Imprimir

FOCO NO MERCADO DE TRABALHO

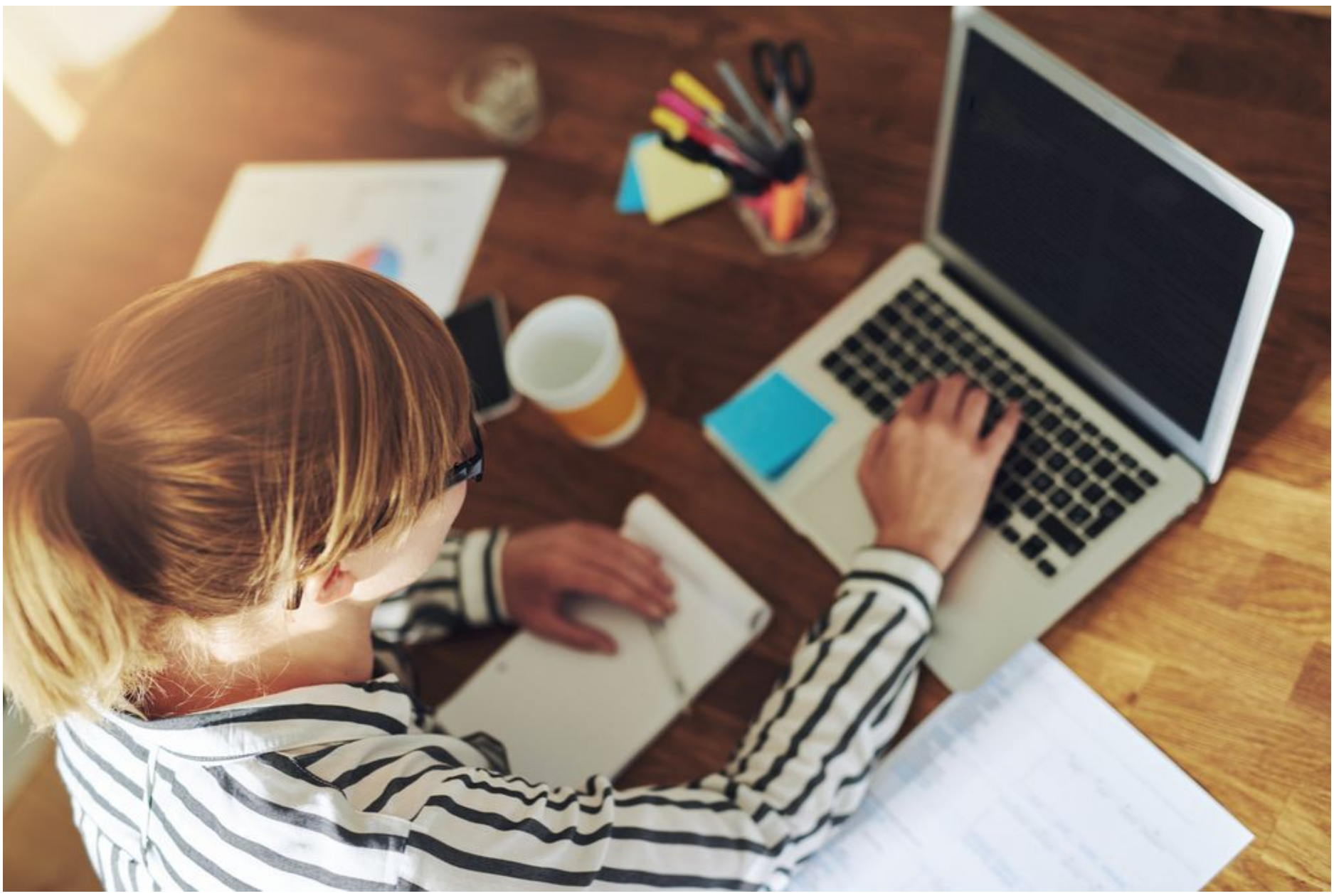
# ESTRUTURAS DE DADOS EM PYTHON

Vanessa Cadan Scheffer

Ver anotações 0

## UTILIZANDO AS ESTRUTURAS DE DADOS OFERECIDAS EM PYTHON

Em Python não existe somente uma forma de implementar uma solução, uma vez que são oferecidas uma série de estruturas de dados.



Fonte: Shutterstock.

**Deseja ouvir este material?**

Áudio disponível no material digital.

DESAFIO

Muitos sistemas disponilizam as informações por meio de uma API (interface de programação de aplicações), as quais utilizam arquivos JSON (notação de objetos JavaScript) para disponibilizar os dados. Um arquivo JSON é composto por

elementos na forma chave-valor, o que torna esse formato leve de ser transferido e fácil de ser manipulado.

Como desenvolvedor em uma empresa de consultoria, você foi alocado para atender um cliente que organiza processos seletivos (concurso público, vestibular, etc.). Essa empresa possui um sistema web no qual os candidatos de um certo processo seletivo fazem a inscrição e acompanham as informações. Um determinado concurso precisou ser adiado, razão pela qual um processo no servidor começou a enviar e-mails para todos os candidatos inscritos. Porém, em virtude da grande quantidade de acessos, o servidor e a API saíram do ar por alguns segundos, o que gerou um rompimento na criação do arquivo JSON com a lista dos candidatos já avisados da alteração.



Por causa do rompimento do arquivo, foram gerados dois novos arquivos, razão pela qual, desde então, não se sabe nem quem nem quantas pessoas já receberam o aviso. Seu trabalho, neste caso, é criar uma função que, com base nas informações que lhe serão fornecidas, retorne uma lista com os e-mails que ainda precisam ser enviados.

Para esse projeto, você e mais um desenvolvedor foram alocados. Enquanto seu colega trabalha na extração dos dados da API, você cria a lógica para gerar a função. Foi combinado, entre vocês, que o programa receberá dois dicionários referentes aos dois arquivos que foram gerados. O dicionário terá a seguinte estrutura: três chaves (nome, email, enviado), cada uma das quais recebe uma lista de informações; ou seja, as chaves *nome* e *email* estarão, respectivamente, associadas a uma lista de nomes e uma de emails. por sua vez, a chave *enviado* estará associada a uma lista de valores booleanos (True-False) que indicará se o e-mail já foi ou não enviado.

Veja um exemplo.

```
dict_1 = {
  'nome': ['nome_1'],
```

```
  'email': ['email_1'],
  'enviado': [False]
}
```

Considerando que você receberá duas estruturas, conforme foi mencionado, crie uma função que trate os dados e retorne uma lista com os e-mails que ainda não foram enviados.



RESOLUÇÃO

Você foi alocado para um projeto no qual precisa implementar uma função que, com base em dois dicionários, retorne uma lista com os e-mails que precisam ser enviados aos condidatos de um concurso. Os dicionários serão fornecidos para você no formato combinado previamente. Ao mesmo tempo em que seu colega trabalha na extração dos dados da API, foi-lhe passado uma amostra dos dados para que você comece a trabalhar na lógica da função. Os dados passados foram:

```
dados_1 = {
  'nome': ['Sonia Weber', 'Daryl Lowe', 'Vernon Carroll', 'Basil Gilliam',
'Mechelle Cobb', 'Edan Booker', 'Igor Wyatt', 'Ethan Franklin', 'Reed
Williamson', 'Price Singleton'],
  'email': ['Lorem.ipsum@cursusvestibulumMauris.com', 'auctor@magnis.org',
'at@magnaUttincidunt.org', 'mauris.sagittis@sem.com',
'nec.euismod.in@mattis.co.uk', 'egestas@massaMaurisvestibulum.edu',
'semper.auctor.Mauris@Crasdolordolor.edu', 'risus.Quisque@condimentum.com',
'Donec@nislMaecenasmalesuada.net', 'Aenean.gravida@atrisus.edu'],
  'enviado': [False, False, False, False, False, False, False, True, False,
False]
}

  dados_2 = {
  'nome': ['Travis Shepherd', 'Hoyt Glass', 'Jennifer Aguirre', 'Cassady Ayers',
'Colin Myers', 'Herrod Curtis', 'Cecilia Park', 'Hop Byrd', 'Beatrice Silva',
'Alden Morales'],
  'email': ['at@sed.org', 'ac.arcu.Nunc@auctor.edu',
'nunc.Quisque.ornare@nibhAliquam.co.uk', 'non.arcu@mauriseu.com',
'fringilla.cursus.purus@erategetipsum.ca', 'Fusce.fermentum@tellus.co.uk',
'dolor.tempus.non@ipsum.net', 'blandit.congue.In@libero.com',
'nec.tempus.mauris@Suspendisse.com', 'felis@urnaconvalliserat.org'],
  'enviado': [False, False, False, True, True, True, False, True, True, False]
}
```

Com a estrutura do dicionário definida e com uma amostra dos dados, é possível começar a implementação. Vale lembrar não existe somente uma forma de implementar uma solução, ainda mais se tratando da linguagem Python, que oferece uma série de estruturas de dados. A seguir trazemos uma possível solução.

A função "extrair_lista_email" recebe como parâmetro dois dicionários de dados. Para que pudéssemos fazer a extração, criamos a lista_1 (linha 2), que consiste em uma lista de tuplas, na qual cada uma destas é composta por *nome*, *email*, *enviado*, exatamente nessa sequência. Para construir essa tupla, usamos a função *zip()*, passando com parâmetro a lista de nomes, de e-mails e o status do enviado, e transformamos seu resultado em uma lista.

Na linha 3 imprimos uma única tupla construída para que pudéssemos checar a construção.

Na linha 5, criamos uma segunda lista de tuplas, agora usando os dados do segundo dicionário.

Para termos a lista completa de dados, foi necessário juntar as duas construções, fizemos isso na linha 7, simplesmente usando o '+' para concatenar as duas listas.

Na linha 12 fizemos a "mágica" de extrair somente os e-mails usando uma *list comprehension*. Vamos entender: "dados" é uma lista de tuplas, conforme amostra que imprimimos. Cada item dessa lista é uma tupla. Quando selecionamos o item[1], estamos pegando o valor que ocupa a posição 1 da tupla, ou seja, o e-mail. Fazendo isso, iterando sobre todos os dados, teremos uma lista com todos os e-mails. No entanto, queremos somente os e-mails que ainda não foram enviados, razão pela qual o valor da posição 2 na tupla tem que ser *Falso*. Para fazer esse filtro, incluímos na listcomp uma estrutura condicional (if) que adiciona somente "if not item[2]", ou seja, somente se o item[2] não tiver valor *True*. Com essa construção, extraímos exatamente o que precisamos.

Na linha 14 retornamos a lista.

Veja que as funções e os objetos em Python facilitaram o trabalho. Com a função zip() conseguimos extrair cada registro do dicionário, depois, com uma simples concatenação, juntamos todos os dados e, com uma única linha, usando a *list comprehension*, criamos a lista com os critérios que foram estabelecidos.

Ver anotações

In [25]:

```
def extrair_lista_email(dict_1, dict_2):
    lista_1 = list(zip(dict_1['nome'], dict_1['email'],
dict_1['enviado']))
    print(f"Amostra da lista 1 = {lista_1[0]}")

    lista_2 = list(zip(dict_2['nome'], dict_2['email'],
dict_2['enviado']))

    dados = lista_1 + lista_2

    print(f"\nAmostra dos dados = \n{dados[:2]}\n\n")

    # Queremos uma lista com o email de quem ainda não recebeu o
aviso
    emails = [item[1] for item in dados if not item[2]]

    return emails


dados_1 = {
    'nome': ['Sonia Weber', 'Daryl Lowe', 'Vernon Carroll', 'Basil
Gilliam', 'Mechelle Cobb', 'Edan Booker', 'Igor Wyatt', 'Ethan
Franklin', 'Reed Williamson', 'Price Singleton'],
    'email': ['Lorem.ipsum@cursusvestibulumMauris.com',
'auctor@magnis.org', 'at@magnaUttincidunt.org',
'mauris.sagittis@sem.com', 'nec.euismod.in@mattis.co.uk',
'egestas@massaMaurisvestibulum.edu',
'semper.auctor.Mauris@Crasdolordolor.edu',
'risus.Quisque@condimentum.com', 'Donec@nislMaecenasmalesuada.net',
'Aenean.gravida@atrisus.edu'],
    'enviado': [False, False, False, False, False, False, False,
True, False, False]
}

dados_2 = {
    'nome': ['Travis Shepherd', 'Hoyt Glass', 'Jennifer Aguirre',
'Cassady Ayers', 'Colin Myers', 'Herrod Curtis', 'Cecilia Park',
'Hop Byrd', 'Beatrice Silva', 'Alden Morales'],
    'email': ['at@sed.org', 'ac.arcu.Nunc@auctor.edu',
'nunc.Quisque.ornare@nibhAliquam.co.uk', 'non.arcu@mauriseu.com',
'fringilla.cursus.purus@erategetipsum.ca',
'Fusce.fermentum@tellus.co.uk', 'dolor.tempus.non@ipsum.net',
'blandit.congue.In@libero.com', 'nec.tempus.mauris@Suspendisse.com',
'felis@urnaconvalliserat.org'],
    'enviado': [False, False, False, True, True, True, False, True,
True, False]
}
```

Ver anotações 0

```
emails = extrair_lista_email(dict_1=dados_1, dict_2=dados_2)
print(f"E-mails a serem enviados = \n {emails}")
```

```
Amostra da lista 1 = ('Sonia Weber',
'Lorem.ipsum@cursusvestibulumMauris.com', False)

Amostra dos dados =
[('Sonia Weber', 'Lorem.ipsum@cursusvestibulumMauris.com',
False), ('Daryl Lowe', 'auctor@magnis.org', False)]



E-mails a serem enviados =
 ['Lorem.ipsum@cursusvestibulumMauris.com',
'auctor@magnis.org', 'at@magnaUttincidunt.org',
'mauris.sagittis@sem.com', 'nec.euismod.in@mattis.co.uk',
'egestas@massaMaurisvestibulum.edu',
'semper.auctor.Mauris@Crasdolordolor.edu',
'Donec@nislMaecenasmalesuada.net',
'Aenean.gravida@atrisus.edu', 'at@sed.org',
'ac.arcu.Nunc@auctor.edu',
'nunc.Quisque.ornare@nibhAliquam.co.uk',
'dolor.tempus.non@ipsum.net', 'felis@urnaconvalliserat.org']
```

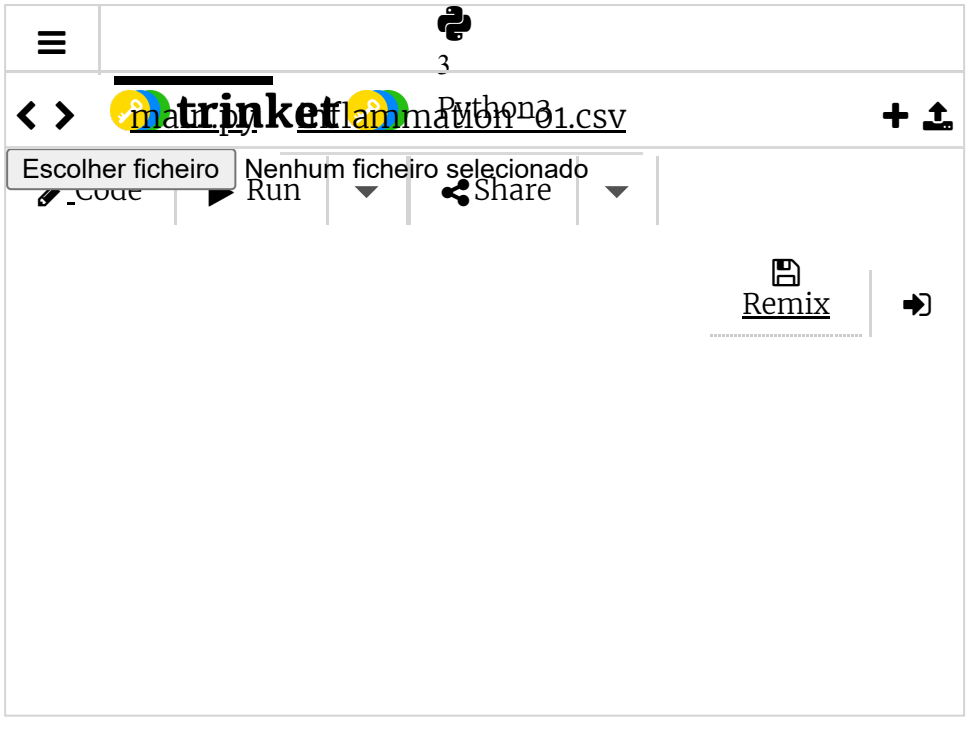
Ver anotações 0

DESAFIO DA INTERNET

Na página 116 do Capítulo 4 da seguinte obra, você encontra o exercício 6 (uso de listas). Nele, o autor propõe a construção de uma progressão aritmética com o uso de listas em Python.

BANIN, S. L. Python 3 - conceitos e aplicações: uma abordagem didática. São Paulo: Érica, 2018. Disponível em: https://bit.ly/2NJnf0w. Acesso em: 30 jun. 2020.

Utilize o emulador a seguir, para resolver o desafio!




Ver anotações 0