 == 0:
        lista_resultante.append(num)
```

PROGRAMANDO EM PYTHON

LIST COMPREHENSION

- **Sintaxe:**

```
<resultado> = [<expressao> for <item> in <iteravel>]
```

- **Exemplos:**

```
>>> nova_lista1 = [letra.upper() for letra in 'abcedefghijkl']
>>> nova_lista1
['A', 'B', 'C', 'E', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L']

>>> nova_lista2 = [float(num) for num in range(1,11)]
>>> nova_lista2
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]
```

PROGRAMANDO EM PYTHON

LIST COMPREHENSION

pyladies São Paulo

- **Sintaxe:**

```
<lista_resultante> = [<comando> for <item> in <iteravel> <filtragem>]
```

- **Exemplos:**

```
>>> nova_lista1 = [letra.upper() for letra in 'abcedefghi' if letra < 'e']
>>> nova_lista1
['A', 'B', 'C', 'E', 'D']


>>> nova_lista2 = [float(num) for num in range(1,11) if num % 2 == 1]
>>> nova_lista2
[1.0, 3.0, 5.0, 7.0, 9.0]
```

AGORA É COM VOCÊ

Em um arquivo **b2_exercicio_4.py**
Dada a tabela abaixo:

Filme	Nota Rotten Tomatoes	Nota IMDB	Nota Google
Roma (2018)	96	78	77
O Clube da Felicidade E da Sorte (1993)	86	76	88
Que Horas Ela Volta? (2015)	97	78	92
Flor do Deserto (2010)	50	74	94

Faça um programa:

1) Represente a tabela em formato de lista de dicionários, incluindo a sua nota pessoal para cada filme
2) Calcule a média das notas de cada filme e insira média como novo dado de cada filme (atualizar dicionários)
3) Selecione os filmes cuja média é **menor que 85** (list comprehension)

LISTAS

RESPOSTA POSSÍVEL

```
filme1 = {'título': 'Roma (2018)', 'tomatoes': 96, 'imdb': 78, 'google': 77,
'minha opinião': 80}
filme2 = {'título': 'O Clube da Felicidade E da Sorte (1993)', 'tomatoes':
89, 'imdb': 76, 'google': 88, 'minha opinião': 100}
filme3 = {'título': 'Que Horas Ela Volta? (2015)', 'tomatoes': 97, 'imdb':
78, 'google': 92, 'minha opinião': 80}
filme4 = {'título': 'Flor do Deserto (2010)', 'tomatoes': 72, 'imdb': 74,
'google': 94, 'minha opinião': 100}

lista_de_filmes = [filme1, filme2, filme3, filme4]

continua
```

LISTAS

RESPOSTAS POSSÍVEIS

```
for filme in lista_de_filmes:
    filme['média'] = (filme['tomatoes'] + filme['imdb'] + filme['google'] +
filme['minha opinião']) / 4
```

ou

LISTAS

RESPOSTAS POSSÍVEIS

```
for filme in lista_de_filmes:
    filme['média'] = 0
    for coluna in ['tomatoes', 'imdb', 'google', 'minha opinião']:
        filme['média'] += filme[coluna]
    filme['média'] = filme['média'] / 4
```

ou

```
for filme in lista_de_filmes:
    filme['média'] = sum([filme[coluna]
                          for coluna in filme.keys()
                          if coluna != 'título'])/4
```

RESPOSTA POSSÍVEL

```
selecao = [filme for filme in lista_de_filmes if filme['média'] < 85]
```

ARQUIVOS: LEITURA E ESCRITA DE ARQUIVO TEXTO

Os programas trabalham com entrada e saída de dados.
Até então usamos `input()` e `print()` para fazer respectivamente entrada e saída.

Arquivos são uma das formas de fazer entrada e saída, ou seja, leitura e escrita de arquivos.

Por exemplo:

Entrada: quando programa lê um texto de um arquivo para contar a quantidade de cada palavra no texto.

Saída: guardar um relatório em arquivo.

PROGRAMANDO EM PYTHON

ARQUIVOS: ESCRITA

Para fazer a escrita de um arquivo:

w é de write (escrita)
apaga o conteúdo que já
está no arquivo antes
de escrever

- **Sintaxe:**

```
with open(<caminho do arquivo>, 'w')  as <variavel arquivo>:
    <variavel arquivo>.write(<conteudo do arquivo>)
```

- **Exemplo:**

```
with open('/Mesa/meuarquivo.txt', 'w') as arquivo:
    arquivo.write('Olá!')
    arquivo.write('Estou aprendendo escrita em arquivo com Python')
```

ARQUIVOS: LEITURA

r é de read (leitura)

Para fazer a leitura de um arquivo:

- **Sintaxe:**

```
with open(<caminho do arquivo>, 'r') as <variavel arquivo>:
    <conteudo> = <variavel arquivo>.read()
```

- **Exemplo:**

```
with open('/Desktop/meuarquivo.txt', 'r') as arquivo:
    conteudo = arquivo.read()
    print(conteudo)
```

csv são arquivos que contém dados tabulados.

ARQUIVOS: ESCRITA DE ARQUIVO CSV

Use `newline=''` para que funcione em qualquer sistema operacional

Para fazer a escrita de um arquivo csv:

- **Sintaxe:**

```
import csv
with open(<caminho do arquivo>, 'w', newline='') as <variavel arquivo>:
    writer = csv.DictWriter(
        <variavel arquivo>, fieldnames=<lista de colunas>)
    writer.writeheader()
    writer.writerow(<dicionario coluna e valor>)
    writer.writerow(<dicionario coluna e valor>)
    writer.writerow(<dicionario coluna e valor>)
```

If newline='' is not specified, newlines embedded inside quoted fields will not be interpreted correctly, and on platforms that use \r\n linendings on write an extra \r will be added. It should always be safe to specify newline='', since the csv module does its own (universal) newline handling.

ARQUIVOS: ESCRITA DE ARQUIVO CSV

pyladies São Paulo

as chaves dos dicionários tem que ter os mesmos valores das colunas

- **Exemplo:**

```
import csv
with open('pessoas.csv', 'w', newline='') as csvfile:
    colunas = ['nome', 'nota1', 'nota2']
    writer = csv.DictWriter(csvfile, fieldnames=colunas)
    writer.writeheader()
    writer.writerow({'nome': 'Annie', 'nota1': 8.5, 'nota2': 7.5,})
    writer.writerow({'nome': 'Claire', 'nota1': 9.5, 'nota2': 10,})
```

ARQUIVOS: LEITURA DE ARQUIVO CSV

Para fazer a leitura de um arquivo csv:

- **Sintaxe:**

```
import csv
with open(<caminho do arquivo>, newline='') as <variavel arquivo>:
    reader = csv.DictReader(<variavel arquivo>)
    for linha in reader:
        print(linha)
        <comandos que usem linha>
```

ARQUIVOS: LEITURA ARQUIVO CSV

Para fazer a leitura de um arquivo csv:

- **Exemplo:**

```
import csv
with open('pessoas.csv', newline='') as arquivo:
    reader = csv.DictReader(arquivo)
    for linha in reader:
        print(linha['nome'], linha['email'])
```

AGORA É COM VOCÊ

Em um módulo chamado **b2_exercicio_escrever_csv.py,**
copie as funções mostradas nos próximos slides,
faça uma **função** que escreva um arquivo csv cujas colunas são:
Filme, Nota 1 e Nota 2,
use todas estas funções e faça um **programa** que pergunte títulos de filmes, pergunte notas para os filmes de duas pessoas e escreva o resultado em um arquivo.
O arquivo csv poderia ficar assim, por exemplo:

Filme	Nota 1	Nota 2
O Nome da Rosa	9.5	4.0
Para sempre Alice	5.5	8.5
Morro dos ventos uivantes	6.3	8.7
Clarice	6.8	5.8

Funções para usar no exercício

```
def pedir_nota_para_filme(titulo):
    nota = input('Entre com uma nota para o filme "{}": '.format(titulo))
    if nota != '':
        return float(nota)
    return 0


def pedir_notas_para_os_filmes(lista_de_titulos):
    lista = []
    for titulo in lista_de_titulos:
        filme = {}
        filme['titulo'] = titulo
        filme['nota1'] = pedir_nota_para_filme(titulo)
        filme['nota2'] = pedir_nota_para_filme(titulo)
        filme['media'] = (filme['nota1'] + filme['nota2']) / 2
        lista.append(filme)
    return lista
```

Funções para usar no exercício

```
def pedir_titulos_de_filmes():
    filmes = []
    while True:
        titulo = input('Entre com o título de um filme: ')
        if titulo == "":
            break
        filmes.append(titulo)
    return filmes
```

RESPOSTA POSSÍVEL (parte 1/3)

```
# importacao dos pacotes e módulos que serão usados

import csv


# aqui ficam as declarações das funções mencionadas anteriormente
```

RESPOSTA POSSÍVEL (parte 2/3)

```
def escrever_dados_em_arquivo_csv(nome_arquivo, lista_filmes):
    colunas = ['Filme', 'Nota 1', 'Nota 2']
    with open(nome_arquivo, 'w', newline='') as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=colunas)
        writer.writeheader()
        for filme in lista_filmes:
            linha = {}
            linha['Filme'] = filme['titulo']
            linha['Nota 1'] = filme['nota1']
            linha['Nota 2'] = filme['nota2']
            writer.writerow(linha)
```

as chaves do dicionário tem que corresponder às colunas

RESPOSTA (parte 3/3)

```
titulos = pedir_titulos_de_filmes()
filmes_com_notas = pedir_notas_para_os_filmes(titulos)

escrever_dados_em_arquivo_csv('filmes.csv', filmes_com_notas)
```

pyladies São Paulo

AGORA É COM VOCÊ

Copie **b2_exercicio_escrever_csv.py, renomeie para b2_exercicio_ler_csv.py.**

Faça ajustes criando mais uma função que leia dados de um arquivo csv (**ler_dados_de_arquivo_csv**) que contenha as colunas: Filme, Nota 1, Nota 2 e retorne a lista dos dados

RESPOSTA POSSÍVEL

```
import csv

def ler_dados_de_arquivo_csv(nome_arquivo):
    resultado = []
    with open(nome_arquivo, newline='') as arquivo:
        reader = csv.DictReader(arquivo)
        for linha in reader:
            resultado.append(linha)
    return resultado

# dados é uma lista de dicionários
filmes = ler_dados_de_arquivo_csv('filmes.csv')
print(filmes)
```

MÓDULOS: são os arquivos .py

PACOTES: contém os módulos e obrigatoriamente um dos módulos tem que ser **__init__.py** mesmo que seu conteúdo é vazio.

Conforme vamos escrevendo as instruções, o programa vai ficando muito longo e difícil de dar manutenção.
Sendo assim, um programa pode ser composto por vários módulos.

A origem dos pacotes e módulos de um programa pode ser:

1. da biblioteca padrão do Python, ou seja, que é instalada ao instalar Python (https://docs.python.org/3/library/);
2. de terceiros, ou seja, que tem que ser instalado a parte (https://pypi.python.org/pypi, github, etc.);
3. os que você criou para seu programa

A PEP8 (pep eight) (www.python.org/dev/peps/pep-0008/) – um guia de estilo de programação para Python – recomenda a ordem de importação acima. E cada grupo separado por uma linha.
Além disso, todas as importações devem estar no início do módulo, como boa prática.

Uma aplicação pode ser composta por mais de um módulo.

`raspagem_de_dados.py`
`relatorios.py`

Em **`raspagem_de_dados.py`** há a função **baixar_dados**.
Em **`relatorios.py`** há a função **gerar_graficos.**

Além destes, há dois outros módulos: **`programa1.py`** e **`programa2.py`**.
Mostram formas diferentes de importá-los e usá-los.

pyladies São Paulo

```
# conteudo do programa1 (por exemplo)

import relatorios
import raspagem_de_dados
dados = raspagem_de_dados.baixar_dados()
relatorios.gerar_graficos(dados)
```

```
# conteudo do programa2 (por exemplo)

from raspagem_de_dados import baixar_dados
from relatorios import gerar_graficos
dados = baixar_dados(10)
gerar_graficos(dados)
```

Para manipular arquivos é necessário usar módulo os da biblioteca padrão do Python.

```
import os
```

Para listar os arquivos de uma pasta. Retorna uma lista com o conteúdo da pasta.

- **Sintaxe:**

```
os.listdir(<caminho da pasta>)
```

- **Exemplo:**

```
os.listdir('/users/pyladies/desktop/')
```

Para criar todas as pastas de um caminho.

- **Sintaxe:**

```
os.makedirs(<caminho da pasta>)
```

- **Exemplo:**

```
os.makedirs('/users/pyladies/desktop/curso básico 2')
```

Para saber se um caminho é uma pasta

- **Sintaxe:**

```
os.path.isdir(<caminho da pasta>)
```

- **Exemplo:**

```
os.path.isdir('/users/pyladies/desktop/curso básico 2')
```

Para saber se um caminho é um arquivo

- **Sintaxe:**

```
os.path.isfile(<caminho da pasta>)
```

- **Exemplo:**

```
os.path.isfile('/users/pyladies/desktop/curso básico 2/texto.txt')
```

ONDE ESTUDAR ONLINE

- www.codecademy.com/pt
- www.sololearn.com/Course/Python
- pythontutor.com
- www.pycursos.com/python-para-zumbis
- coursera.org
- www.urionlinejudge.com.br/judge/pt/login

REFERÊNCIAS

- http://wiki.python.org.br/PrincipiosFuncionais
- Curso Python para Zumbis
- Curso “An Introduction to Interactive Programming in Python” - Coursera
- http://www.peachpit.com/articles/article.aspx?p=13'Básico 1'792&seqNum=6
- https://docs.python.org/release/2.3.5/whatsnew/section-slices.html
- http://www.bbc.co.uk/education/guides/zqh49j6/revision/3
- https://www.youtube.com/watch?v=SYioCdLPmfw
- https://pt.wikibooks.org/wiki/Python/Conceitos_b%C3%A1sicos/Tipos_e_operadores
- http://www.dcc.ufrj.br/~fabiom/mab'Básico 2'5/02tipos.pdf
- http://pt.stackoverflow.com/questions/62844/como-se-insere-n%C3%BAmeros-complexosem-python
- www.arquivodecodigos.net/principal/dicas_truques_categoria2.php?linguagem='Básico 1'&categoria1=1&categoria2=59

REFERÊNCIAS

- www.arquivodecodigos.net/principal/dicas_truques_categoria2.php?linguagem='Básico 1'&categoria1=1&categoria2=51
- www.dotnetperls.com/lower-python
- https://pt.wikipedia.org/wiki/Algoritmo
- http://wiki.python.org.br/SoftwarePython
- http://wiki.python.org.br/EmpresasPython
- https://powerpython.wordpress.com/20'Básico 1'/03/16/programas-e-jogos-feitos-em-python/
- http://tutorial.djangogirls.org/pt/python_installation/index.html
- https://powerpython.wordpress.com/20'Básico 1'/03/19/aula-python-17-estrutura-de-decisao/
- https://under-linux.org/entry.php?b=1371
- http://aprenda-python.blogspot.com.br/2009/10/nova-formatacao-de-strings.html
- https://docs.python.org/3/library/string.html#formatspec
- https://www.python.org/dev/peps/pep-3101/
- https://novatec.com.br/livros/automatize-tarefas-macantes-com-python/

MÃO NA MASSA

Procurem trabalhar em grupo e
trocar informações.

Tendo dúvidas, estamos à disposição

PyLadiesSP

PyLadiesSãoPaulo

PyLadiesSP

@PyLadiesSP

Meetup PyLadiesSP

saopaulo@pyladies.com

Mulheres que amam programar e ensinar Python